Oracle® Fusion Middleware

Application Adapter for J.D. Edwards OneWorld User's Guide for Oracle WebLogic Server

11*g* Release 1 (11.1.1)

E17058-01

April 2010

ORACLE®

Oracle Fusion Middleware Application Adapter for J.D. Edwards OneWorld User's Guide for Oracle WebLogic Server, 11*g* Release 1 (11.1.1)

E17058-01

Primary Author: Stefan Kostial

Contributing Authors: Sunil Gopal, Marian Jones, Vikas Anand, Sunil Wadhwa, Vishal Saxena, Vimmika Dinesh

Contents

7 Advanced User Tools

A Configuring J.D. Edwards OneWorld for Outbound and Inbound Processing

B Sample Files

Glossary

Index

Preface

Oracle Fusion Middleware Application Adapter for J.D. Edwards OneWorld User's Guide for Oracle WebLogic Server describes how to provide connectivity and integrate with J.D. Edwards OneWorld systems.

Audience

Oracle Fusion Middleware Application Adapter for J.D. Edwards OneWorld User's Guide for Oracle WebLogic Server is intended for those who integrate with J.D. Edwards OneWorld systems and develop applications.

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Enterprise Repository 11*g* Release 1 (11.1.1) documentation set:

- *Oracle Fusion Middleware Application Adapters Installation Guide for Oracle WebLogic Server*
- *Oracle Fusion Middleware Application Adapter Upgrade Guide for Oracle WebLogic Server*
- *Oracle Fusion Middleware Application Adapter Best Practices Guide for Oracle WebLogic Server*
- Oracle's Unified Method (OUM)

  A wealth of additional Governance information can be found within Oracle's Unified Method (OUM). OUM can be used by Oracle employees, Oracle Partner Network Certified Partners or Certified Advantage Partners, and Clients who either participate in the OUM Customer Program or are engaged on projects where Oracle provides consulting services. OUM is a web-deployed toolkit for planning, executing and controlling software development and implementation projects.

  For more information about OUM, see the OUM FAQ at

  http://my.oracle.com/portal/page/myo/ROOTCORNER/KNOWLEDGEAREAS1/BUSINESS_PRACTICE/Methods/Learn_about_OUM.html

Conventions

The following text conventions are used in this document:

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

1 Introduction

Oracle WebLogic Server connects to a J.D. Edwards OneWorld system through Oracle Application Adapter for J.D. Edwards OneWorld. Oracle Application Adapter for J.D. Edwards OneWorld provides connectivity and carries out interactions on a J.D. Edwards OneWorld system. This chapter discusses the following topics:

This chapter discusses the following topics:

1.1 Adapter Features

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

Oracle Application 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 Oracle Adapter J2CA. It can also be deployed as a Web services servlet and is referred to as Oracle Adapter Business Services Engine (BSE).

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

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

To support event functionality, channels are supported. A **channel** represents configured connections to particular instances of back-end or other types of systems.

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, which enables the user to choose the optimal channel component based on deployment requirements.

Oracle Application Adapter for J.D. Edwards OneWorld provides:

- XML schemas and WSDLs for the J2CA 1.0 and 1.5 resource adapter.
- Web services for BSE.

See Also: *Oracle Application Server Adapter Concepts Guide*

1.2 J.D. Edwards OneWorld Platforms, Products, and Releases Supported

The following table indicates which combinations of adapter platforms and J.D. Edwards OneWorld platforms are supported, and for each combination, which J.D. Edwards OneWorld products and releases are supported.

Adapter Platform	J.D. Edwards OneWorld Platform	J.D. Edwards OneWorld Product and Release
Windows, AS400, HP 9000/B, Sun or IBM RS/6000	Windows, AS400, HP 9000/B, Sun or IBM RS/6000	■ XE (B7333) from SP19 to SP23 ■ ERP 8.0 (B7334) ■ EnterpriseOne B9 (8.9) ■ EnterpriseOne 8.10 (with Tools release 8.93 and 8.94) ■ EnterpriseOne 8.11 (SP1 and Tools Release 8.95) ■ EnterpriseOne 8.12 (Tools Release 8.96 2.0)

1.2.1 J.D. Edwards OneWorld Inbound WSDL Generation on UNIX Platforms

On UNIX platforms, manual inbound WSDL documents can be generated for J.D. Edwards OneWorld.

1.2.2 J.D. Edwards OneWorld Unit Of Work (UOW)

The following section provides information on the J.D. Edwards OneWorld Unit of Work (UOW).

1. iWay recommends the customer to generate individual business functions of J.D. Edwards OneWorld and then group them together.
2. Generating the individual business functions and grouping them together is completely based on the experience and knowledge related to the business functions of J.D. Edwards OneWorld. There is no documentation on generating the UOW.
3. To create the XML Schema Document (XSD) files that can be used for UOW, perform the following steps:
   a. Trigger the appropriate event from the J.D. Edwards OneWorld GUI and generate an XML output file based on the event.
   b. Using the XML file, create an XSD file using an XML editor, such as XMLSPY.

c. When creating the XSD, ensure that the XSD satisfies the SOA 11*g* namespace requirements. Manually add the namespace, target namespace, and other items that are required.

4. Once generated, copy the XSD files for the UOW into the repository folder. This repository folder is automatically configured on your file system when a J.D. Edwards OneWorld target is created using Application Explorer.

1.2.3 J.D. Edwards OneWorld Versions and Library Files

The following table lists the library files by version that are required for the Oracle Application Adapter for J.D. Edwards OneWorld.

J.D. Edwards OneWorld Version	Required Library Files
XE (B7333)	Connector.jar and Kernel.jar
ERP 8.0 (B7334)	Connector.jar and Kernel.jar
EnterpriseOne 8.9 (B9)	Connector.jar, Kernel.jar, jdeutil.jar, and log4j.jar
EnterpriseOne 8.10	Connector.jar, Kernel.jar, jdeutil.jar, and log4j.jar
EnterpriseOne 8.11 (SP1 and Tools Release 8.95)	Base_JAR.jar, Connector.jar, JdeNet_JAR.jar, log4j.jar, and System_JAR.jar
EnterpriseOne 8.12 (Tools Release 8.96 2.0)	Connector.jar, log4j.jar, Base_JAR.jar, EventProcesser_EJB.jar, EventProcesser_ JAR.jar. JdeNet_JAR.jar, and System_JAR.jar

On the J.D. Edwards OneWorld system, these library files are located in the following folder:

```
\\system\classes
```

1.3 J.D. Edwards OneWorld Concepts

You can use Oracle Application 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 J.D. Edwards OneWorld with non-J.D. Edwards OneWorld systems.

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

1.4 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 adapter 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.

Oracle Application 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). Oracle Application Adapter for J.D. Edwards OneWorld acts as a consumer of request messages and provides a response. An adapter 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 that is generated by the adapter.
  The schema is based on metadata in the EIS.

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

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

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

Propagating Internal Listeners out of J.D. Edwards OneWorld

Integrating a J.D. Edwards OneWorld listener with external systems is similar to the outbound 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 writes 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 inbound transactions. The J. D. Edwards 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 framework and leverages various integration access methods to provide the greatest amount of flexibility and functionality.

Oracle Application 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 OneWorld unedited transaction tables (Z tables)

Figure 1–1 illustrates the outbound processing framework.

The adapter uses the J.D. Edwards OneWorld ThinNet API to communicate with the J.D. Edwards OneWorld application. Using the ThinNet API, the adapter 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 J.D. Edwards OneWorld application because the adapter runs the MBF.

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

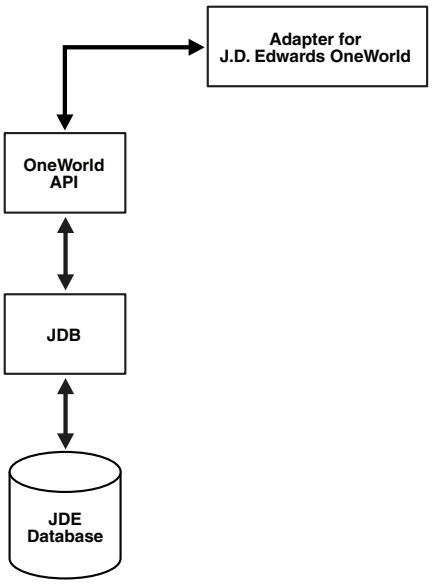
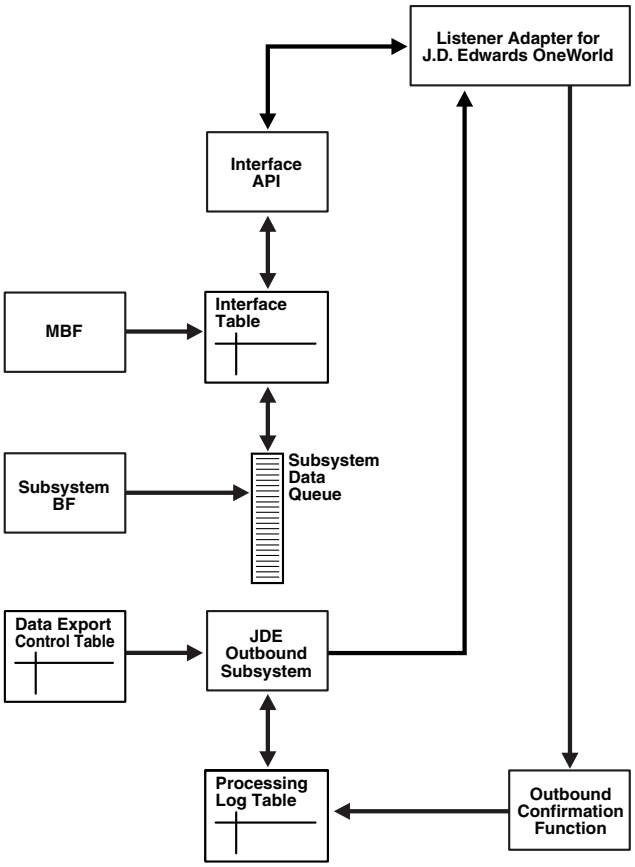


Figure 1–2 illustrates the inbound processing framework.

Figure 1–2 J.D. Edwards OneWorld Inbound 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 J.D. Edwards OneWorld outbound subsystem retrieves the data queue entry and looks in the Data Export Control table for the external processes to notify. The J.D. Edwards OneWorld outbound subsystem then calls the Oracle Application 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.

1.5 Adapter Architecture

Oracle Application Adapter for J.D. Edwards OneWorld uses Application Explorer with one of the following components:

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

Oracle 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 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 system for business functions. Application Explorer enables you to create XML schemas and Web services for the associated business function.

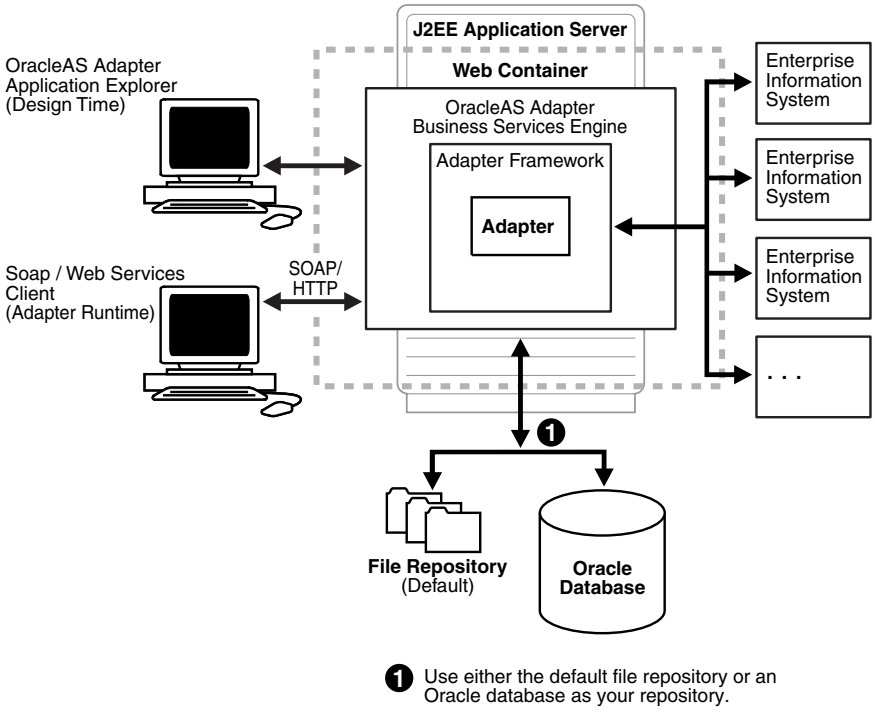
Resource Adapters

Oracle Application 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), and receive notification of events occurring in an EIS.

Oracle WebLogic 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 with BSE, as deployed to a Web container in a J2EE application server.

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



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

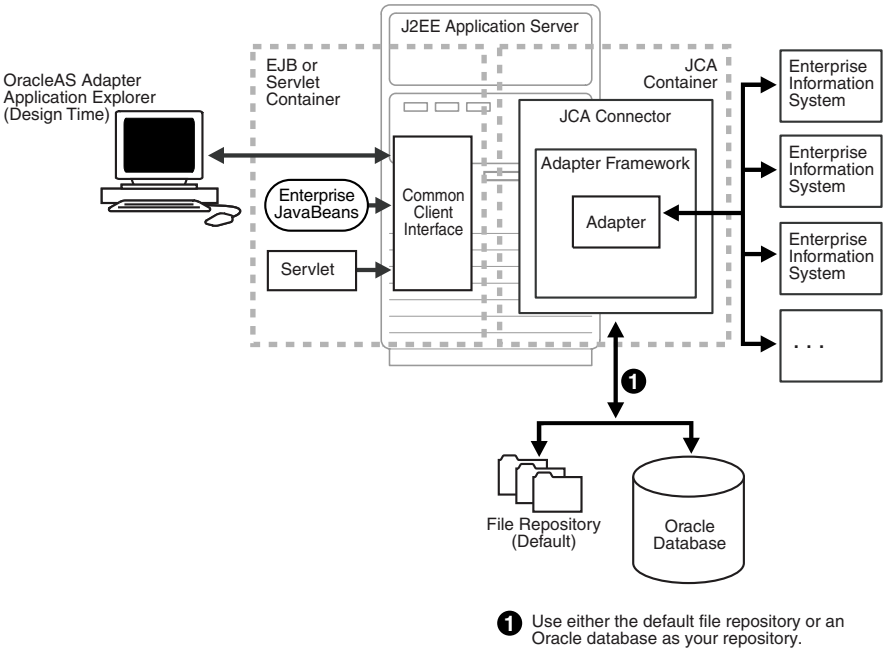
Application Explorer, a design-time tool deployed along with BSE, is used to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. Metadata created while you perform these operations are stored in the repository by BSE.

BSE uses SOAP as a protocol for receiving requests from clients, interacting with the EIS, and sending responses from the EIS back to clients.

Oracle WebLogic 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 Oracle Adapter Generic J2CA Architecture



Application Explorer, a design tool that works 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 Oracle WebLogic Server deployment descriptor `ra.xml`. See Chapter 3, "Oracle WebLogic Server Deployment and Integration" for more information.

Processing Business Functions

Oracle Application Adapter for J.D. Edwards OneWorld enables the processing of J.D. Edwards OneWorld business functions through the J.D. Edwards 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 a listener is defined through a TCP connection.

External applications that access J.D. Edwards OneWorld through Oracle Application 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 Oracle

Application Adapter for J.D. Edwards OneWorld" 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.

1.6 BSE Versus Oracle Adapter J2CA Deployment

If you are using Oracle Application Adapter for J.D. Edwards OneWorld with Business Process Execution Language (BPEL) Process Manager, please note that:

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

The following three factors explain the differences between deploying BSE and Oracle 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 WebLogic 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. Oracle Adapter J2CA provides slightly better performance.

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

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

2

Configuring Oracle Application Adapter for J.D. Edwards OneWorld

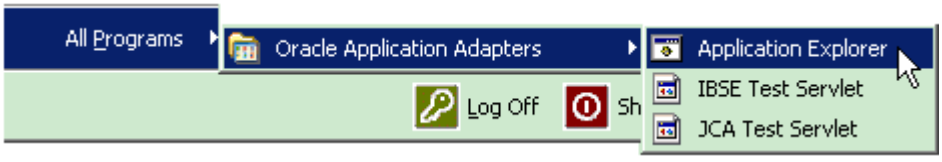
This chapter describes how to use Oracle Adapter Application Explorer (Application Explorer) to define a target to connect to a J.D. Edwards OneWorld system, view system objects, and create XML schemas and Web services. This chapter also explains how to configure an event adapter.

This chapter discusses the following topics:

2.1 Starting Application Explorer

To start Application Explorer:

1. Ensure that Oracle WebLogic Server is started, which is where Application Explorer is deployed.
2. Start Application Explorer by clicking the Windows **Start** menu, selecting **All Programs**, **Oracle Application Adapters**, and clicking **Application Explorer**.



You can also start Application Explorer by executing the **ae.bat** file, which is located in the following directory:

```
C:\oracle\Middleware\home_0309\Oracle_
```

```
SOA1\soa\thirdparty\ApplicationAdapters\tools\iwae\bin\ae.bat
```

It is a good practice to create a shortcut for the **ae.bat** file on your desktop.

If you are using a UNIX or Linux platform you can start Application Explorer by executing the **iwae.sh** file.

Application Explorer starts. You can now define new targets to a J.D. Edwards OneWorld system.

2.2 Configuring Repository Settings

You need not configure BSE for a file-based repository because it is configured during the installation.

2.2.1 Configuring the Database Repository for BSE

After BSE is deployed to Oracle WebLogic Server, you can configure it through the BSE configuration page. This configuration is required only when using a database repository with BSE.

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

To configure BSE:

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

   The `iwse.ora` SQL script is located in the following directory:

   ```
   C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\etc
   ```

   This script creates the required tables that are used to store the adapter configuration information in the database. These tables are used by Application Explorer and by adapters during design time and runtime. It is recommended that you use the same credentials to create the database repository and also in the `web.xml` file for database user credentials.

   ```
   C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\etc>sqlplus

   SQL*Plus: Release 10.1.0.2.0 - Production on Tue Dec 27 18:10:44 2005
   Copyright (c) 1982, 2004, Oracle.  All rights reserved.

   Enter user-name: scott
   Enter password: scott1

   Connected to:
   Oracle Database 11g Enterprise Edition Release 11.1.1.2.0 - Production
   With the Partitioning, OLAP and Data Mining options

   SQL>@ iwse.ora
   ```

2. Open the following page in your browser:

   ```
   http://host name:port/ibse
   ```

   Where *host name* is the system where BSE is installed and *port* is the HTTP port for a managed Oracle WebLogic server (for example, soa_server1).

For example,

```
http://localhost:8001/ibse
```

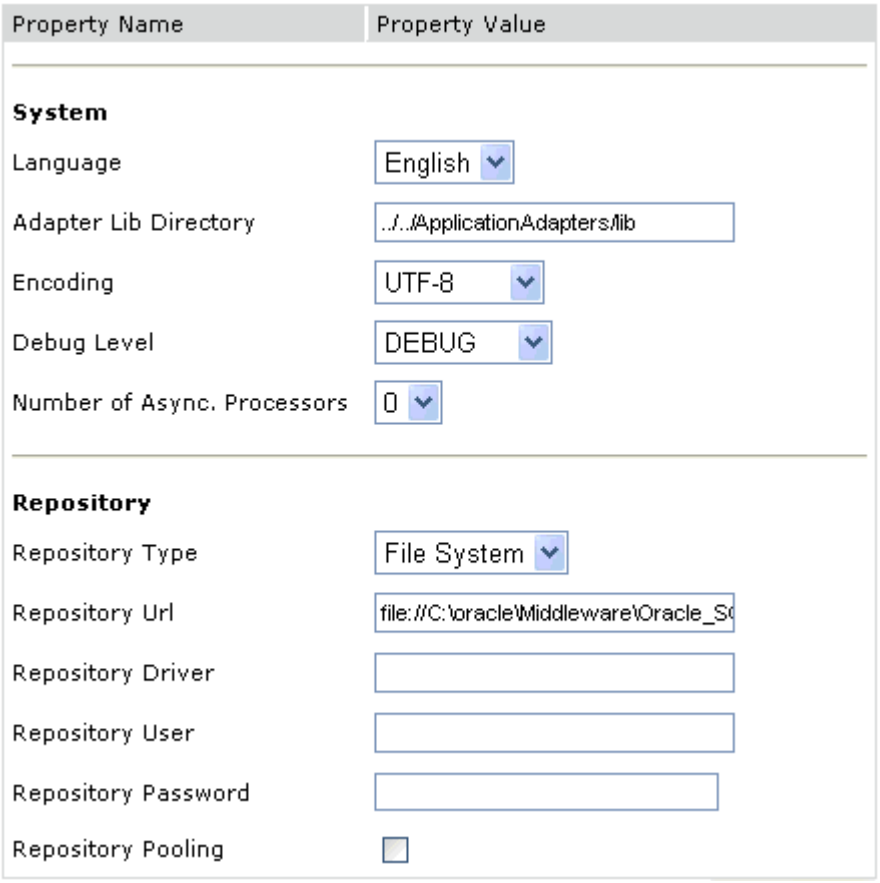
> **Note:** If you are accessing this page for the first time, it may take longer to load.

3. Log on when prompted.

   Enter the user ID and password, for example:

   - User name: weblogic
   - Password: welcome1

   The BSE configuration page is displayed.



4. The **ojdbc14.jar** file must be copied to the following directory:

   ```
   C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\lib
   ```

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

   ```
   C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\lib
   ```

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

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.

6. Click **Save**.

2.2.1.1 Configuring BSE System Settings

To configure BSE system settings:

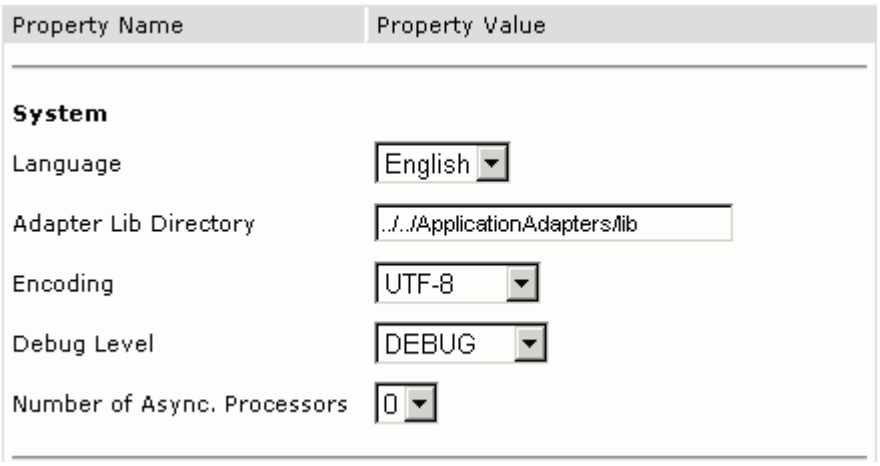
1. Display the **BSE configuration** page in a browser:

   `http://host name:port/ibse/IBSEConfig`

   Where *host name* is the system 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 settings pane is displayed, as shown in the following image.

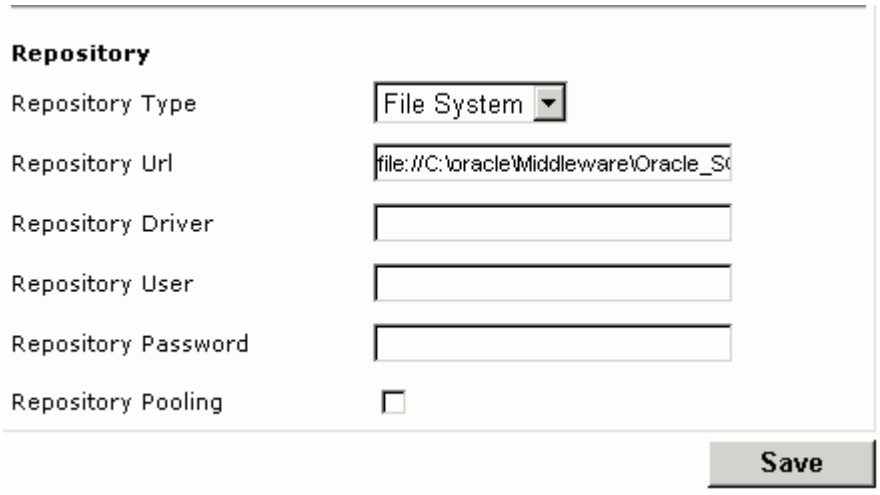


2. Configure the system settings.

   The following table lists the parameters with descriptions of the information to provide.

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 all fields and check boxes for the Repository pane.



3. Configure the repository settings.

   BSE requires a repository to store transactions and metadata required for the delivery of Web services.

   See Configuring a File System Repository on page 2-6 for more information.

   The following table lists the parameters with descriptions of the information to provide.

Parameter	Description
Repository Type	Select one of the following repositories from the list: ■ Oracle ■ File (Do not use for BSE in production environments.)
Repository URL	Enter the JDBC URL to use when opening a connection to the database. For example, the following repository URL format is used when connecting to Oracle: `jdbc:oracle:thin:@host name:port;SID`

Parameter	Description
Repository Driver	Provide the JDBC driver class to use when opening a connection to the database (optional). For example, the following repository driver format is used when connecting to Oracle: `oracle.jdbc.driver.OracleDriver`
Repository User	Enter a valid user ID to use when opening a connection to the database.
Repository Password	Enter a valid password that is associated with the user ID.
Repository Pooling	If selected, repository pooling is used. This option is disabled by default.

4. 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 system. 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:

```
C:\oracle\Middleware\user_projects\domains\base_domain\servers\soa_
server1\stage\ibse\ibse.war
```

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.

2.2.2 Configuring the Database Repository for J2CA

During the J2CA deployment of Oracle Application Adapter for J.D. Edwards OneWorld, Oracle WebLogic Server generates a deployment descriptor called `ra.xml`. This descriptor provides OracleWLS-specific deployment information for resource adapters. See Chapter 3, "Oracle WebLogic Server Deployment and Integration" for more information.

No configuration changes are necessary if you are using the default file based repository with J2CA deployment. This section describes how to configure the database repository for J2CA.

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

   The `iwse.ora` SQL script is located in the following directory:

   ```
   C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\etc
   ```

   This script creates the required tables that are used to store the adapter configuration information in the database. These tables are used by Application Explorer and by adapters during design time and runtime. It is recommended that you use the same credentials to create the database repository and also in the `ra.xml` file for database user credentials.

```
C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\etc>sqlplus

SQL*Plus: Release 10.1.0.2.0 - Production on Tue Dec 27 18:10:44 2005
Copyright (c) 1982, 2004, Oracle.  All rights reserved.

Enter user-name: scott
Enter password: scott1

Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.1.2.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL>@ iwse.ora
```

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

   ```
   C:\oracle\Middleware\Oracle_
   SOA1\soa\thirdparty\ApplicationAdapters\config\J2CA_SampleConfig
   ```

   Note: The `jcatransport.properties` file is required for each J2CA configuration that is created using Application Explorer. The J2CA configuration folder, for example, J2CA_SampleConfig, is named according to the configuration name that is specified in Application Explorer.

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

   ```
   iwafjca.repo.url=jdbc:oracle:thin:@90.0.0.51:1521:orcl

   iwafjca.repo.user=scott

   iwafjca.repo.password=scott1
   ```

   The following table lists the parameters with descriptions of the information to provide.

Parameter	Description
iwafjca.repo.url	Specify the JDBC URL to use when opening a connection to the database. For example, the following repository URL format is used when connecting to Oracle: `jdbc:oracle:thin:@host name:port;SID`
iwafjca.repo.user	Specify a valid user ID to use when opening a connection to the database.
iwafjca.repo.password	Specify a valid password that is associated with the user ID.

4. The **ojdbc14.jar** file must be copied to the following directory:

   ```
   C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\lib
   ```

5. Navigate to the following directory:

   ```
   C:\oracle\Middleware\Oracle_
   SOA1\soa\thirdparty\ApplicationAdapters\iwafjca.rar\META-INF
   ```

6. Open the `ra.xml` file in a text editor.
7. Provide the JDBC connection information as a value for the `IWAYRepo_URL` property.
8. Provide a valid user name for the `IWAYRepo_User` property.
9. Provide a valid password for the `IWAYRepo_Password` property.
10. Save your changes to the `ra.xml` file.

2.2.2.1 Password Encryption

When creating J2CA configurations, you can also encrypt a password using Application Explorer and use this value in the jcatransport.properties and 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 is displayed.
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 only if you are using a database repository.

   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 (3189319731832973218312931643233321232 27)
   ```
6. In the ra.xml file, which is used during run time, replace the existing password with the encrypted value for the IWayRepoPassword element. This is applicable for file system and database repositories.
7. Restart the Oracle WebLogic Server.

2.3 Creating a Repository Configuration

Before you use Application Explorer with Oracle Application 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 run-time.

Web services and BSE refer to the same type of deployment. See "Adapter Features" on page 1-1 for more information.

2.3.1 Creating a Configuration for BSE

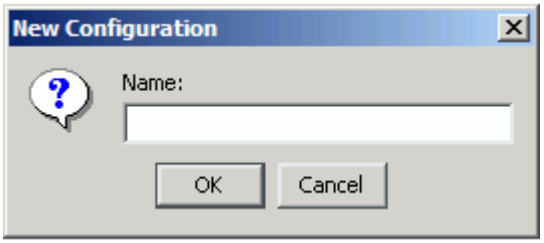
To create a repository configuration for BSE using Application Explorer, you must first define a new configuration.

Defining a New Configuration for BSE

To define a new configuration for BSE:

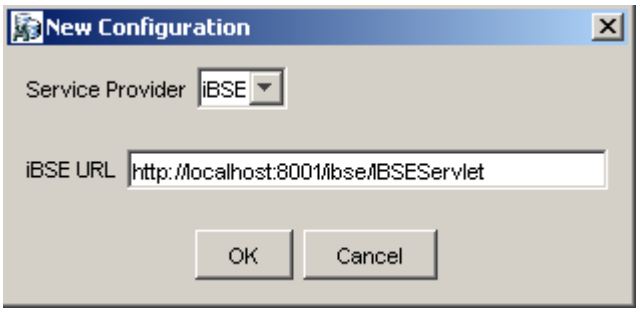
1. Right-click **Configurations** and select **New**.

   The New Configuration dialog is displayed.



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

   The New Configuration dialog is displayed.



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://host name:port/ibse/IBSEServlet`

   Where *host name* is the system where your Oracle WebLogic Server resides and *port* is the HTTP port for a managed Oracle WebLogic server (for example, soa_server1).
5. Click **OK**.

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

2.3.2 Creating a Configuration for J2CA

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

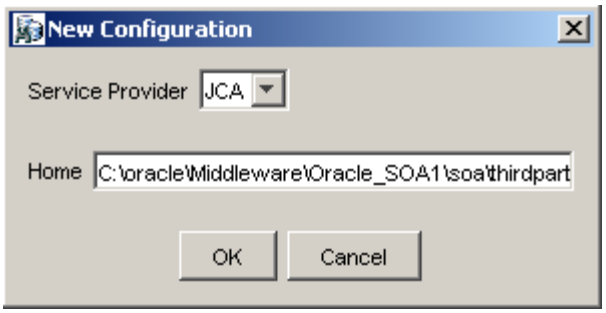
Defining a New Configuration for J2CA

To define a new configuration for J2CA:

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

   The New Configuration dialog is displayed.

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



3. From the **Service Provider** list, select **JCA**.
4. Click **OK**.

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

The Oracle Adapter J2CA configuration file is stored in:

`C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\config\`*`configuration_name`*

Where *`configuration_name`* is the name of the configuration you created (for example, myConfig).

HTTP Repository Connection

You can create an HTTP repository connection using J2CA, which enables them to generate and store WSDL documents remotely. Perform the following steps to create an HTTP repository connection in Application Explorer. To use the HTTP repository, ensure that the iwjcaivp test tool(jca-app-adapter-test) is successfully deployed and running.

1. Start the Application Explorer.
2. Right-click the **Configurations** node in the left pane and select **New**.

   The New Configuration dialog opens.

3. Type a name for the configuration and click **OK**.
4. Select **JCA** from the Service Provider list box and enter an HTTP target value in the Home field.

   Use the following format for the HTTP target value:

   `http://`**`host name`**`:`**`port`**`/iwafjca/JCAServlet`

   For example:

   `http://iwserv14:8001/iwafjca/JCAServlet`

5. Click **OK**.

   The new HTTP repository connection is added to the Configurations node.

   Once you connect to the remote server, you can create new Adapter targets, generate WSDL documents, and store them in the remote server.

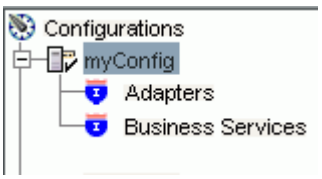
   Note: When you configure an Adapter target with the J2CA HTTP repository, you are not required to restart Oracle WebLogic Server for run time purposes.

2.3.3 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, the Business Services node is not shown. The following is an example of a BSE configuration named myConfig:



- Use the **Adapters** folder to create inbound interaction with J.D. Edwards OneWorld. For example, you use the JDEdwards node in the Adapters folder to configure a service that updates J.D. Edwards OneWorld.
- Use the **Events** folder (available for J2CA configurations only) 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.

2.4 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.

2.4.1 Defining a Target to J.D. Edwards OneWorld

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

Note: Before you create a new target, you must obtain the required library files for your J.D. Edwards OneWorld system and copy them to the appropriate location where the Oracle Application Adapter for J.D. Edwards OneWorld is deployed. For more information, see the *Oracle Fusion Middleware Application Adapters Installation Guide for Oracle WebLogic Server*.

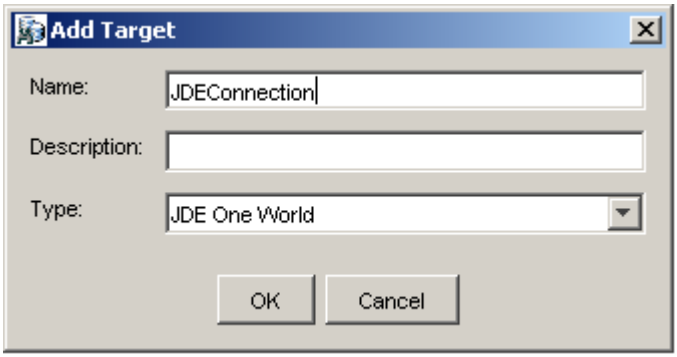
To define a target:

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 is displayed.



   Perform the following steps:

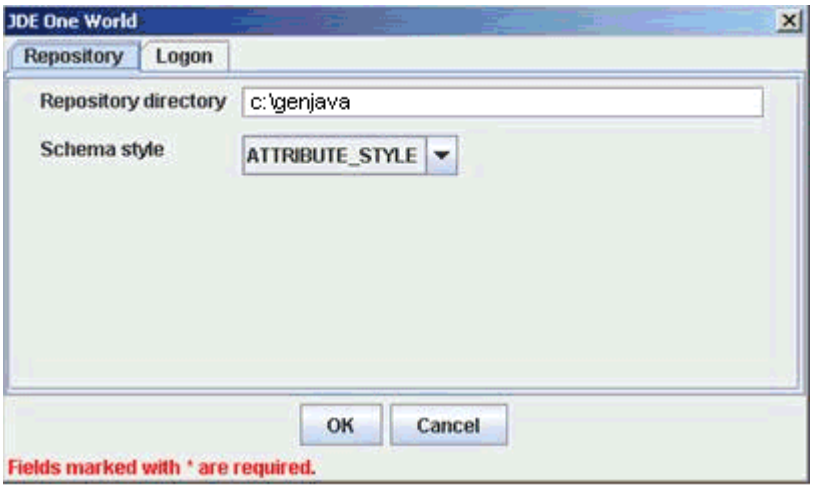
   a. In the **Name** field, enter a descriptive name, for example, JDEConnection.

   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 appears.



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

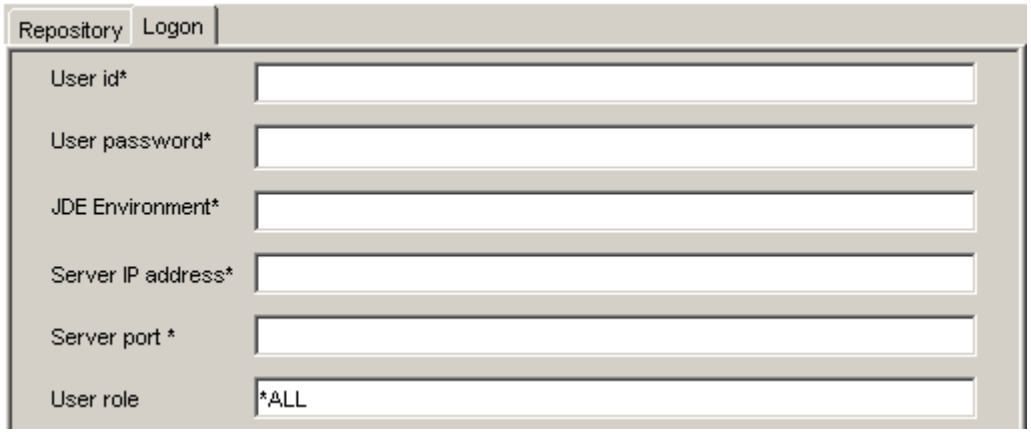
   This is the location of the Java wrappers for accessing the J.D. Edwards OneWorld business functions, which are created by the GenJava development tool. Please note that this is a prerequisite step, which must be performed before a new target is created using Application Explorer.

   Note: Generating schemas requires the GenJava repository. For more comprehensive information on building the J.D. Edwards OneWorld Master Business Function repository, see the *J.D. Edwards Interoperability Guide for OneWorld Xe.* For information on how to use the GenJava program, see Using the GenJava Development Tool (Outbound Processing) in Appendix A, "Configuring J.D. Edwards OneWorld for Outbound and Inbound Processing".

   ---

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

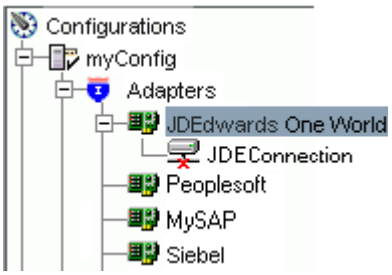
c. Click the **Logon** tab and enter the appropriate information for your target type based on the information in the following table. Fields marked with an asterisk are required.



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 J.D. Edwards OneWorld system administrator.
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 OneWorld system version. ■ For **B7333**: You can leave this field blank. ■ For **EnterpriseOne 8.10**: You must specify ***ALL**.

4. Click **OK**.

   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 2-15.

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 want 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*.

2.5 Creating an XML Schema

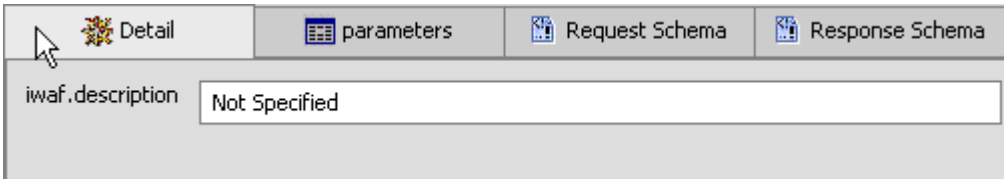
To execute an MBF, the adapter must receive a request document through the J.D. Edwards OneWorld ThinNet API. The agent processes the request and sends an XML response document indicating the result. Application Explorer creates both the XML request schema and the XML response schema.

2.5.1 Creating a Request and a Response Schema

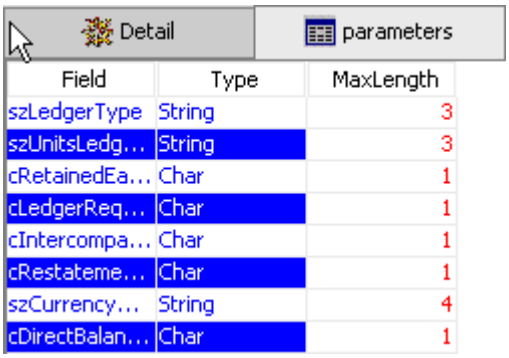
The following procedure explains how to create request and response schemas for a J.D. Edwards OneWorld business function. Application Explorer enables you to create XML schemas for this function.

1. Connect to a J.D. Edwards OneWorld target as described in "Connecting to a Defined J.D. Edwards OneWorld Target" on page 2-14.
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.

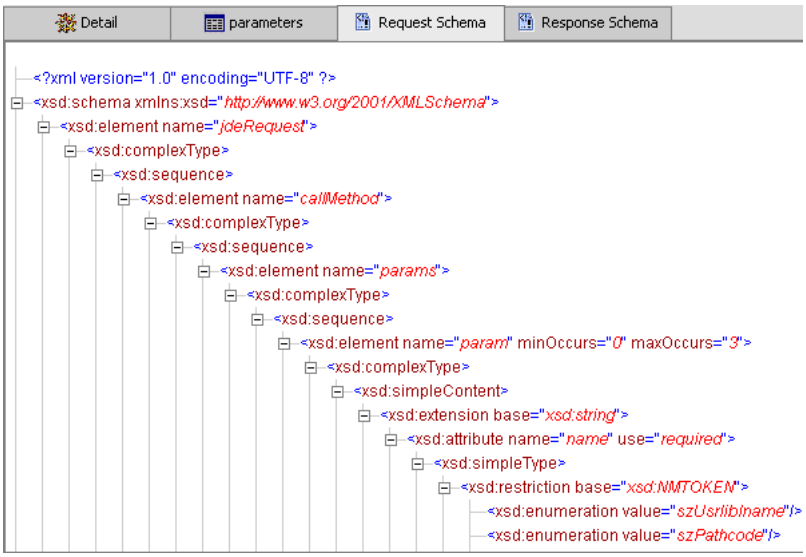
   The following image shows the tabs that appear on the right.



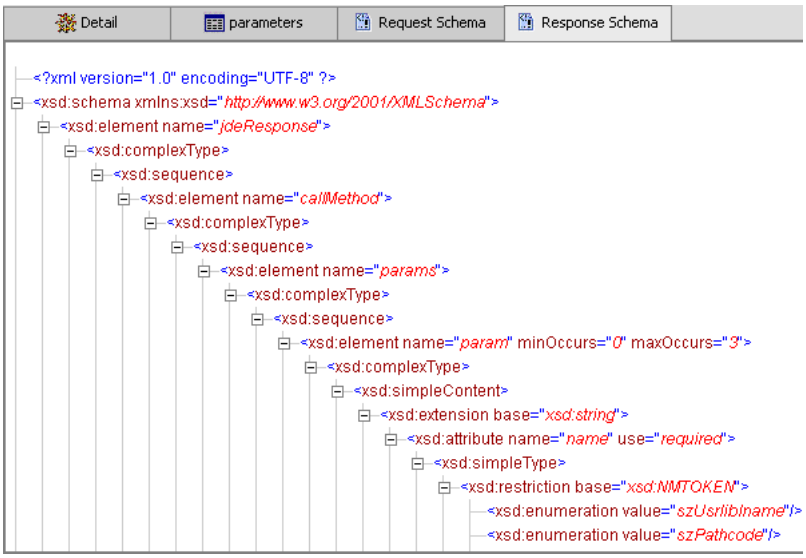
5. Click the **parameters** tab to view the parameter information.



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



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



2.6 Generating WSDL (J2CA Configurations Only)

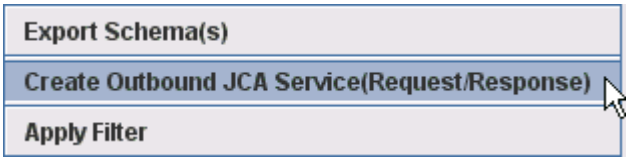
The procedure for generating WSDL (Web Service Definition Language) for request-response (outbound) services differs from that of generating WSDL for event notification (inbound) J2CA services of the adapter. The following sections describe each procedure.

Generating WSDL for Outbound Interaction

To generate a WSDL file for request-response service:

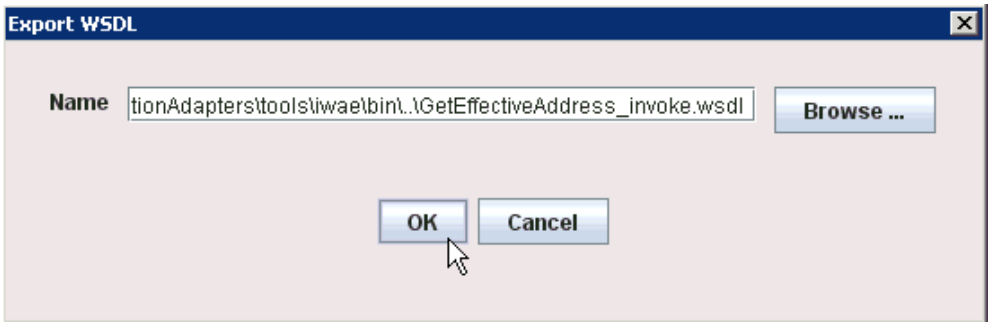
1. Start Application Explorer and connect to a defined J.D. Edwards OneWorld target.
2. Expand **Services**, **JDEJAVA_CFIN**, and then **B0100033**. Select **GetEffectiveAddress**.
3. Right-click GetEffectiveAddress.

   The following menu is displayed.



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

   The Export WSDL dialog is displayed.



5. Accept the default name 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`.
6. Click **OK**.

   The WSDL file is saved in the specified location.

Generating WSDL for Inbound Interaction

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

You can create an inbound J2CA service only if the node that you selected supports events.

1. Create a channel using Application Explorer under the J.D. Edwards Events node.

2. Start the channel.

   Do not restart Oracle WebLogic 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, which is located in the following directory:

   ```
   C:\oracle\Middleware\Oracle_
   SOA1\soa\thirdparty\ApplicationAdapters\config\config_name\log\iwaf_jca1500.log
   ```

   Alternatively, you can create a port using the File protocol under the Events node in Application Explorer, which disposes the event message to the file system.

5. Use a third party tool (for example, XMLSpy) to create the XML schema (.xsd file) using the XML payload that was captured in the previous step.

6. In the generated XML schema (.xsd file) perform the following modifications:

   a. Search for `Schemas-jdedwards-com` and replace it with `iwaysoftware`.

      ```
      <xs:schema
      targetNamespace="urn:Schemas-jdedwards-com:trans.response.JDESOOUT"
            xmlns="urn:Schemas-jdedwards-com:trans.response.JDESOOUT"
       xmlns:xs=http://www.w3.org/2001/XMLSchema elementFormDefault="qualified">
      ```

      to:

      ```
      <xs:schema
            targetNamespace="urn:iwaysoftware:trans.response.JDESOOUT"
            xmlns="urn:iwaysoftware:trans.response.JDESOOUT"
      xmlns:xs=http://www.w3.org/2001/XMLSchema elementFormDefault="qualified">
      ```

   b. Cut the following syntax:

      ```
      <xs:element name="jdeResponse">
      <xs:complexType>
      </xs:complexType>
      </xs:element>
      ```

   c. Paste it before the following line:

      ```
      <xs:element name="transaction">
      ```

7. Copy the XML schema (.xsd file) from the following directory:

   ```
   C:\oracle\Middleware\Oracle_
   SOA1\soa\thirdparty\ApplicationAdapters\config\config_name\JDEdwards\schemas
   ```

   Note: Edit the created channel by providing the location of the schema (.xsd) file (as mentioned in step 7) in the PreParser tab of Application Explorer. For example:

   ```
   C:\oracle\Middleware\Oracle_
   SOA1\soa\thirdparty\ApplicationAdapters\config\config_
   name\JDEdwards\schemas\jde-schema.xsd
   ```

8. Open a command prompt and navigate to the following directory:

   ```
   C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\tools
   ```

9. Execute the **obadapter.bat** file to set the environment.
10. Navigate to the following directory where the XML schema (.xsd file) is copied:

```
C:\oracle\Middleware\Oracle_
SOA1\soa\thirdparty\ApplicationAdapters\config\config_name\JDEdwards\schemas
```

11. Enter the following command to generate a WSDL:

```
 java -Diway.oem=oracle11g
com.iwaysoftware.af.container.tools.wsdl.IWayWSILBrowser adapterhome adapter
target channel schemaPrefix wsdlFileName
```

where:

adapterhome is the path to your ApplicationAdapters home. For example:

```
C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\
```

adapter is the name of the adapter. For example, JDEdwards.

target is the name of the adapter target you created in Application Explorer.

channel is the name of the channel you created in Application Explorer.

schemaPrefix is the prefix for the XSD schema. The schema file must be in the same directory where the Java command is executed, for example:

```
java -Diway.oem=oracle11g
com.iwaysoftware.af.container.tools.wsdl.IWayWSILBrowser
C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\JDEdwards
jde812_tgt jde_ch Jde812_Schema Jde812_salesorder_receive.wsdl
```

Once the command is executed, the following is displayed in the command window:

```
Running Inbound WSDL generation tool...
-> user.dir = java com.iwaysoftware.af.container.tools.wsdl.IWayWSILBrowser
C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\JDEdwards
jde812_tgt jde_ch Jde812_Schema Jde812_salesorder_receive.wsdl

-> Generating WSDL...
-> Done.
-> Writing WSDL 'C:\oracle\Middleware\Oracle_
SOA1\soa\thirdparty\ApplicationAdapters\wsdls\Jde812_salesorder_receive.wsdl '
to disk...
-> Done.
```

> **Note:** It is good practice to append **_receive** to the names of WSDL files that are generated for event notification services. This allows you to easily distinguish between them and those generated for request-response services.

12. Stop the channel in Application Explorer.

2.7 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.

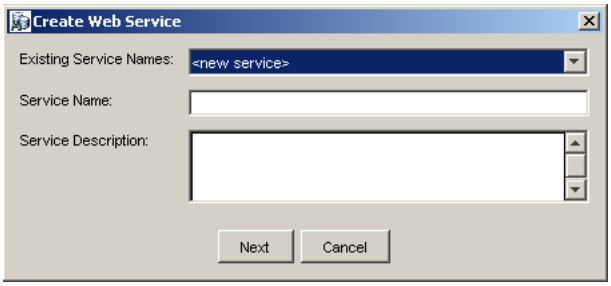
Note: In a J2EE Connector Architecture (J2CA) implementation, Web services are not available. When the adapters are deployed to use J2CA, the Common Client Interface (CCI) provides integration services.

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 is displayed.



   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 are saved. If you want 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.

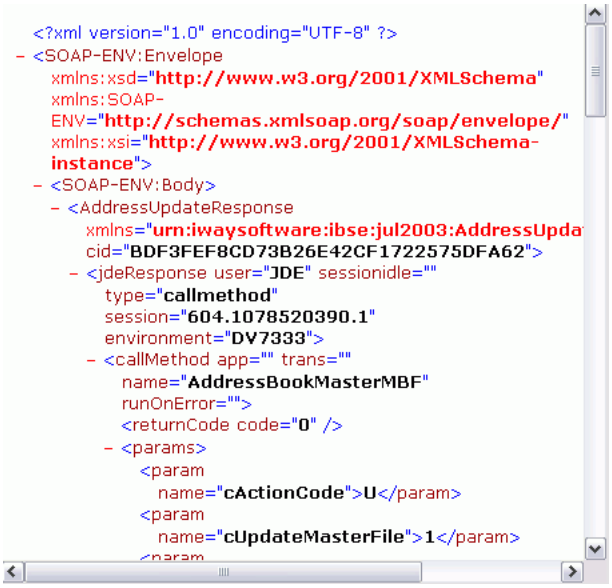
To test a Web service:

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.



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.

2.8 Configuring an Event Adapter

Events are generated by 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 channel.

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

A **channel** represents configured connections to particular instances of back-end systems. See "Creating and Editing a Channel" on page 2-22 for more information.

2.8.1 Creating and Editing a Channel

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

When you create, modify, or delete a channel, you must restart the Oracle WebLogic Server to recognize the change and update the repository for run time purposes. After successfully creating the channel and inbound WSDL file, close Application Explorer before you restart the application server.

Note: If you are planning to integrate Oracle Application Adapter for J.D. Edwards OneWorld with BPEL inbound or Mediator inbound process components, do not start the channel. The channel is managed by the run-time server after the BPEL or Mediator process component is deployed. If you start the channel from Application Explorer for testing and debugging purposes, stop it before run-time (when working with BPEL or Mediator process components).

Three channel types are available:

- HTTP
- TCP
- File

Note: Channels can be configured only on the system where the Oracle Application Adapter for J.D. Edwards OneWorld is installed.

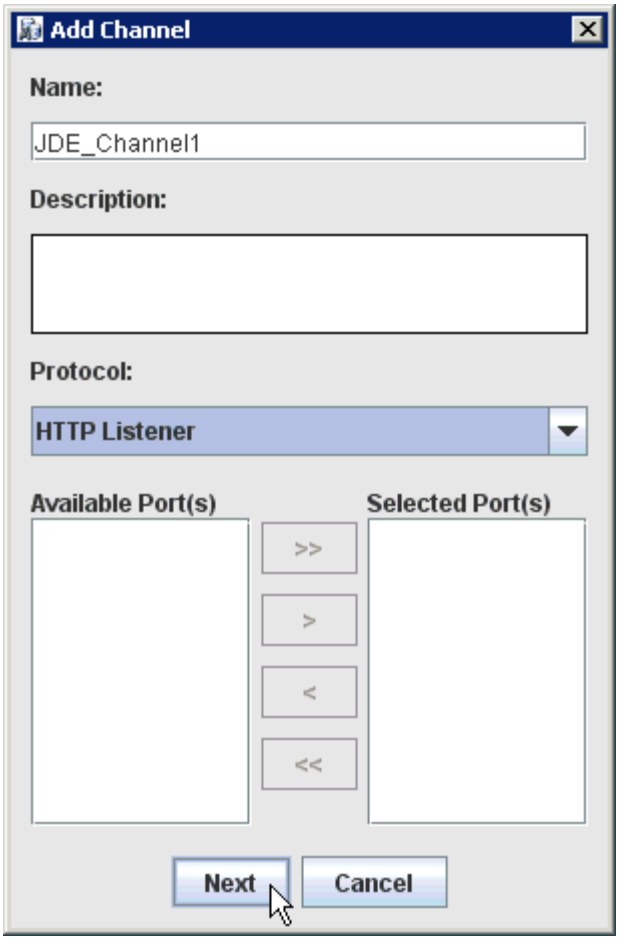
Creating an HTTP Channel

To create an HTTP Channel:

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

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

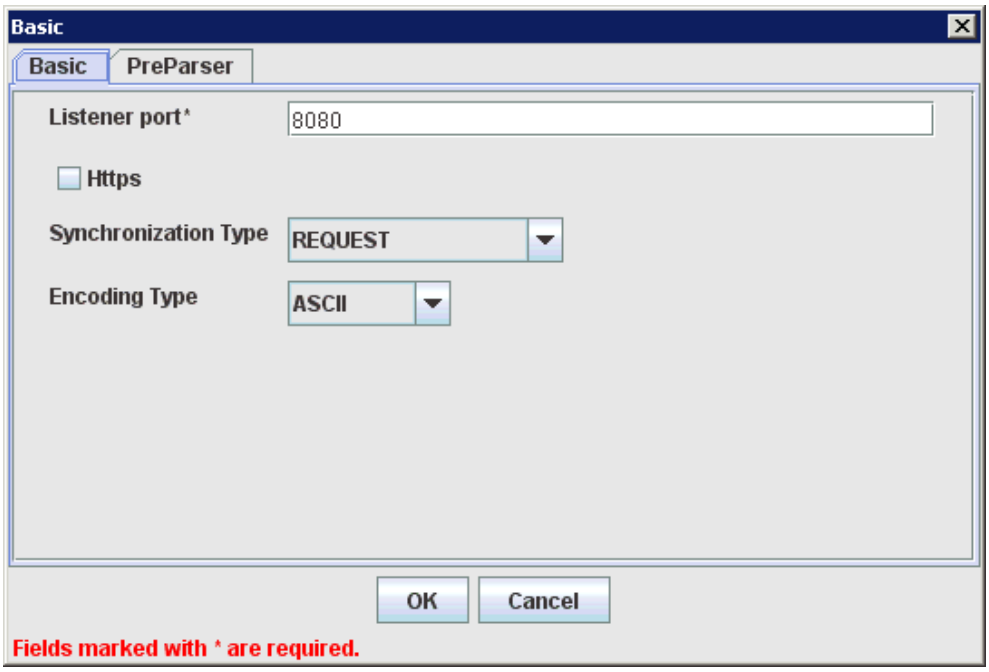
   The Add Channel dialog is displayed.



   Provide the following information:

   a. Enter a name for the channel, for example, **JDE_Channel1**.
   b. Enter a brief description.
   c. From the **Protocol** list, select **HTTP Listener**.
4. Click **Next**.

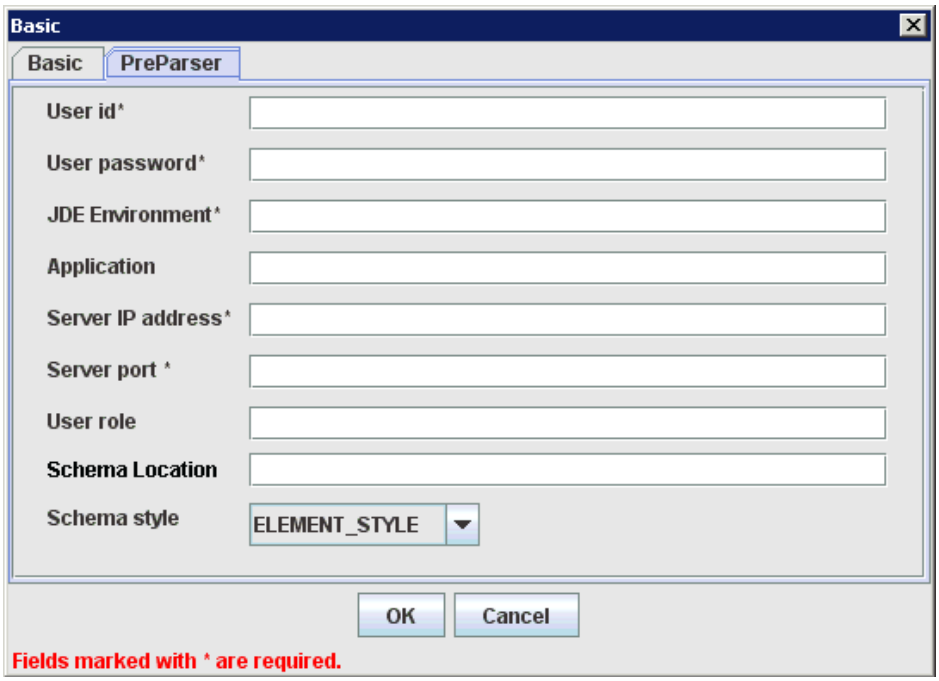
   The Basic dialog is displayed.



5. Enter the system information as specified in the following table:

Parameter	Description
Listener port	Port on which to listen for J.D. Edwards OneWorld event data.
Https	For a secure HTTP connection, select the **Https** check box. This option is currently not supported.
Synchronization Type	Choose from the following synchronization options: ■ REQUEST_RESPONSE ■ REQUEST_ACK **Important:** The J.D. Edwards OneWorld channel does not work if the synchronization type is set to REQUEST.
Encoding Type	Choose an encoding type to be used from the list. By default, ASCII is selected.

6. Click the **PreParser** tab.



7. Enter the system information as specified in the following table:

Parameter	Description
User id	A valid user ID for J.D. Edwards OneWorld.
User password	The password associated with the J.D. Edwards OneWorld 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 J.D. Edwards OneWorld system administrator.
Application	The application that is defined in the J.D. Edwards OneWorld environment.
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 OneWorld system version. ■ For **B7333**: You can leave this field blank. ■ For **EnterpriseOne 8.10**: You must specify ***ALL**.
Schema Location	The location of the XML schema (.xsd file) that was generated from the event output. For example: `C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\config\config_name\JDEdwards\schemas\jde-schema.xsd` For more information, see "Generating WSDL for Inbound Interaction" on page 2-17.
Schema Style	Choose from one of the following options: ■ ELEMENT_STYLE (default) ■ ATTRIBUTE_STYLE

8. Click **OK**.

   A summary pane is displayed, providing the channel description, channel status, and available ports. All the information is associated with the channel you created.

   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.

9. Right-click the channel and select **Start**.

   The channel you created becomes active. The X over the icon in the left pane disappears.

10. To stop the channel, right-click the channel and select **Stop**.

Creating a TCP Channel

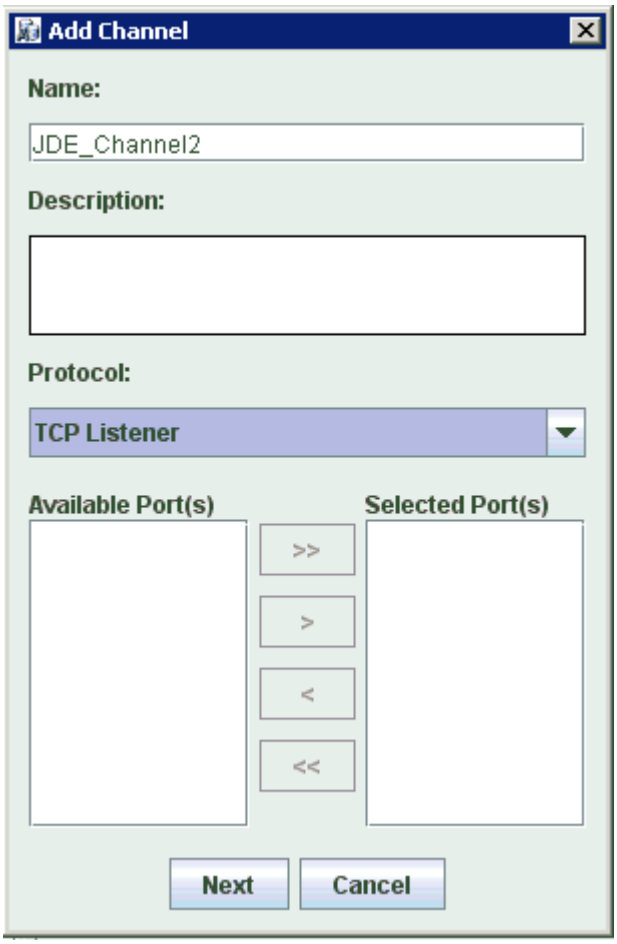
To create a TCP Channel:

1. Click the **Events** node.
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 is displayed.



Provide the following information:

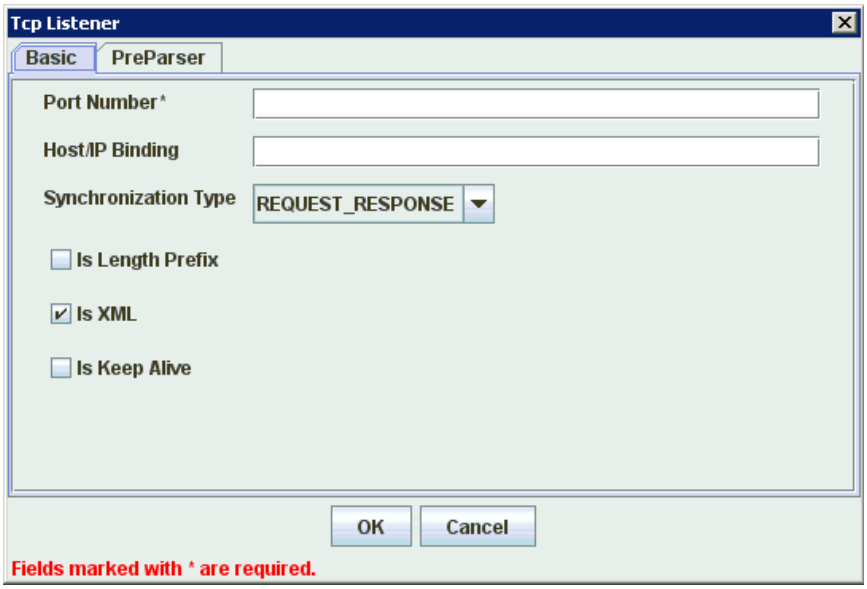
a. Enter a name for the channel, for example, **JDE_Channel2**.

b. Enter a brief description.

c. From the **Protocol** list, select **TCP Listener**.

4. Click **Next**.

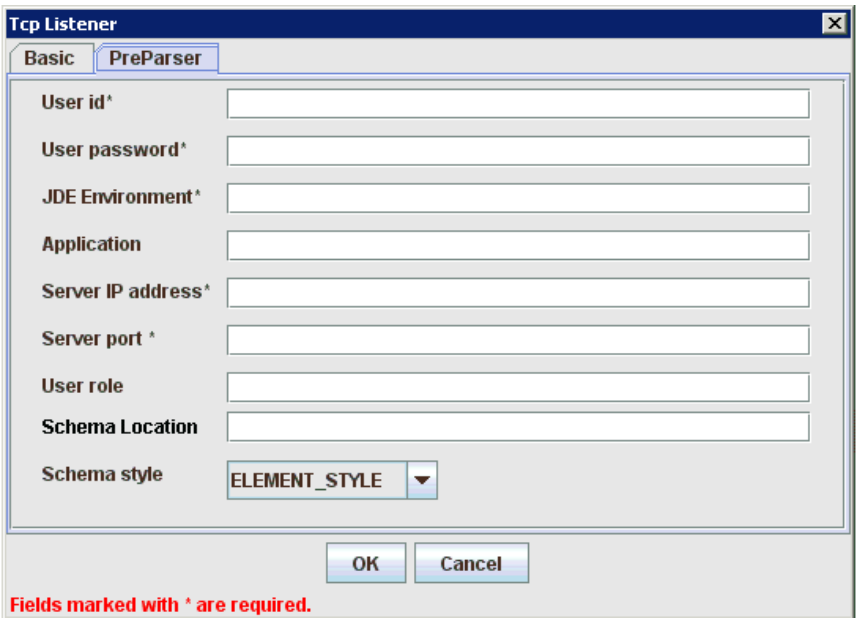
The Tcp Listener dialog is displayed.



5. Enter the system information as specified in the following table:

Parameter	Description
Port Number	Port on which the Host database is listening.
Host/IP Binding	Name or URL of the system where the database resides.
Synchronization Type	Choose from the following synchronization options: ■ REQUEST_RESPONSE ■ REQUEST_ACK **Important:** The J.D. Edwards OneWorld channel does not work if the synchronization type is set to REQUEST.
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.

6. Click the **PreParser** tab.



7. Enter the system information as specified in the following table:

Parameter	Description
User id	A valid user ID for J.D. Edwards OneWorld.
User password	The password associated with the J.D. Edwards OneWorld 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 J.D. Edwards OneWorld system administrator.
Application	The application that is defined in the J.D. Edwards OneWorld environment.
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 OneWorld system version. ■ For **B7333**: You can leave this field blank. ■ For **EnterpriseOne 8.10**: You must specify ***ALL**.
Schema Location	The location of the XML schema (.xsd file) that was generated from the event output. For example: `C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\config\config_name\JDEdwards\schemas\jde-schema.xsd` For more information, see "Generating WSDL for Inbound Interaction" on page 2-17.
Schema Style	Choose from one of the following options: ■ ELEMENT_STYLE (default) ■ ATTRIBUTE_STYLE

8. Click **OK**.

A summary pane is displayed, providing the channel description, channel status, and available ports. All the information is associated with the channel you created.

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.

9. Right-click the channel and select **Start**.

   The channel you created becomes active. The X over the icon in the left pane disappears.

10. To stop the channel, right-click the channel and select **Stop**.

Creating a File Channel

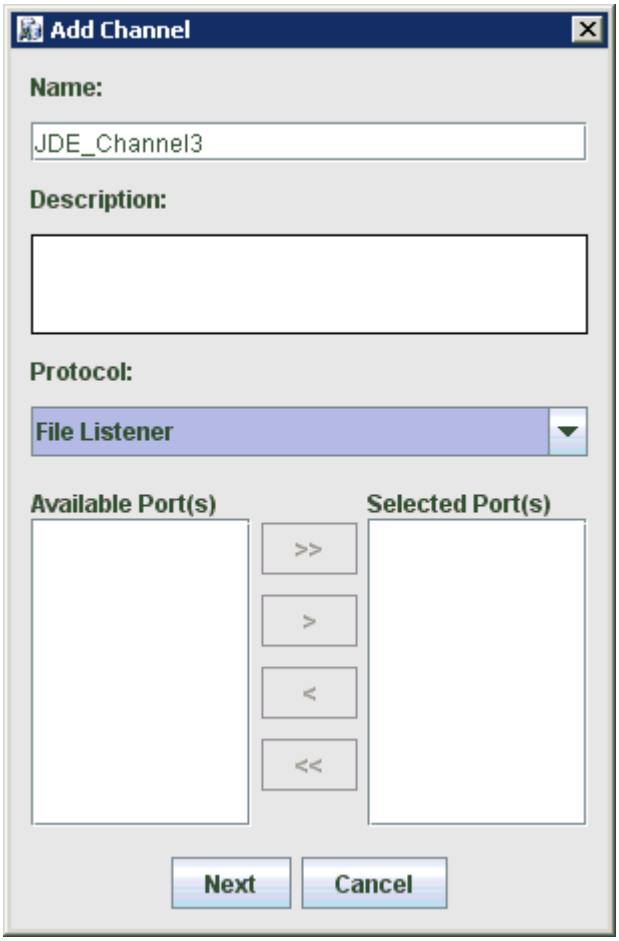
To create a File Channel:

1. Click the **Events** node.
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 is displayed.



Provide the following information:

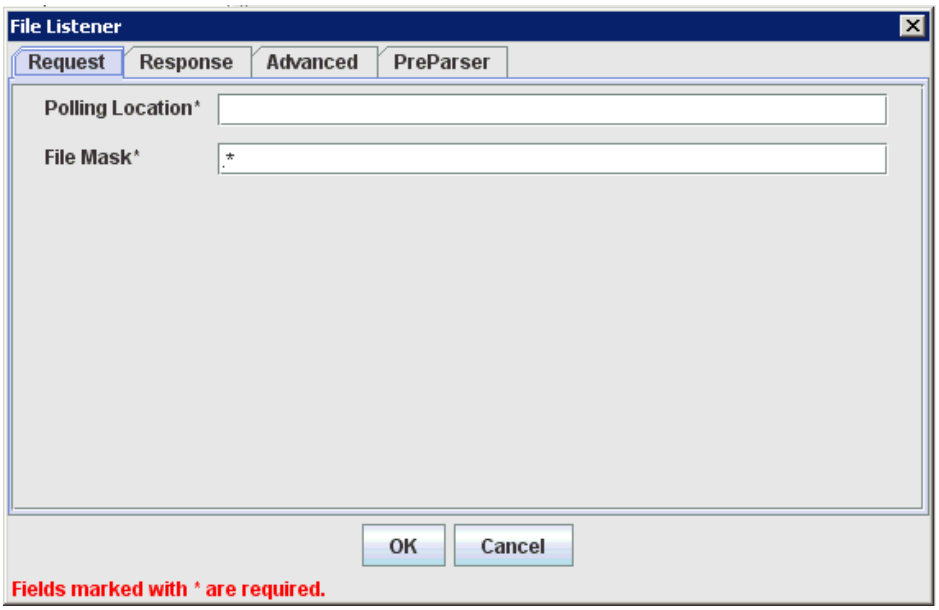
a. Enter a name for the channel, for example, **JDE_Channel3**.

b. Enter a brief description.

c. From the **Protocol** list, select **File Listener**.

4. Click **Next**.

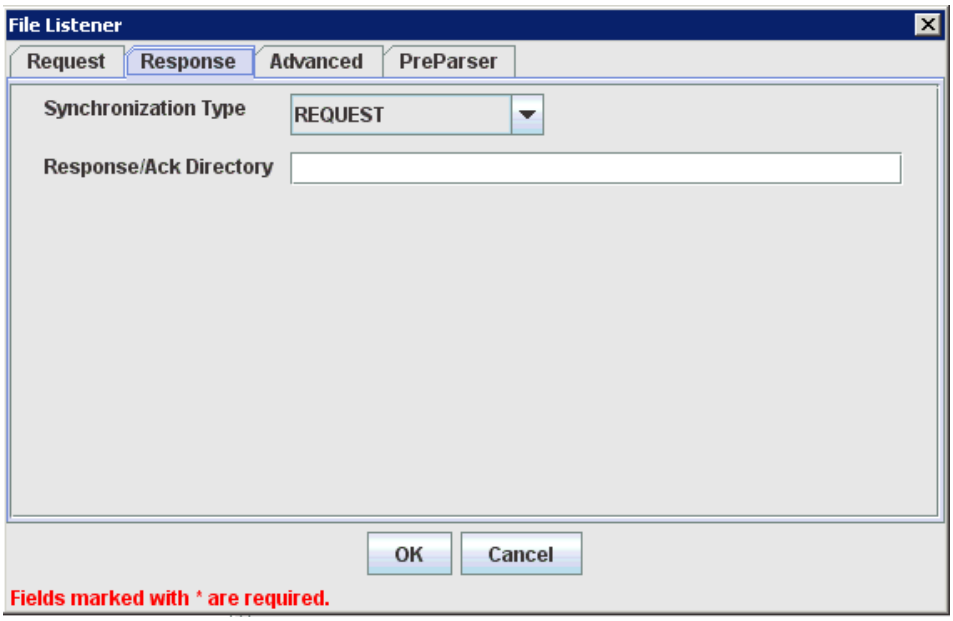
   The File Listener dialog is displayed.



5. Enter the system information in the Request tab as specified in the following table:

Parameter	Description
Polling Location	Target file system location for the J.D. Edwards OneWorld XML file.
File Mask	File name to be used for the output file generated by the operation.

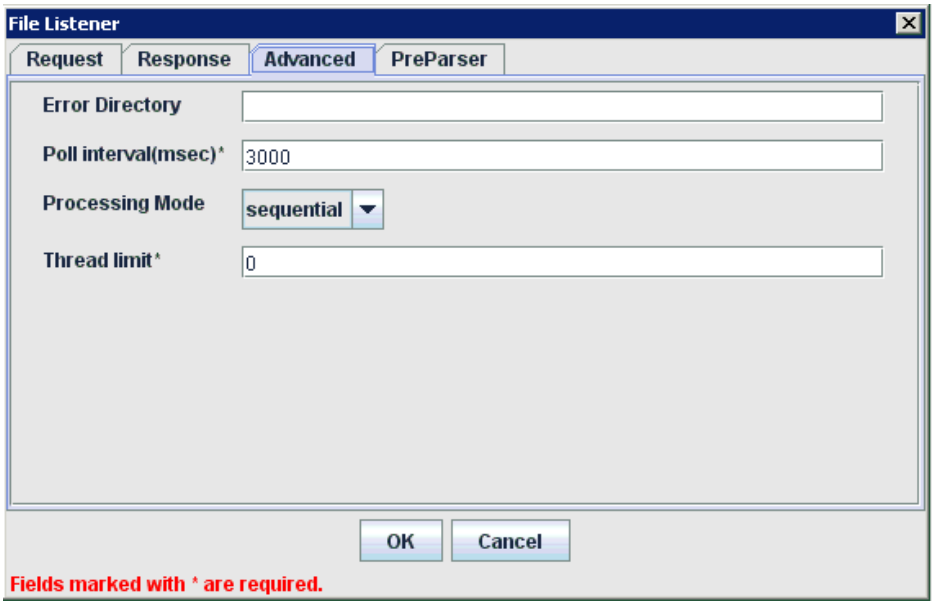
6. Click the **Response** tab.



7. Enter the system information in the Response tab as specified in the following table:

Parameter	Description
Synchronization Type	Choose from the following synchronization options: ■ REQUEST_RESPONSE ■ REQUEST_ACK **Important:** The J.D. Edwards OneWorld channel does not work if the synchronization type is set to REQUEST.
Response/Ack Directory	Target file system location for the J.D. Edwards OneWorld XML file.

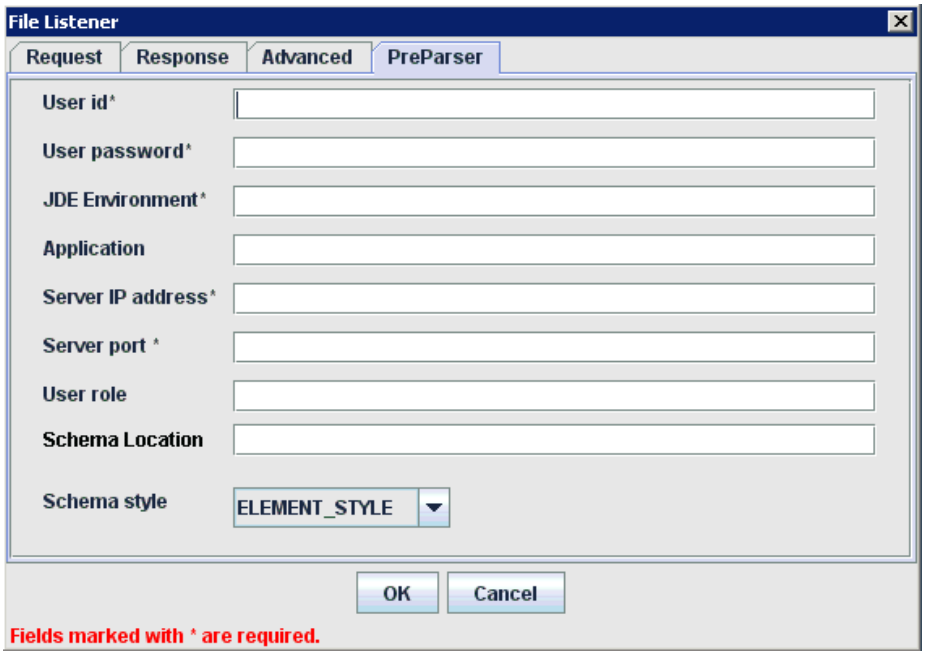
8. Click the **Advanced** tab.



9. Enter the system information in the Advanced tab as specified in the following table:

Parameter	Description
Error directory	Directory to which documents with errors are written.
Poll interval (msec)	Interval (in milliseconds) when to check for new input. The default is three seconds. Optional.
Processing Mode	**Sequential** indicates single processing of requests. **Threaded** indicates processing of multiple requests simultaneously.
Thread limit	If you selected threaded processing, indicate the maximum number of requests that can be processed simultaneously.

10. Click the **PreParser** tab.



11. Enter the system information as specified in the following table:

Parameter	Description
User id	A valid user ID for J.D. Edwards OneWorld.
User password	The password associated with the J.D. Edwards OneWorld 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 J.D. Edwards OneWorld system administrator.
Application	The application that is defined in the J.D. Edwards OneWorld environment.
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.

Parameter	Description
User role	This property setting depends on your J.D. Edwards OneWorld system version. ■ For **B7333**: You can leave this field blank. ■ For **EnterpriseOne 8.10**: You must specify ***ALL**.
Schema Location	The location of the XML schema (.xsd file) that was generated from the event output. For example: `C:\oracle\Middleware\Oracle_ SOA1\soa\thirdparty\ApplicationAdapters\config\config_ name\JDEdwards\schemas\jde-schema.xsd` For more information, see "Generating WSDL for Inbound Interaction" on page 2-17.
Schema Style	Choose from one of the following options: ■ ELEMENT_STYLE (default) ■ ATTRIBUTE_STYLE

12. Click **OK**.

    A summary pane is displayed, providing the channel description, channel status, and available ports. All the information is associated with the channel you created.

    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.

13. Right-click the channel and select **Start**.

    The channel you created becomes active. The X over the icon in the left pane disappears.

14. To stop the channel, right-click the channel and select **Stop**.

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 is displayed.

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

   The channel disappears from the list in the left pane.

2.8.2 The J.D. Edwards OneWorld Event Listener

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

The J.D. Edwards 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 J.D. Edwards OneWorld environment, see the *J.D. Edwards Interoperability Guide for OneWorld.*

The J.D. Edwards OneWorld Event Listener is called directly from the J.D. Edwards 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.

2.8.3 Configuring the J.D. Edwards OneWorld Event Listener

The J.D. Edwards OneWorld Event Listener is installed as part of the basic installation. The J.D. Edwards 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 OneWorld application server, you must configure the J.D. Edwards OneWorld Event Listener.

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

The J.D. Edwards OneWorld Event Listener includes the following components:

- The listener exit (`IWOEvent`), located under *`adapters_home`*`\etc\jde`, where `adapters_home` is `C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters`. For example:

  ```
  C:\oracle\Middleware\Oracle_
  SOA1\soa\thirdparty\ApplicationAdapters\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`), which must be created by the user.

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

1. Create a folder called Outbound under the JDE structure on the JDE Enterprise Sever, for example:

   ```
   \\JDEdwards\E812\DDP\Outbound
   ```

2. Copy the `iwoevent.dll` file in the new Outbound folder.

3. Create an environment variable, *`IWOEVENT_HOME`*, to point to the directory containing the `iwoevent.dll` 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
     ```

4. On the J.D. Edwards OneWorld Server, create an `iwoevent.cfg` file in the defined directory, *`IWOEVENT_HOME`*.

   The J.D. Edwards 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 J.D. Edwards OneWorld Event Listener requires connection information for the associated integration server to function properly. This information is contained in the `iwoevent.cfg` file. 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.

     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.

   - **Alias**

     The alias section of the configuration file contains the connection information required to send transactions to specific servers. Currently, the Oracle Application Adapter for J.D. Edwards OneWorld supports 100 entries (alias names) in the configuration file.

     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 Oracle Application Adapter for J.D. Edwards OneWorld (required).

     `port` is the port defined for Oracle Application Adapter for J.D. Edwards OneWorld in the TCP channel configuration (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.

The following is a sample entry for `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.JDESOOUT=edamcs1t,edamcs2
trans.JDEPOOUT=edamcs1
```

5. Create a folder using the alias names that are specified in the iwoevent.cfg file under the defined directory, *IWOEVENT_HOME*. For example:

```
\\JDEdwards\E812\DDP\Outbound\edamcs1
```

2.9 Runtime Overview

After J.D. Edwards OneWorld starts the J.D. Edwards 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. All log information is saved in a file called `iwoevent.log`. The `iwoevent.log` file is created in the outbound folder where the `iwoevent.dll` and `iwoevent.cfg` files are located.

2.10 Modifying the JDE.INI File for Outbound and Inbound Processing

This section describes the settings that are required in the JDE.INI file for the XML call object kernel (outbound and inbound processing).

Open the JDE.INI file and modify the **[JDENET_KERNEL_DEF6]** and **[JDENET_KERNEL_DEF15]** sections as follows:

```
[JDENET_KERNEL_DEF6]
krnlName=CALL OBJECT KERNEL
dispatchDLLName=XMLCallObj.dll
dispatchDLLFunction=_XMLTransactionDispatch@28
maxNumberOfProcesses=1
numberOfAutoStartProcesses=1

[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 NT 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

Note: The J.D. Edwards OneWorld installation for version B7333(XE) does not include **[JDENET_KERNEL_DEF15]**. As a result, if you are using version B7333(XE), you must manually add it to the jde.ini file. For all other J.D. Edwards OneWorld versions, **[JDENET_KERNEL_DEF15]** is included with the installation.

3

Oracle WebLogic Server Deployment and Integration

This chapter describes Oracle WebLogic Server (OracleWLS) deployment and integration with Oracle Application Adapter for J.D. Edwards OneWorld.

This chapter discusses the following topics:

- Adapter Integration with Oracle WebLogic Server
- Deployment of Adapter
- Updating Adapter Configuration

See Also:

- *Oracle Application Server Adapter Concepts Guide*

3.1 Adapter Integration with Oracle WebLogic Server

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

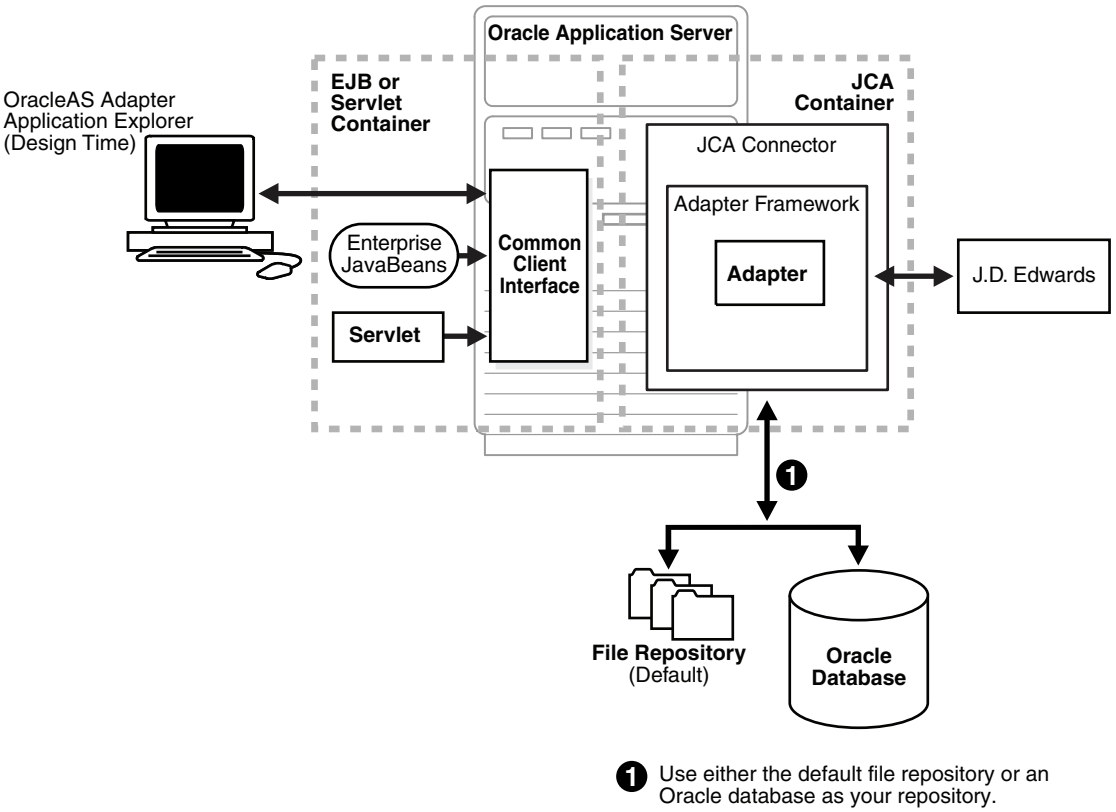
See Also:

- *Oracle Application Server Adapter Concepts Guide*

3.2 Deployment of Adapter

Figure 3–1 shows deployment of the J2CA Connector to the Oracle Application Server. In a run-time service scenario, an Enterprise Java Bean, 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 J2CA Architecture



See Also:

- *Oracle Application Server Adapter Concepts Guide*

3.3 Updating Adapter Configuration

During the J2CA deployment of Oracle Application Adapter for J.D. Edwards OneWorld, OracleWLS generates a deployment descriptor called `ra.xml`, located in:

```
C:\oracle\Middleware\Oracle_
SOA1\soa\thirdparty\ApplicationAdapters\iwafjca.rar\META-INF
```

Your installation contains more than one file named `ra.xml`. The OracleWLS deployment descriptor that is described in this section is located in the directory specified above.

> **Note:** Multiple managed connection factories are supported only for outbound processing (services).

Creating a Managed Connector Factory Object

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE connector PUBLIC '-//Sun Microsystems, Inc.//DTD Connector 1.0//EN'
'http://java.sun.com/dtd/connector_1_0.dtd'>
```

```
<connector>
  <display-name>IWAFJCA10</display-name>
  <vendor-name>IWAY Software</vendor-name>
  <spec-version>1.0</spec-version>
  <eis-type>IWAF</eis-type>
  <version>1.0</version>
  <license>
    <license-required>false</license-required>
  </license>
  <resourceadapter>

<managedconnectionfactory-class>com.ibi.afjca.spi.IWAFManagedConnectionFactory</ma
nagedconnectionfactory-class>

<connectionfactory-interface>javax.resource.cci.ConnectionFactory</connectionfacto
ry-interface>

<connectionfactory-impl-class>com.ibi.afjca.cci.IWAFConnectionFactory</connectionf
actory-impl-class>
    <connection-interface>javax.resource.cci.Connection</connection-interface>

<connection-impl-class>com.ibi.afjca.cci.IWAFConnection</connection-impl-class>
    <transaction-support>NoTransaction</transaction-support>
    <config-property>
      <config-property-name>AdapterName</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value></config-property-value>
    </config-property>
    <config-property>
      <config-property-name>Config</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value></config-property-value>
    </config-property>
    <config-property>
      <config-property-name>IWayHome</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value>C:\oracle\Middleware\Oracle_
SOA1\soa\thirdparty\ApplicationAdapters</config-property-value>
    </config-property>
    <config-property>
      <config-property-name>IWayConfig</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value>jca_sample</config-property-value>
    </config-property>
    <config-property>
      <config-property-name>IWayRepoDriver</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value></config-property-value>
    </config-property>
    <config-property>
      <config-property-name>IWayRepoURL</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value></config-property-value>
    </config-property>
    <config-property>
      <config-property-name>IWayRepoUser</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value></config-property-value>
    </config-property>
    <config-property>
```

```
      <config-property-name>IWayRepoPassword</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value></config-property-value>
    </config-property>
    <config-property>
      <config-property-name>LogLevel</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value>DEBUG</config-property-value>
    </config-property>
    <authentication-mechanism>
      <authentication-mechanism-type>BasicPassword</authentication-mechanism-type>

<credential-interface>javax.resource.spi.security.PasswordCredential</credential-i
nterface>
    </authentication-mechanism>
    <reauthentication-support>true</reauthentication-support>
  </resourceadapter>
</connector>
```

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

Parameter Name	Description
IWayHome	The base installation directory for the OracleWLS packaged application adapter.
IWayConfig	The adapter configuration name as defined in Application Explorer. For example, Oracle Application 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 repository.
IWayRepoUser	User name to use when connecting to the database. This is necessary only when using an Oracle database as the repository.
IWayRepoPassword	Password. If provided, it overwrites configuration. This is necessary only when using an Oracle database as the repository.
loglevel	It overwrites the level set by the ManagedConnectorFactory property.

Creating Multiple Managed Connector Factory Objects

To establish multiple managed connector factory objects, you must edit the `weblogic-ra.xml` file and add more `<connection-instance>` nodes. This file is located in:

```
C:\oracle\Middleware\Oracle_
SOA1\soa\thirdparty\ApplicationAdapters\iwafjca.rar\META-INF
```

For example, the first jca_configuration in Application Explorer is represented in the `weblogic-ra.xml` file as follows:

```
<?xml version="1.0"?>
<weblogic-connector xmlns="http://www.bea.com/ns/weblogic/90">
    <enable-access-outside-app>true</enable-access-outside-app>
    <enable-global-access-to-classes>true</enable-global-access-to-classes>
    <outbound-resource-adapter>
```

```
        <default-connection-properties>
        <pool-params>
        <initial-capacity>0</initial-capacity>
        </pool-params>
        <transaction-support>LocalTransaction</transaction-support>
        </default-connection-properties>
        <connection-definition-group>

<connection-factory-interface>javax.resource.cci.ConnectionFactory</connection-fac
tory-interface>
            <connection-instance>
                <jndi-name>eis/OracleJCAAdapter/DefaultConnection</jndi-name>
            </connection-instance>
        </connection-definition-group>
    </outbound-resource-adapter>
</weblogic-connector>
```

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

```
<?xml version="1.0"?>
<weblogic-connector xmlns="http://www.bea.com/ns/weblogic/90">

    <enable-access-outside-app>true</enable-access-outside-app>
    <enable-global-access-to-classes>true</enable-global-access-to-classes>

    <outbound-resource-adapter>
        <default-connection-properties>
        <pool-params>
        <initial-capacity>0</initial-capacity>
        </pool-params>
        <transaction-support>LocalTransaction</transaction-support>
        </default-connection-properties>
        <connection-definition-group>

<connection-factory-interface>javax.resource.cci.ConnectionFactory</connection-fac
tory-interface>
            <connection-instance>
                <jndi-name>eis/OracleJCAAdapter/DefaultConnection</jndi-name>
            </connection-instance>
            <connection-instance>
                <jndi-name>eis/OracleJCAAdapter/DefaultConnection1</jndi-name>
                <connection-properties>
                <properties>
                <property>
<name>IWayHome</name>
<value>C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters</value>
                </property>
                <property>
                <name>IWayConfig</name>
                <value>jca_sample2</value>
                </property>
                <property>
    <name>IWayRepoURL</name>
    <value></value>
                </property>
                <property>
      <name>IWayRepoUser</name>
      <value></value>
                </property>
                <property>
```

```
      <name>IWayRepoPassword</name>
      <value></value>
                </property>
                <property>
      <name>LogLevel</name>
      <value>Debug</value>
                </property>
                </properties>
                </connection-properties>
              </connection-instance>
            </connection-definition-group>
    </outbound-resource-adapter>
</weblogic-connector>
```

If you do not specify a `<property>` element in the `<connection-instance>` section, the value is taken from the `ra.xml` file. You can specify the default properties in the `ra.xml` file and then override them as required in the `weblogic-ra.xml` file. In addition, note that the J2CA configuration (for example, jca_sample2) must already be created in Application Explorer.

Note: When you modify the `ra.xml` and `weblogic-ra.xml` files, the application server must be restarted. If the application server is already running, stop the application server and then restart it.

In addition, the `iwafjca.rar` file must be redeployed in the Oracle WebLogic Administration Console to activate these changes.

Modifying WSDL Files for Additional Connection Factory Values

Application Explorer generates the J2CA properties file using the default connection factory name `eis/OracleJCAAdapter/DefaultConnection`. If you created additional connection factories, the WSDLs generated for the additional configuration and connection factory should be changed to reflect the location field of the `jca:address` section in the J2CA properties file. The default J2CA properties file for the Oracle Application Adapter for J.D. Edwards OneWorld with a configuration of `isdsrv2_conn2` is shown in the following example.

Notice that the J2CA properties file has the following default connection factory: `eis/OracleJCAAdapter/DefaultConnection`

```
<jca:address location="eis/OracleJCAAdapter/DefaultConnection"
        ConnectionSpec="com.ibi.afjca.cci.IWAFConnectionSpec"
        cs.AdapterName="JDEdwards" cs.Config="isdsrv2_conn2"
UIConnectionName="Connection1"/>
```

The connection factory value must be changed to the following: `eis/OracleJCAAdapter/DefaultConnection1`

For example:

```
<jca:address location="eis/OracleJCAAdapter/DefaultConnection1"
                        ConnectionSpec="com.ibi.afjca.cci.IWAFConnectionSpec"
                        cs.AdapterName="JDEdwards" cs.Config="isdsrv2_conn2"
UIConnectionName="Connection1"/>
```

Note that only the value for the location field in the `jca:address` section should be modified. Do not modify any other field or section.

4

Integration With BPEL Service Components in the Oracle SOA Suite

Oracle Application 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:

4.1 Overview

To integrate with Oracle BPEL Process Manager, Oracle Application Adapter for J.D. Edwards OneWorld must be deployed in the same WLS 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 Adapter Application Explorer (Application Explorer) for both request-response (outbound) and event notification (inbound) services of the adapter. See Chapter 2, "Configuring Oracle Application Adapter for J.D. Edwards OneWorld" 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 Oracle JDeveloper and deployed to a BPEL server. Upon deployment to the BPEL server, every newly built process is automatically deployed to the Oracle Enterprise Manager console, where you run, monitor, administer BPEL processes, and listen to adapter events.

4.2 Deployment of Adapter

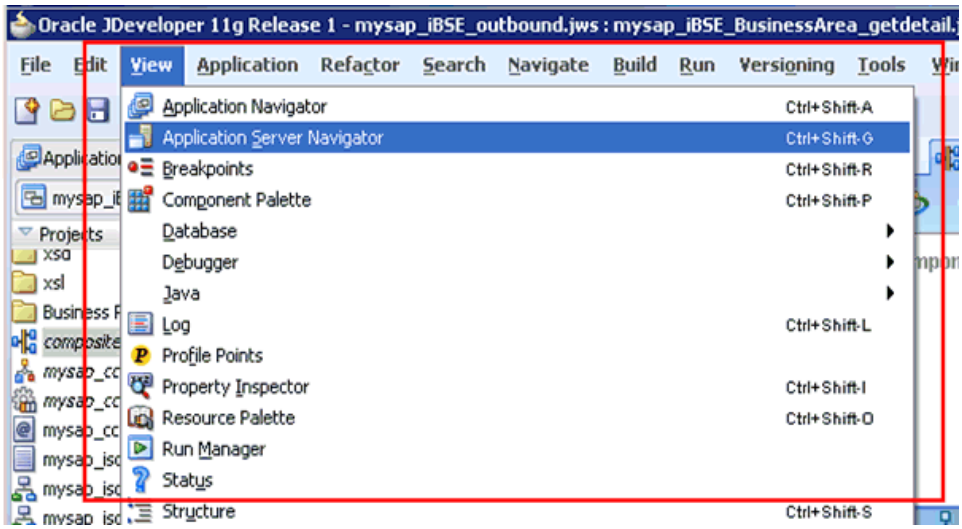
During installation, Oracle Application Adapter for J.D. Edwards OneWorld is deployed as a J2CA 1.0 resource adapter within the WLS container. The adapter must be deployed in the same WLS container as Oracle BPEL Process Manager.

See Also: *Oracle Application Server Adapter Concepts Guide*

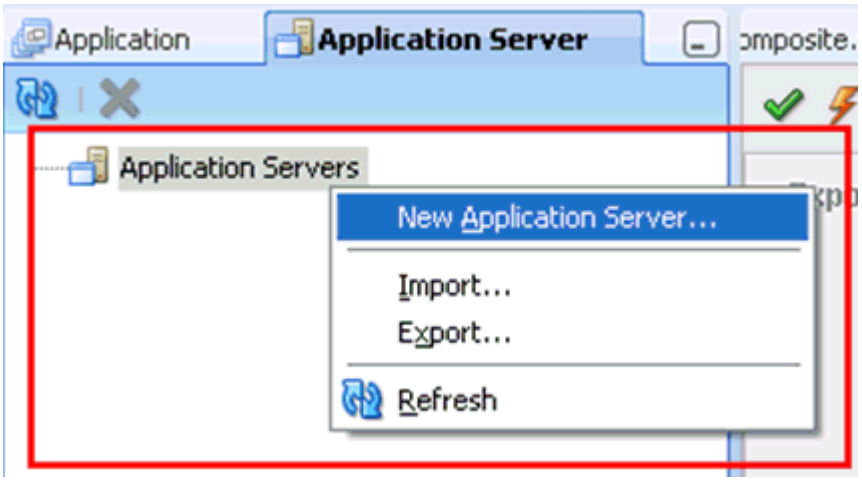
4.3 Configuring a New Application Server Connection

To configure a new Application Server connection in Oracle JDeveloper:

1. Open Oracle JDeveloper on your system.
2. From the menu bar, click **View** and select **Application Server Navigator**.

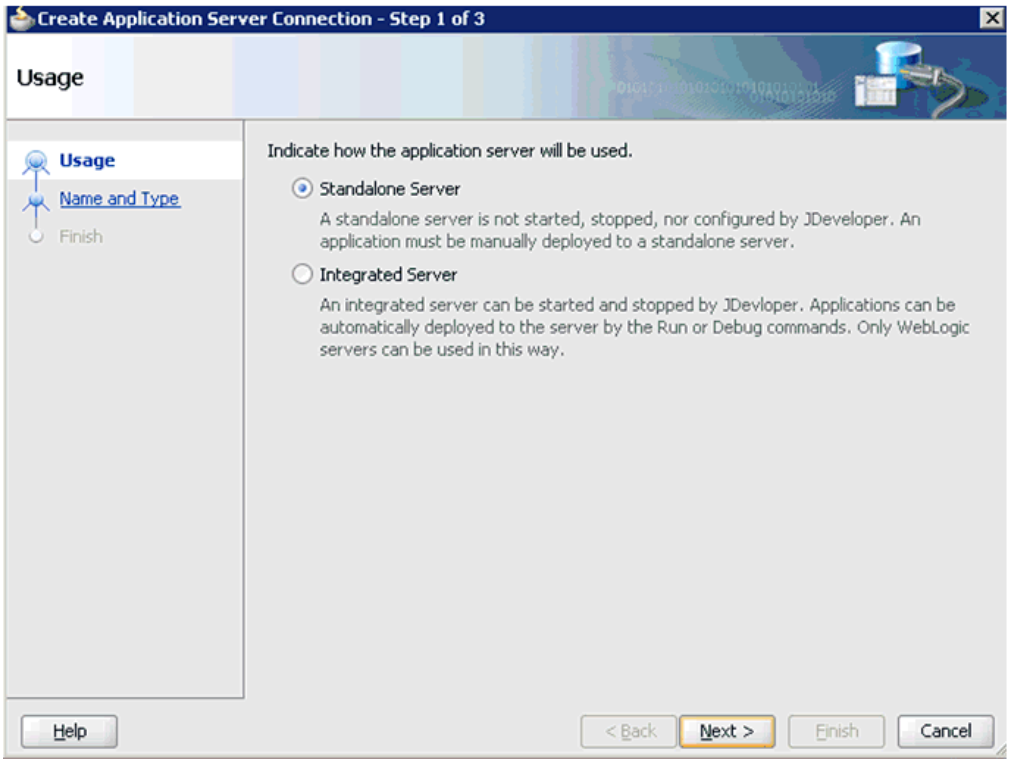


The Application Server tab is displayed.



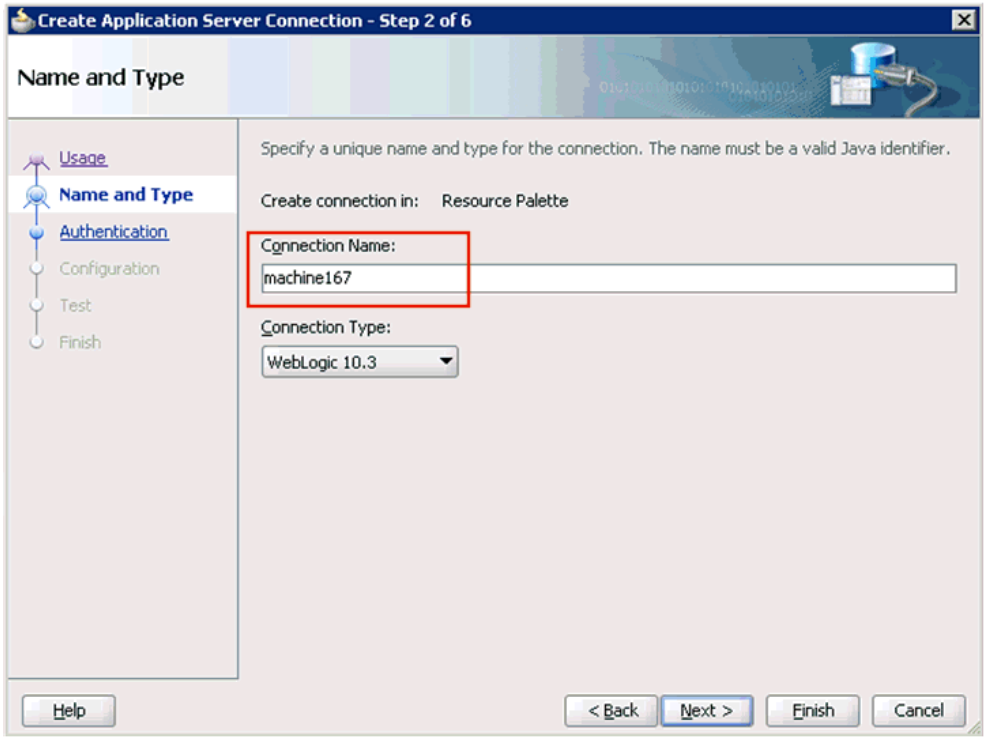
3. Right-click **Application Servers**, and then select **New Application Server**.

   The Create Application Server Connection Wizard is displayed.



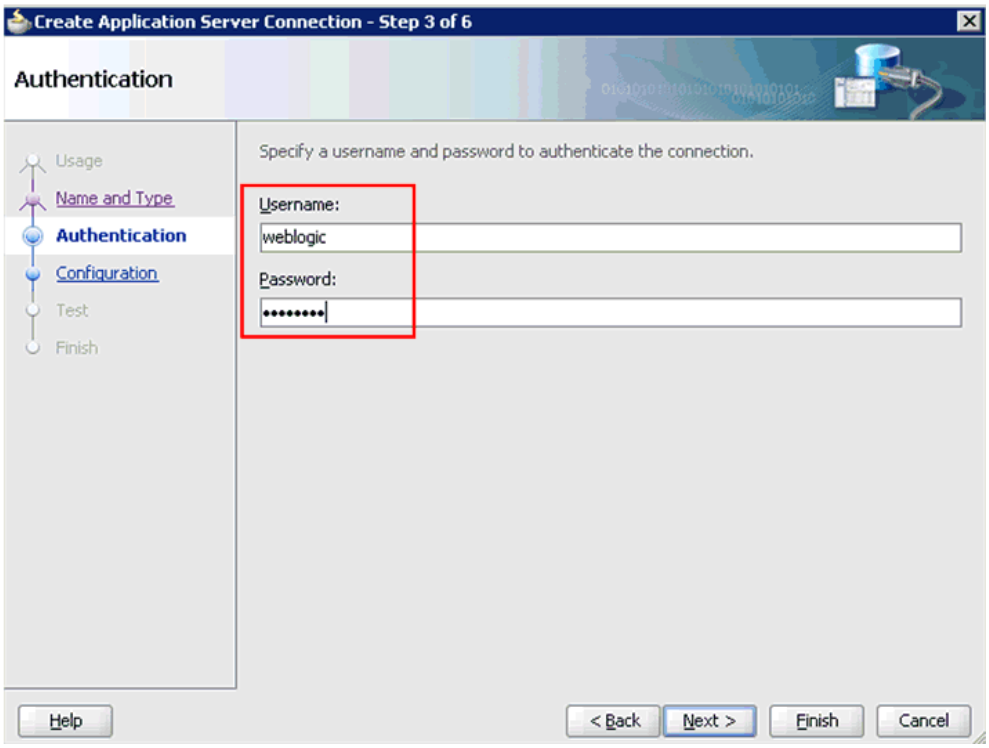
4. Accept the default selection (Standalone Server) and click **Next**.

   The Name and Type page is displayed.



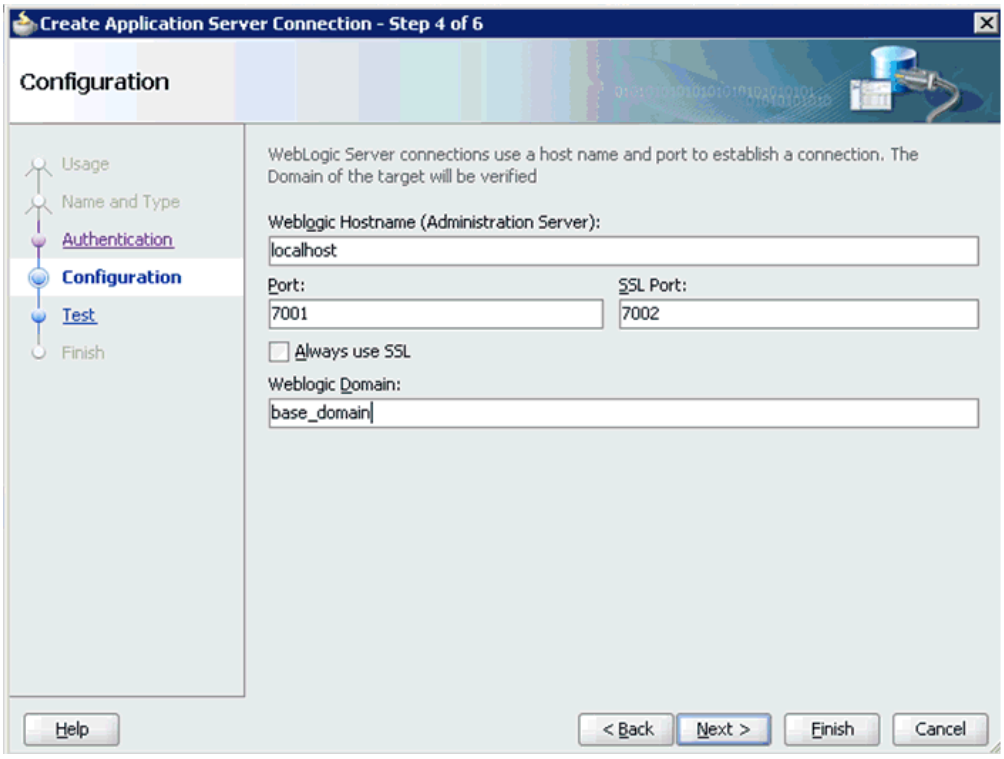
5. Specify a new name for the Application Server connection and click **Next**.

   The Authentication page is displayed.



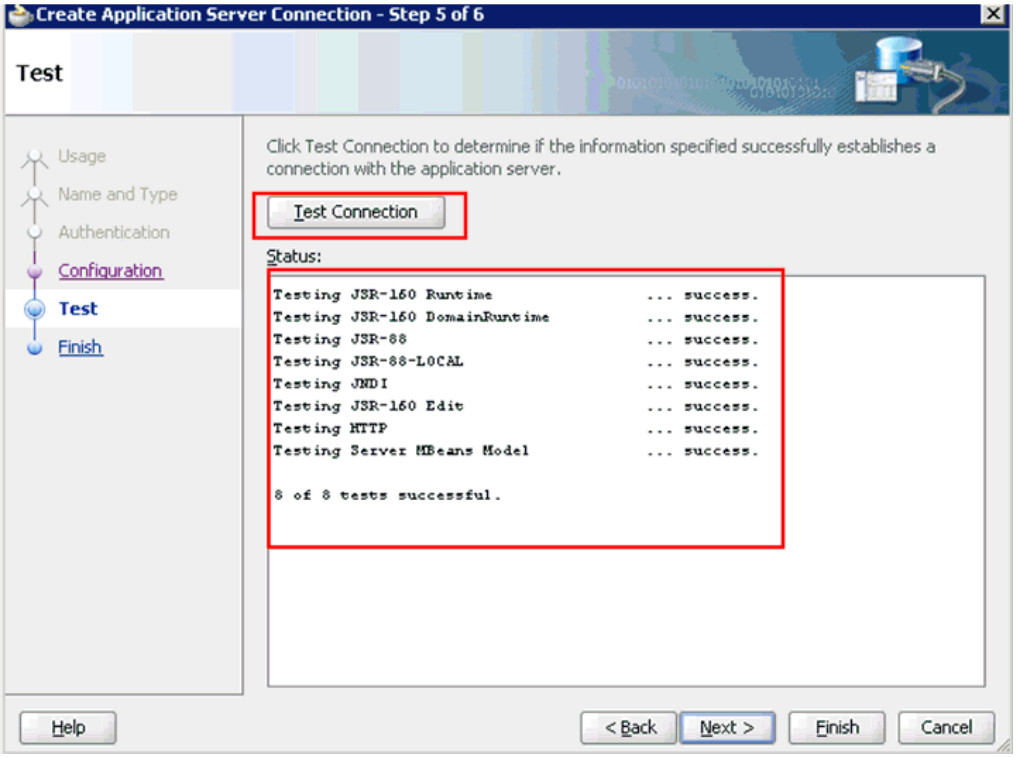
6. Specify a valid user name (for example, weblogic) and a password (for example, welcome1) for your new connection.
7. Click **Next**.

   The Configuration page is displayed.



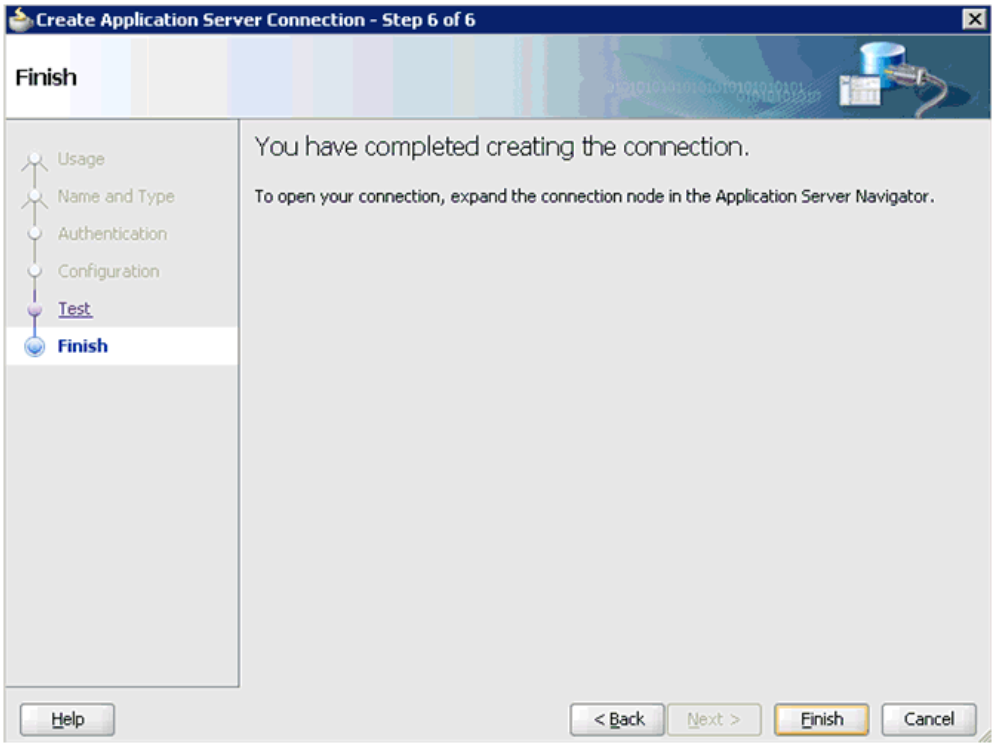
8. Specify the Oracle WebLogic host name (for example, localhost), which is the machine IP where the process needs to deploy and Oracle WebLogic domain (for example, base_domain).
9. Click **Next**.

   The Test page is displayed.



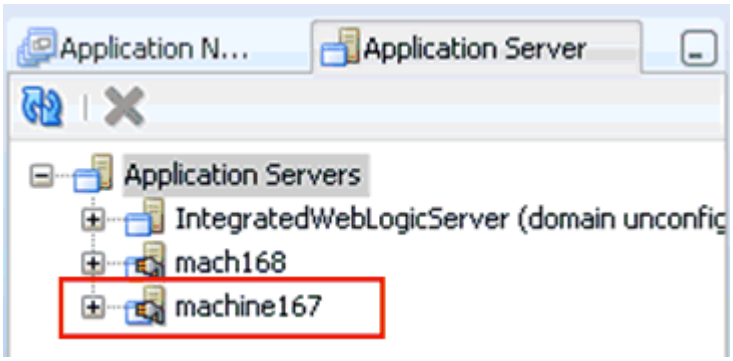
10. Click **Test Connection**.
11. Ensure that the test status is successful.
12. Click **Next**.

    The Finish page is displayed.



13. Click **Finish**.

    The new Application Server connection is listed in the left pane (Application Server tab), as shown in the following image.



4.4 Designing an Outbound BPEL Process for Service Integration

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

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

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

Before you design a BPEL process, you must generate WSDL using Application Explorer. See "Generating WSDL for Request/Response Service" on page 4-7 for more

information. The WSDL generated in Application Explorer is used during the BPEL process configuration.

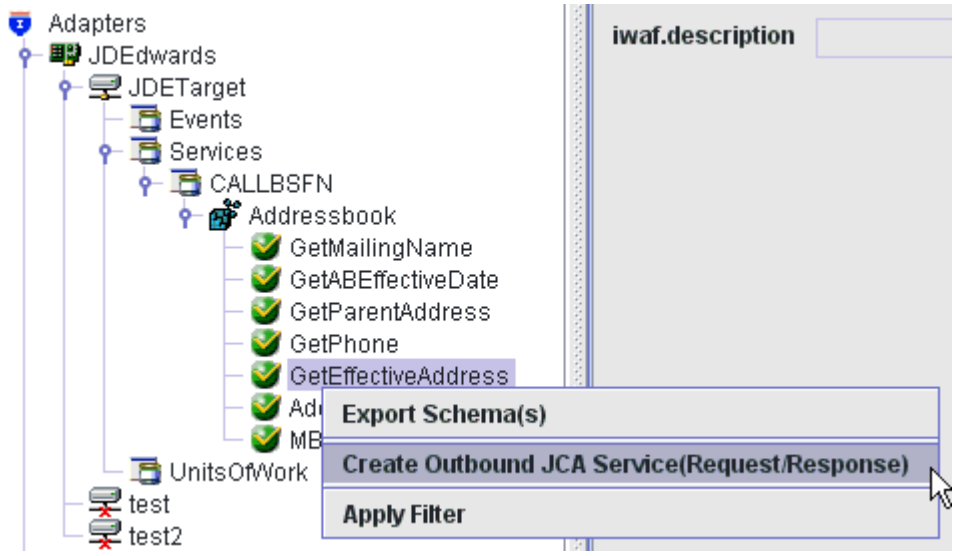
4.4.1 Generating WSDL for Request/Response Service

Perform the following steps to generate a WSDL for request/response service:

1. Start Application Explorer and connect to a defined J.D. Edwards OneWorld target.

   For more information, see "Defining a Target to J.D. Edwards OneWorld" on page 2-11.

2. Expand **Services**, **CALLBSFN**, and **Addressbook**.



3. Right-click **GetEffectiveAddress**, and then select **Create Outbound JCA Service (Request/Response)**.

   The Export WSDL dialog is displayed.

4. Accept the default name 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**.

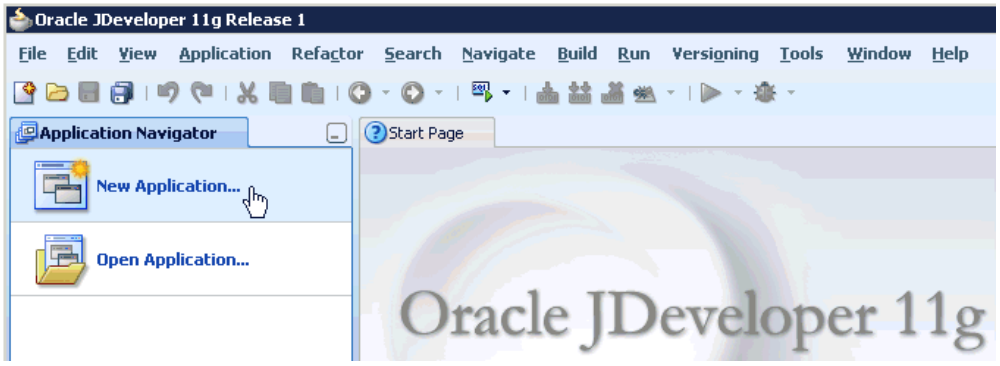
5. Click **OK**.

You can now create a new SOA application, which is the first step that is required to define a BPEL outbound process in Oracle JDeveloper.

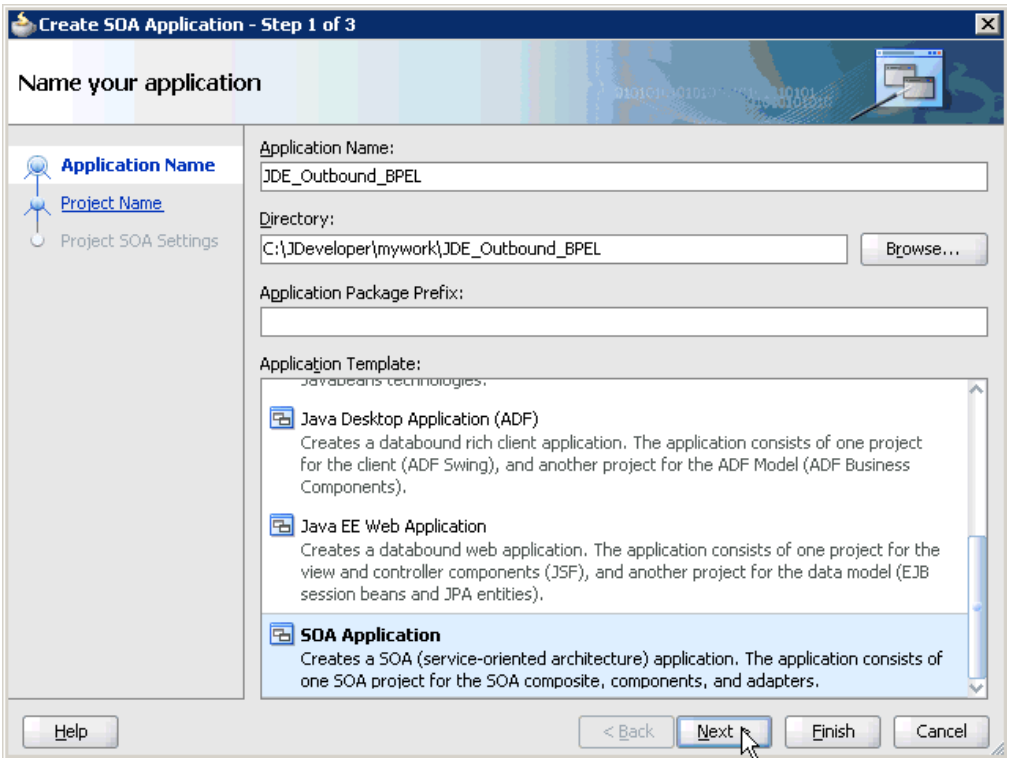
4.4.2 Creating a New SOA Application for the Outbound BPEL Process

Perform the following steps to create a new SOA application for the outbound BPEL process:

1. Open Oracle JDeveloper on your system.
2. In the Application Navigator tab, click **New Application**.

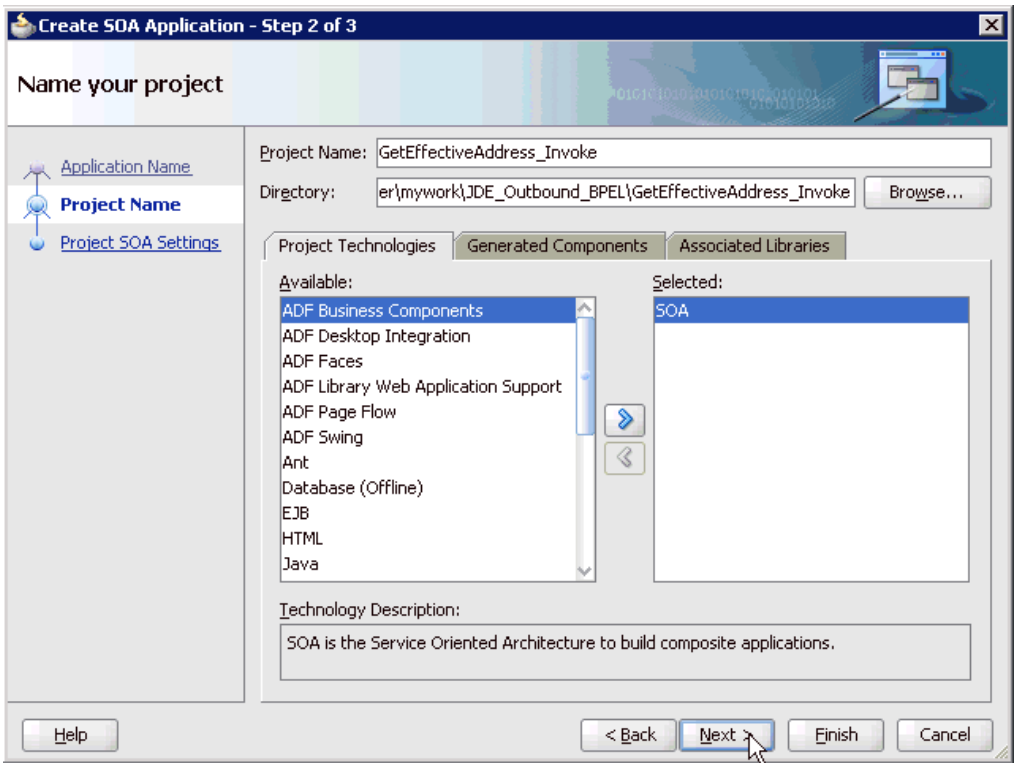


The Create SOA Application wizard is displayed.

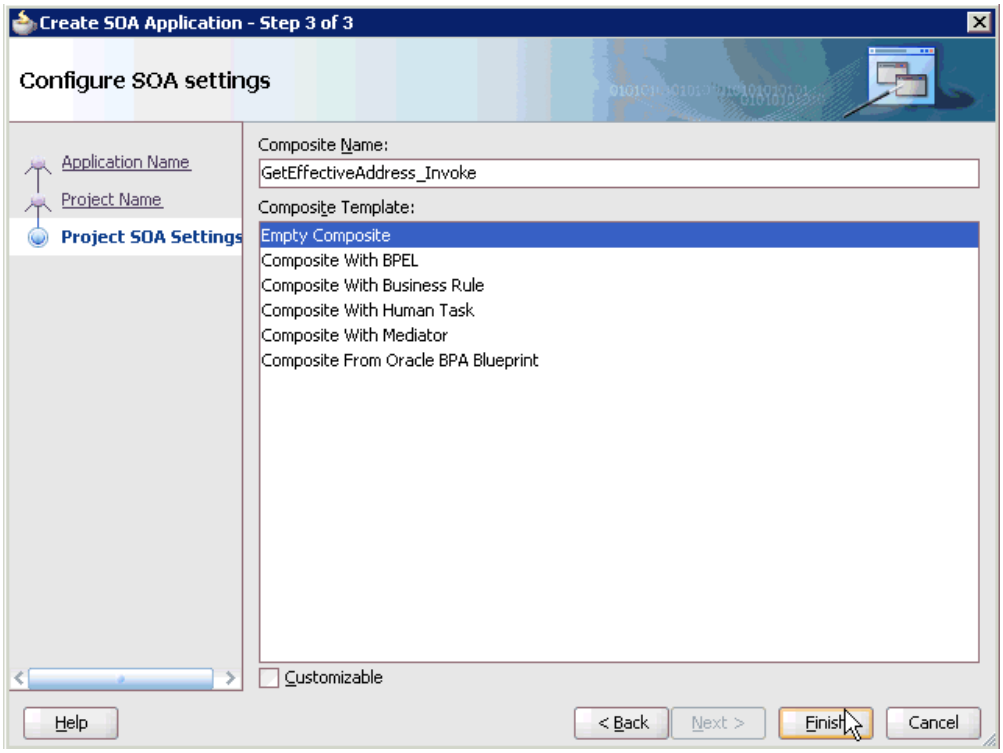


3. From the Application Template list, click **SOA Application**.
4. Enter name for the new SOA application (for example, JDE_Outbound_BPEL) and click **Next**.

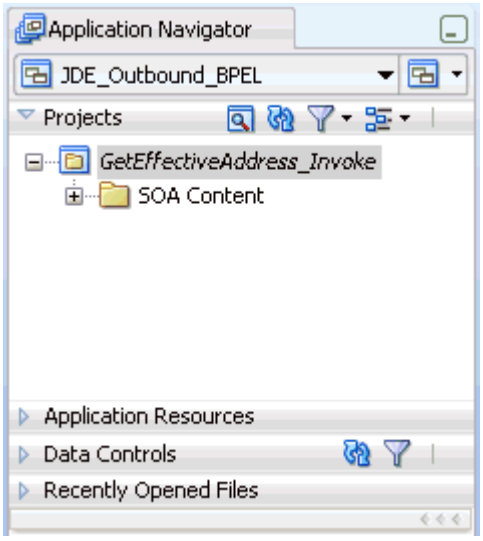
   The Name your project page is displayed.



5. Enter a project name (for example, GetEffectiveAddress_Invoke) and click **Next**. The Configure SOA settings page is displayed.



6. From the Composite Template list, select **Empty Composite** and click **Finish**.



The new SOA application (JDE_Outbound_BPEL) and associated project (GetEffectiveAddress_Invoke) are added to the Application Navigator tab in the left pane.

4.4.3 Defining a BPEL Outbound Process

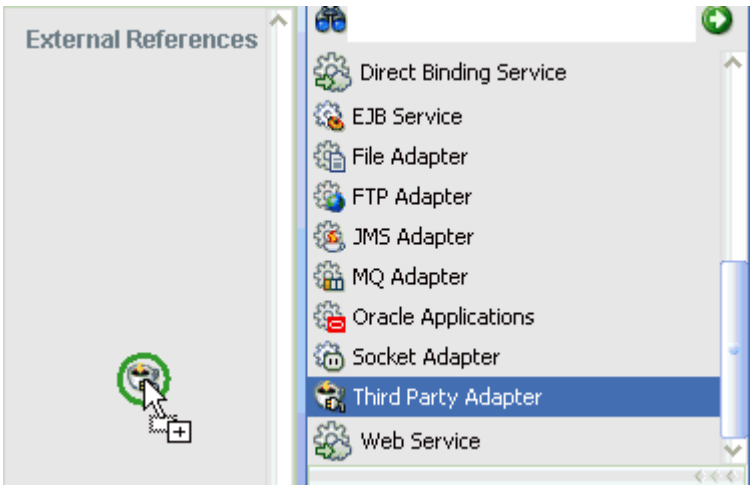
This section describes how to define a BPEL outbound process, which consists of the following stages:

1. Configuring a Third Party Adapter Service Component
2. Configuring an Outbound BPEL Process Component

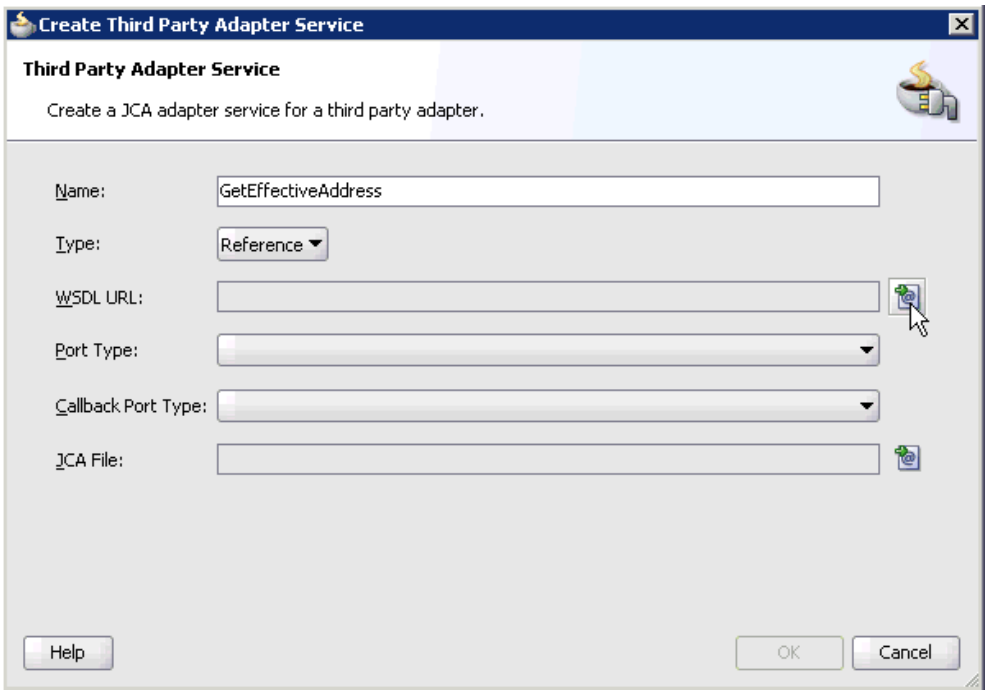
Configuring a Third Party Adapter Service Component

Perform the following steps to create a third party adapter service component:

1. Drag and drop the **Third Party Adapter** component from the Component Palette tab (Service Adapters section) to the External References pane.

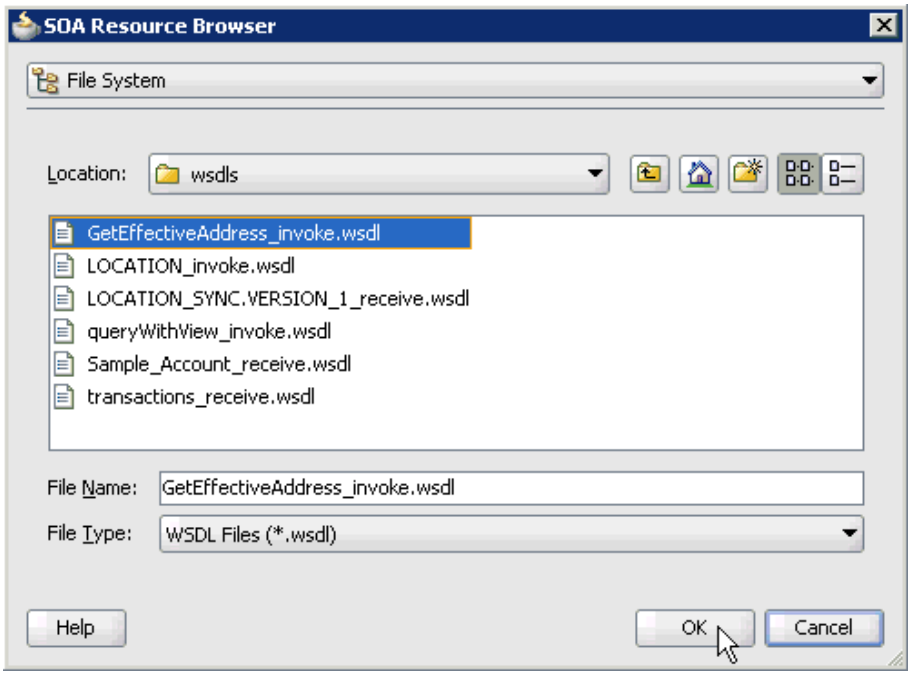


The Create Third Party Adapter Service dialog is displayed.



2. Enter a name for the new third party adapter service.
3. Ensure that **Reference** is selected from the Type list (default).
4. Click the **Find existing WSDLs** icon, which is located to the right of the WSDL URL field.

   The SOA Resource Browser dialog is displayed.

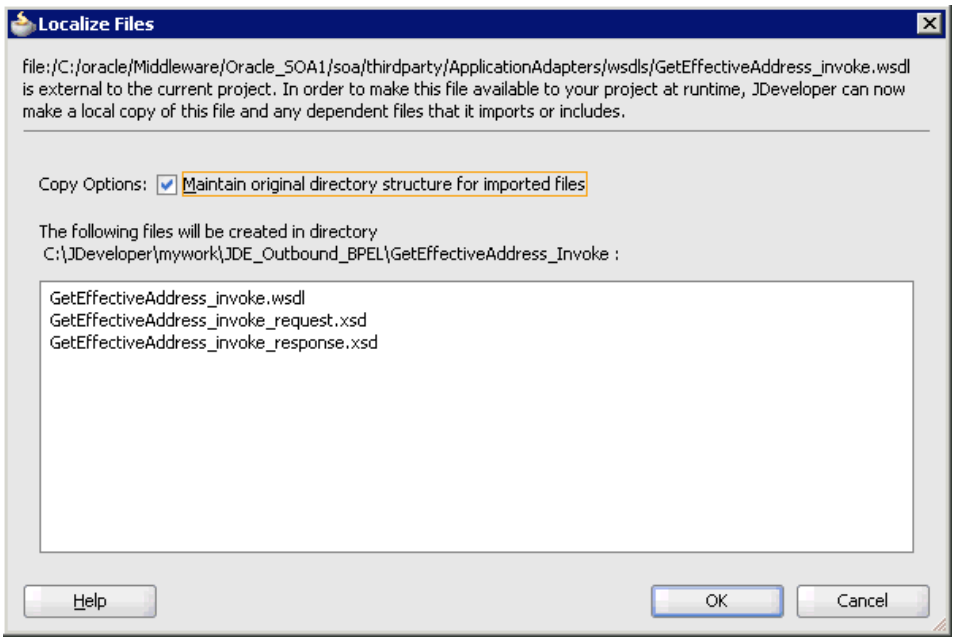


5. Browse and select an outbound WSDL file (for example, GetEffectiveAddress_invoke.wsdl) from the following directory:

   `C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\wsdls`

6. Click **OK**.

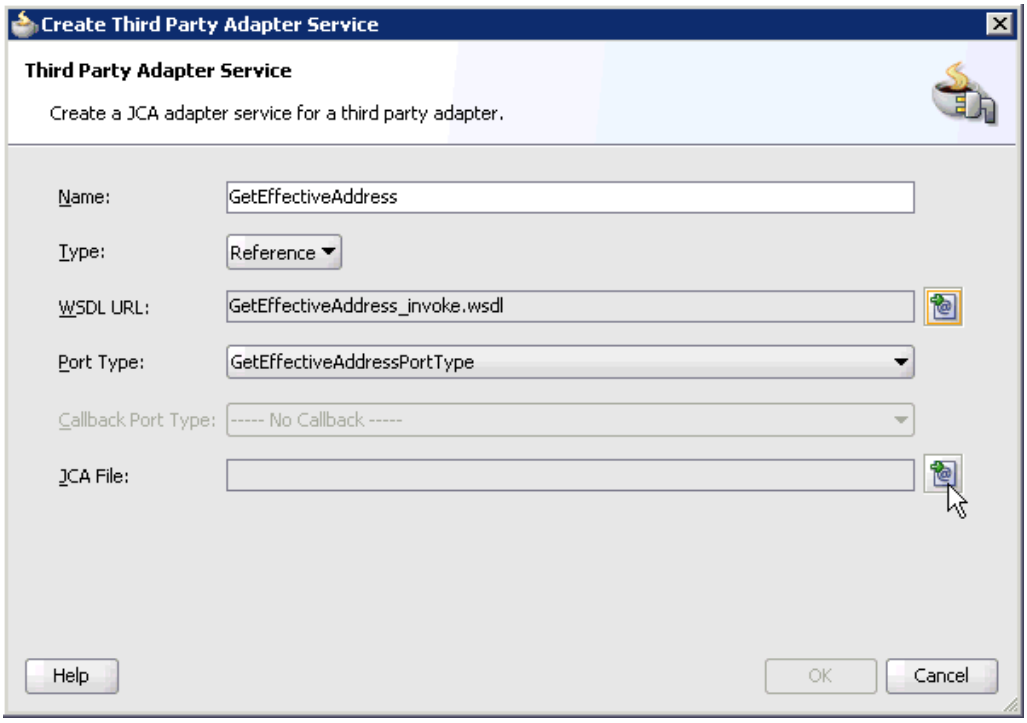
   The Localize Files dialog is displayed.



7. Click **OK**.

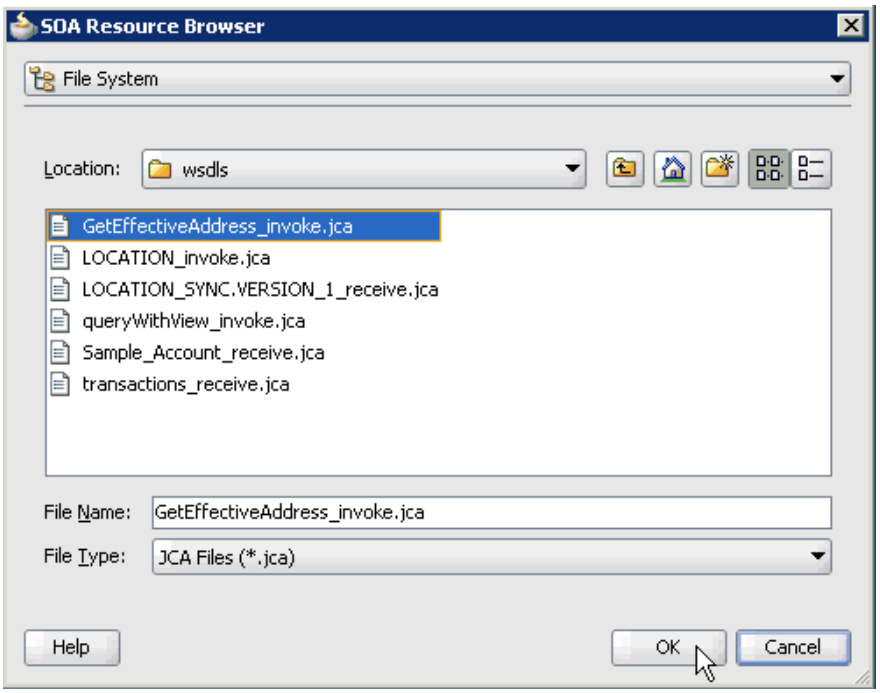
   The outbound WSDL file and associated request and response XML schema files (.xsd) are imported to the project folder that has been created.

   You are returned to the Create Third Party Adapter Service dialog.



8. Click the **Find JCA Files** icon, which is located to the right of the JCA File field.

The SOA Resource Browser dialog is displayed.

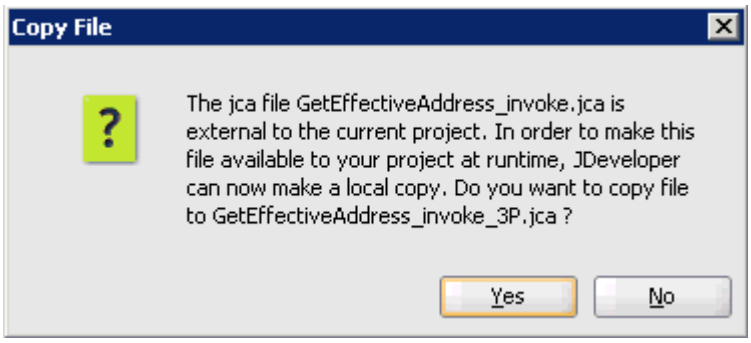


9. Browse and select the JCA properties file (for example, GetEffectiveAddress_invoke.jca) from the following directory:

   `C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\wsdls`

10. Click **OK**.

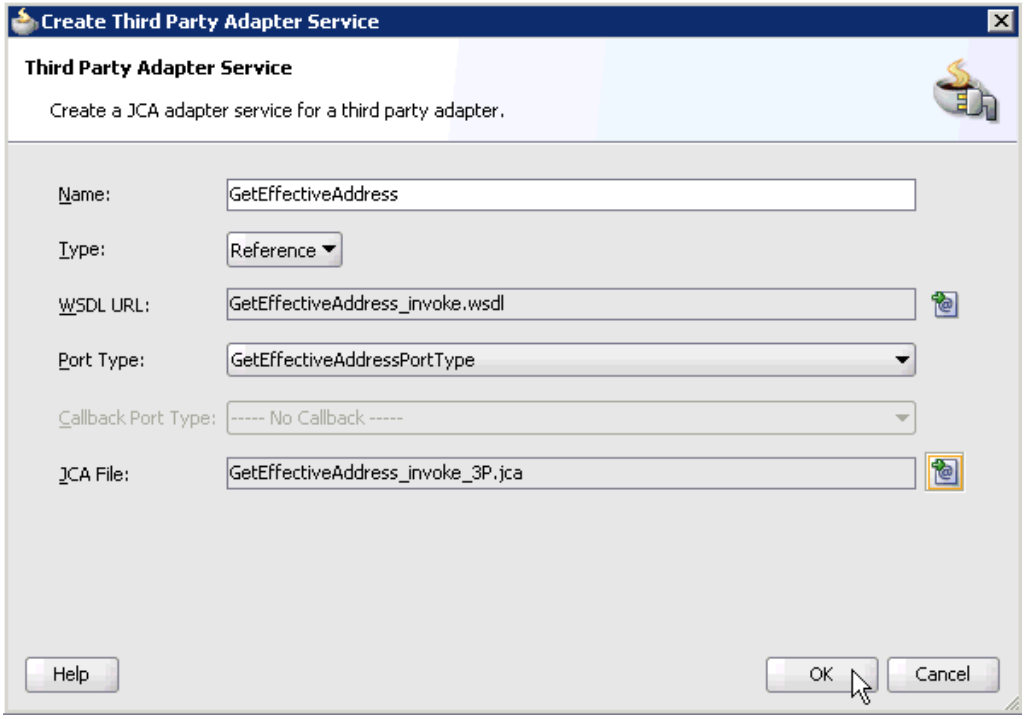
    The following message is displayed.



11. Click **Yes**.

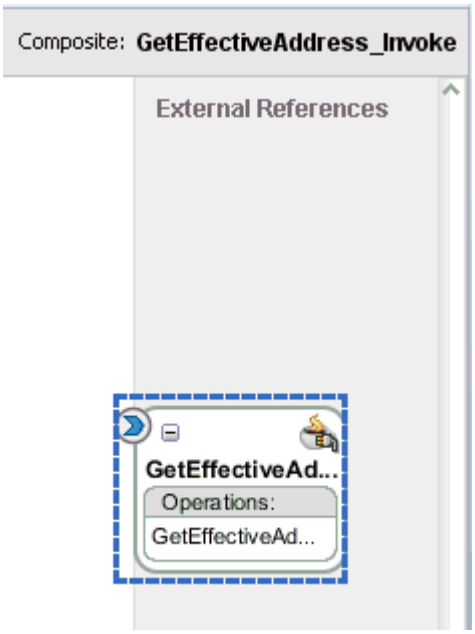
    A copy of the JCA properties file is made in the project folder.

    You are returned to the Create Third Party Adapter Service dialog.



12. Click **OK**.

    The third party adapter service component (GetEffectiveAddress) is created in the External References pane, as shown in the following image.

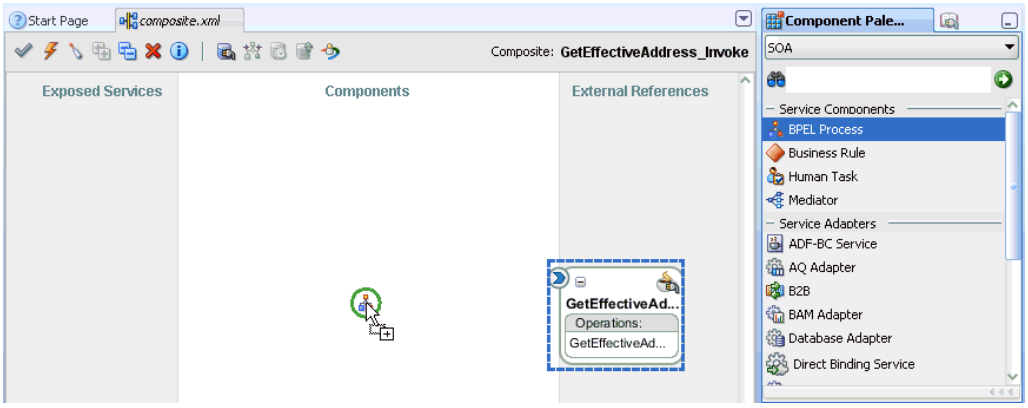


    You are now ready to configure an outbound BPEL process component.

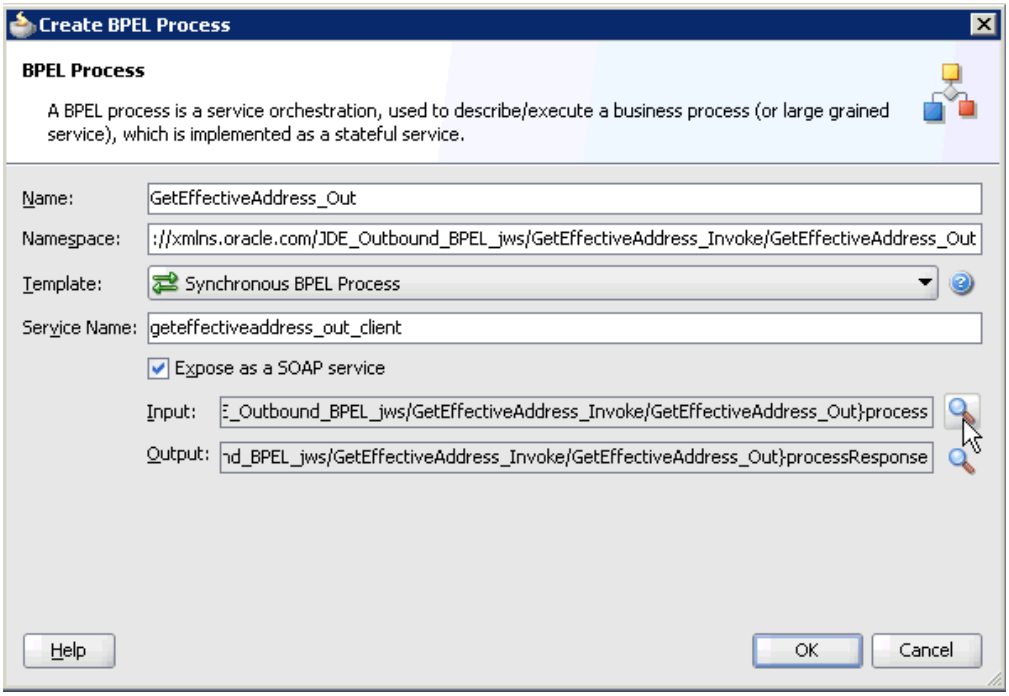
Configuring an Outbound BPEL Process Component

Perform the following steps to configure an outbound BPEL process component:

1. Drag and drop the **BPEL Process** component from the Component Palette tab (Service Components section) to the Components pane.

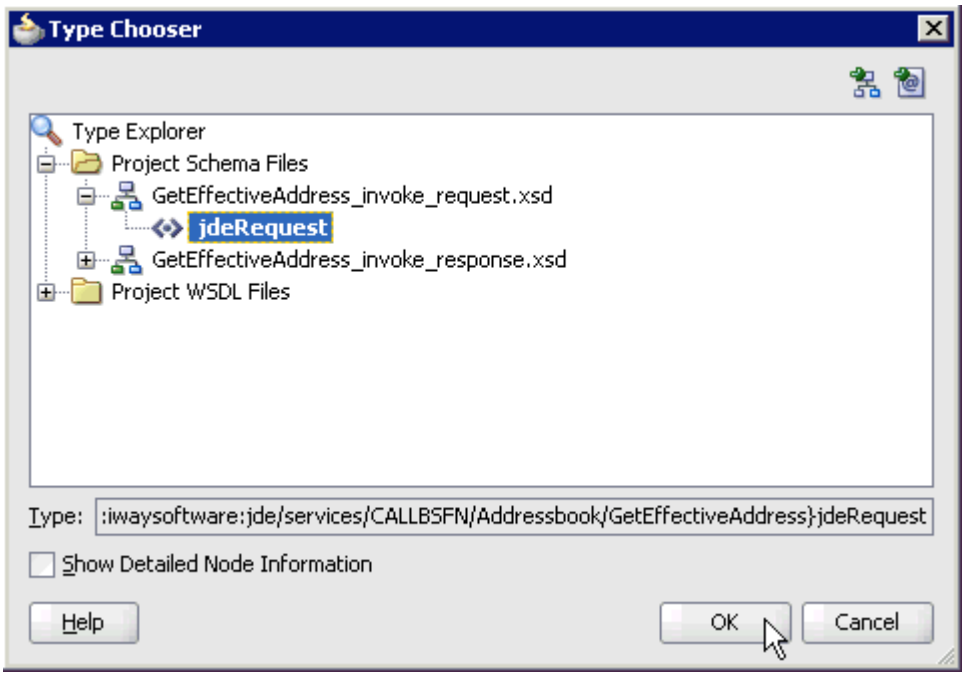


The Create BPEL Process dialog is displayed.



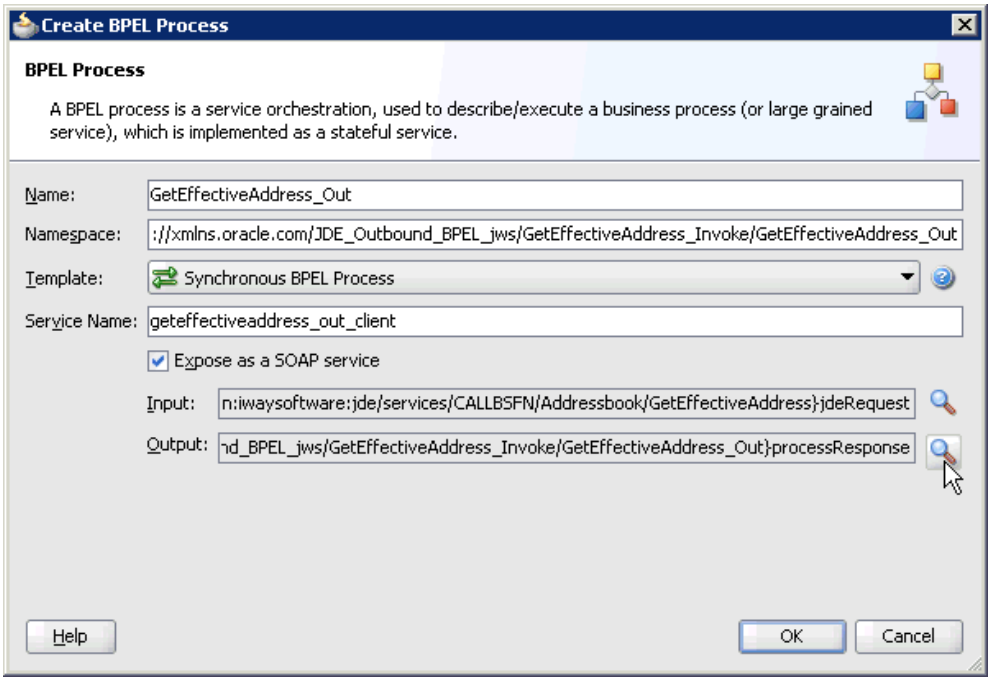
2. In the Name field, enter a name to identify the new outbound BPEL process component (for example, GetEffectiveAddress_Out).
3. From the Template list, select **Synchronous BPEL Process**.
4. Click the **Browse Input Elements** icon, which is located to the right of the Input field to select the associated XML request schema file.

   The Type Chooser dialog is displayed.



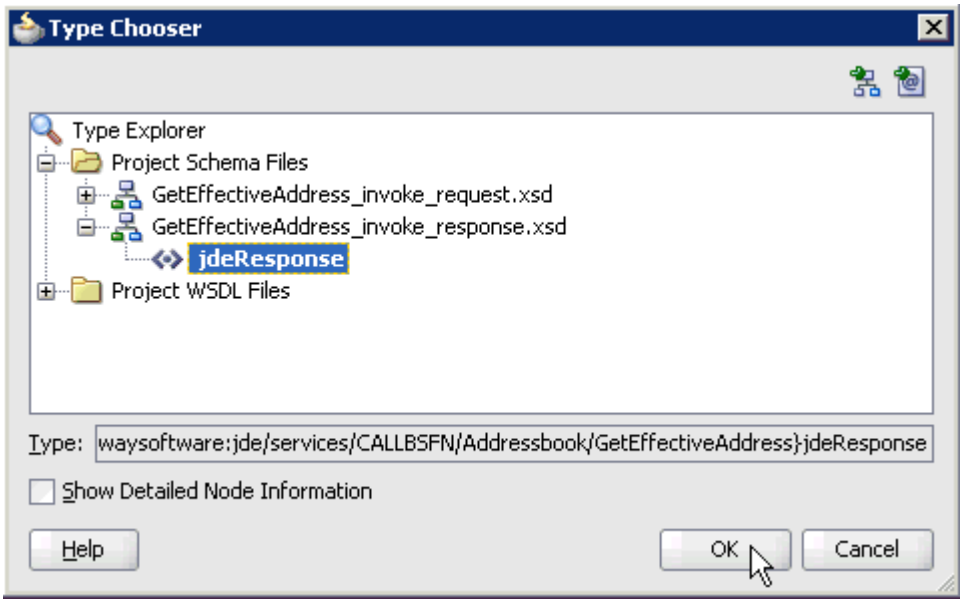
5. Expand **Project Schema Files**, **GetEffectiveAddress_invoke_request.xsd**, and select **jdeRequest**.
6. Click **OK**.

   You are returned to the Create BPEL Process dialog.



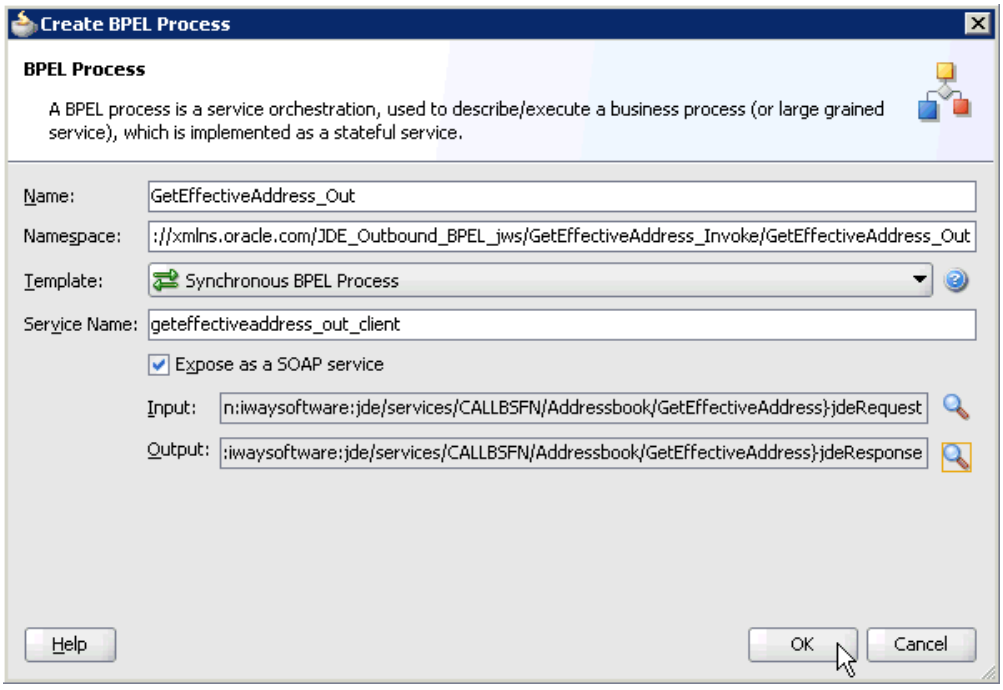
7. Click the **Browse Output Elements** icon, which is located to the right of the Output field to select the associated XML response schema file.

   The Type Chooser dialog is displayed.

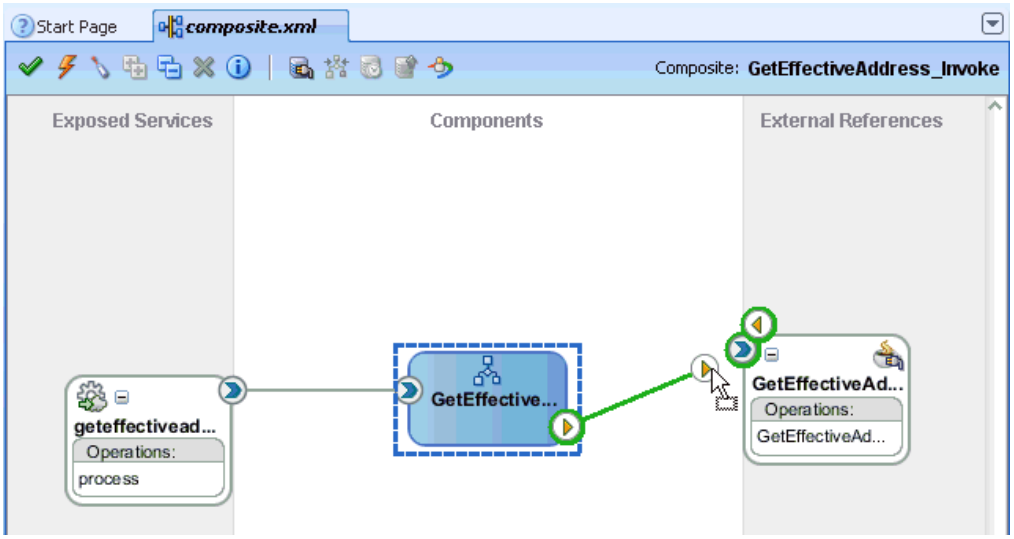


8. Expand **Project Schema Files**, **GetEffectiveAddress_invoke_response.xsd**, and select **jdeResponse**.
9. Click **OK**.

   You are returned to the Create BPEL Process dialog.

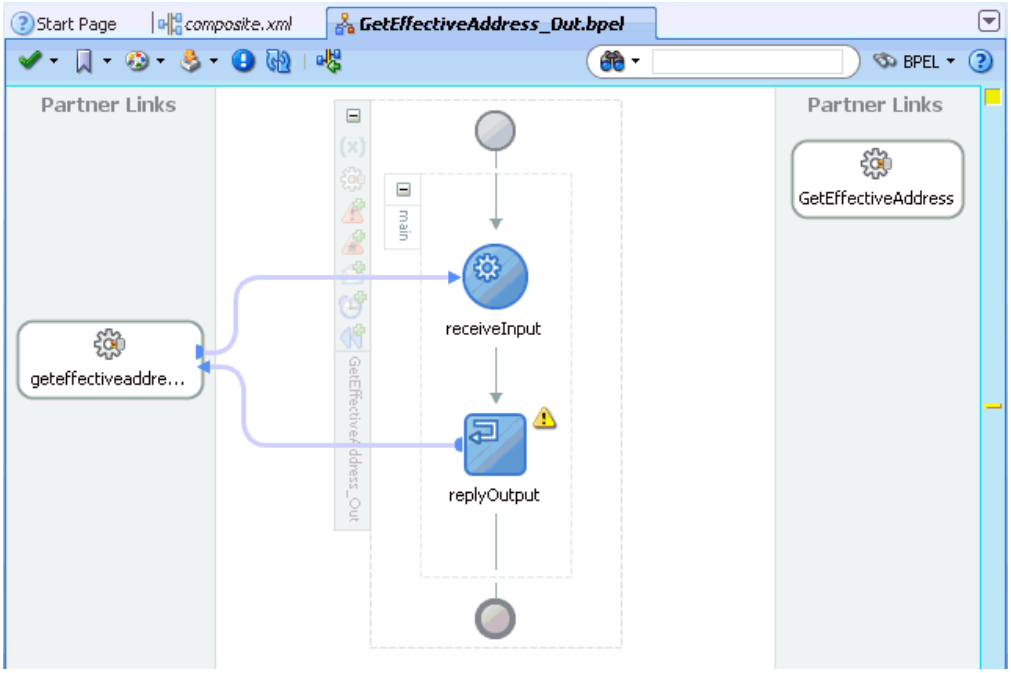


10. Click **OK**.
11. Create a connection between the outbound BPEL process component (GetEffectiveAddress_Out) and the third party adapter service component (GetEffectiveAddress).

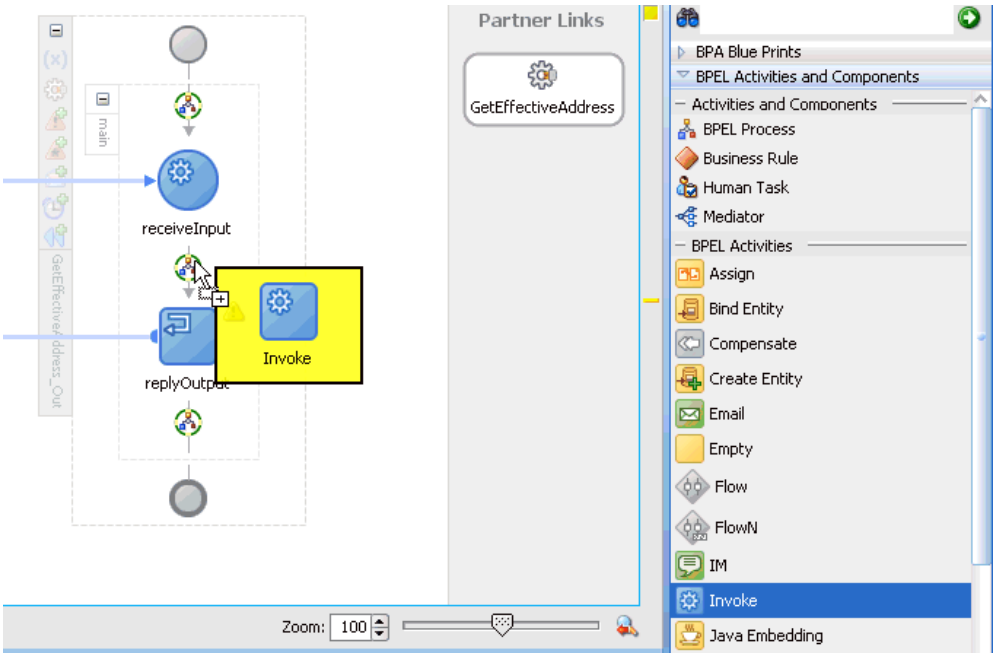


12. Double-click the outbound BPEL process component (GetEffectiveAddress_Out) in the Components pane.

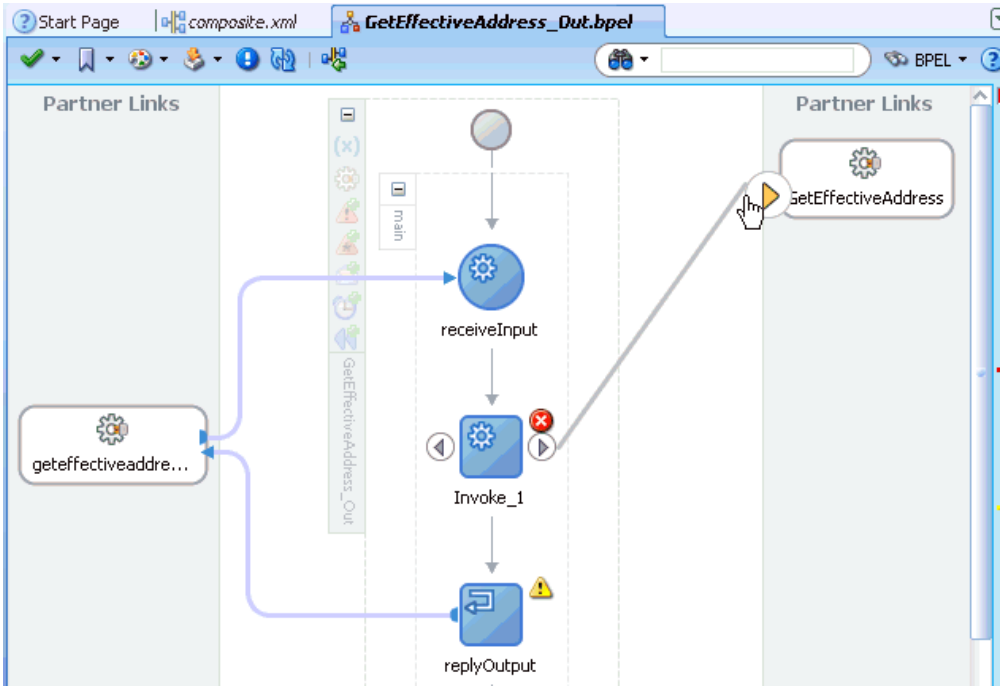
    The following is displayed.



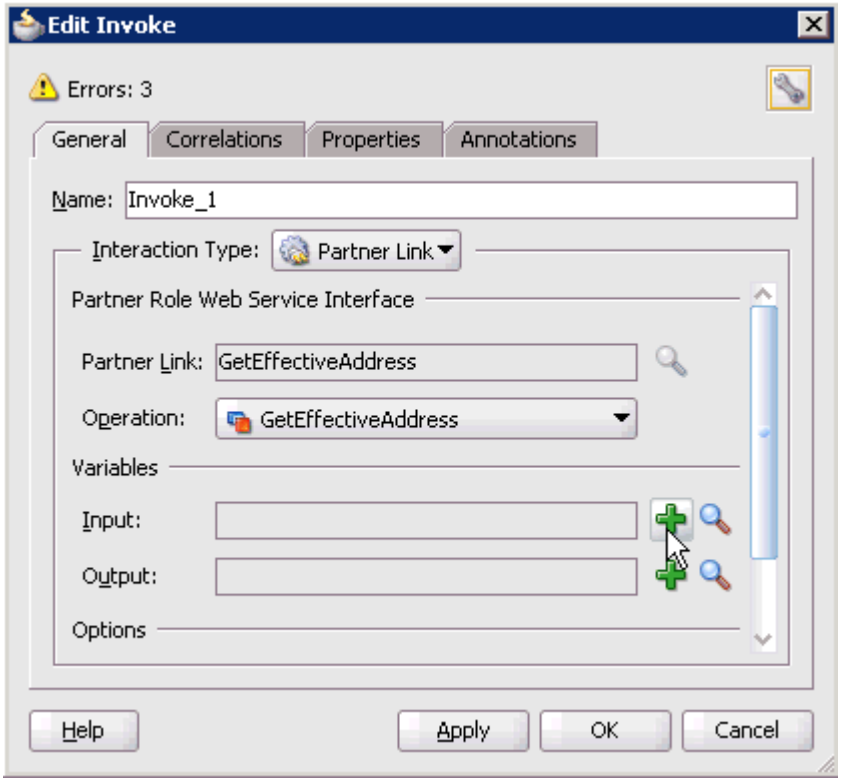
13. Drag and drop the **Invoke** activity component to the Components pane and place it between the **receiveInput** activity component and the **replyOutput** activity component.



14. Create a connection between the new Invoke activity component (Invoke_1) and the third party adapter service component (GetEffectiveAddress).

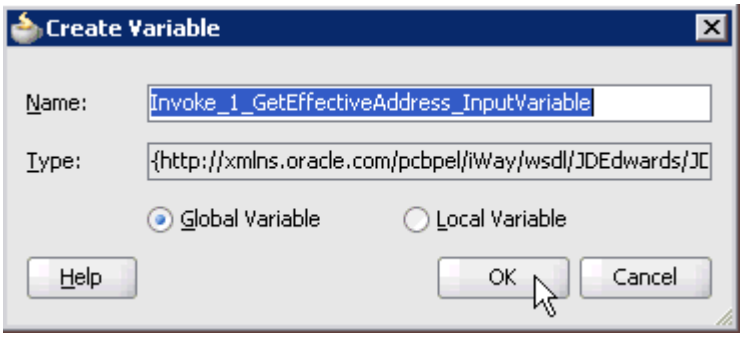


The Edit Invoke dialog is displayed.



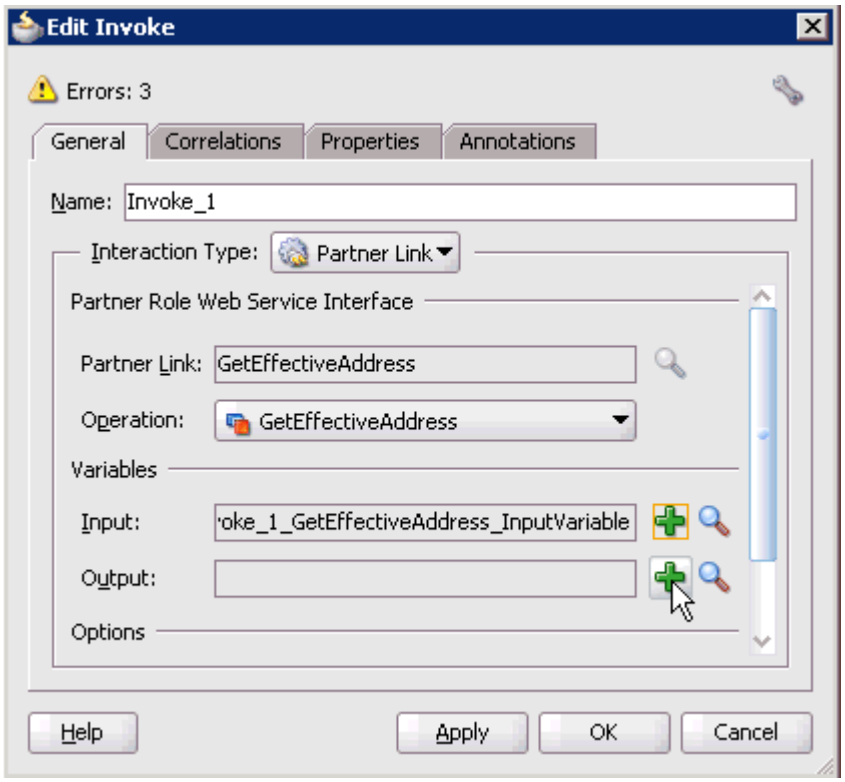
15. Click the **Automatically Create Input Variable** icon, which is located to the right of the Input field to configure a new input variable.

    The Create Variable dialog is displayed.



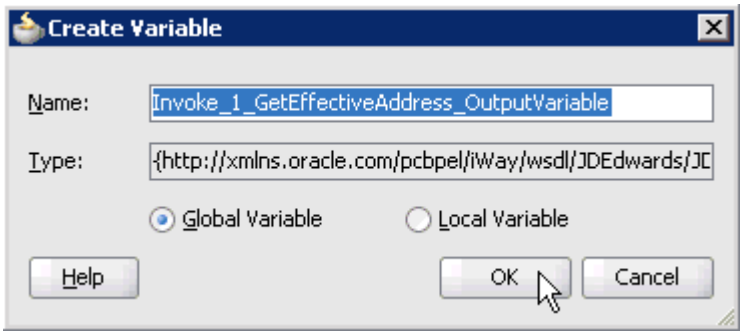
16. Accept the default values that are provided for the new input variable and click **OK**.

    You are returned to the Edit Invoke dialog.



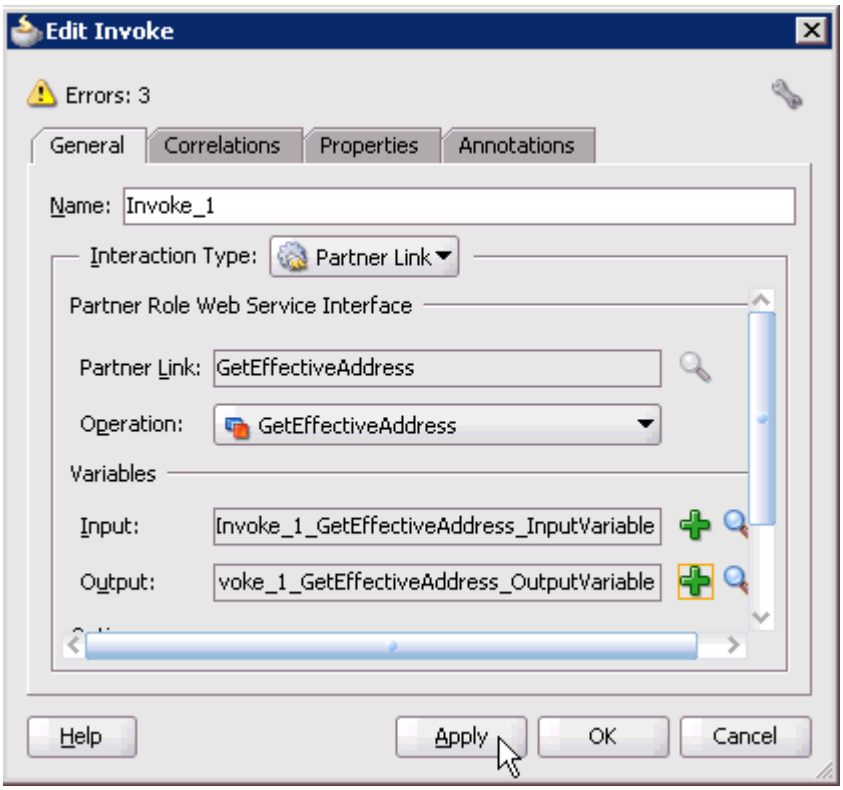
17. Click the **Automatically Create Output Variable** icon, which is located to the right of the Output field to configure a new output variable.

    The Create Variable dialog is displayed.



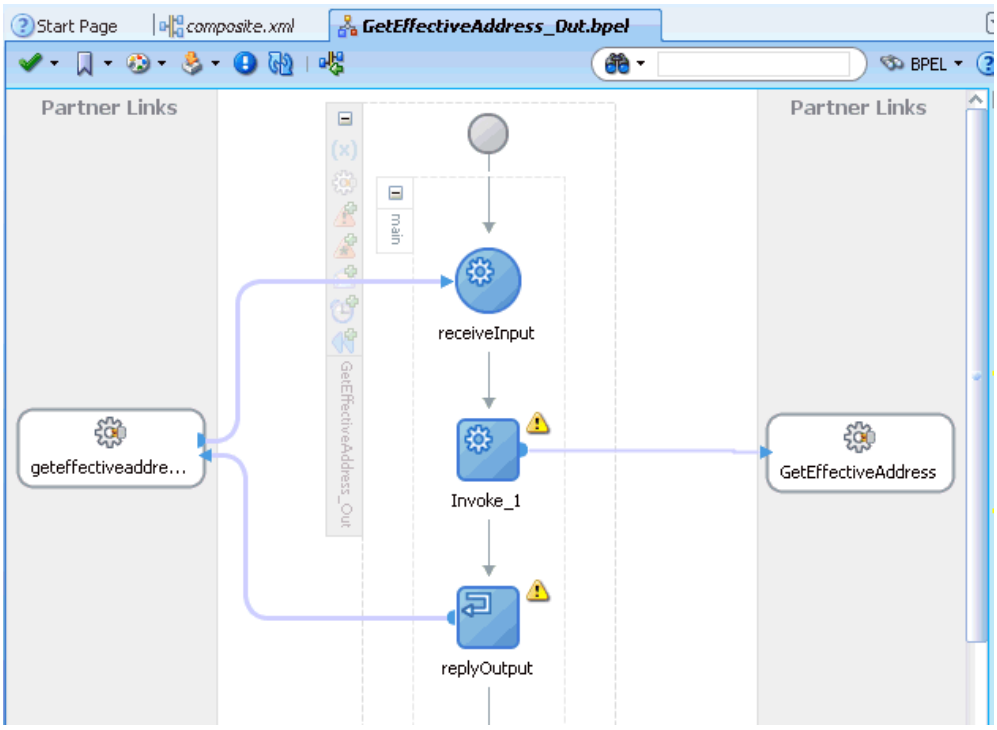
18. Accept the default values that are provided for the new output variable and click **OK**.

    You are returned to the Edit Invoke dialog.

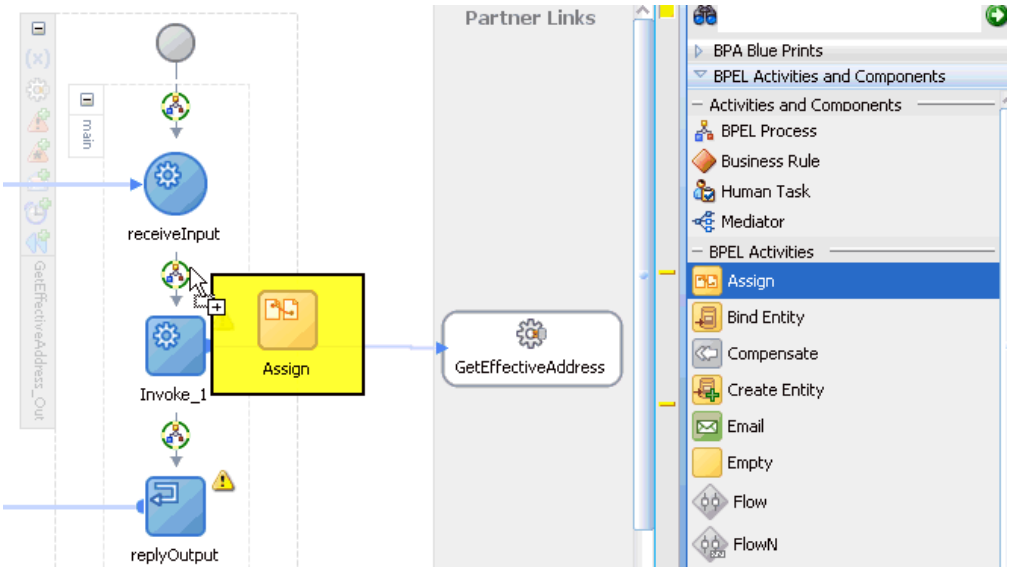


19. Click **Apply** and then **OK**.

    The Invoke activity component (Invoke_1) is updated accordingly.

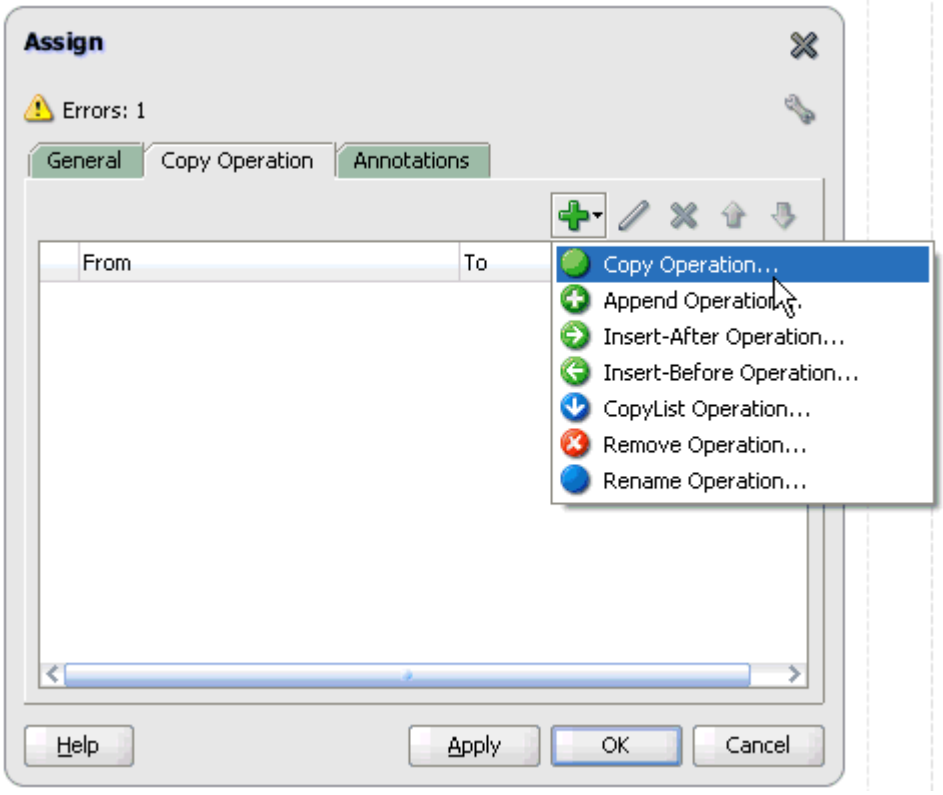


20. Drag and drop the **Assign** activity component to the Components pane and place it between the Receive activity component (receiveInput) and the Invoke activity component (Invoke_1).



21. Double-click the new Assign activity component (**Assign_1**).

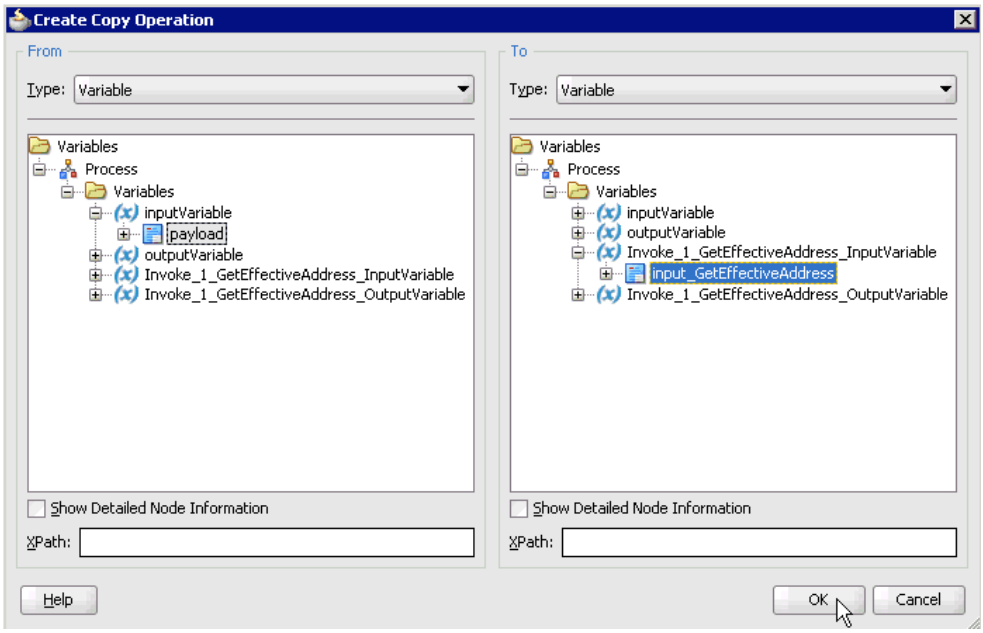
    The Assign dialog is displayed.



22. Click the **Copy Operation** tab.

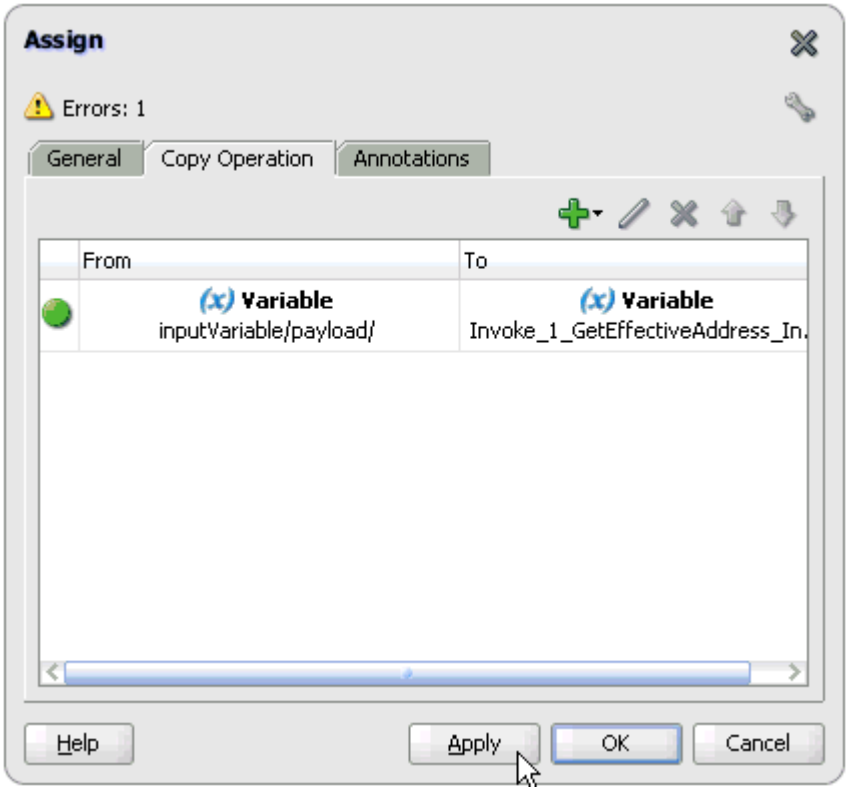
23. Click the **Plus sign** icon and select **Copy Operation** from the list of available operations.

    The Create Copy Operation dialog is displayed.



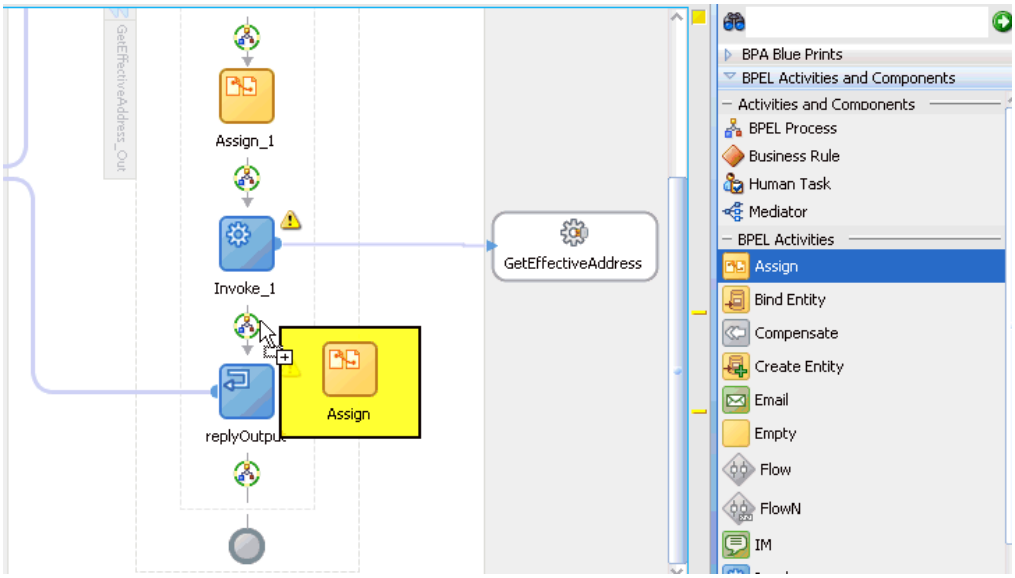
24. In the From pane, expand **Variables**, **InputVariable**, and then select **payload**.
25. In the To pane, expand **Variables**, **Invoke_1_GetEffectiveAddress_InputVariable**, and then select **input_GetEffectiveAddress**.
26. Click **OK**.

    You are returned to the Assign dialog.



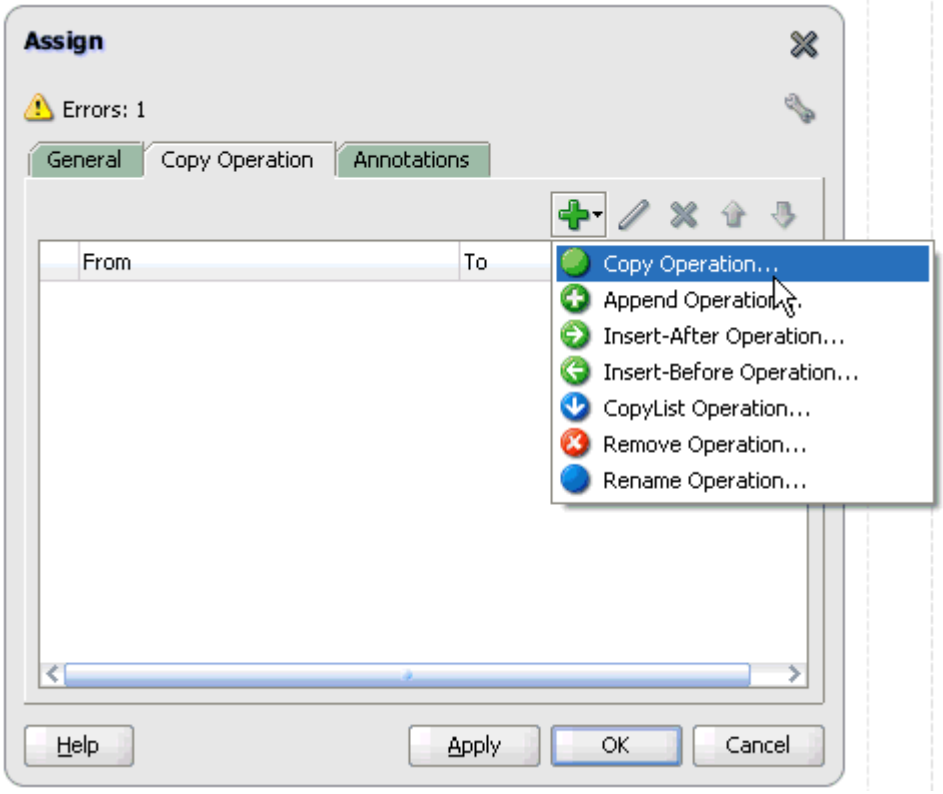
27. Click **Apply** and then **OK**.

28. Drag and drop the **Assign** activity component to the Components pane and place it between the Invoke activity (Invoke_1) and the Reply activity (replyOutput).



29. Double-click the new Assign activity component (**Assign_2**).

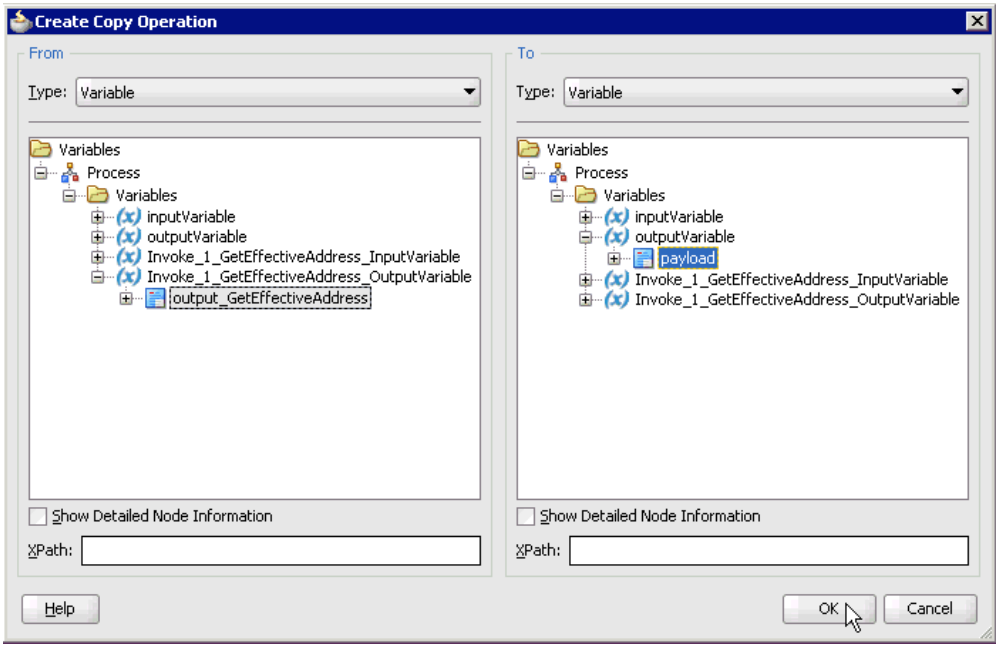
    The Assign dialog is displayed.



30. Click the **Copy Operation** tab.

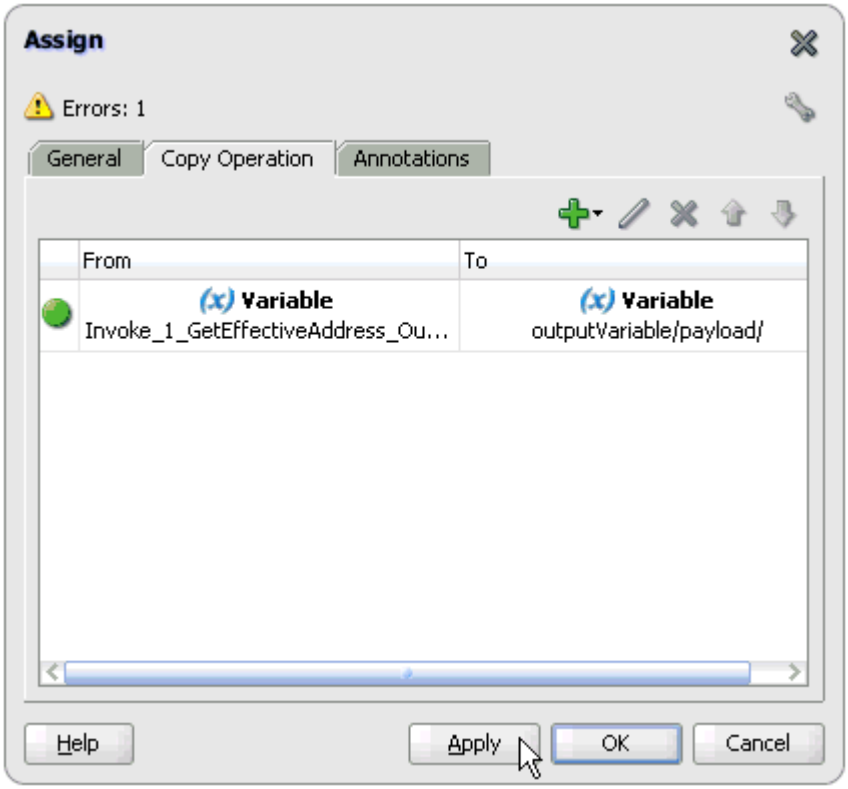
31. Click the **Plus sign** icon and select **Copy Operation** from the list of available operations.

    The Create Copy Operation dialog is displayed.



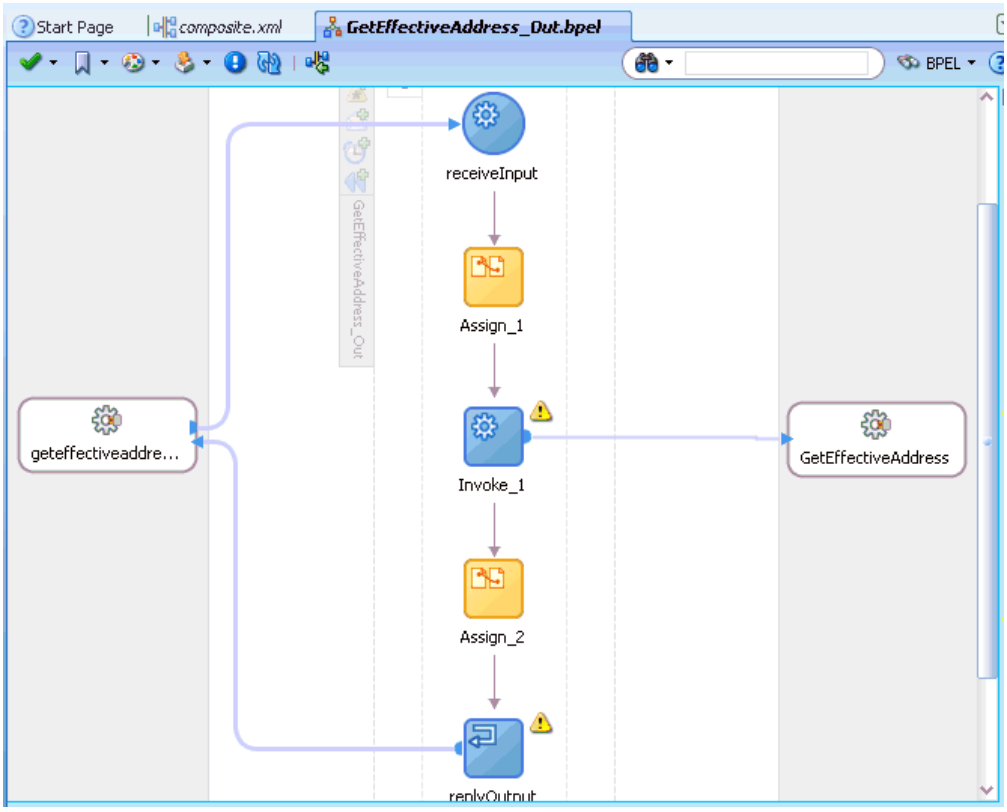
32. In the From pane, expand **Variables**, **Invoke_1_GetEffectiveAddress_OutputVariable**, and then select **output_GetEffectiveAddress**.
33. In the To pane, expand **Variables**, **outputVariable**, and then select **payload**.
34. Click **OK**.

    You are returned to the Assign dialog.

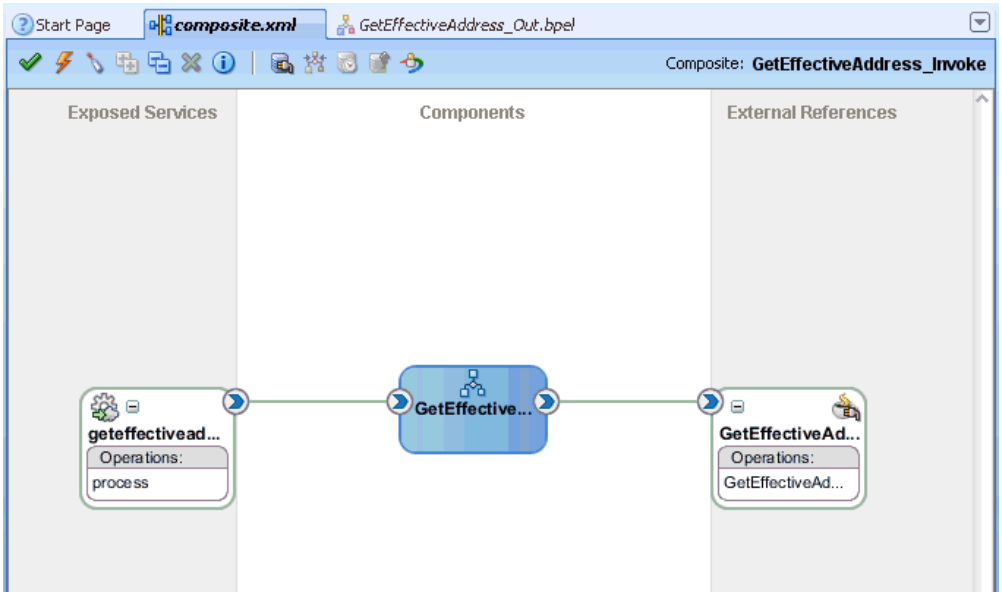


35. Click **Apply** and then **OK**.

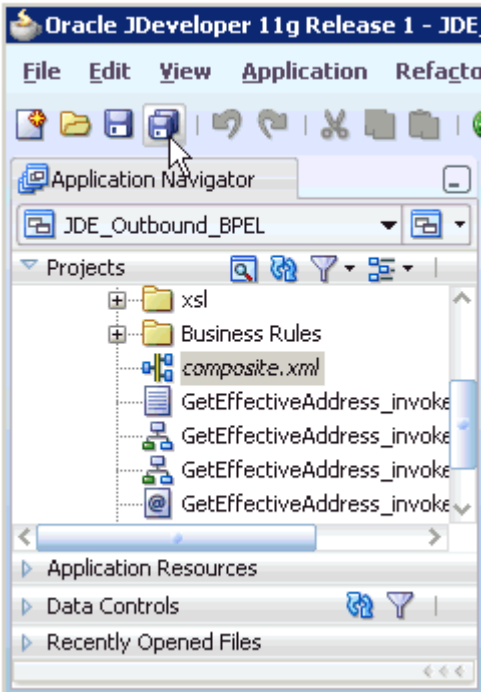
The completed activity flow is now displayed.



36. Double-click **composite.xml** in the left pane.



37. Click the **Save All** icon in the menu bar to save the new outbound BPEL process component that was configured.

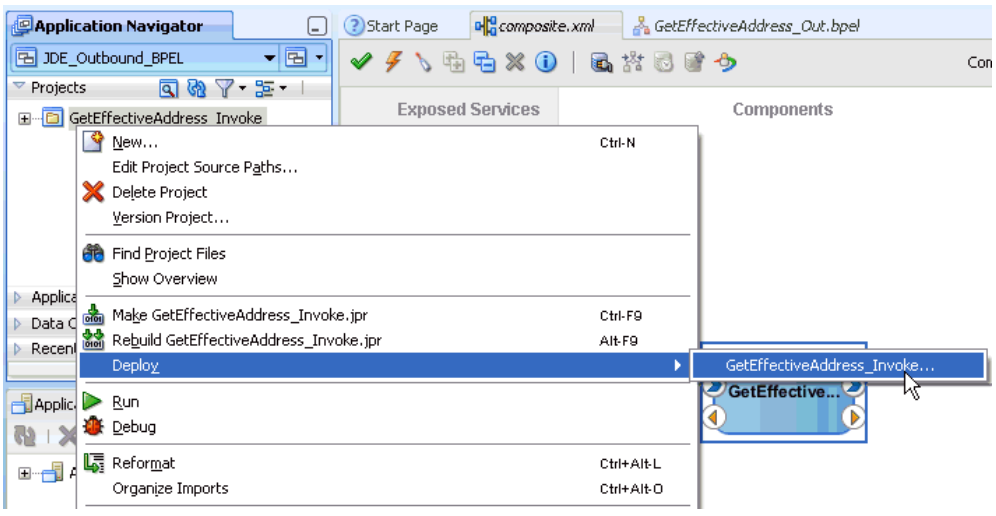


You are now ready to deploy the BPEL outbound process.

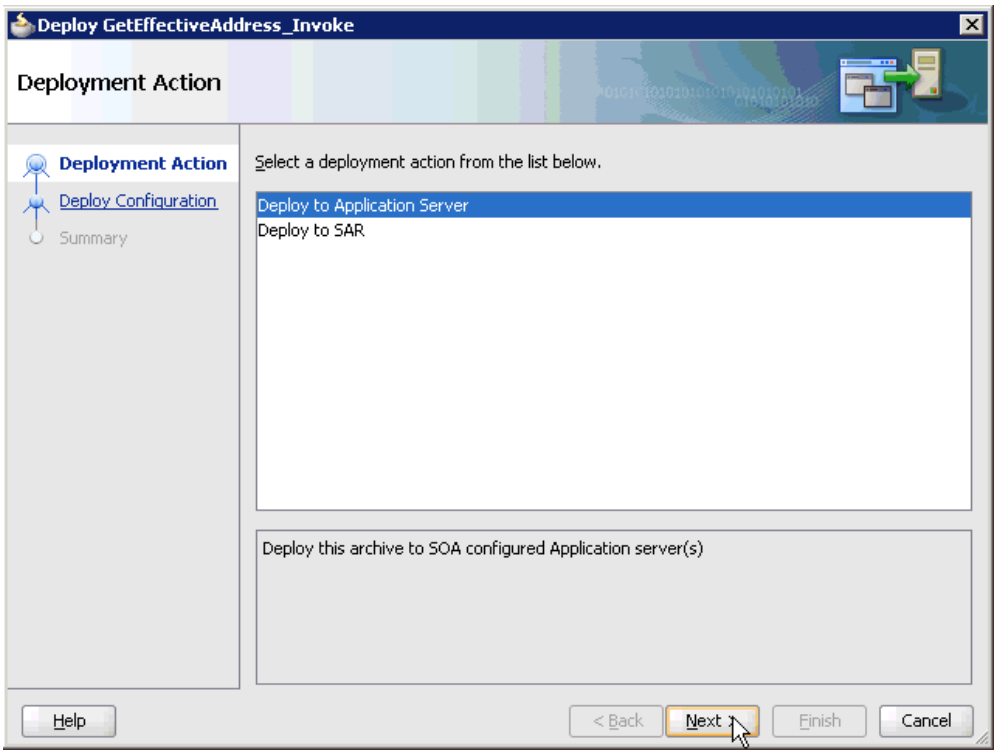
4.4.4 Deploying the BPEL Outbound Process

Perform the following steps to deploy the BPEL outbound process.

1. Right-click the project name in the left pane (for example, **GetEffectiveAddress_Invoke**), select **Deploy**, and then click **GetEffectiveAddress_Invoke**.

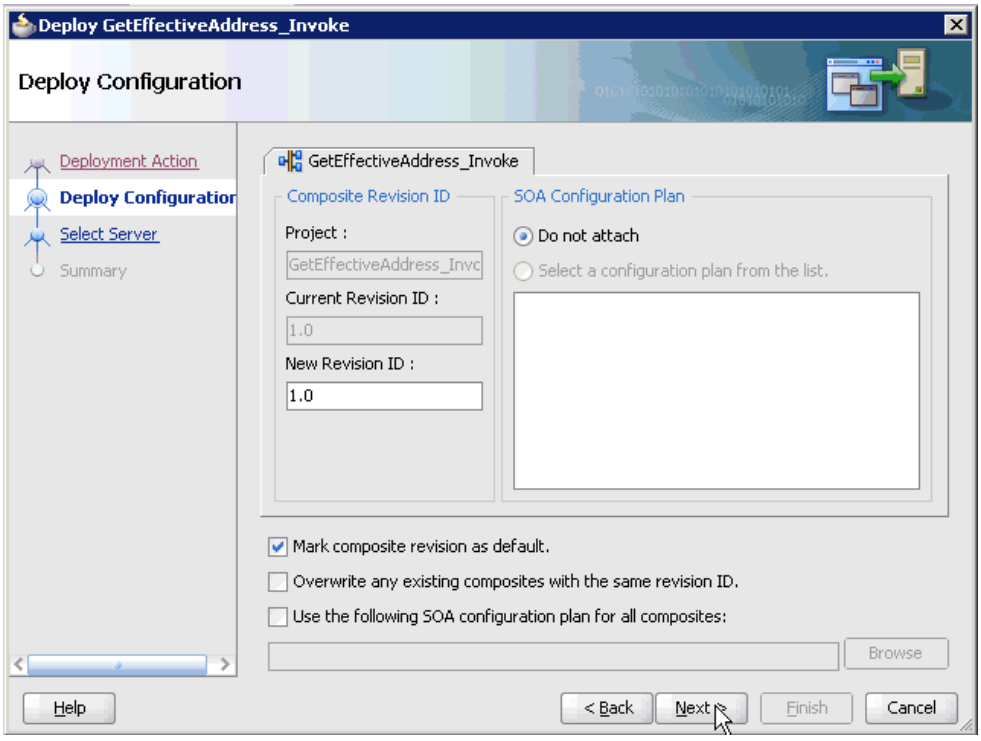


The Deployment Action page is displayed.



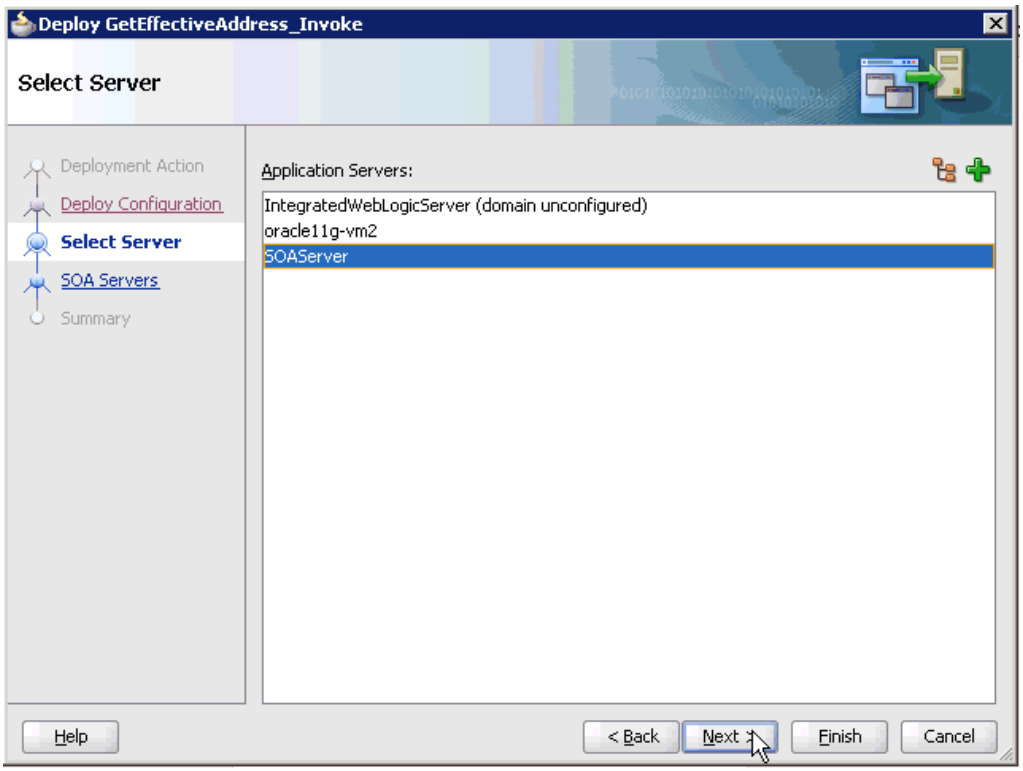
2. Ensure that **Deploy to Application Server** is selected.
3. Click **Next**.

   The Deploy Configuration page is displayed.



4. Leave the default values selected and click **Next**.

   The Select Server page is displayed.

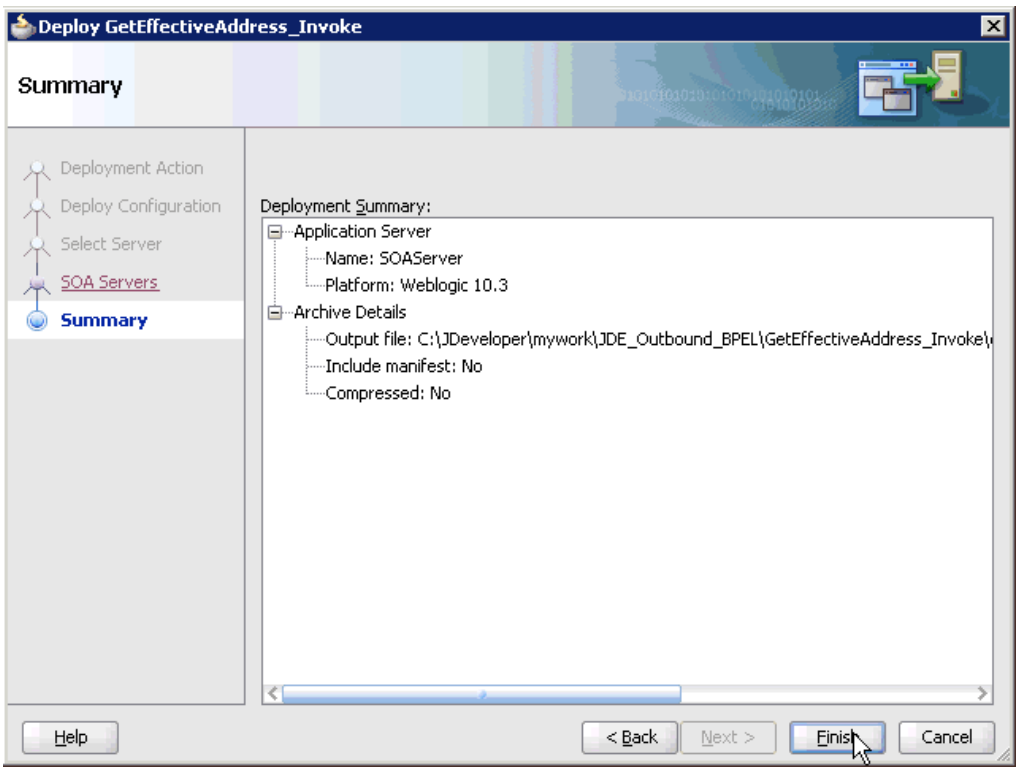


5. Select an available application server that was configured and click **Next**.

   The SOA Servers dialog is displayed.

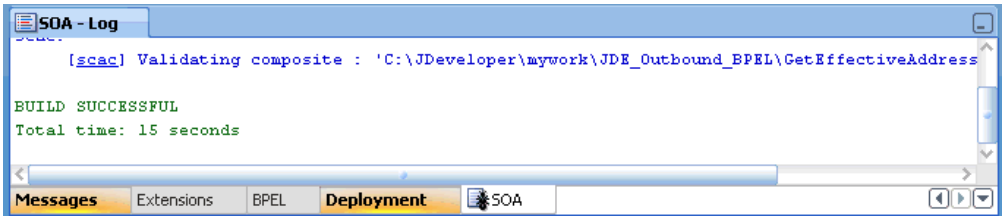
6. Select a target SOA server and click **Next**.

   The Summary page is displayed.



7. Review and verify all the available deployment information for your project and click **Finish**.

   The process is deployed successfully.



   If an Authorization Request dialog is displayed during the deployment process, provide the required user name and password and click **OK**.

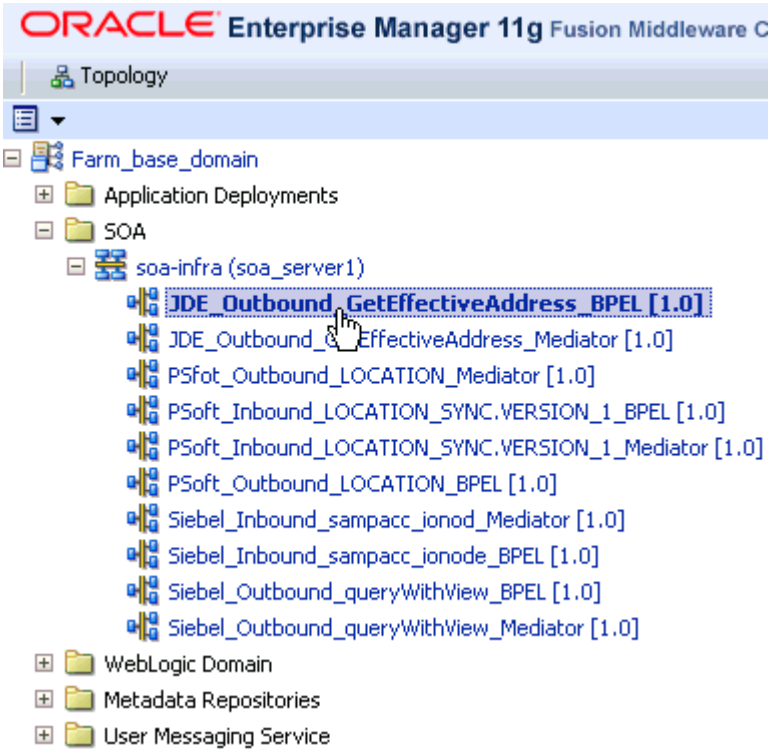
4.4.5 Invoking the Input XML Document in the Oracle Enterprise Manager Console

Perform the following steps to invoke the input XML document in the Oracle Enterprise Manager console.

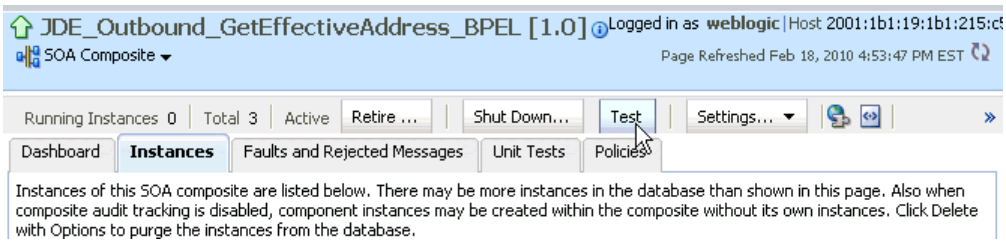
1. Log in to the Oracle Enterprise Manager console by using the following URL:

   ```
   http://localhost:7001/em
   ```

2. Expand your domain in the left pane followed by the **SOA** folder.



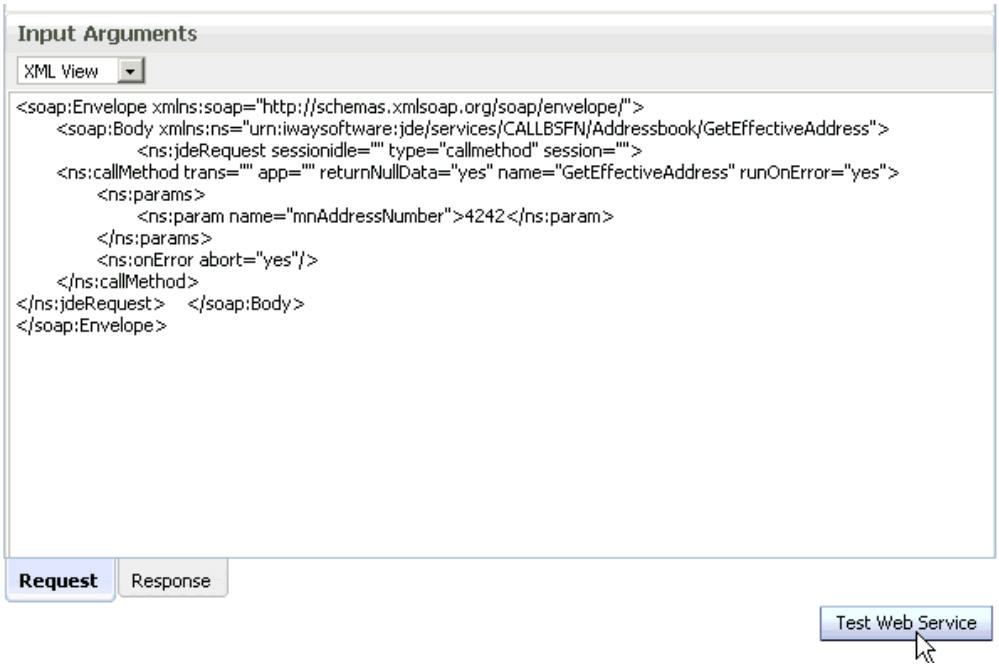
3. Select an available project (for example, JDE_Outbound_GetEffectiveAddress_BPEL).



4. Click **Test** in the right pane.

   The Test Web Service page is displayed.

5. Click the **Request** tab.
6. Scroll down to the Input Arguments section.



7. Select **XML View** from the list in the upper-left corner.
8. Provide an appropriate input XML document in the Input Arguments area.

   For example:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body
xmlns:ns="urn:iwaysoftware:jde/services/CALLBSFN/Addressbook/GetEffectiveAddres
s">
        <ns:jdeRequest sessionidle="" type="callmethod" session="">
<ns:callMethod trans="" app="" returnNullData="yes" name="GetEffectiveAddress"
runOnError="yes">
<ns:params>
<ns:param name="mnAddressNumber">4242</ns:param>
</ns:params>
<ns:onError abort="yes"/>
</ns:callMethod>
</ns:jdeRequest>    </soap:Body>
</soap:Envelope>
```

9. Click **Test Web Service**.

   The output response is received in the Response tab of the Oracle Enterprise Manager console.

4.4.6 Testing Outbound BPEL and Mediator Processes

When testing an outbound BPEL process or an outbound Mediator process from the Oracle Enterprise Manager 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 Mediator data flows can be tested using the Enterprise Manager console. When creating a Mediator data flow and interactions, the Web services are created and registered with the Oracle Application Server. For more information on creating a Mediator outbound process, see Chapter 5, "Integration With Mediator Service Components in the Oracle SOA Suite".

4.5 Designing an Inbound BPEL Process for Event Integration

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

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

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

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

Before you design a BPEL process, you must generate the respective WSDL file using Application Explorer. See "Generating WSDL for Event Integration" on page 4-34 for more information.

4.5.1 Generating WSDL for Event Integration

Before you design a BPEL process using Oracle 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.

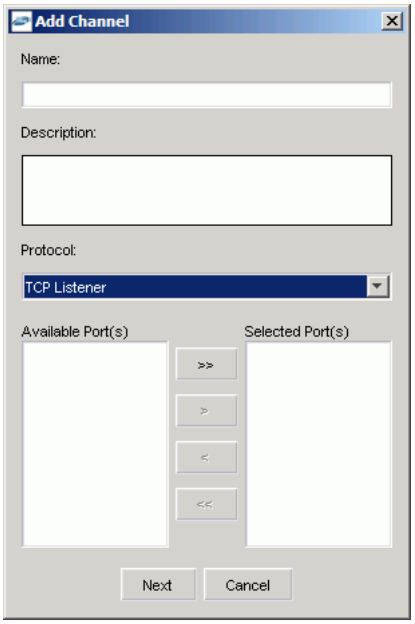
> **Note:** If two or more events share the same channel, event messages may not be delivered to the right BPEL process.

Creating a Channel in Application Explorer

To create a channel:

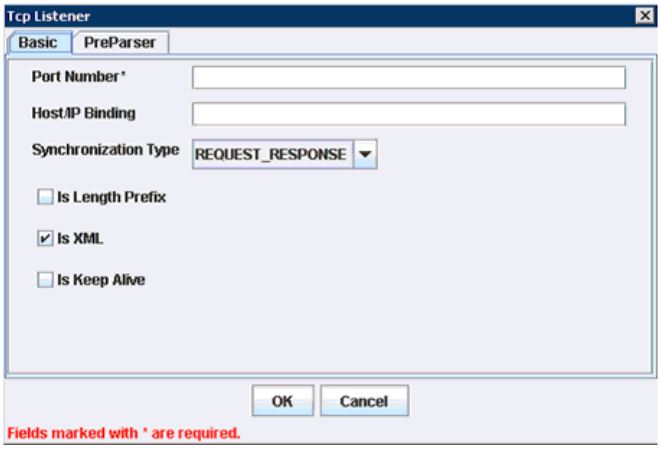
1. In Application Explorer, expand the **JDEdwards** node.
2. Right-click the **Channels** node, and select **Add Channels**.

   The Add Channel dialog is displayed.

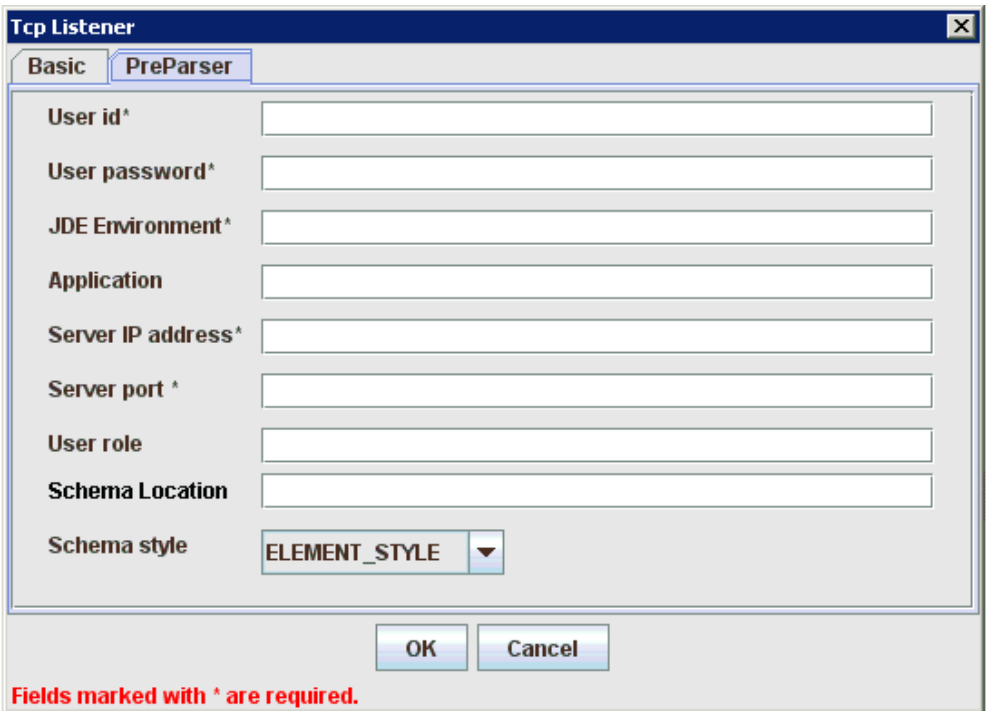


3. In the **Name** field, enter a descriptive name for the channel.
4. In the **Description** field, enter a description (optional).
5. From the **Protocol** list, choose a protocol for your channel.
6. Click **Next**.

   The dialog is displayed for the selected listener.



7. Enter the port number of the channel in the **Port Number** field.
8. Enter the location of the server in the **Host/IP Binding** 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.
User Role	Define a user role according to your requirements.
Schema Location	The location of the XML schema (.xsd file) that was generated from the event output. For example: `C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\config\config_name\JDEdwards\schemas\jde-schema.xsd` For more information, see "Generating WSDL for Inbound Interaction" on page 2-17.
Schema style	Choose from one of the following options: ■ ELEMENT_STYLE (default) ■ ATTRIBUTE_STYLE

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 run-time.

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.

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

Note: The schema validation options (Root, Namespace, Schema) are not applicable for the Oracle Application Adapter for J.D. Edwards OneWorld.

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.
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:

   ```
   C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\config\jca_
   sample\log\iwaf_jca1500.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. In the generated XML schema (.xsd file) perform the following modifications:
   a. Search for `Schemas-jdedwards-com` and replace it with `iwaysoftware`.
   b. Cut the following syntax:

      ```
      <xs:element name="jdeResponse">
      <xs:complexType>
      </xs:complexType>
      </xs:element>
      ```

   c. Paste it before the following line:

      ```
      <xs:element name="transaction">
      ```
7. Copy the XML schema (.xsd file) from the following directory:

   ```
   C:\oracle\Middleware\Oracle_
   SOA1\soa\thirdparty\ApplicationAdapters\config\config_name\JDEdwards\schemas
   ```

Note: Edit the created channel by providing the location of the schema (.xsd) file (as mentioned in step 7) in the PreParser tab of Application Explorer. For example:

```
C:\oracle\Middleware\Oracle_
SOA1\soa\thirdparty\ApplicationAdapters\config\config_
name\JDEdwards\schemas\jde-schema.xsd
```

8. Open a command prompt and navigate to the following directory:

```
C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\tools
```

9. Execute the **obadapter.bat** file to set the environment.
10. Navigate to the following directory where the XML schema (.xsd file) is copied:

```
C:\oracle\Middleware\Oracle_
SOA1\soa\thirdparty\ApplicationAdapters\config\config_name\JDEdwards\schemas
```

11. Enter the following command to generate a WSDL:

```
 java -Diway.oem=oracle11g
com.iwaysoftware.af.container.tools.wsdl.IWayWSILBrowser adapterhome adapter
target channel schemaPrefix wsdlFileName
```

where:

adapterhome is the path to your ApplicationAdapters home. For example:

```
C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\
```

adapter is the name of the adapter. For example, JDEdwards.

target is the name of the adapter target you created in Application Explorer.

channel is the name of the channel you created in Application Explorer.

schemaPrefix is the prefix for the XSD schema. The schema file must be in the same directory where the Java command is executed, for example:

```
java -Diway.oem=oracle11g
com.iwaysoftware.af.container.tools.wsdl.IWayWSILBrowser
C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\JDEdwards
jde812_tgt jde_ch Jde812_Schema Jde812_salesorder_receive.wsdl
```

Once the command is executed, the following is displayed in the command window:

```
Running Inbound WSDL generation tool...
-> user.dir = java com.iwaysoftware.af.container.tools.wsdl.IWayWSILBrowser
C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\JDEdwards
jde812_tgt jde_ch Jde812_Schema Jde812_salesorder_receive.wsdl

-> Generating WSDL...
-> Done.
-> Writing WSDL 'C:\oracle\Middleware\Oracle_
SOA1\soa\thirdparty\ApplicationAdapters\wsdls\Jde812_salesorder_receive.wsdl '
to disk...
-> Done.
```

Note: It is good practice to append **_receive** to the names of WSDL files that are generated for event notification services. This allows you to easily distinguish between them and those generated for request-response services.

12. Stop the channel in Application Explorer.

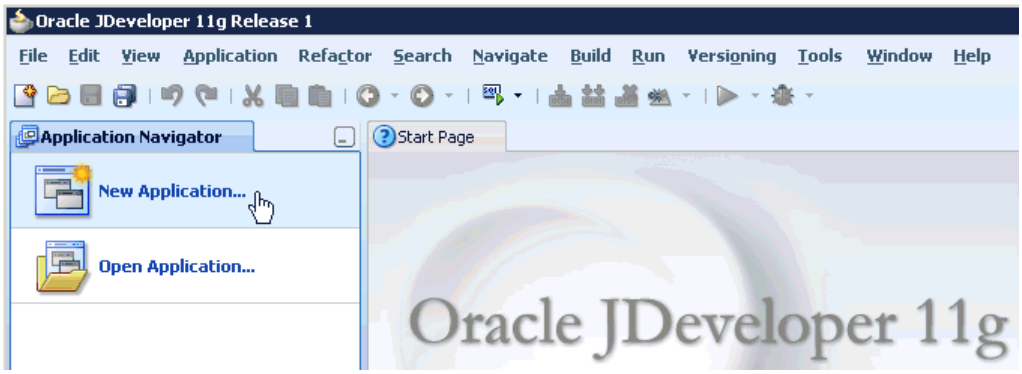
Note: You can organize your WSDL files in subfolders, creating your own WSDL hierarchy structure. Create the folders under `C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\wsdls`. The WSIL browser in Oracle JDeveloper displays the full tree structure of your WSDL hierarchy.

You can now create a new SOA application, which is the first step that is required to define a BPEL inbound process in Oracle JDeveloper.

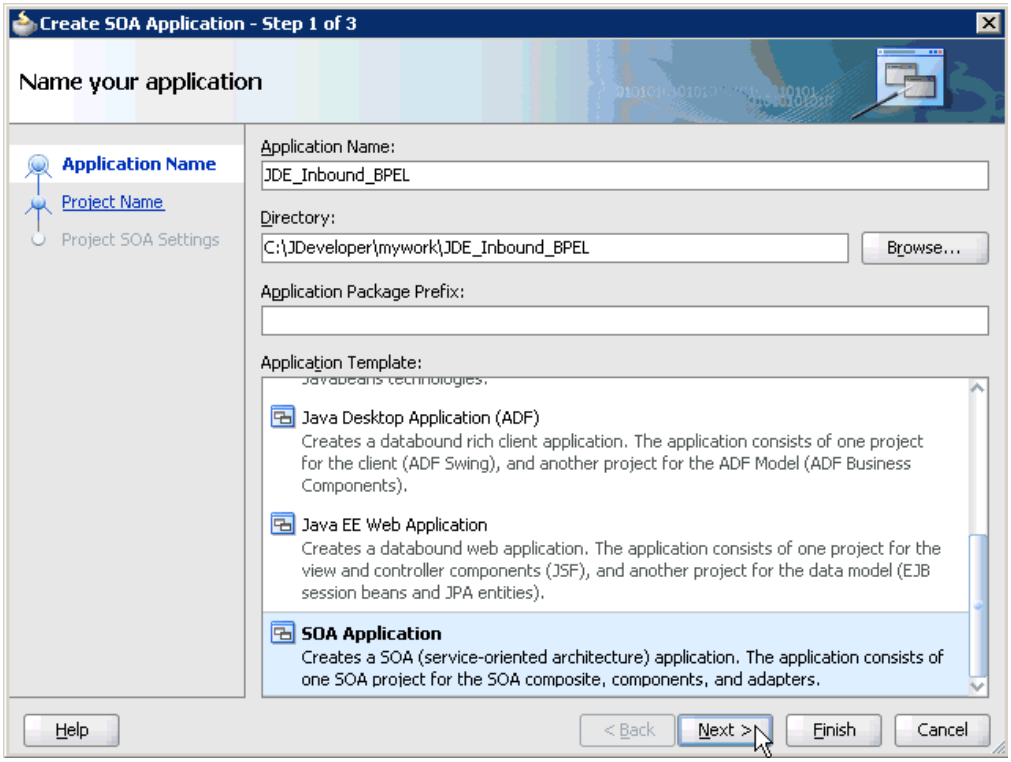
4.5.2 Creating a New SOA Application for the Inbound BPEL Process

Perform the following steps to create a new SOA application for the inbound BPEL process:

1. Open Oracle JDeveloper on your system.
2. In the Application Navigator tab, click **New Application**.

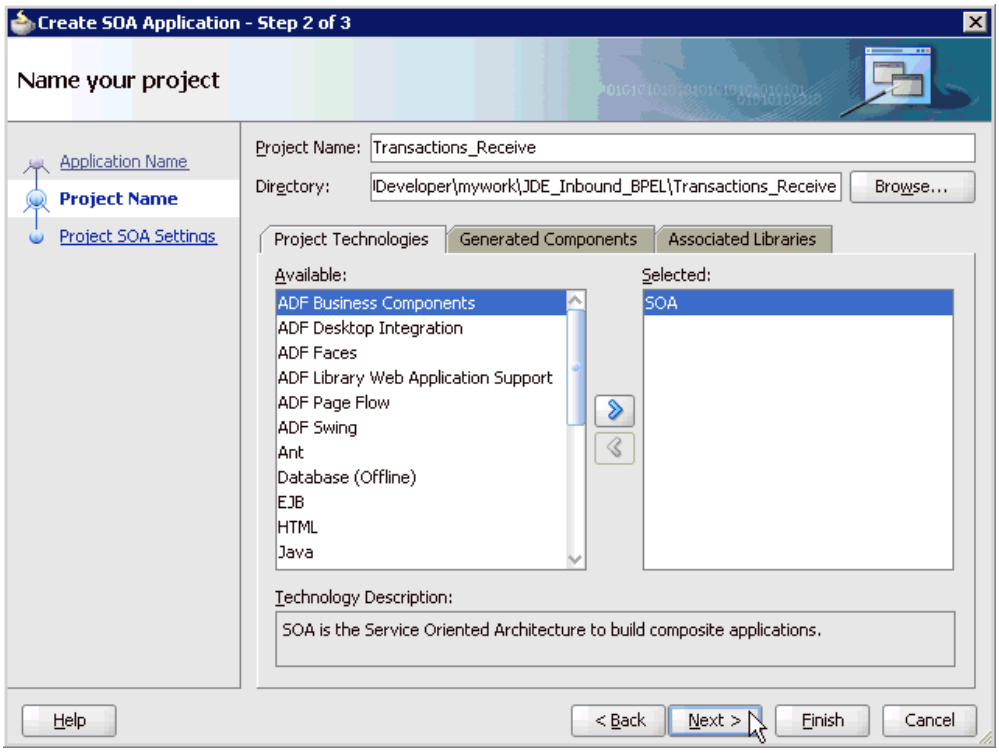


The Create SOA Application wizard is displayed.



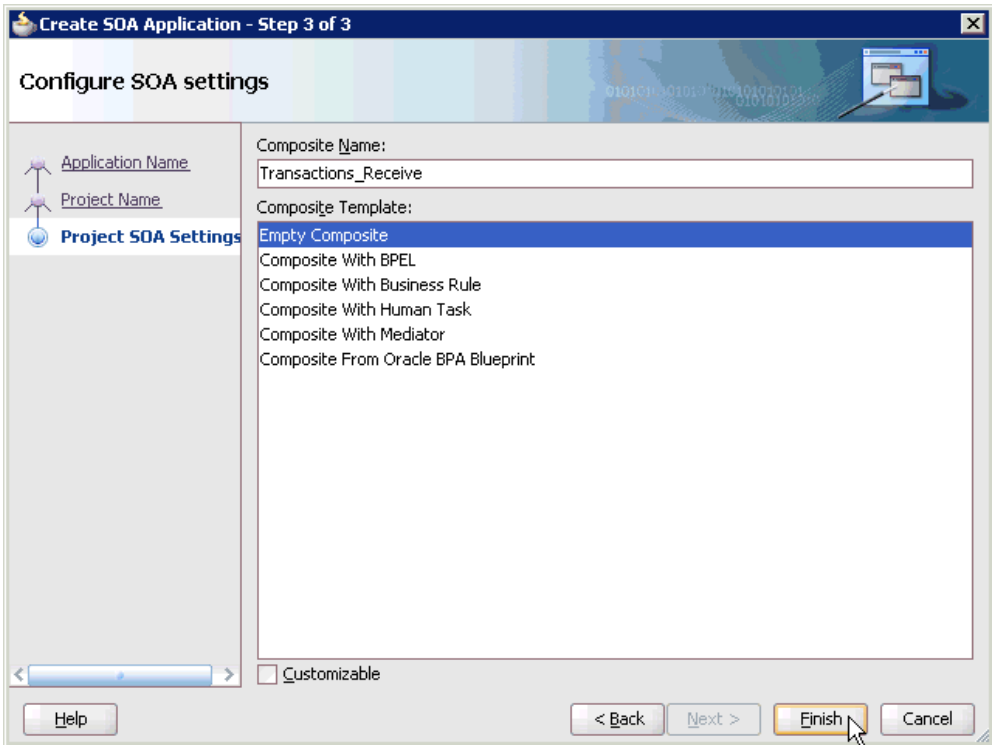
3. From the Application Template list, click **SOA Application**.
4. Enter name for the new SOA application (for example, JDE_Inbound_BPEL) and click **Next**.

   The Name your project page is displayed.

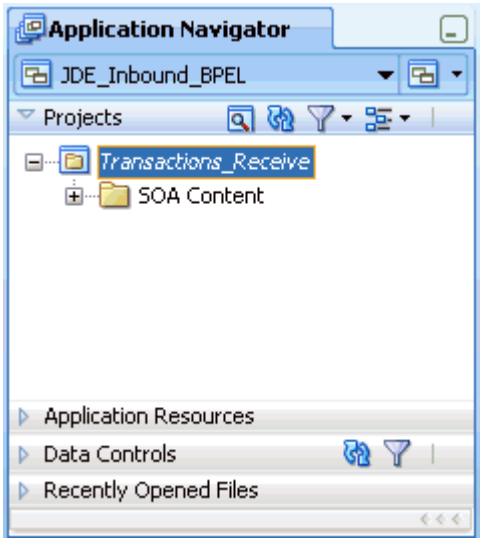


5. Enter a project name (for example, Transactions_Receive) and click **Next**.

   The Configure SOA settings page is displayed.



6. From the Composite Template list, select **Empty Composite** and click **Finish**.



   The new SOA application (JDE_Inbound_BPEL) and associated project (Transactions_Receive) are added to the Application Navigator tab in the left pane.

4.5.3 Defining a BPEL Inbound Process

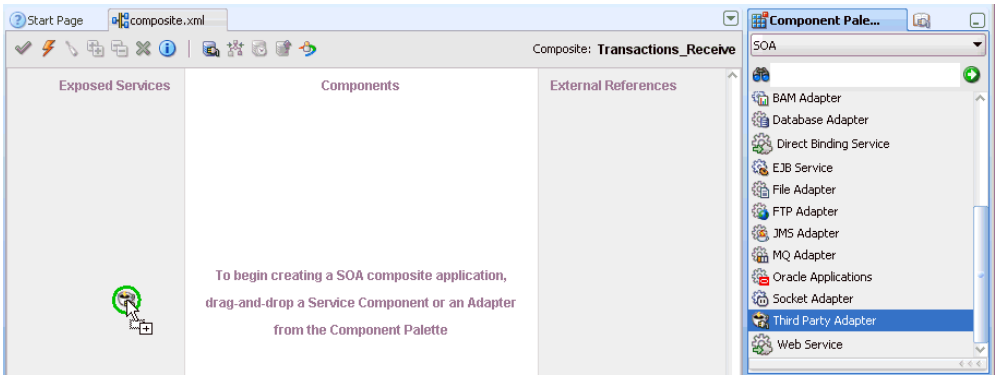
This section describes how to define a BPEL inbound process, which consists of the following stages:

1. Configuring a Third Party Adapter Service Component
2. Configuring an Inbound BPEL Process Component

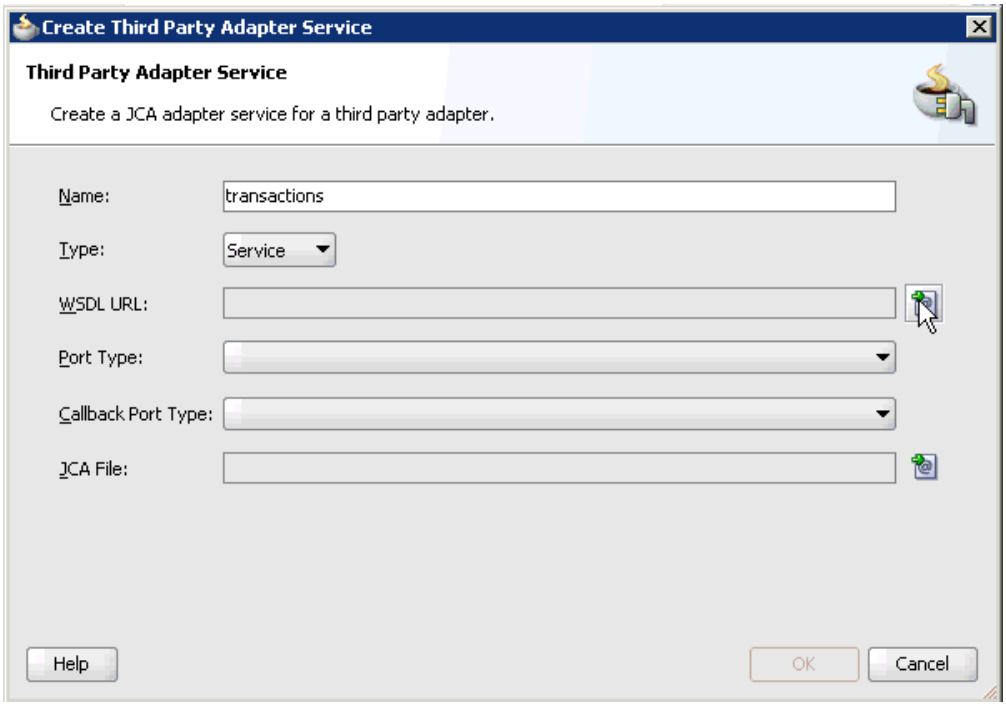
Creating a Third Party Adapter Service Component

Perform the following steps to create a third party adapter service component:

1. Drag and drop the **Third Party Adapter** component from the Component Palette tab (Service Adapters section) to the Exposed Services pane.

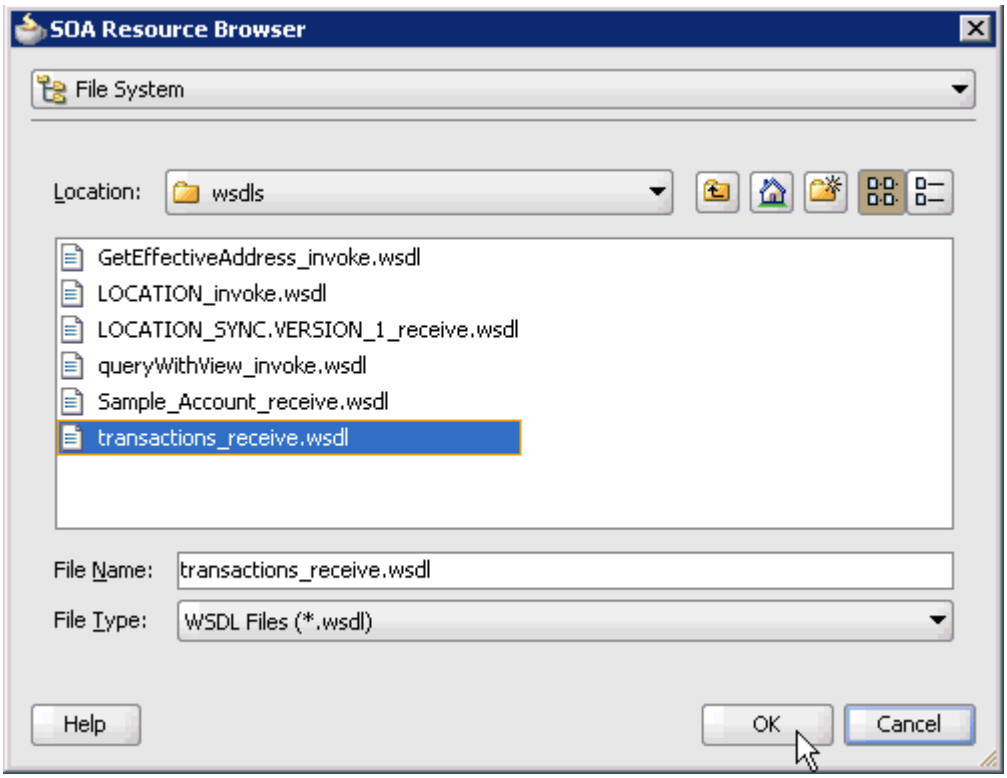


The Create Third Party Adapter Service dialog is displayed.



2. Enter a name for the third party adapter service.
3. Ensure that **Service** is selected from the Type list (default).
4. Click the **Find existing WSDLs** icon, which is located to the right of the WSDL URL field.

   The SOA Resource Browser dialog is displayed.

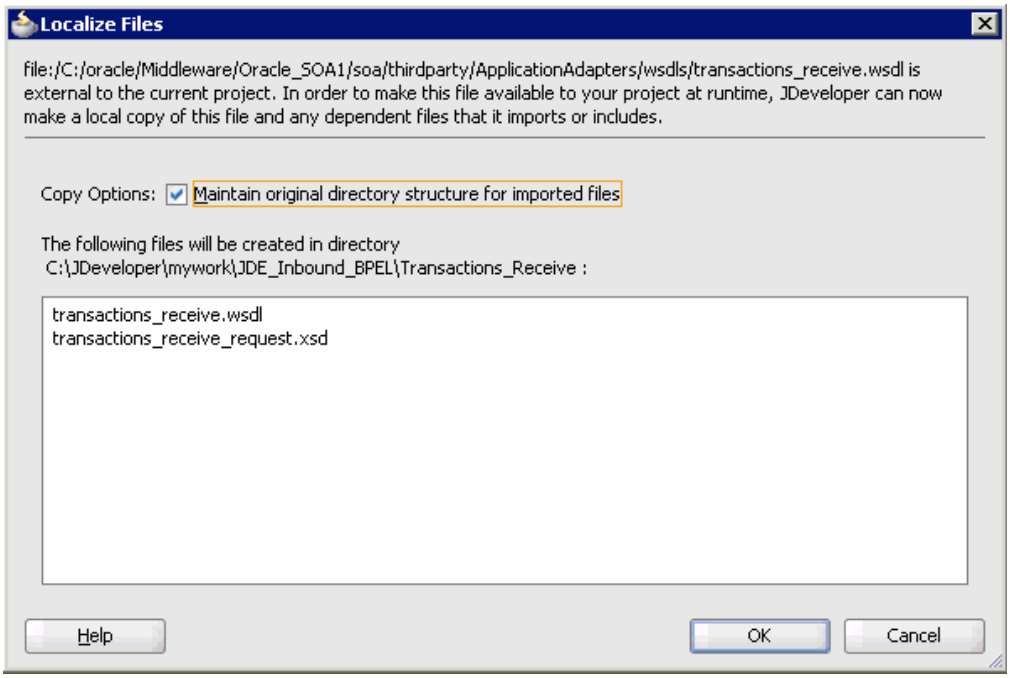


5. Browse and select an inbound WSDL file from the following directory:

   ```
   C:\oracle\Middleware\home_GA\Oracle_
   SOA1\soa\thirdparty\ApplicationAdapters\wsdls
   ```

6. Click **OK**.

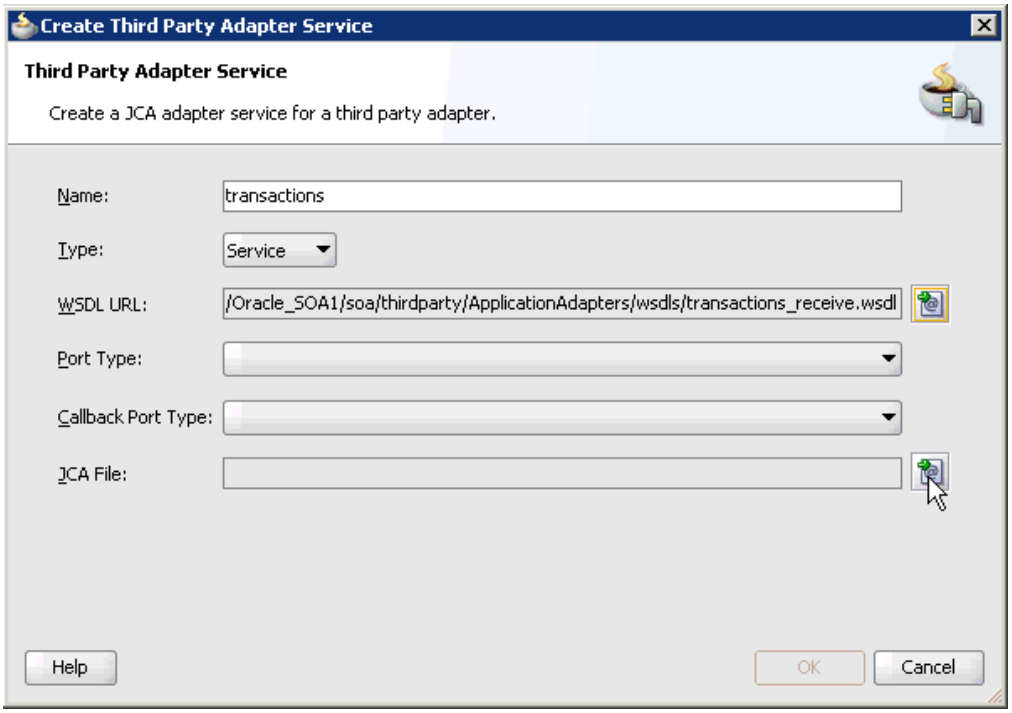
   The Localize Files dialog is displayed.



7. Click **OK**.

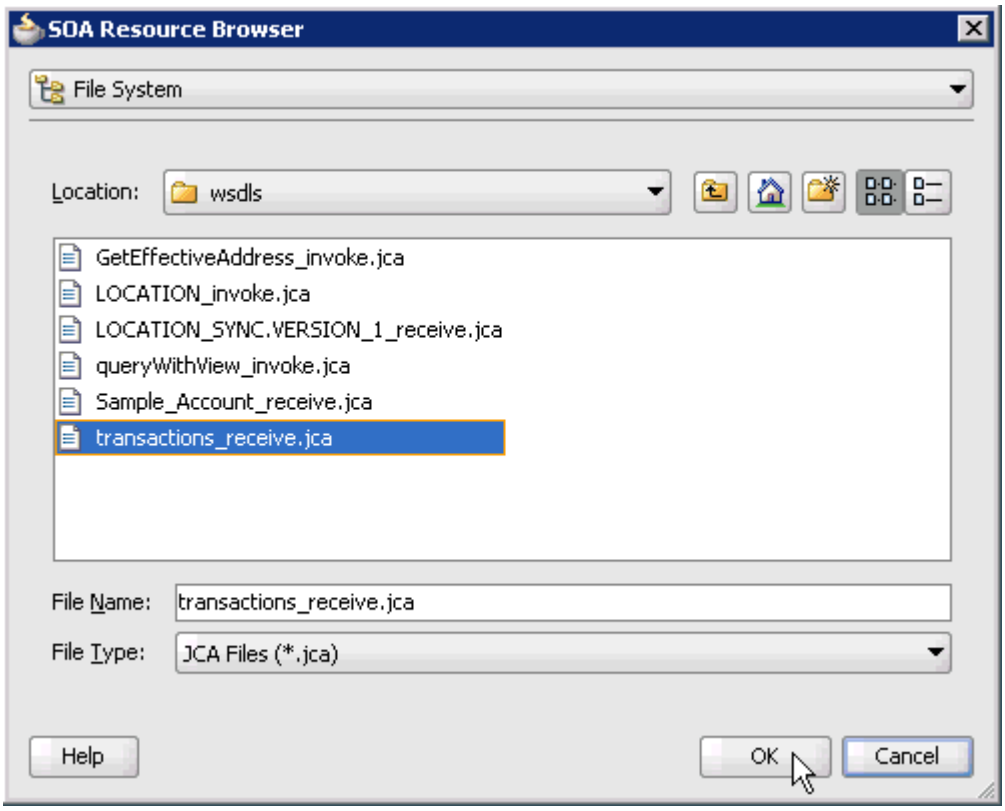
   The inbound WSDL file and associated receive/request XML schema file (.xsd) are imported to the project folder that has been created.

   You are returned to the Create Third Party Adapter Service dialog.



8. Click the **Find JCA Files** icon, which is located to the right of the JCA File field.

   The SOA Resource Browser dialog is displayed.

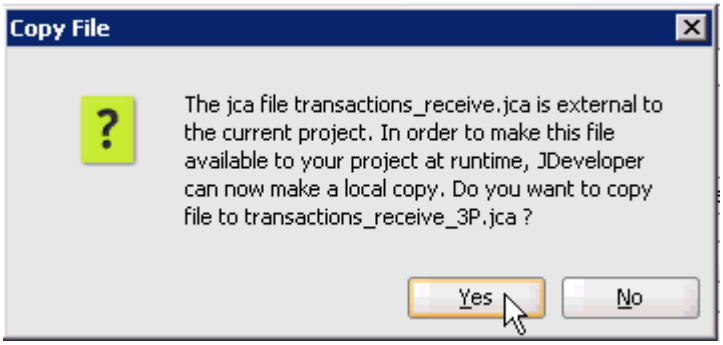


9. Browse and select the JCA properties file from the following directory:

   ```
   C:\oracle\Middleware\home_GA\Oracle_
   SOA1\soa\thirdparty\ApplicationAdapters\wsdls
   ```

10. Click **OK**.

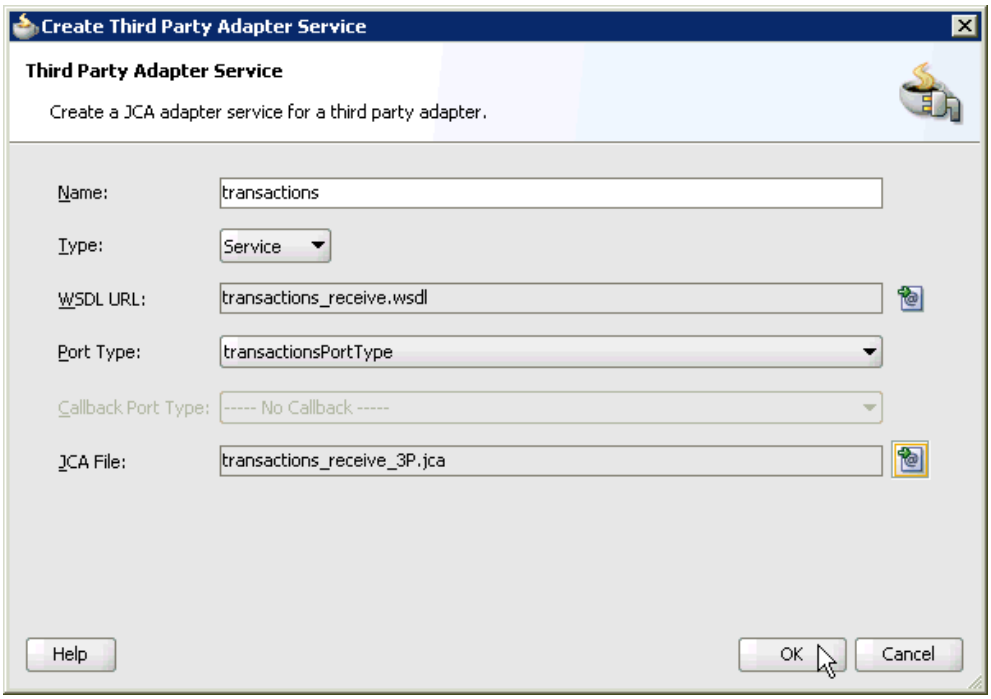
    The following message is displayed.



11. Click **Yes**.

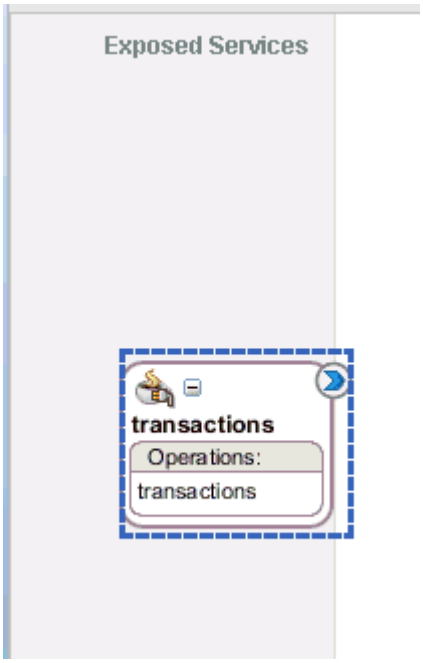
    A copy of the JCA properties file is made in the project folder.

    You are returned to the Create Third Party Adapter Service dialog.



12. Click **OK**.

    The third party adapter service component (transactions) is created in the Exposed Services pane, as shown in the following image.

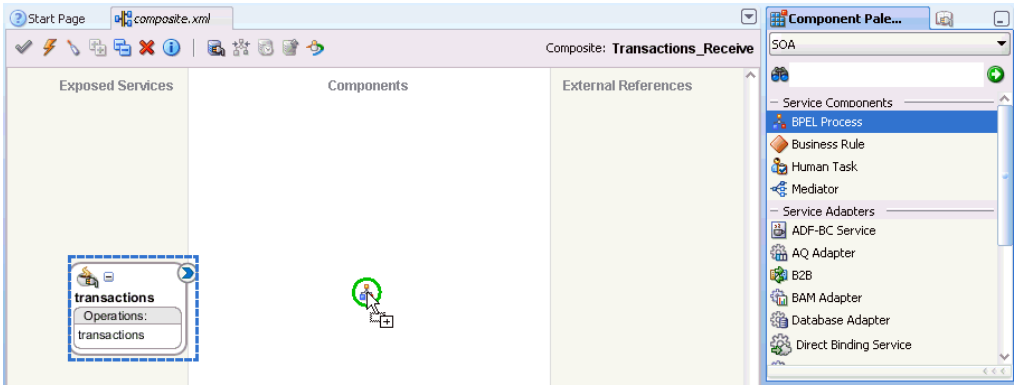


    You are now ready to configure an inbound BPEL process component.

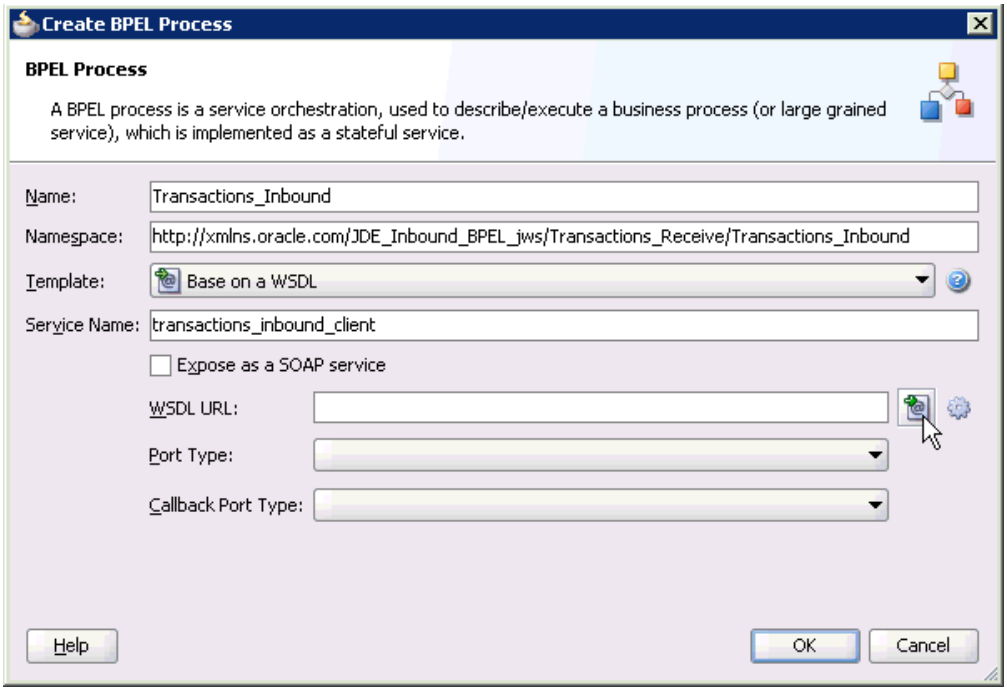
Creating an Inbound BPEL Process Component

Perform the following steps to create an inbound BPEL process component:

1. Drag and drop the **BPEL Process** component from the Component Palette tab (Service Components section) to the Components pane.

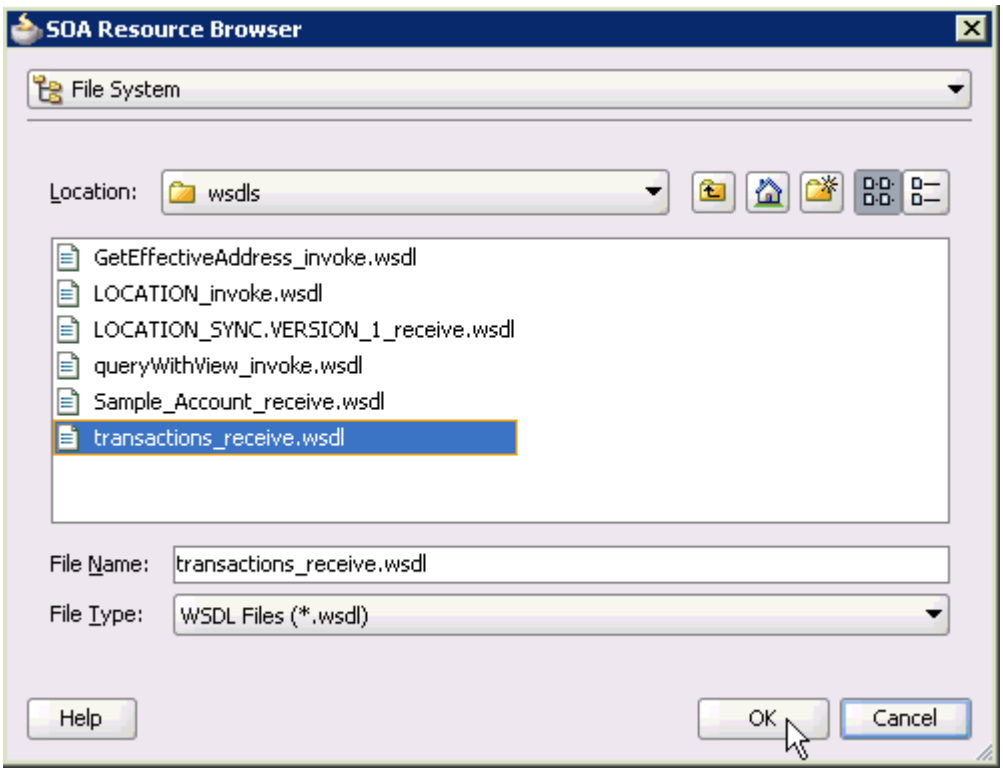


The Create BPEL Process dialog is displayed.



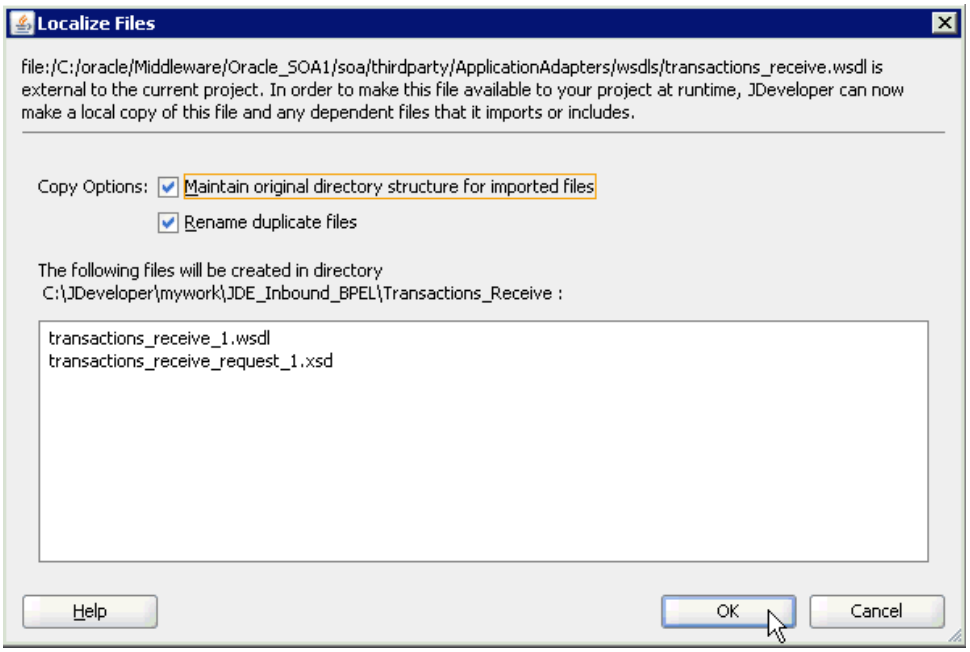
2. In the Name field, enter a name to identify the new inbound BPEL process component (for example, Transactions_Inbound).
3. From the Template list, select **Base on a WSDL**.
4. Uncheck the **Expose as a SOAP Service** check box.
5. Click the **Find existing WSDLs** icon, which is located to the right of the WSDL URL field.

   The SOA Resource Browser dialog is displayed.



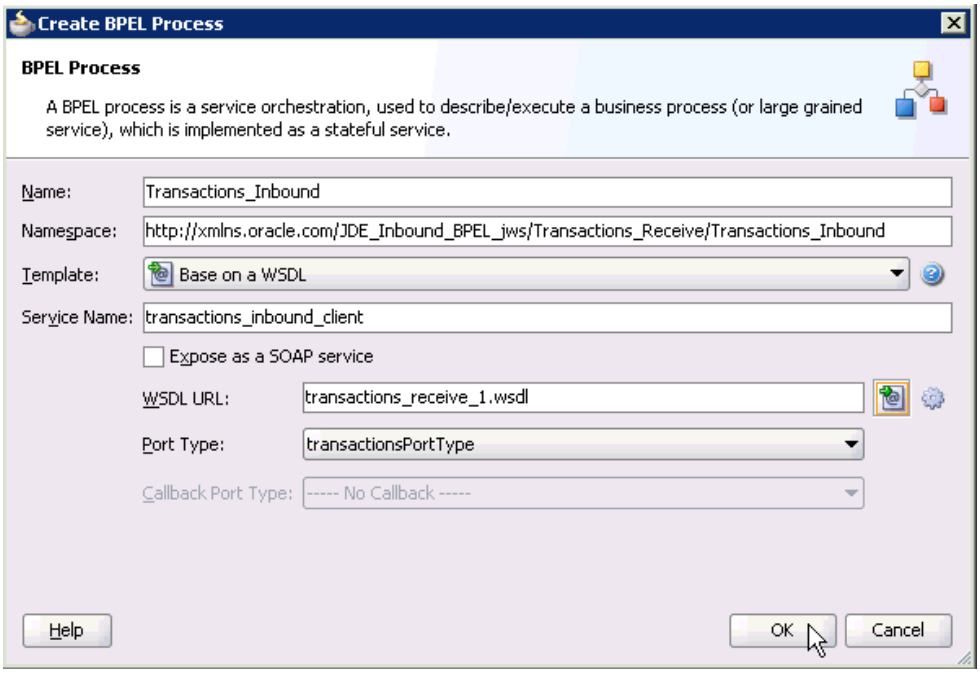
6. Browse and select an inbound WSDL file (for example, transactions_receive.wsdl) from the project folder.
7. Click **OK**.

   The Localize Files dialog is displayed.

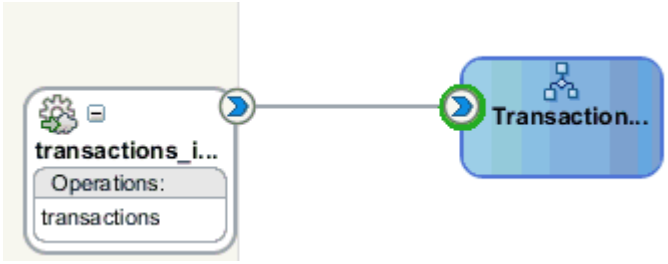


8. Click **OK**.

   You are returned to the Create BPEL Process dialog.

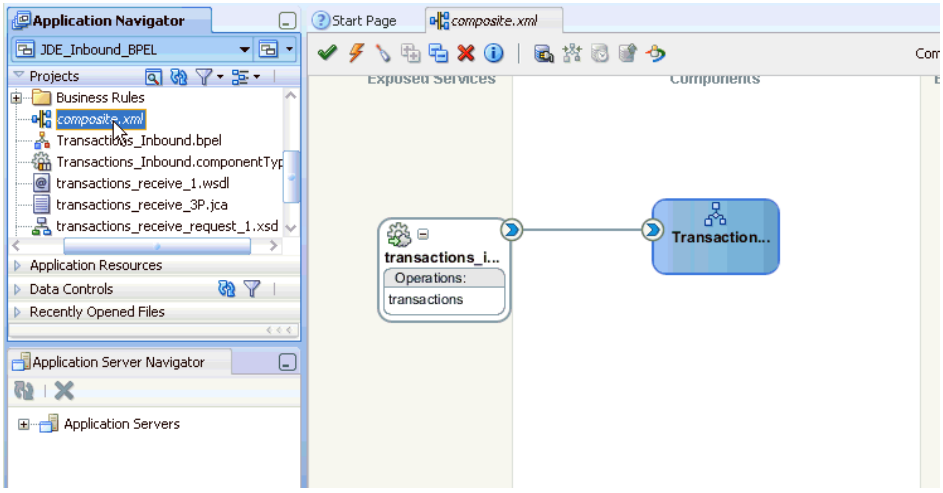


9. Click **OK**.

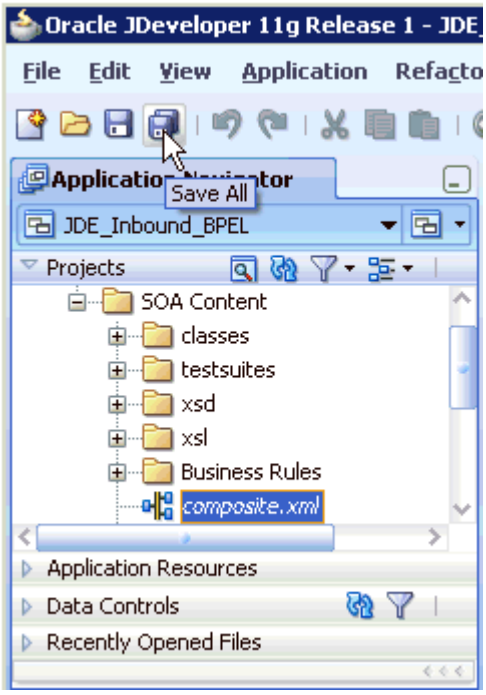


10. Create a connection between the third party adapter service component (transactions) and the inbound BPEL process component (Transactions_Inbound).

11. Double-click **composite.xml** in the left pane.



12. Click the **Save All** icon in the menu bar to save the new inbound BPEL process component that was configured.

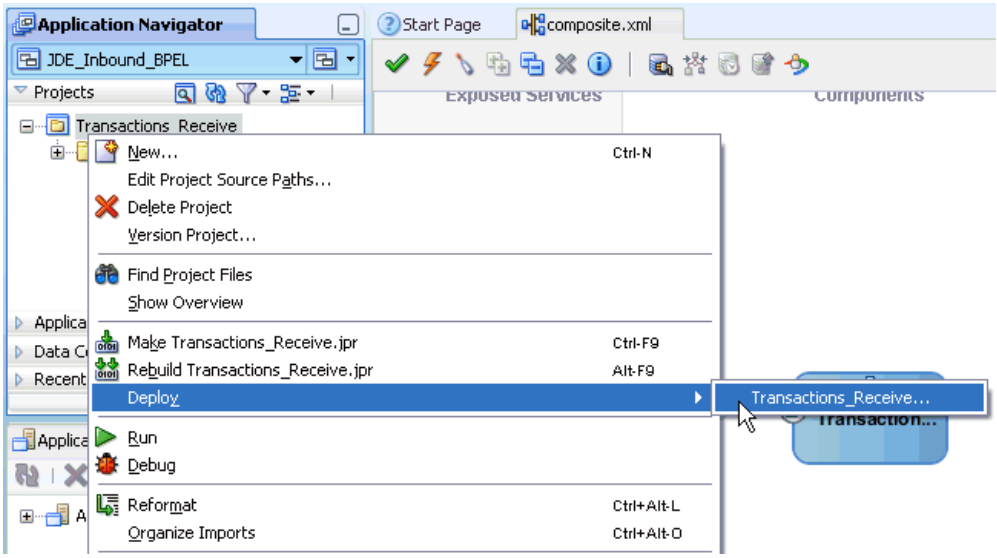


You are now ready to deploy the BPEL inbound process.

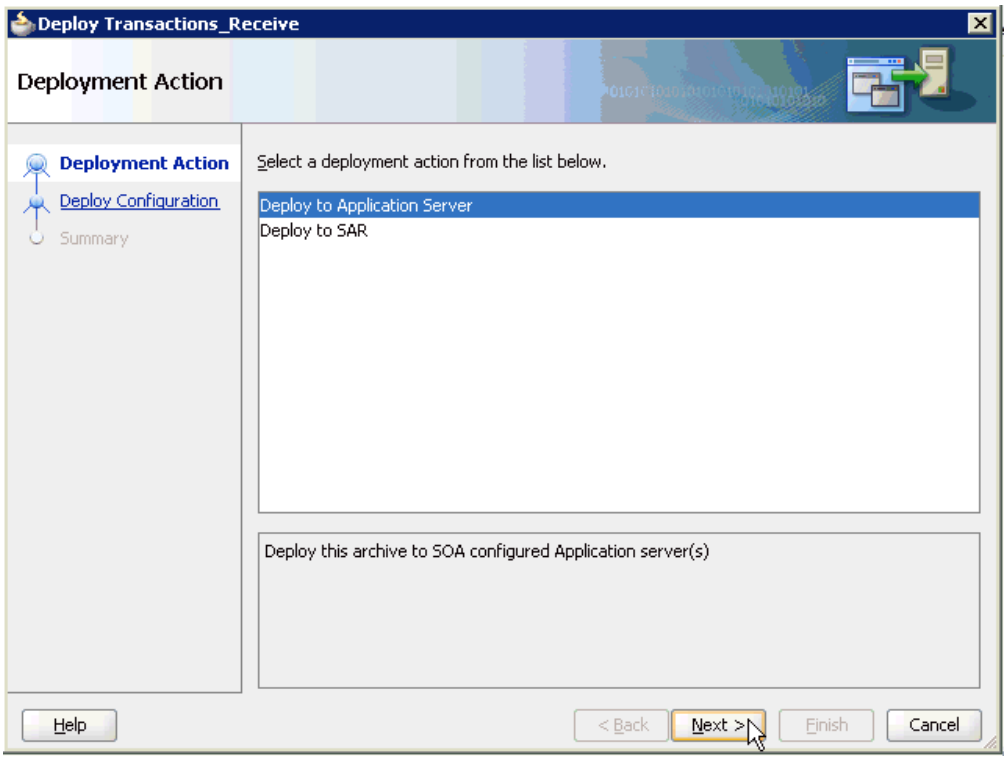
4.5.4 Deploying the BPEL Inbound Process

Perform the following steps to deploy the BPEL inbound process.

1. Right-click the project name in the left pane (for example, **Transactions_Receive**), select **Deploy**, and then click **Transactions_Receive**.

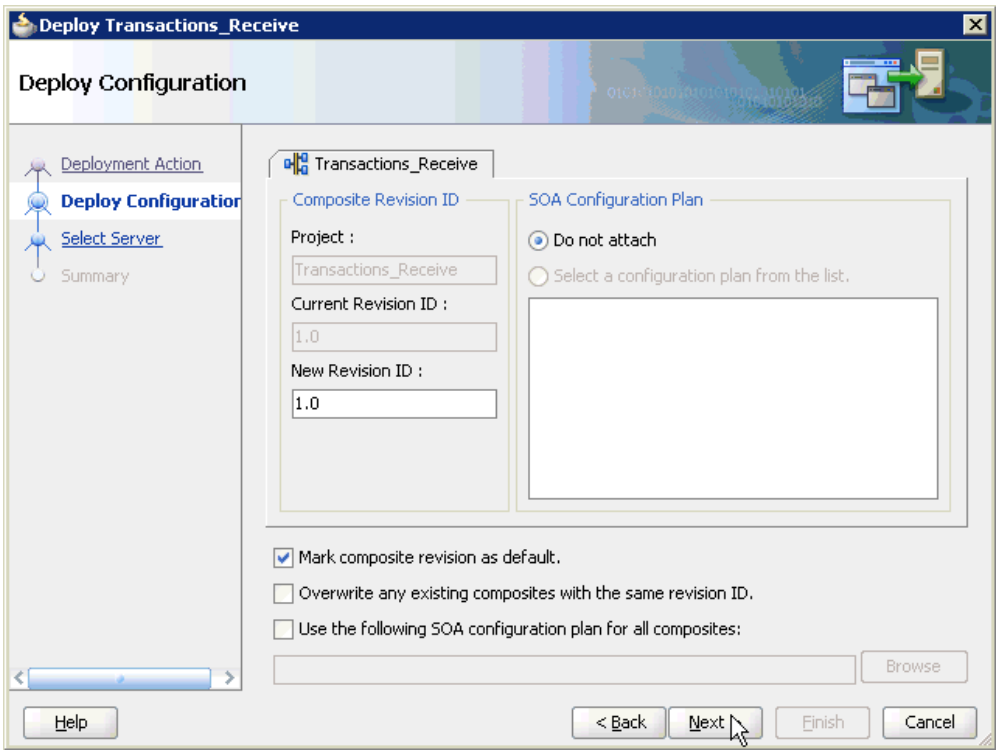


The Deployment Action page is displayed.



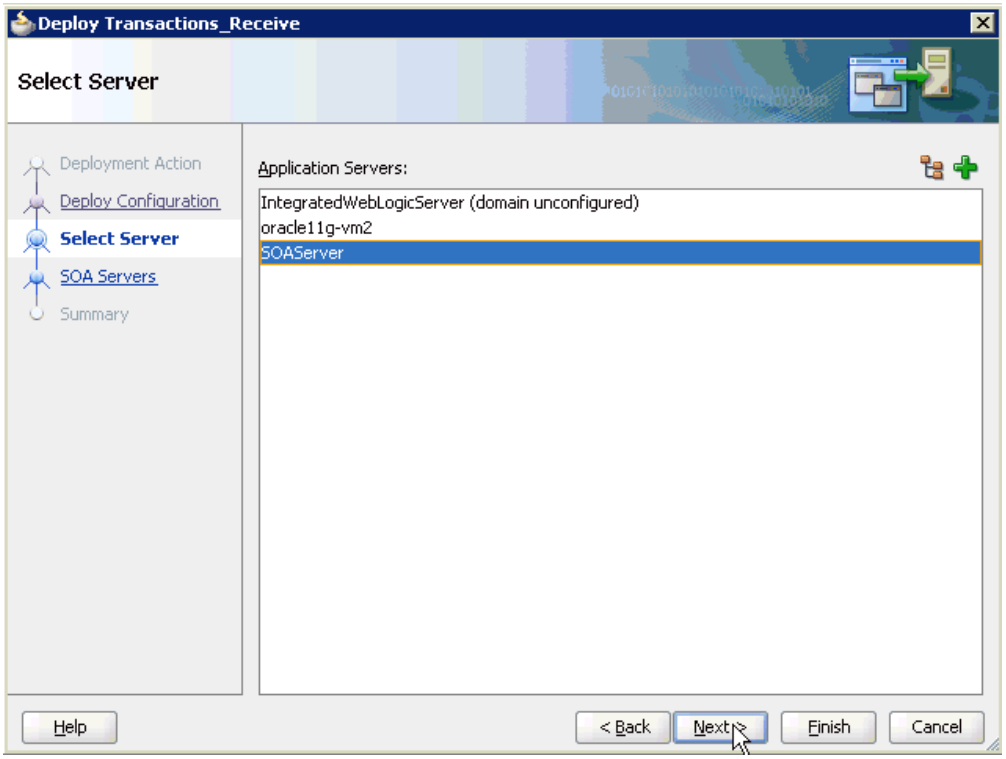
2. Ensure that **Deploy to Application Server** is selected.
3. Click **Next**.

   The Deploy Configuration page is displayed.



4. Leave the default values selected and click **Next**.

The Select Server page is displayed.

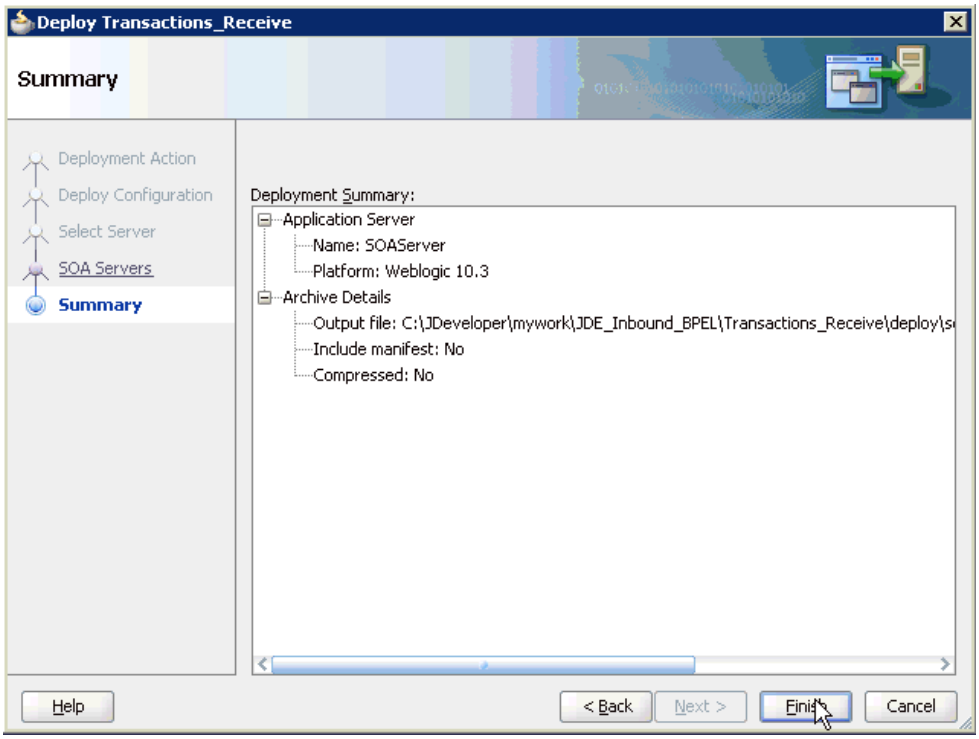


5. Select an available application server that was configured and click **Next**.

   The SOA Servers dialog is displayed.

6. Select a target SOA server and click **Next**.

   The Summary page is displayed.



7. Review and verify all the available deployment information for your project and click **Finish**.

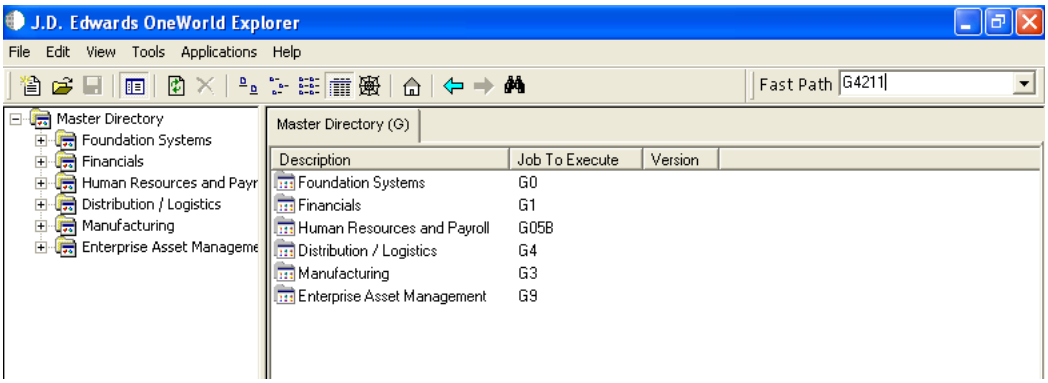
   If an Authorization Request dialog is displayed during the deployment process, provide the required user name and password and click **OK**.

   Once event messages are triggered through J.D. Edwards OneWorld, successful instances are received in the Oracle Enterprise Manager console.

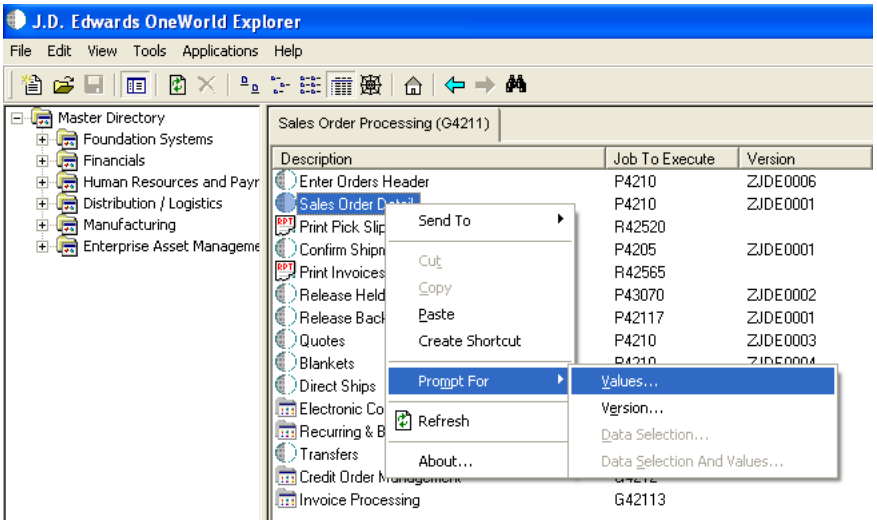
4.5.5 Triggering an Event in J.D. Edwards OneWorld

Events are generated by activity in a database or in an application system. You can use events to trigger an action in your application. 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**.

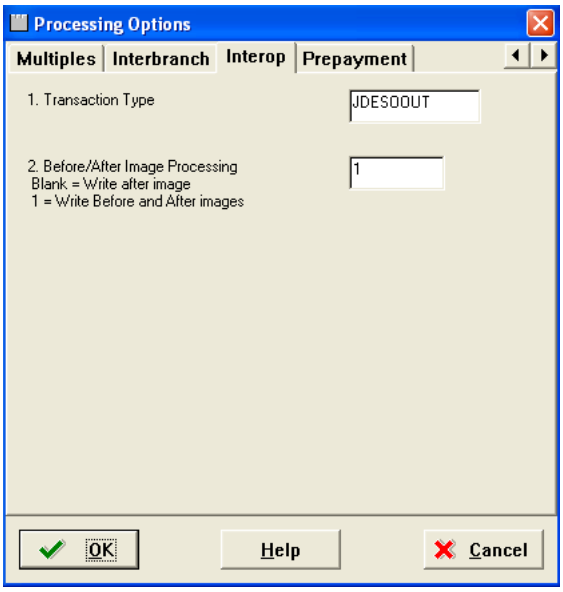


3. Right-click **Sales Order Detail** (P4210).



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

   The Processing Options dialog is displayed.



Perform the following steps:

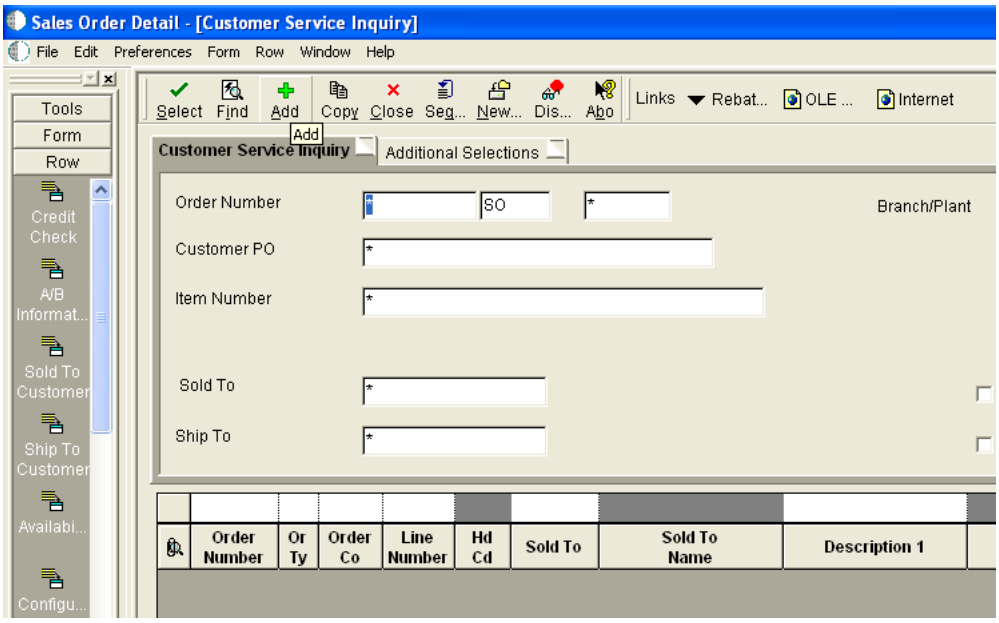
a. Click the **Interop** tab.

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**.

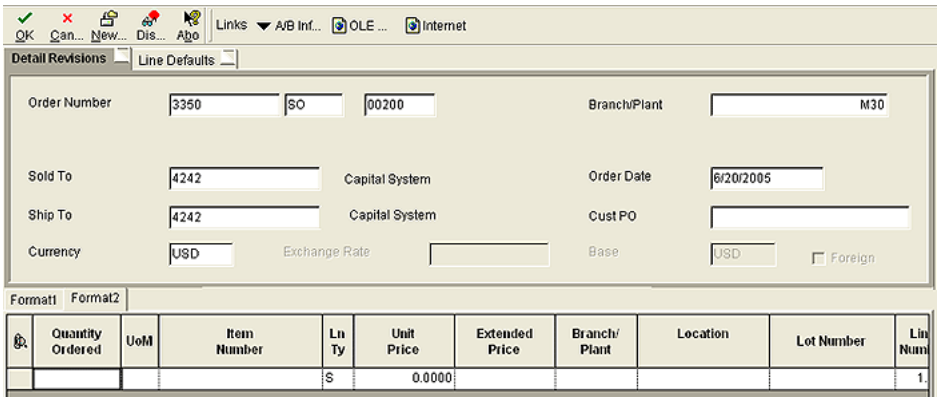
   The **Sales Order Detail - (Customer Service Inquiry)** window is displayed.



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

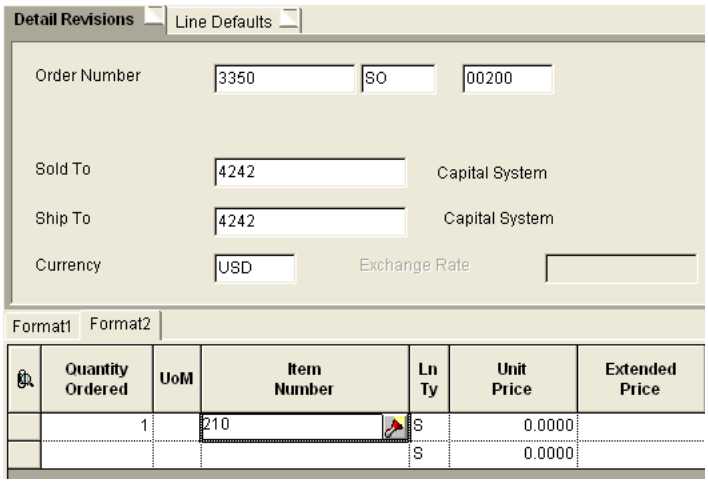
7. Enter the values as shown in the following screen.

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

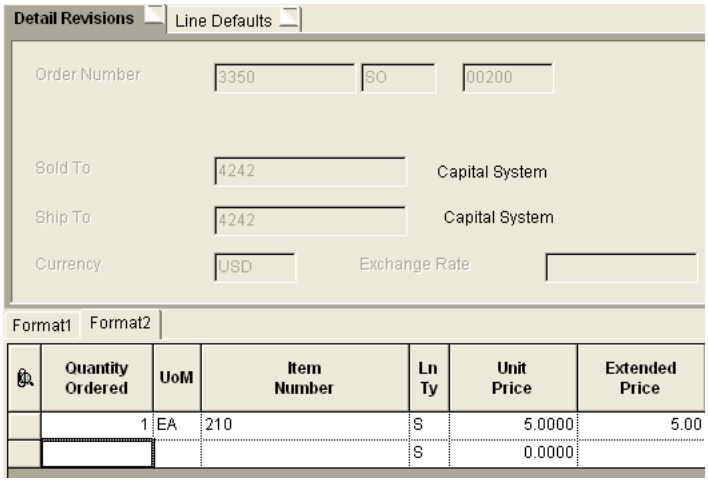


8. Enter a value for **Quantity Ordered** and **Item Number**.

For example:



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



10. Click **OK**.

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

Verifying the Results

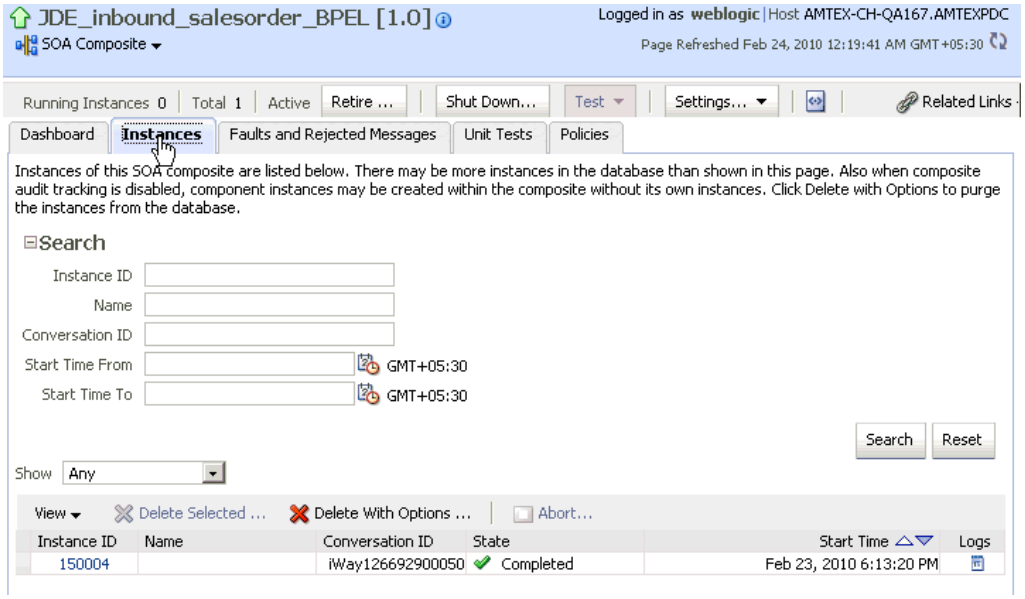
To verify your results:

1. Log in to the Oracle Enterprise Manager console by using the following URL:

   `http://localhost:7001/em`

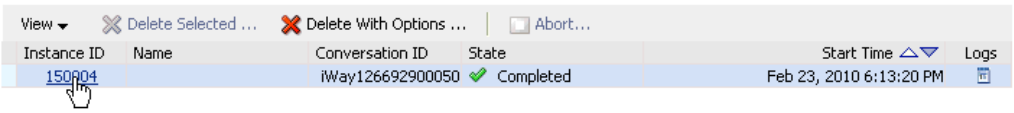
2. Expand your domain in the left pane followed by the **SOA** folder.

3. Select an available project (for example, JDE_Inbound_salesorder_BPEL).



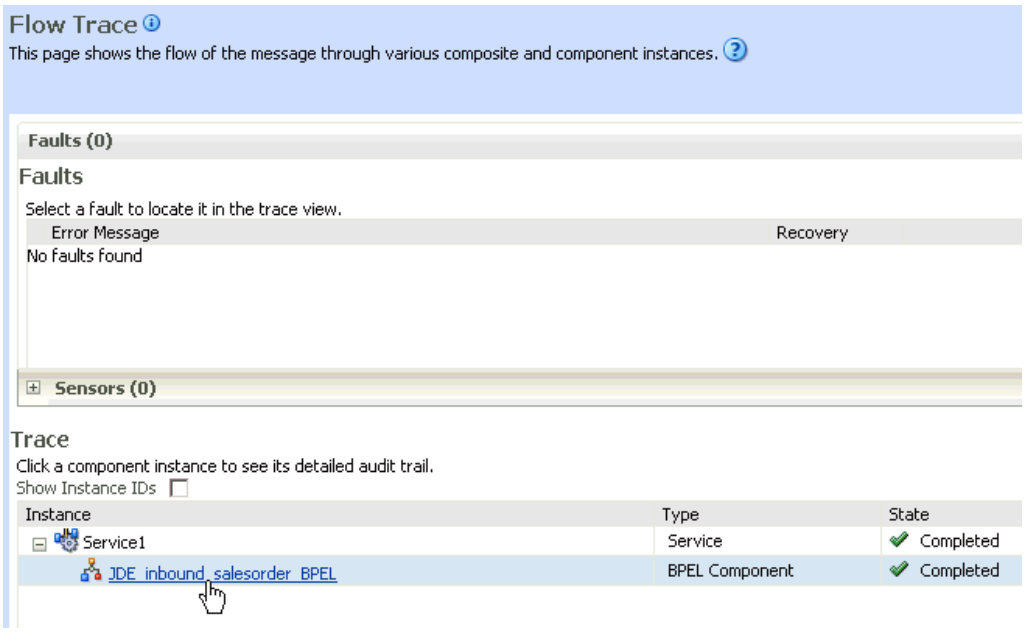
4. Click the **Instances** tab in the right pane.

   Recently received run-time events are displayed in the Instances tab.



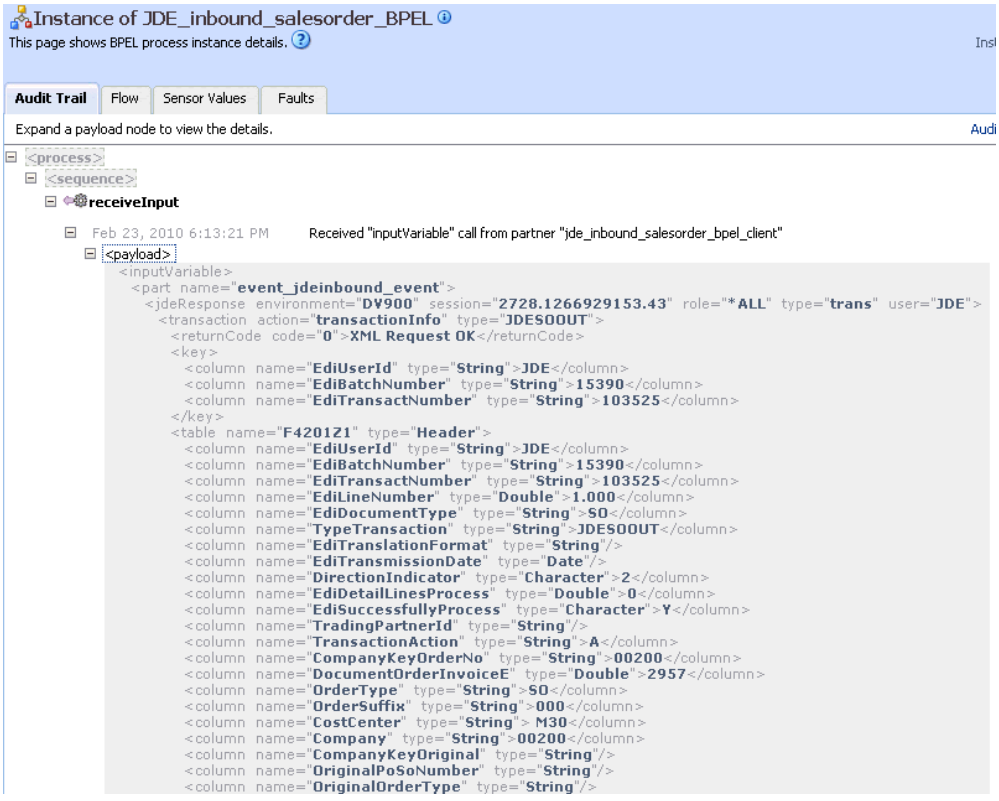
5. Select a J.D. Edwards OneWorld instance ID.

   The Flow Trace page is displayed.



6. Select a component instance to view its detailed audit trail.

   The Instance page for the selected component is displayed.



7. Click the **Audit Trail** tab to view the event message.

   The message received from the J.D. Edwards OneWorld system is displayed in the Audit Trail tab.

5

Integration With Mediator Service Components in the Oracle SOA Suite

This chapter contains the following examples:

- Configuring a New Application Server Connection
- Configuring a Mediator Outbound Process
- Configuring a Mediator Inbound Process

The scenarios shown in this chapter require the following prerequisites.

Prerequisites

The following are installation and configuration requirements:

- Oracle Application Adapter for J.D. Edwards OneWorld must be installed on Oracle WebLogic Server.
- J.D. Edwards OneWorld must be configured for inbound and outbound processing.

See Also: *Oracle Application Server Adapter Concepts 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 Oracle Application Adapter for J.D. Edwards OneWorld for services and events. For more information, see Chapter 2, "Configuring Oracle Application Adapter for J.D. Edwards OneWorld".
- How to configure Oracle JDeveloper. For more information, see Chapter 4, "Integration With BPEL Service Components in the Oracle SOA Suite".

Overview of Mediator Integration

Mediator provides a comprehensive application integration framework. Oracle Application Adapter for J.D. Edwards OneWorld used with Mediator 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.

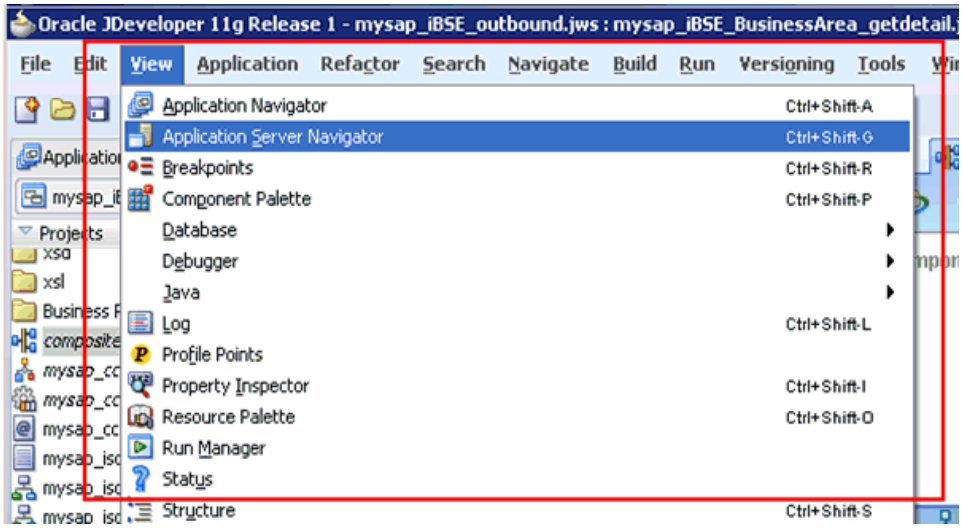
Adapter integration with Oracle WebLogic Server, Mediator is a two-step process:

1. **Design Time:** Oracle Application Adapter for J.D. Edwards OneWorld is configured in Application Explorer for services and events, as described in Chapter 2, "Configuring Oracle Application Adapter for J.D. Edwards OneWorld". Integration logic is modeled in iStudio. Metadata are stored in repositories.
2. **Runtime:** The underlying platform treats this metadata as run-time instructions to enable the communication between participating applications.

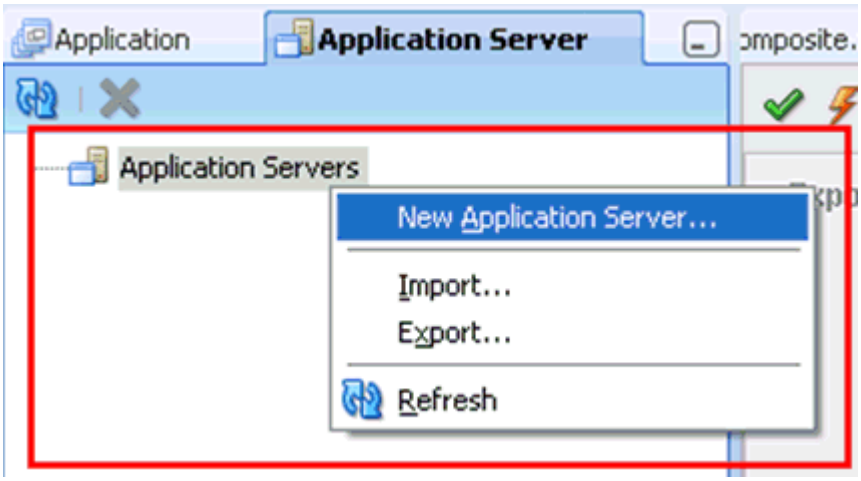
5.1 Configuring a New Application Server Connection

To configure a new Application Server connection in Oracle JDeveloper:

1. Open Oracle JDeveloper on your system.
2. From the menu bar, click **View** and select **Application Server Navigator**.

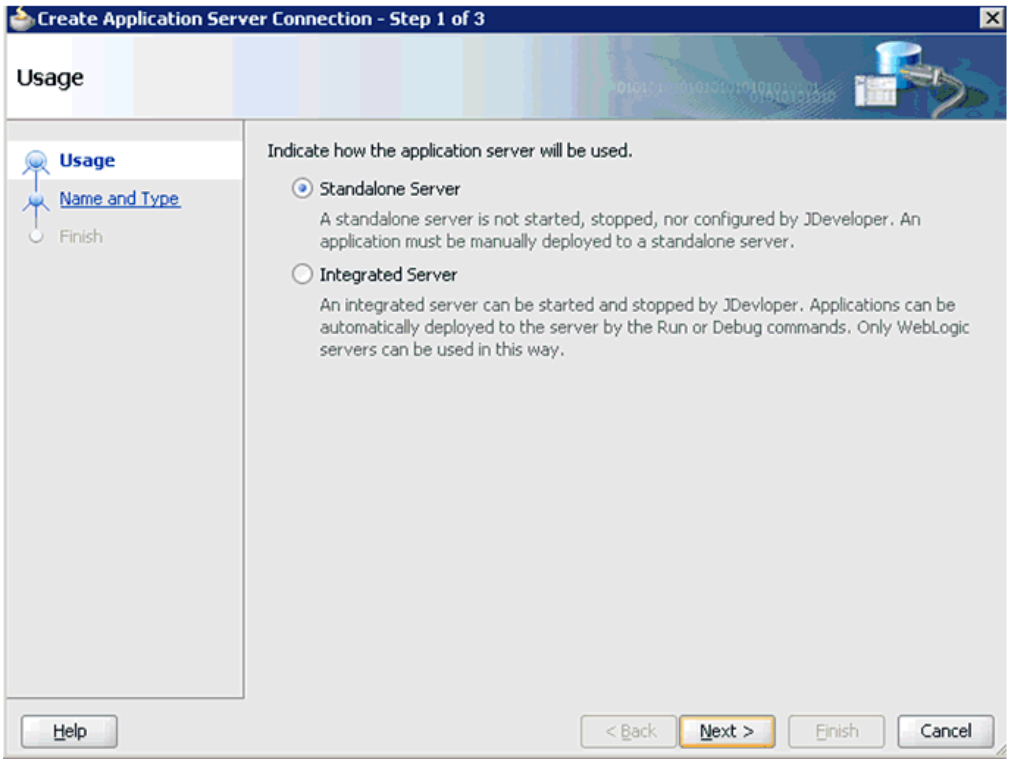


The Application Server tab is displayed.



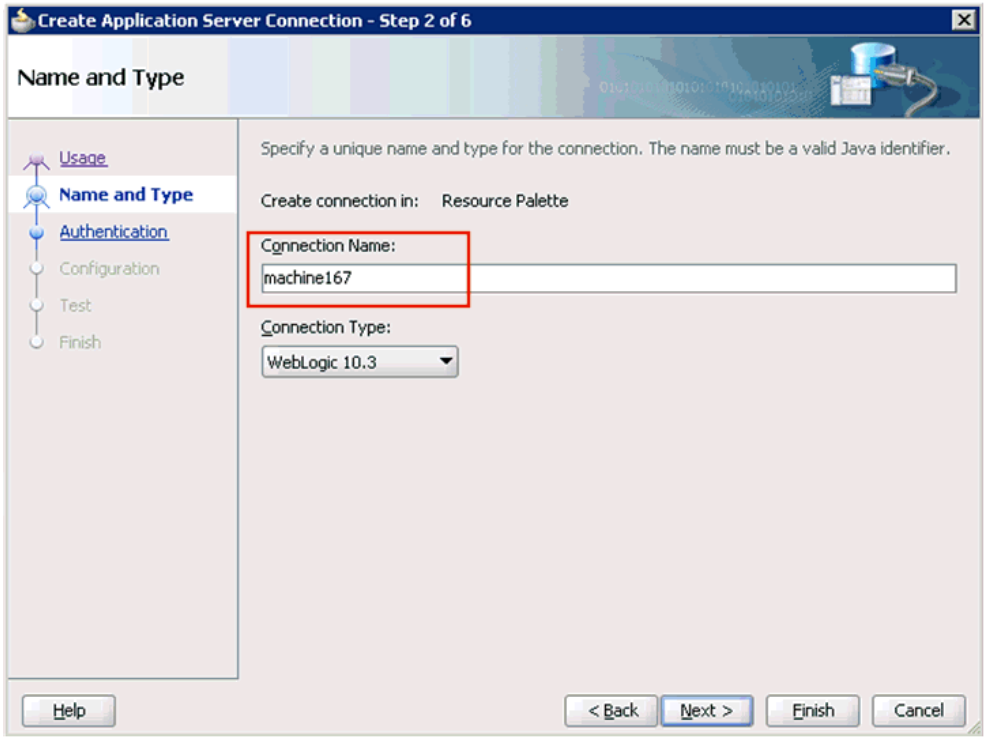
3. Right-click **Application Servers**, and then select **New Application Server**.

   The Create Application Server Connection Wizard is displayed.



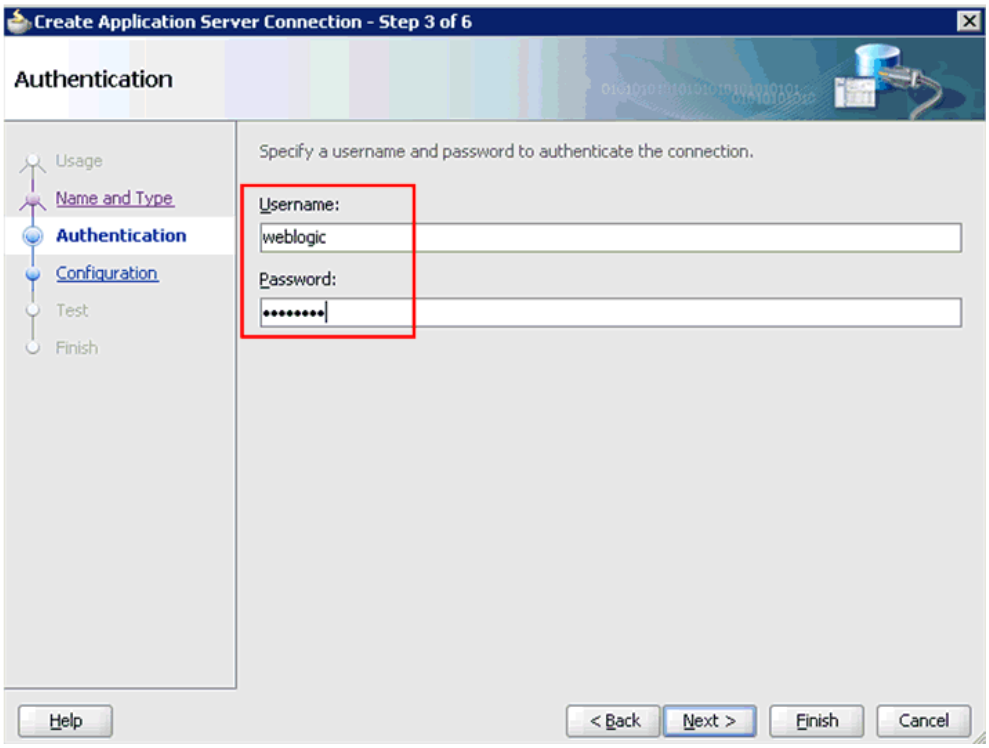
4. Accept the default selection (Standalone Server) and click **Next**.

   The Name and Type page is displayed.



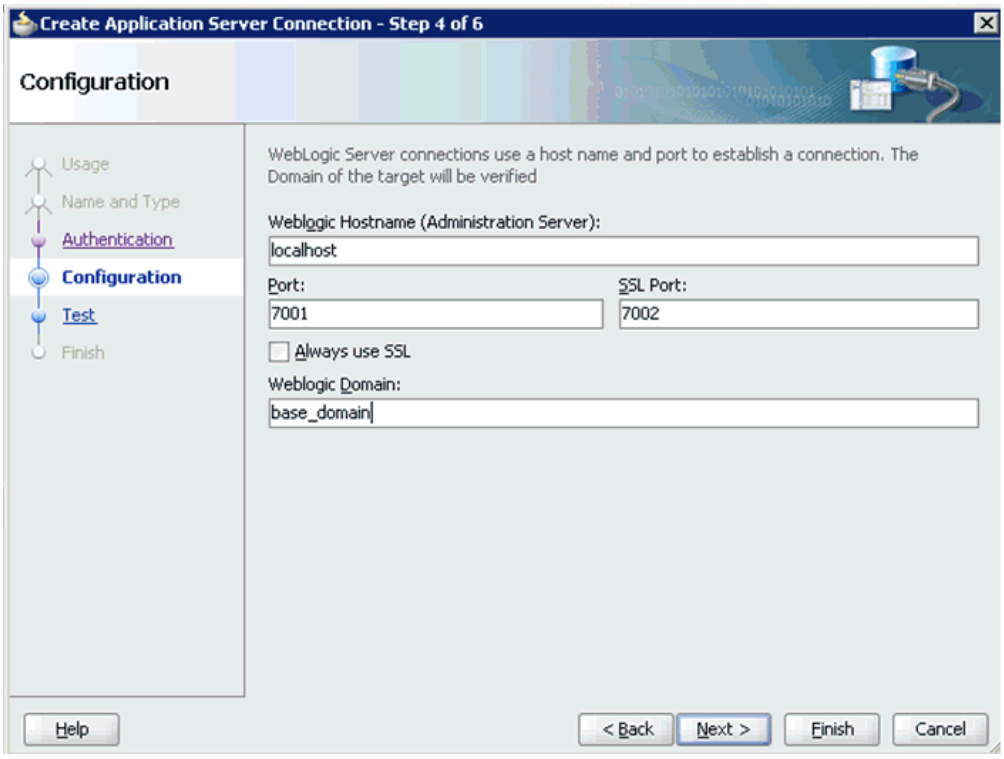
5. Specify a new name for the Application Server connection and click **Next**.

   The Authentication page is displayed.



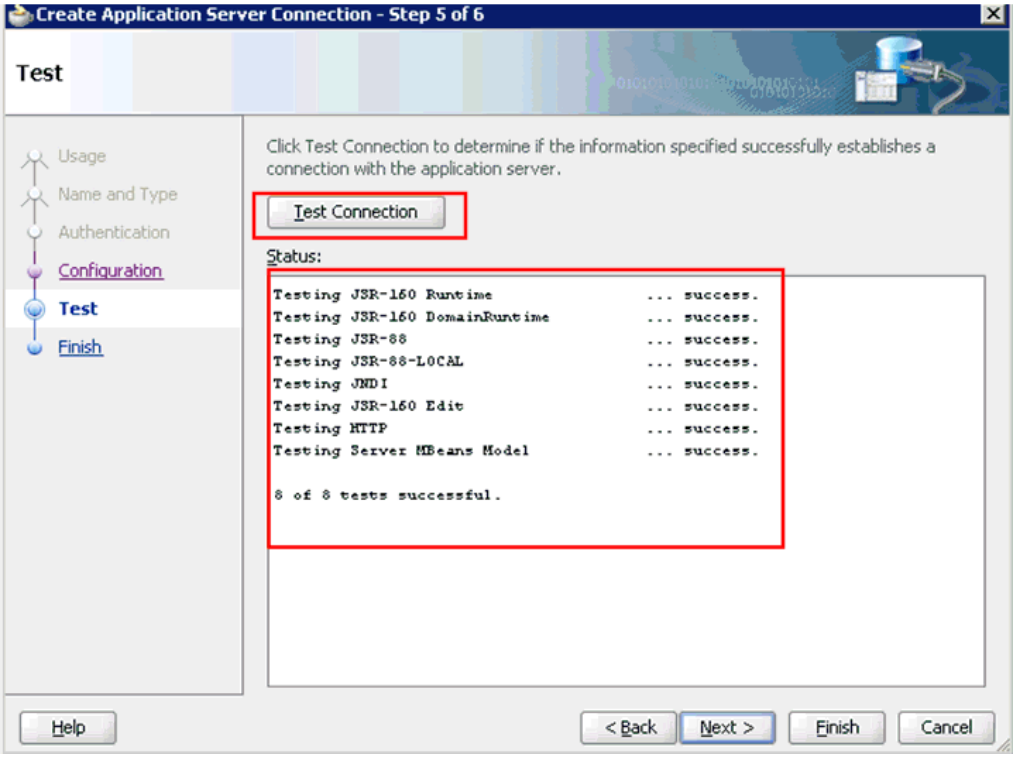
6. Specify a valid user name (for example, weblogic) and a password (for example, welcome1) for your new connection.
7. Click **Next**.

   The Configuration page is displayed.



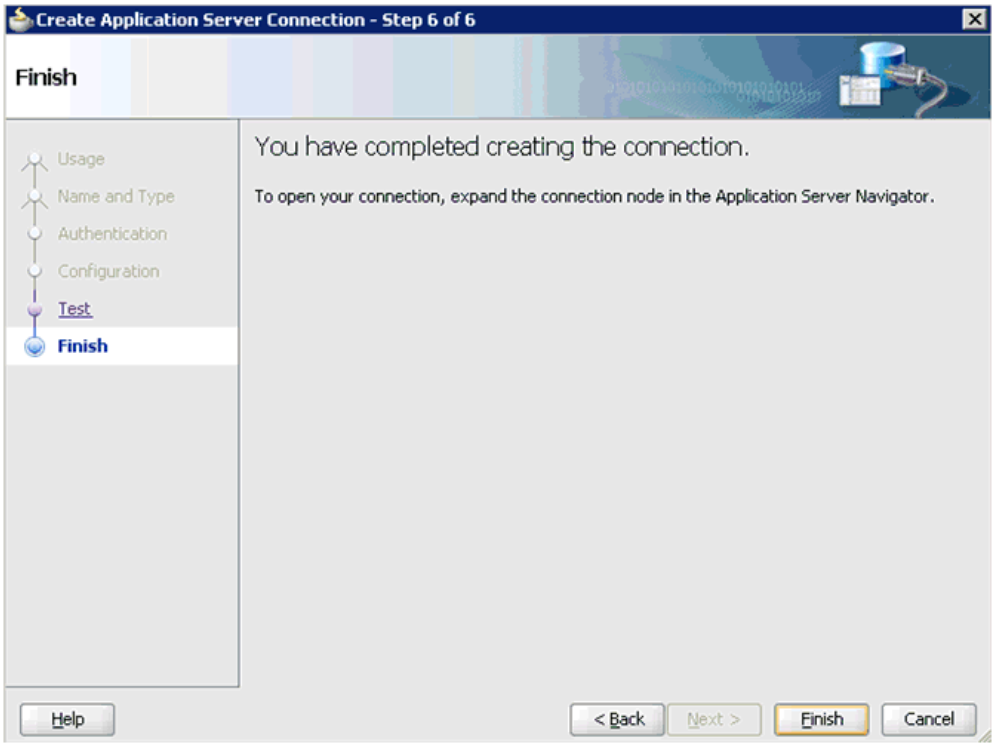
8. Specify the Oracle WebLogic host name (for example, localhost), which is the machine IP where the process needs to deploy and Oracle WebLogic domain (for example, base_domain).
9. Click **Next**.

   The Test page is displayed.



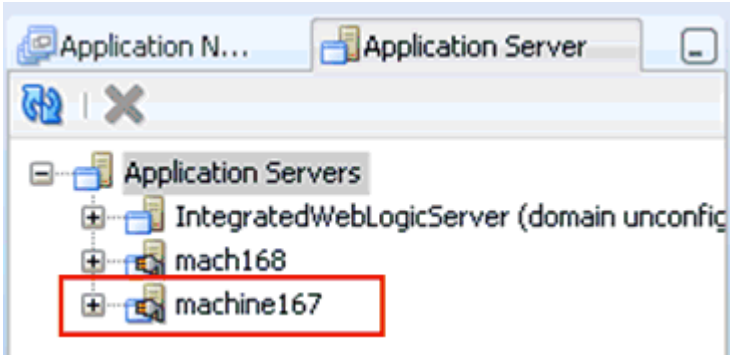
10. Click **Test Connection**.
11. Ensure that the test status is successful.
12. Click **Next**.

    The Finish page is displayed.



13. Click **Finish**.

    The new Application Server connection is listed in the left pane (Application Server tab), as shown in the following image.



5.2 Configuring a Mediator Outbound Process

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

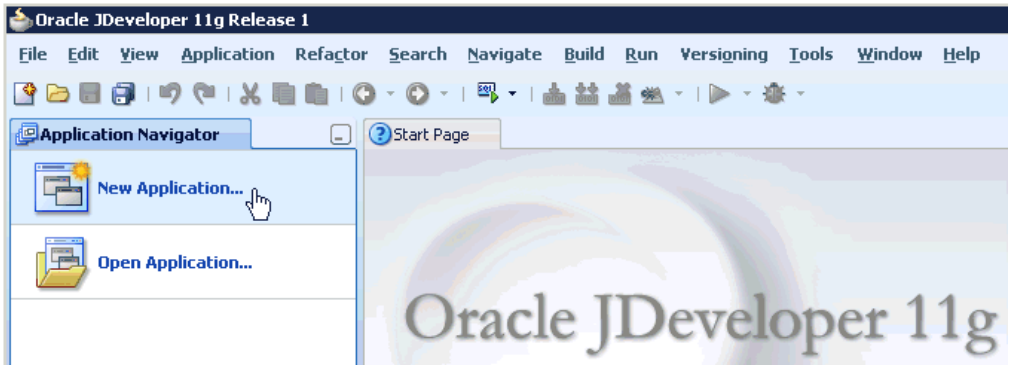
Prerequisites

Before you design a Mediator outbound process, you must generate the respective WSDL file using Application Explorer. See "Generating WSDL for Request/Response Service" on page 4-7 for more information.

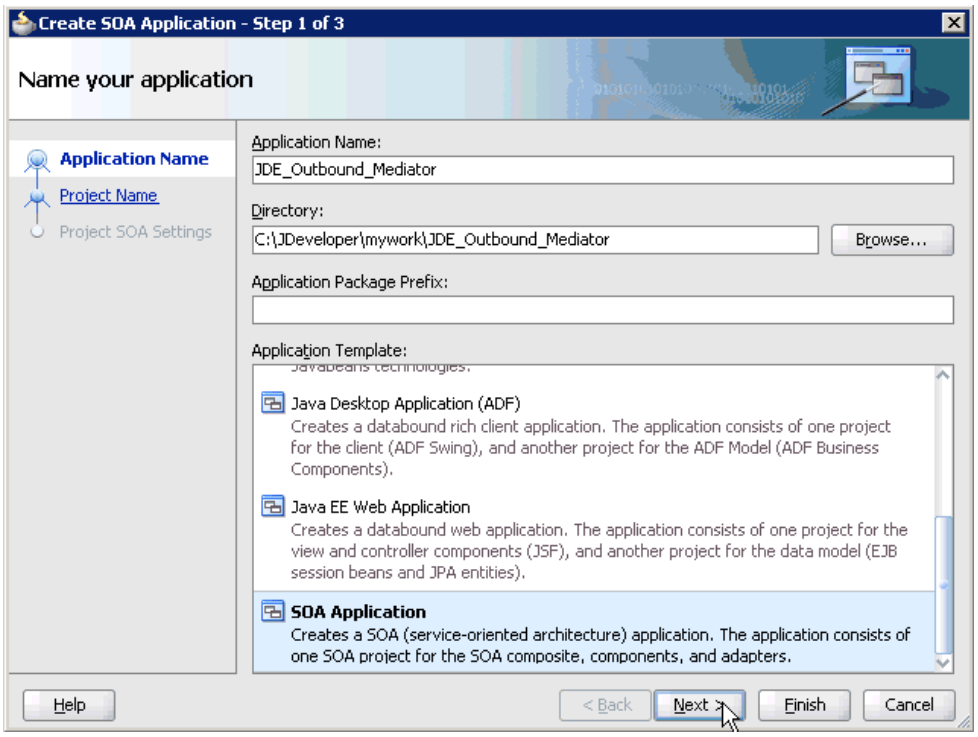
5.2.1 Creating a New SOA Application for the Outbound Mediator Process

Perform the following steps to create a new SOA application for the outbound Mediator process:

1. Open Oracle JDeveloper on your system.
2. In the Application Navigator tab, click **New Application**.

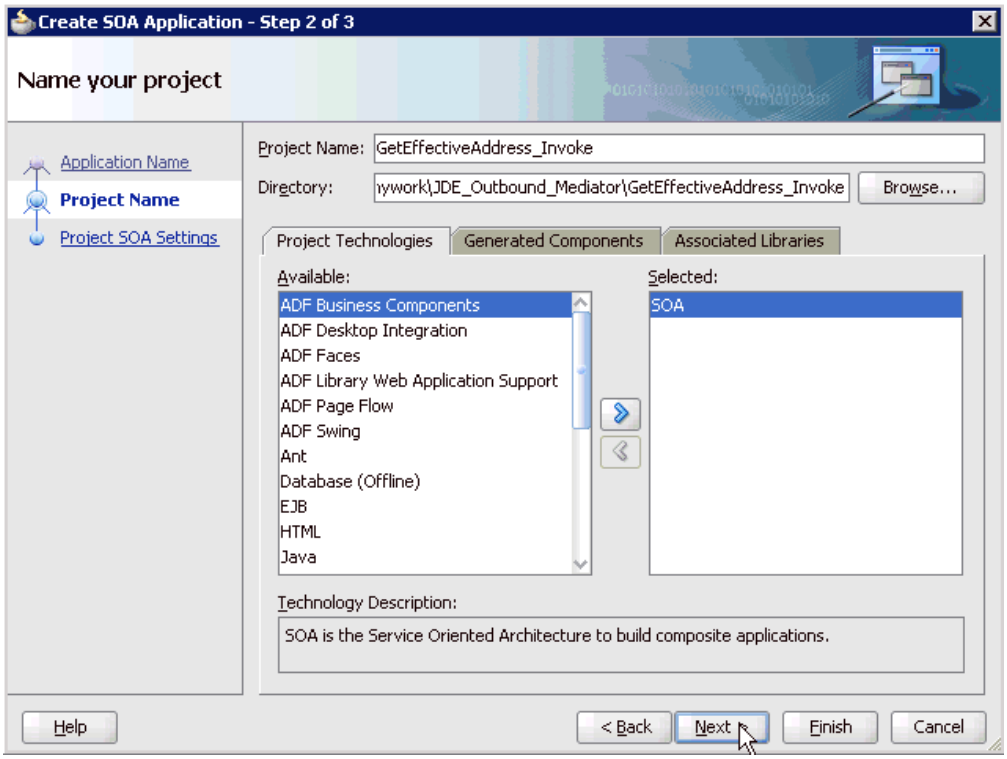


The Create SOA Application wizard is displayed.

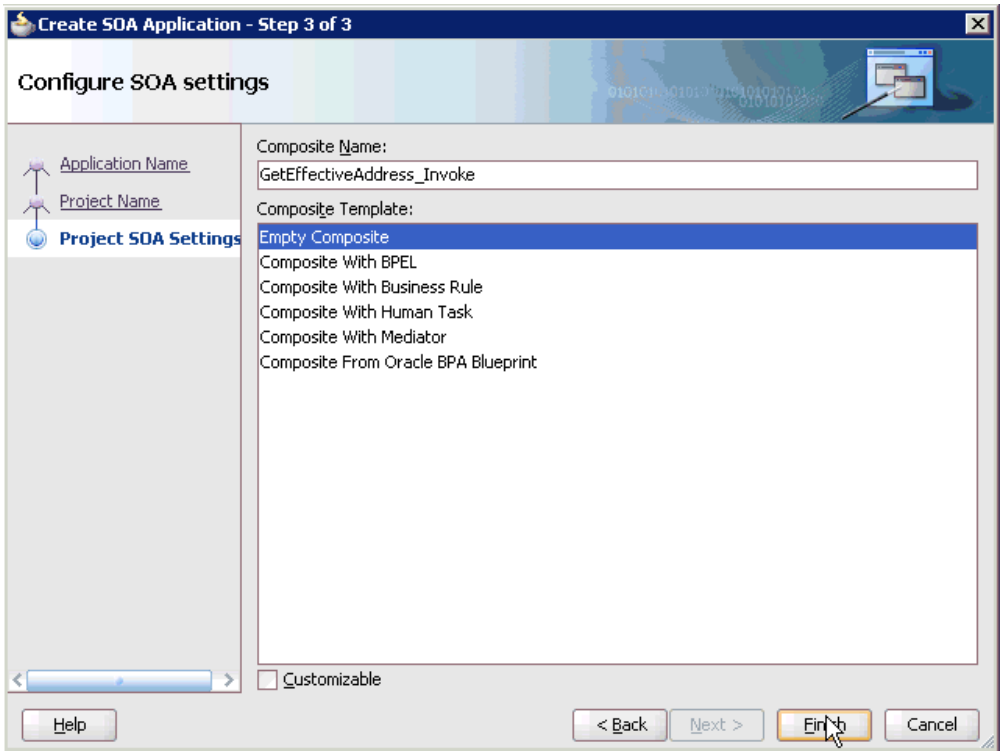


3. From the Application Template list, click **SOA Application**.
4. Enter name for the new SOA application (for example, JDE_Outbound_Mediator) and click **Next**.

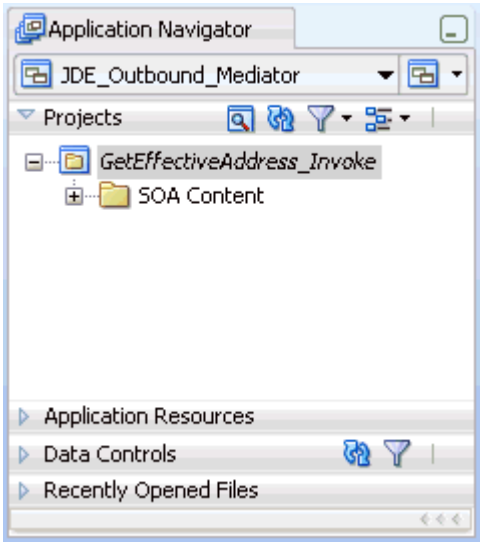
   The Name your project page is displayed.



5. Enter a project name (for example, GetEffectiveAddress_Invoke) and click **Next**. The Configure SOA settings page is displayed.



6. From the Composite Template list, select **Empty Composite** and click **Finish**.



The new SOA application (JDE_Outbound_Mediator) and associated project (GetEffectiveAddress_Invoke) are added to the Application Navigator tab in the left pane.

5.2.2 Defining a Mediator Outbound Process

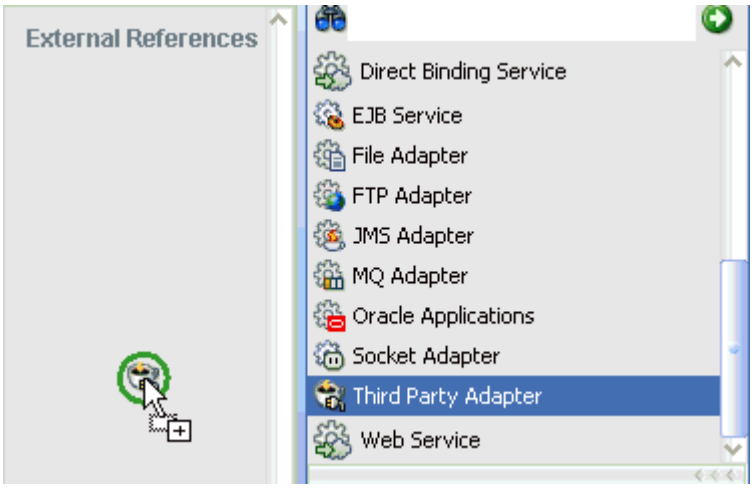
This section describes how to define a Mediator outbound process, which consists of the following stages:

1. Configuring a Third Party Adapter Service Component
2. Configuring an Outbound Mediator Process Component
3. Configuring the Routing Rules

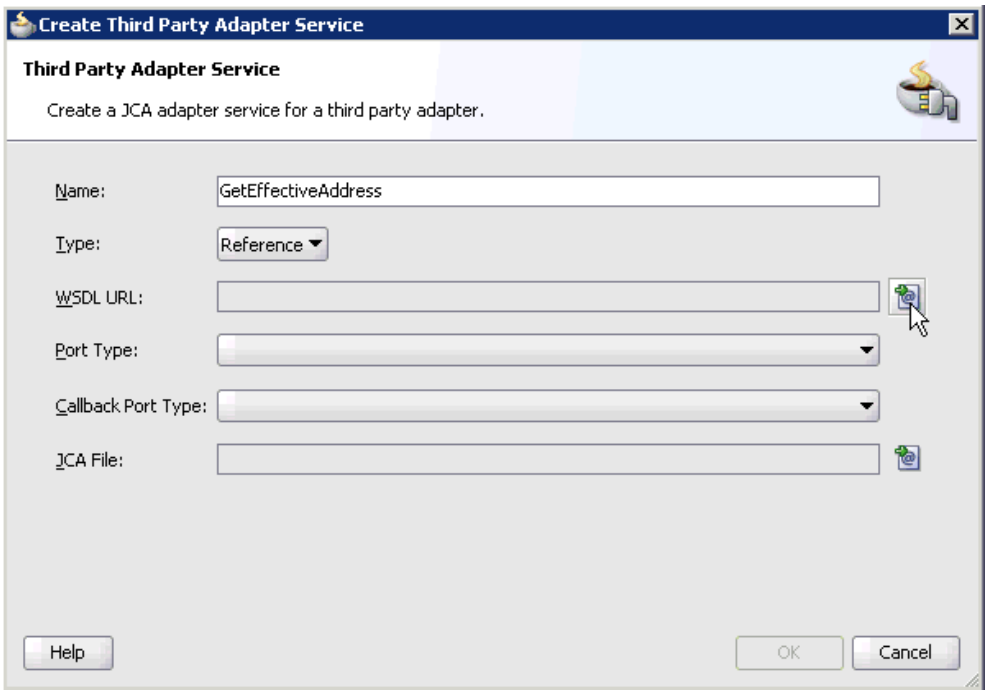
Configuring a Third Party Adapter Service Component

Perform the following steps to create a third party adapter service component:

1. Drag and drop the **Third Party Adapter** component from the Component Palette tab (Service Adapters section) to the External References pane.

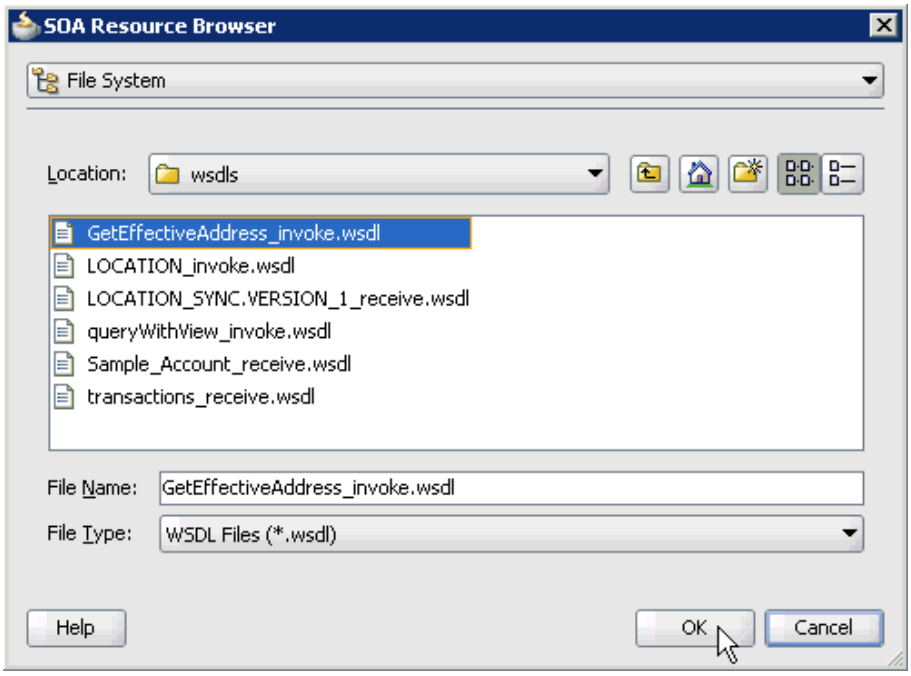


The Create Third Party Adapter Service dialog is displayed.



2. Enter a name for the new third party adapter service.
3. Ensure that **Reference** is selected from the Type list (default).
4. Click the **Find existing WSDLs** icon, which is located to the right of the WSDL URL field.

   The SOA Resource Browser dialog is displayed.

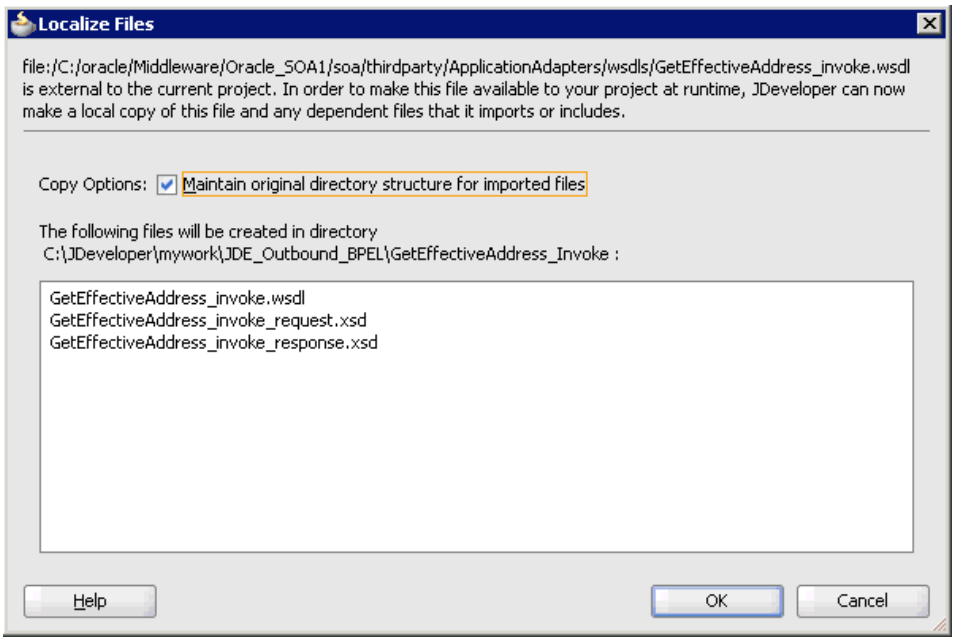


5. Browse and select an outbound WSDL file (for example, GetEffectiveAddress_invoke.wsdl) from the following directory:

   `C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\wsdls`

6. Click **OK**.

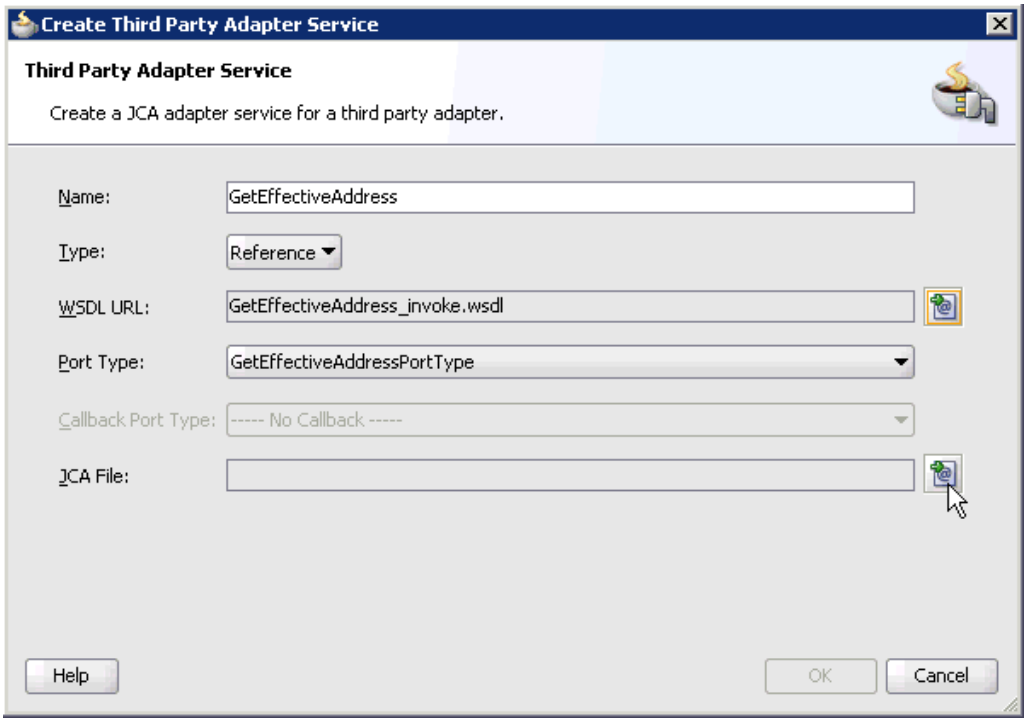
   The Localize Files dialog is displayed.



7. Click **OK**.

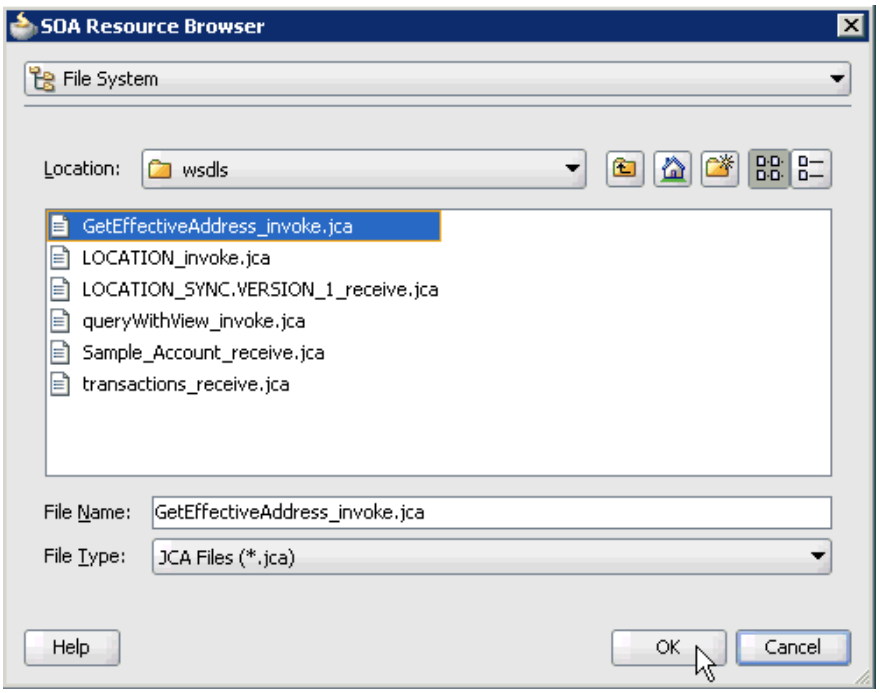
   The outbound WSDL file and associated request and response XML schema files (.xsd) are imported to the project folder that has been created.

   You are returned to the Create Third Party Adapter Service dialog.



8. Click the **Find JCA Files** icon, which is located to the right of the JCA File field.

The SOA Resource Browser dialog is displayed.

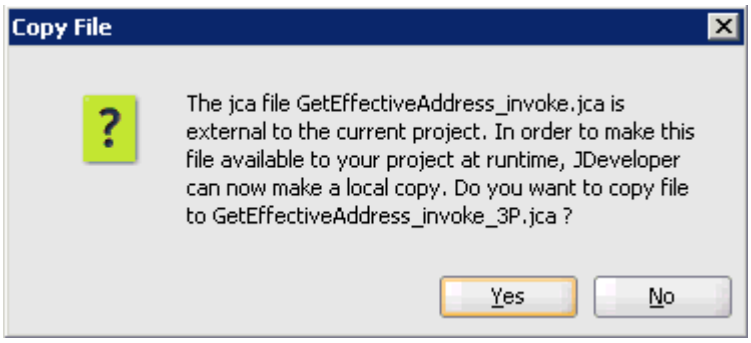


9. Browse and select the JCA properties file (for example, GetEffectiveAddress_invoke.jca) from the following directory:

   `C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\wsdls`

10. Click **OK**.

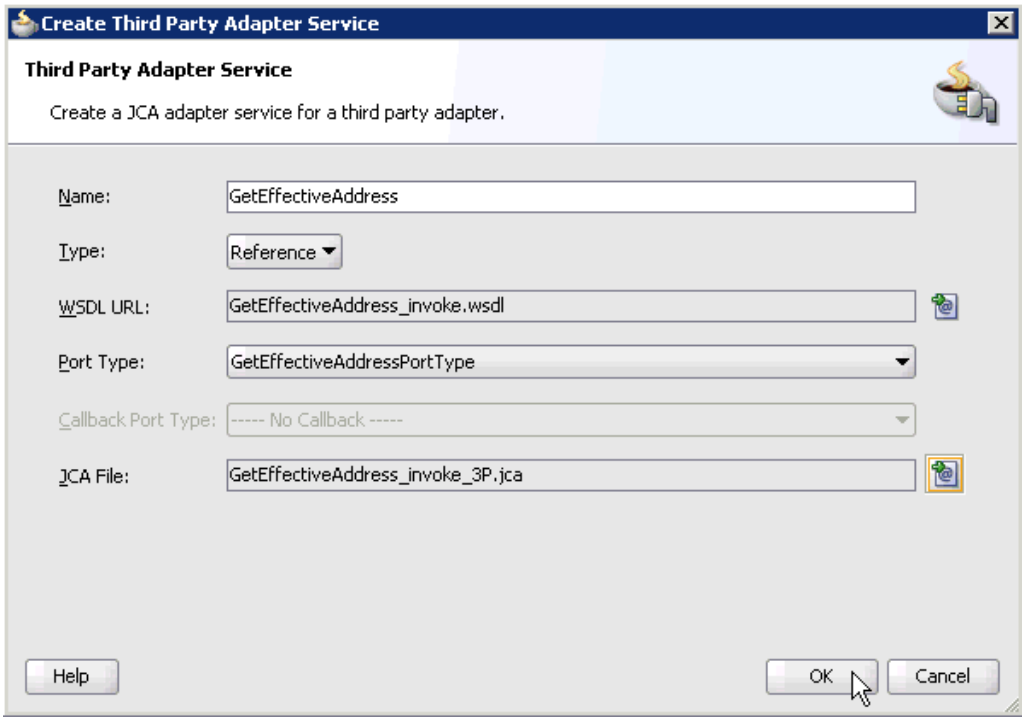
    The following message is displayed.



11. Click **Yes**.

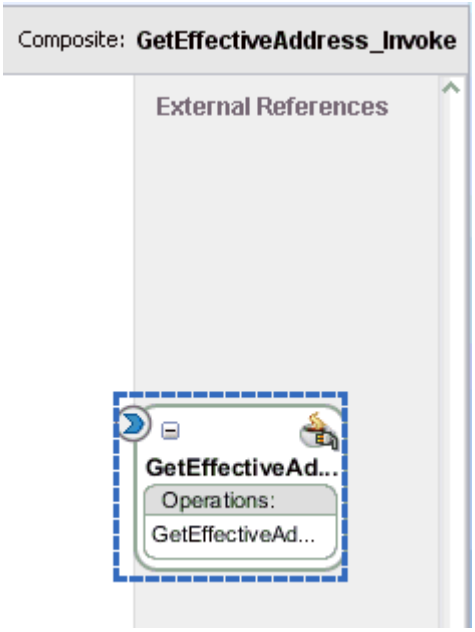
    A copy of the JCA properties file is made in the project folder.

    You are returned to the Create Third Party Adapter Service dialog.



12. Click **OK**.

    The third party adapter service component (GetEffectiveAddress) is created in the External References pane, as shown in the following image.

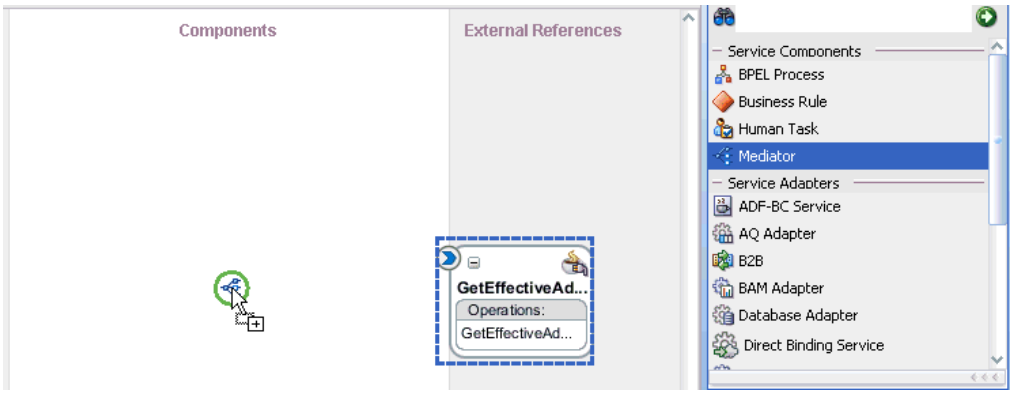


    You are now ready to configure an outbound Mediator process component.

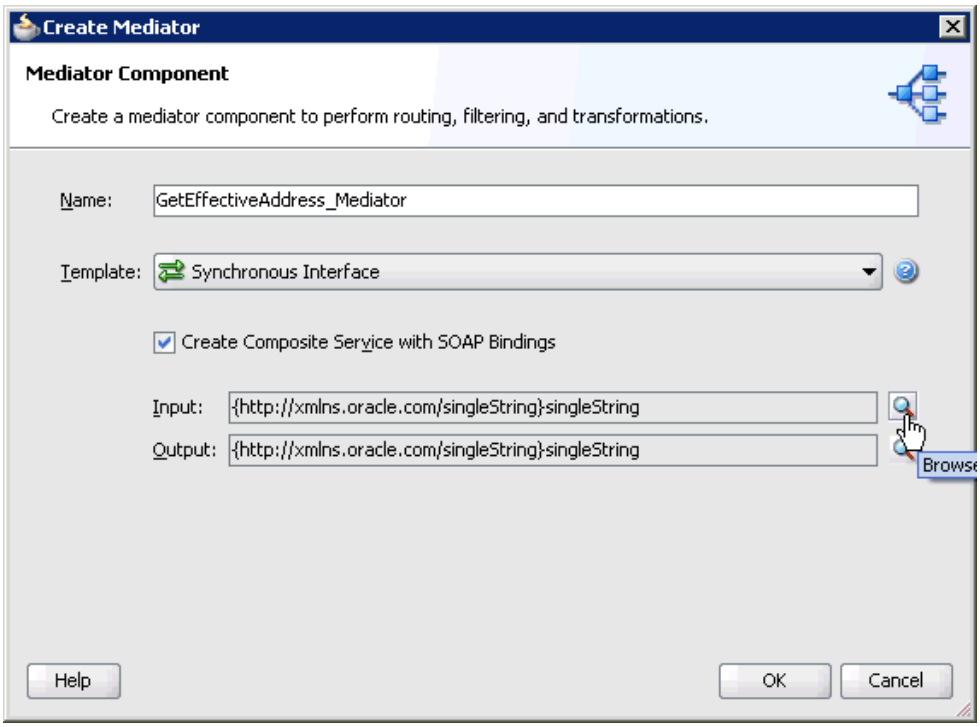
Configuring an Outbound Mediator Process Component

Perform the following steps to configure an outbound Mediator process component:

1. Drag and drop the **Mediator** component from the Component Palette tab (Service Components section) to the Components pane.

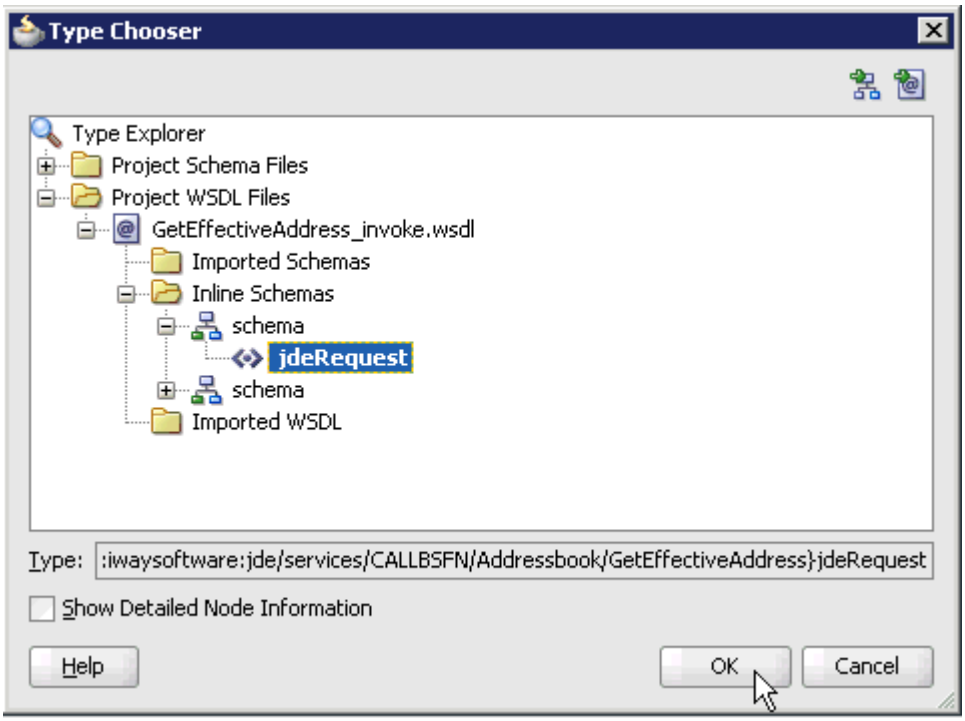


The Create Mediator dialog is displayed.



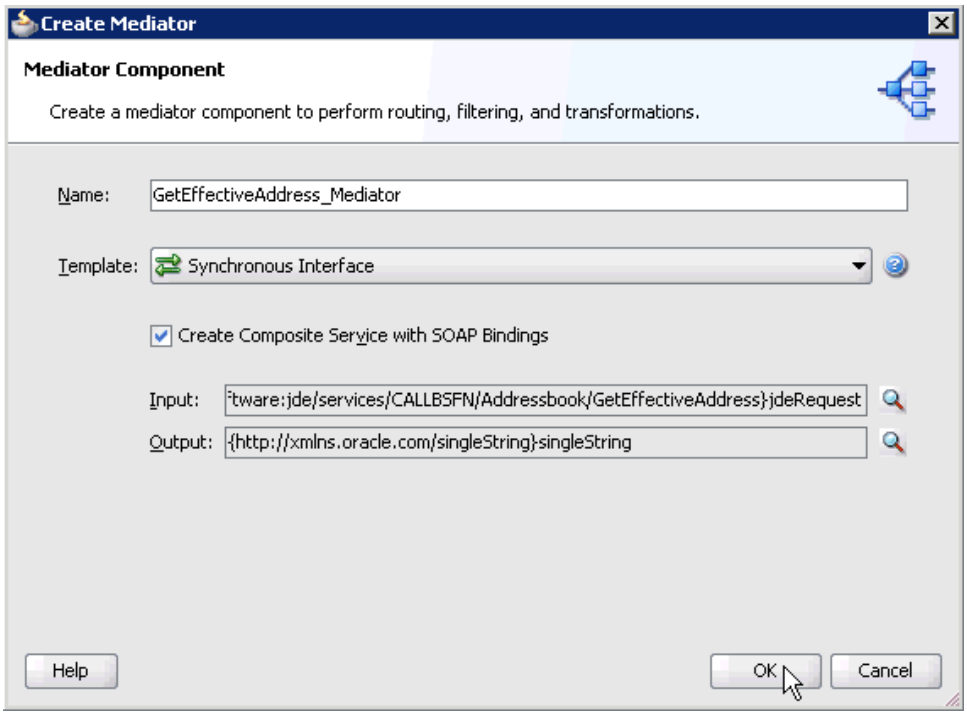
2. In the Name field, enter a name to identify the new outbound Mediator process component (for example, GetEffectiveAddress_Mediator).
3. From the Template list, select **Synchronous Interface**.
4. Click the **Browse Input Elements** icon, which is located to the right of the Input field to select the associated XML request schema file.

   The Type Chooser dialog is displayed.



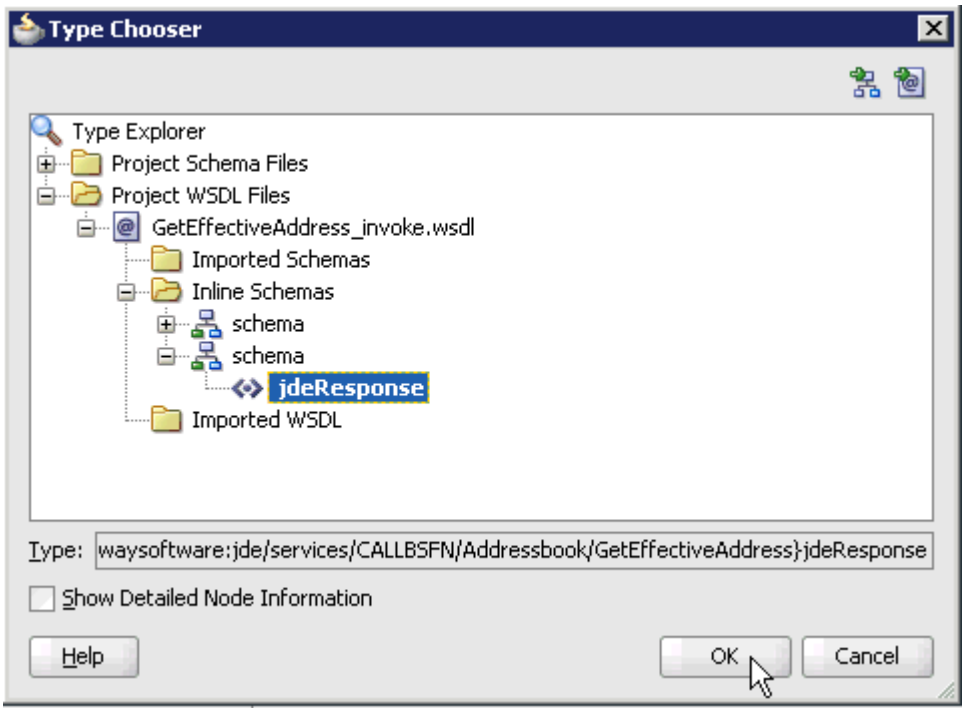
5. Expand **Project WSDL Files**, **GetEffectiveAddress_invoke.wsdl**, **Inline Schemas**, **schema**, and select **jdeRequest**.
6. Click **OK**.

   You are returned to the Create Mediator dialog.



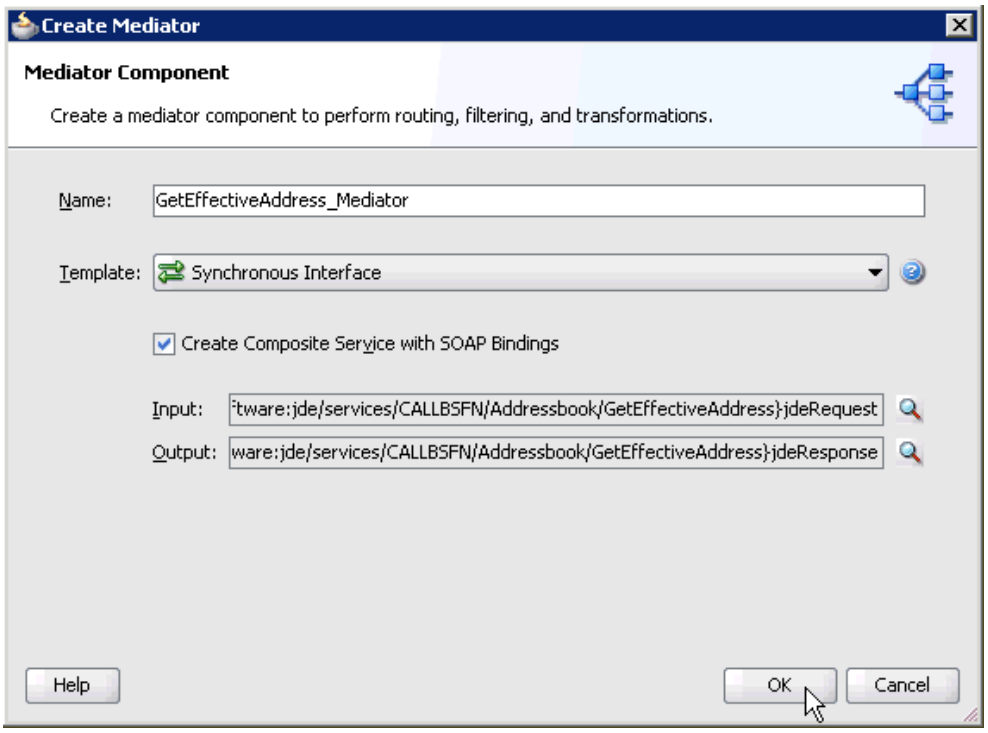
7. Click the **Browse Output Elements** icon, which is located to the right of the Output field to select the associated XML response schema file.

The Type Chooser dialog is displayed.



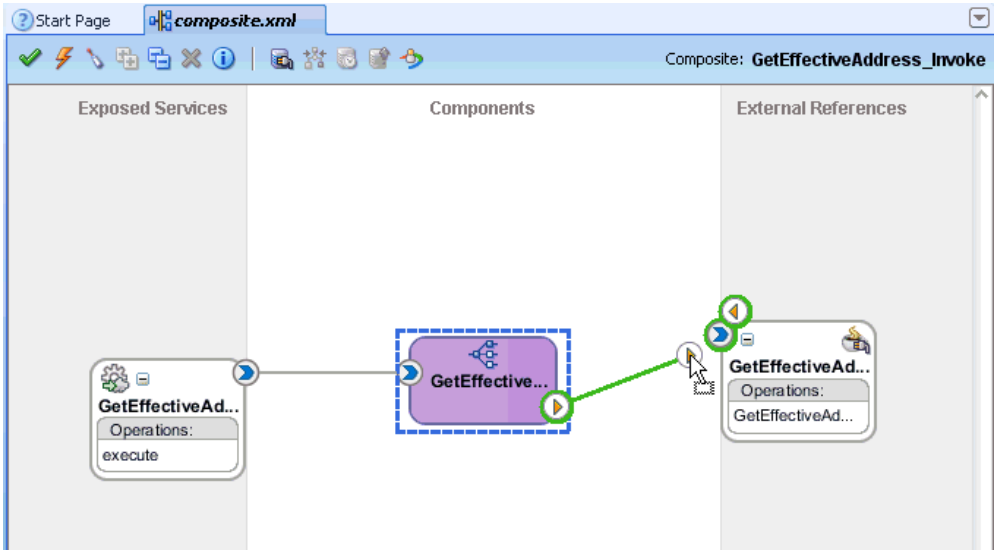
8. Expand **Project WSDL Files**, **GetEffectiveAddress_invoke.wsdl**, **Inline Schemas**, **schema**, and select **jdeResponse**.
9. Click **OK**.

   You are returned to the Create Mediator dialog.



10. Click **OK**.

11. Create a connection between the outbound Mediator process component (GetEffectiveAddress_ Mediator) and the third party adapter service component (GetEffectiveAddress).

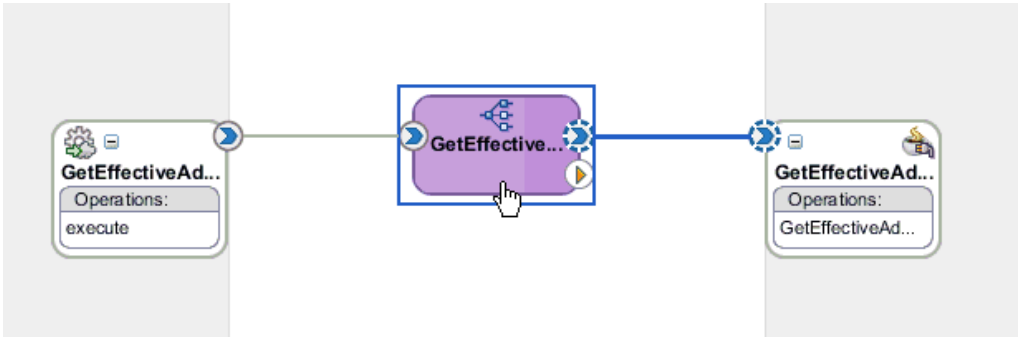


You are now ready to configure the routing rules.

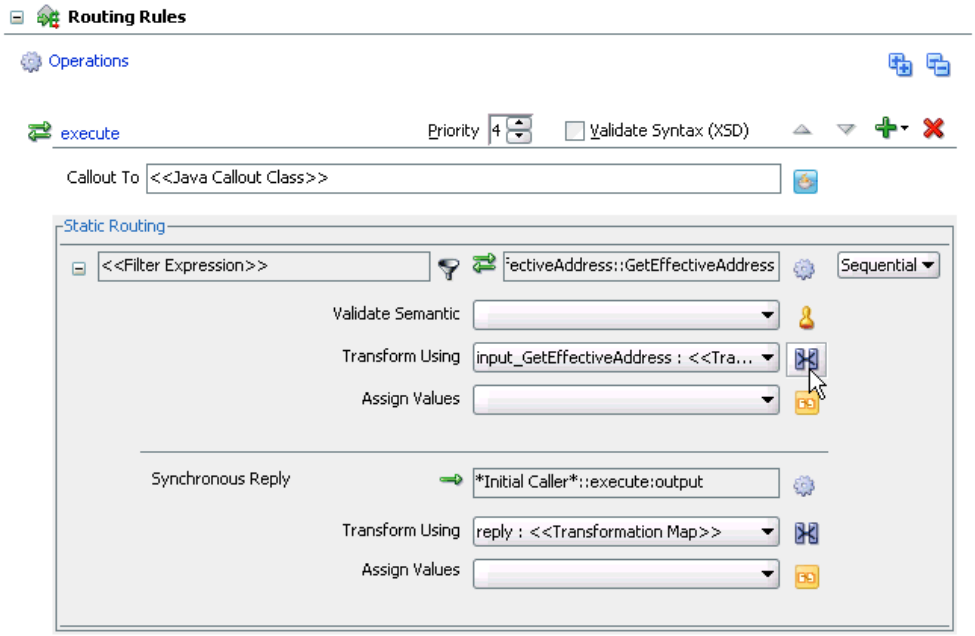
Configuring the Routing Rules

Perform the following steps to configure routing rules for the Mediator outbound process component:

1. Double-click the outbound Mediator process component (GetEffectiveAddress_ Mediator) in the Components pane.

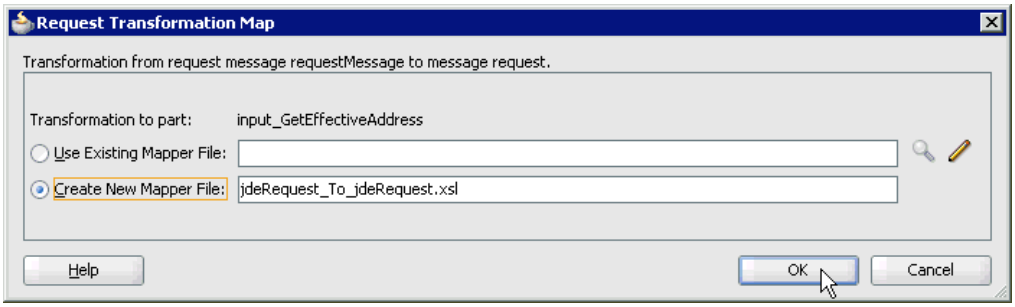


The Routing Rules dialog is displayed.



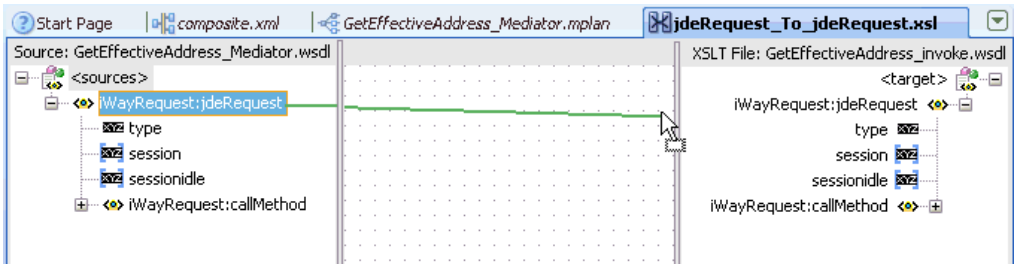
2. In the <<Filter Expression>> area, click the **Select an existing mapper file or create a new one** icon to the right of the Transform Using field.

   The Request Transformation Map dialog is displayed.



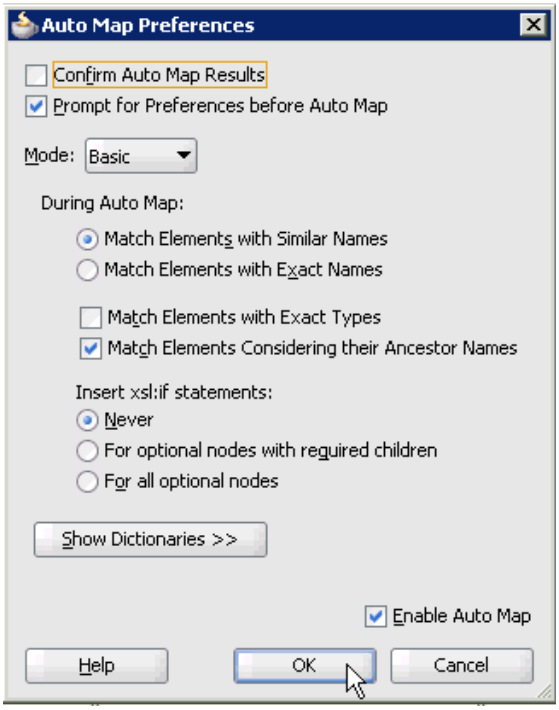
3. Select the **Create New Mapper File** option and click **OK**.

   The jdeRequest_To_jdeRequest.xsl tab is displayed.

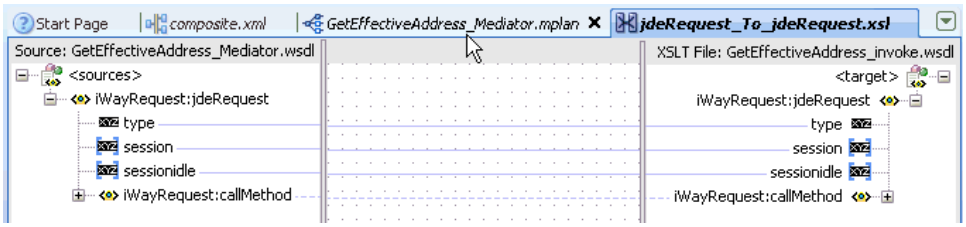


4. Map the **iWayRequest:jdeRequest** source element to the **iWayRequest:jdeRequest** target element.

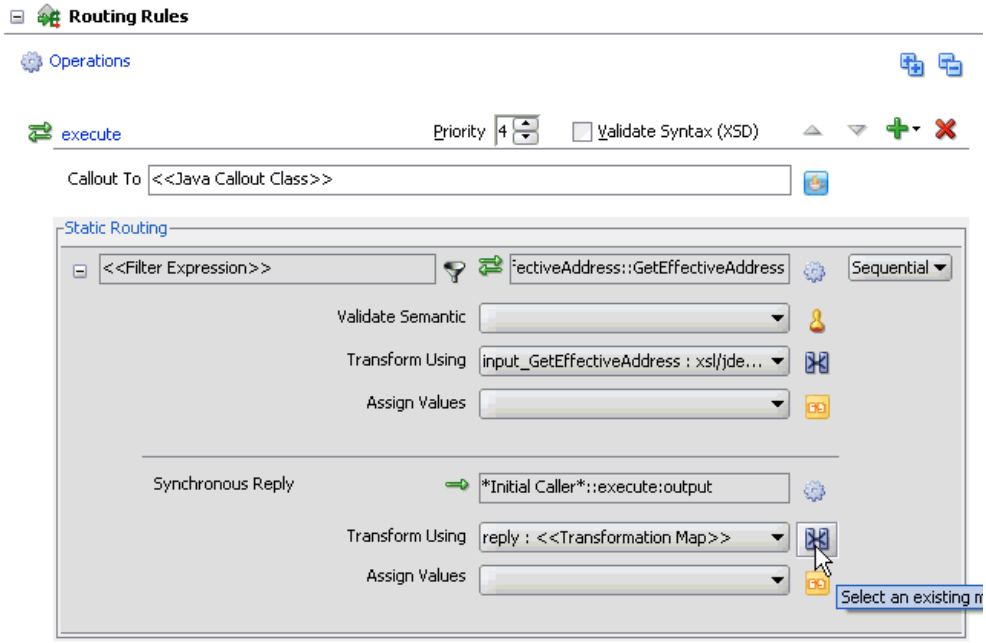
   The Auto Map Preferences dialog is displayed.



5. Retain the default values and click **OK**.
6. Click the **GetEffectiveAddress_Mediator.mplan** tab.

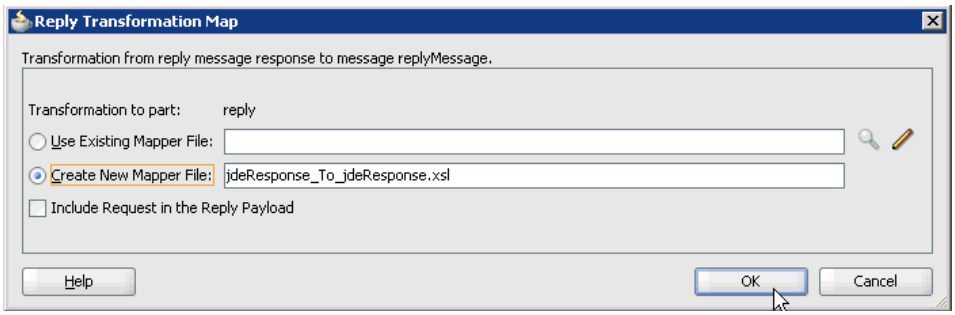


You are returned to the Routing Rules dialog.



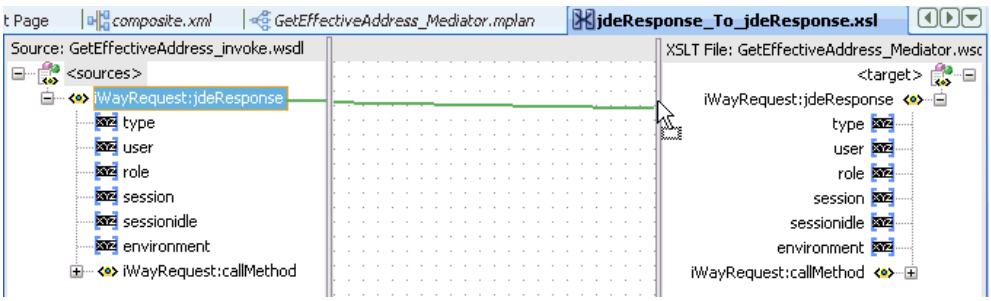
7. In the Synchronous Reply area, click the **Select an existing mapper file or create a new one** icon to the right of the Transform Using field.

   The Reply Transformation Map dialog is displayed.



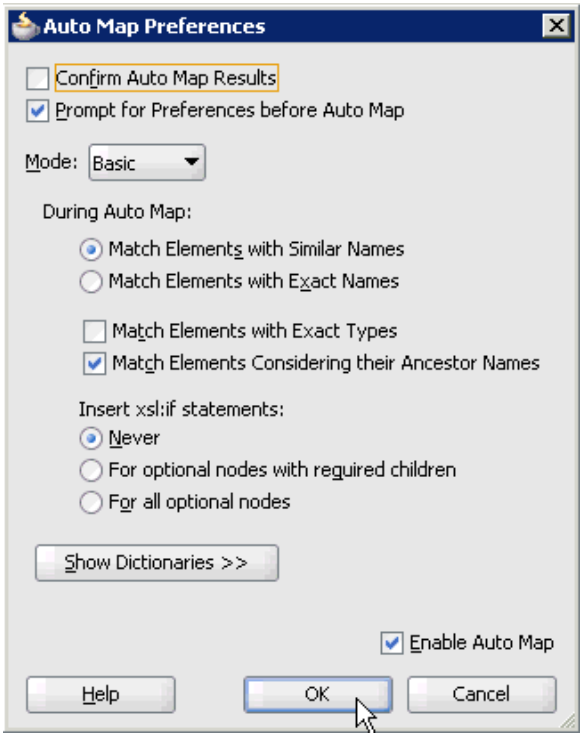
8. Select the **Create New Mapper File** option and click **OK**.

   The jdeResponse_To_jdeResponse.xsl tab is displayed.



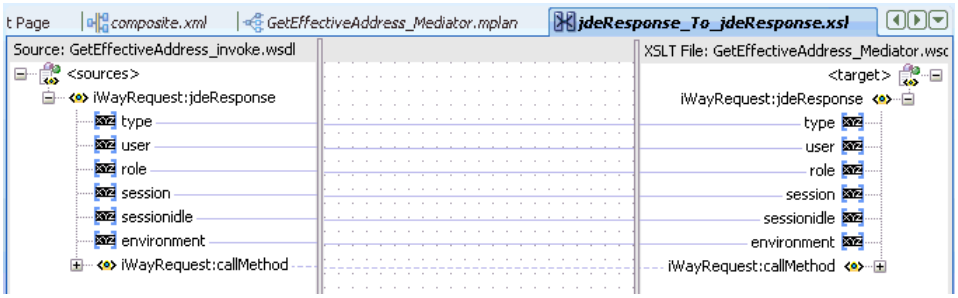
9. Map the **iWayRequest:jdeResponse** source element to the **iWayRequest:jdeResponse** target element.

   The Auto Map Preferences dialog is displayed.

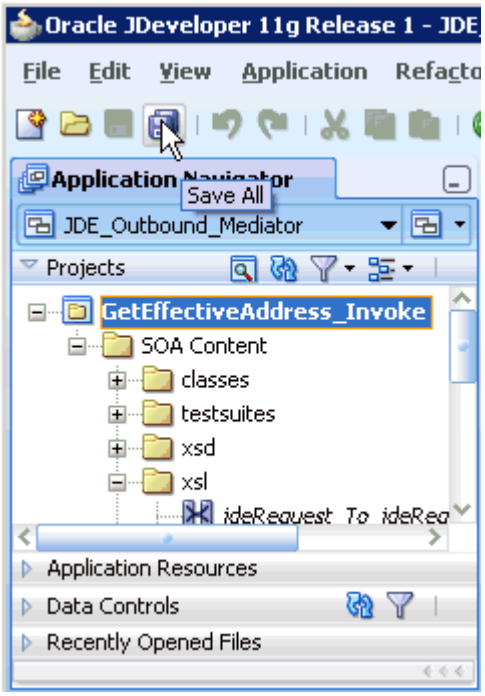


10. Retain the default values and click **OK**.

The mapping is complete, as shown in the following image.



11. Click the **Save All** icon in the menu bar to save the new outbound Mediator process component that was configured.

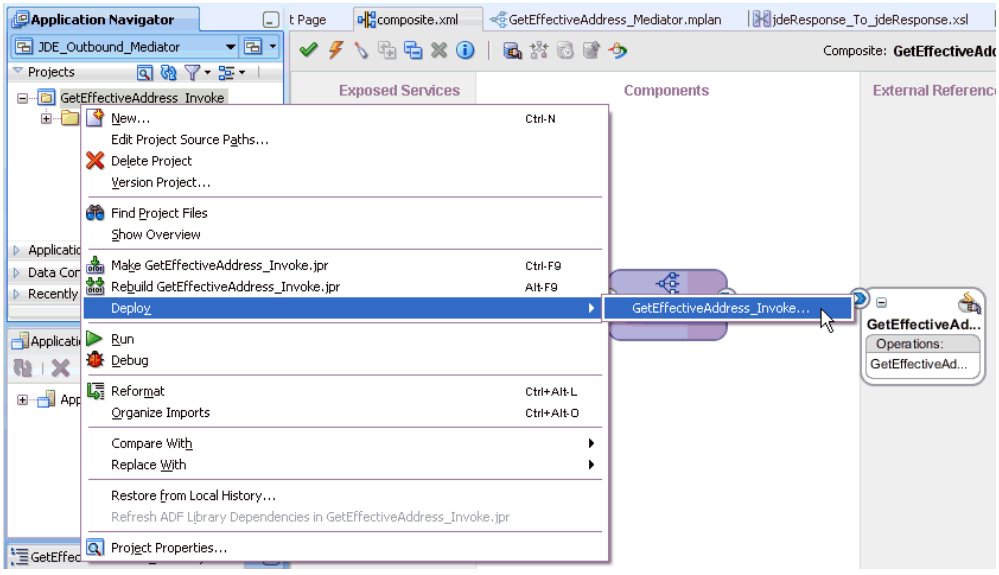


You are now ready to deploy the Mediator outbound process.

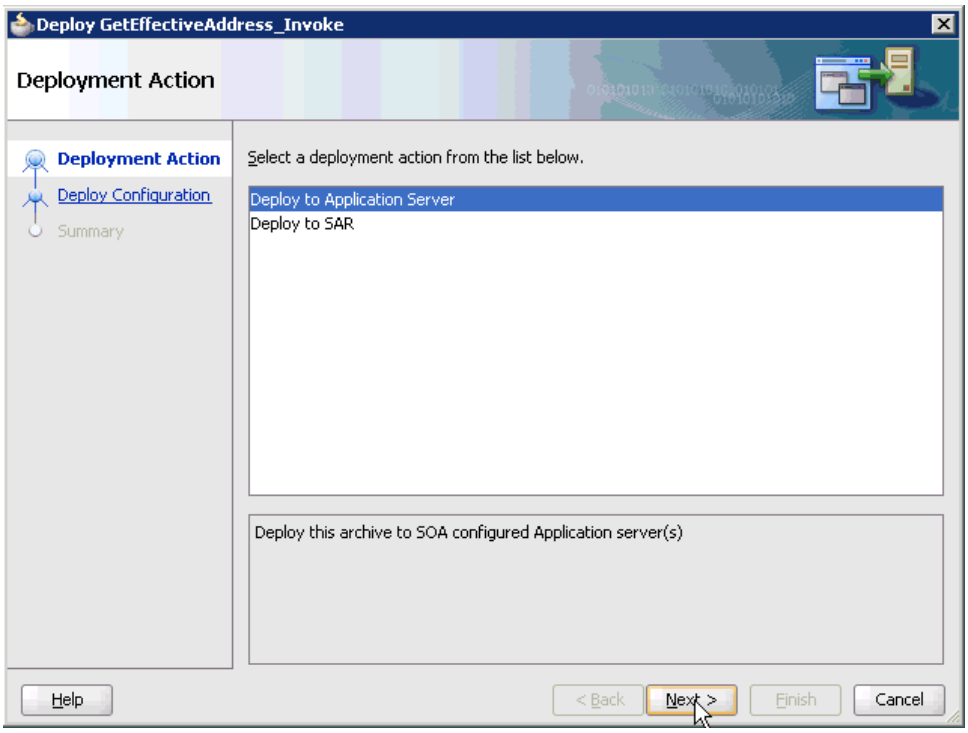
5.2.3 Deploying the Mediator Outbound Process

Perform the following steps to deploy the Mediator outbound process.

1. Right-click the project name in the left pane (for example, **GetEffectiveAddress_Invoke**), select **Deploy**, and then click **GetEffectiveAddress_Invoke**.

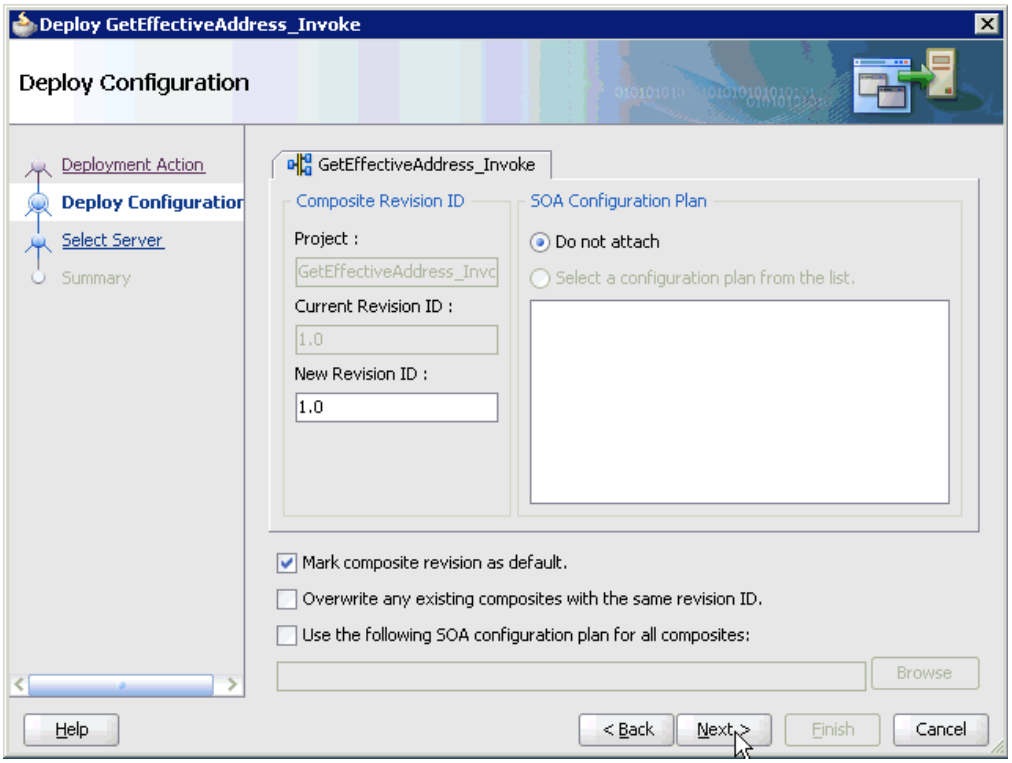


The Deployment Action page is displayed.



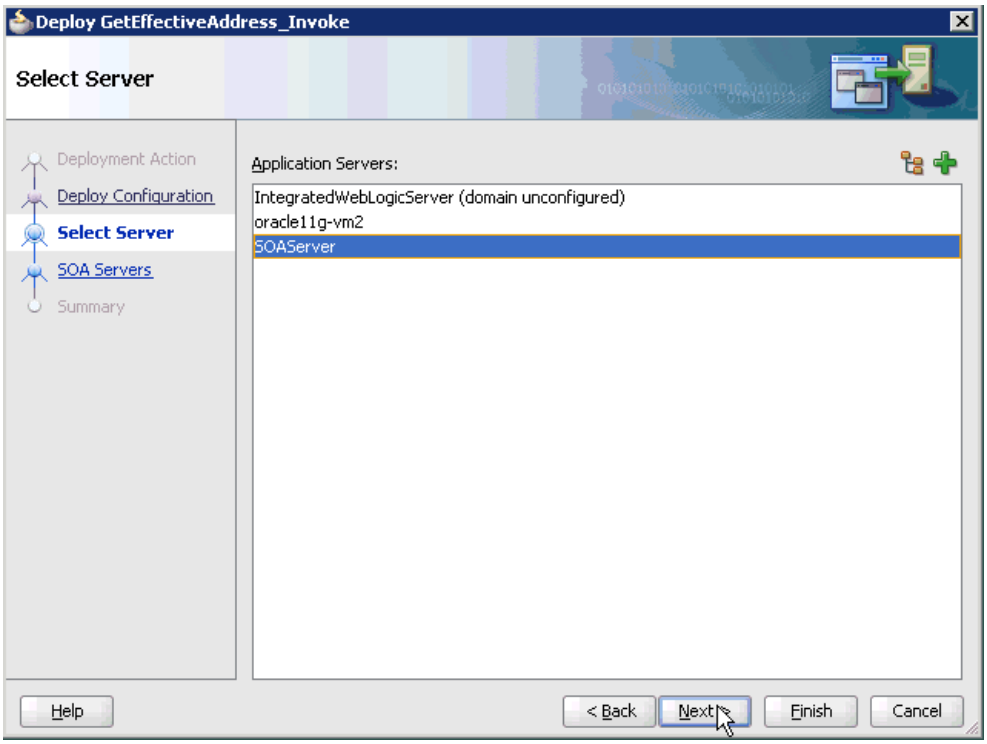
2. Ensure that **Deploy to Application Server** is selected.
3. Click **Next**.

   The Deploy Configuration page is displayed.



4. Leave the default values selected and click **Next**.

The Select Server page is displayed.

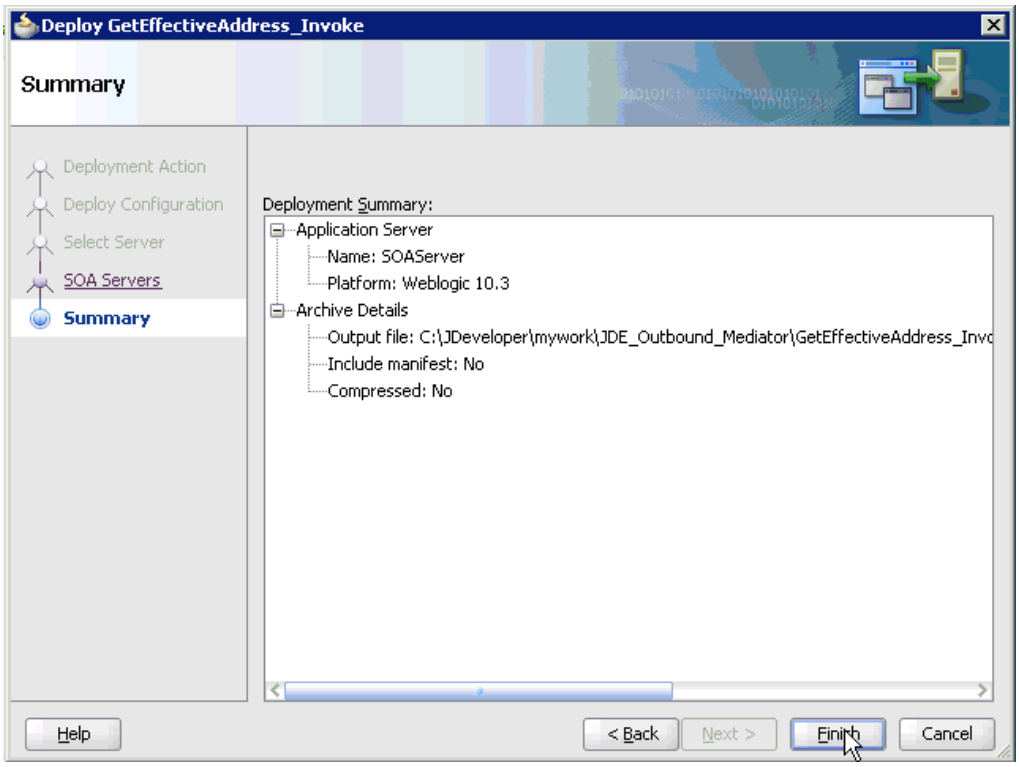


5. Select an available application server that was configured and click **Next**.

   The SOA Servers dialog is displayed.

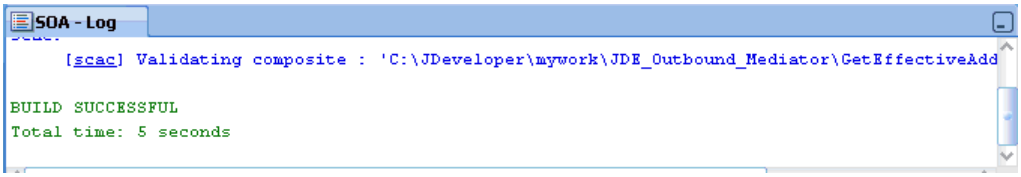
6. Select a target SOA server and click **Next**.

   The Summary page is displayed.



7. Review and verify all the available deployment information for your project and click **Finish**.

   The process is deployed successfully.



   If an Authorization Request dialog is displayed during the deployment process, provide the required user name and password and click **OK**.

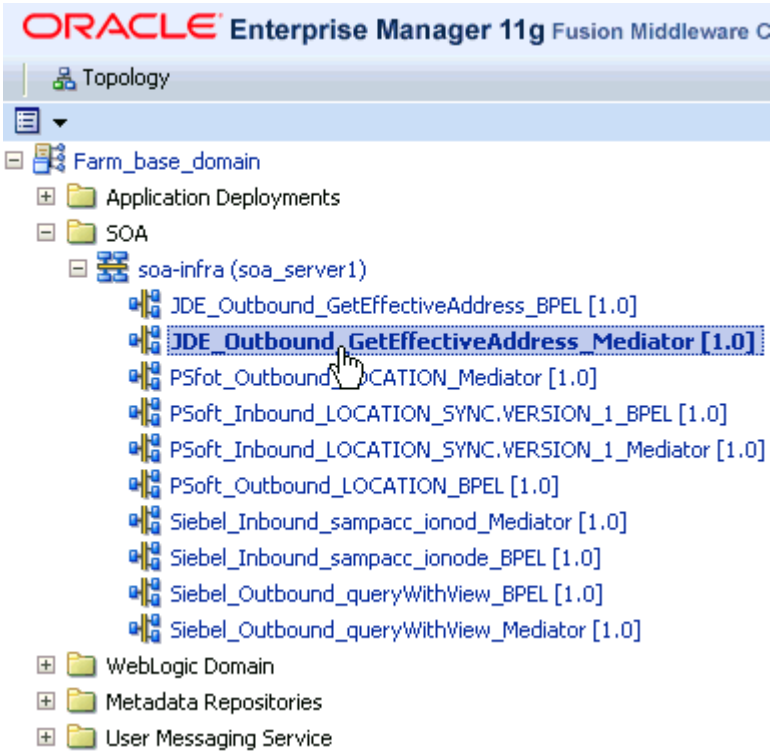
5.2.4 Invoking the Input XML Document in the Oracle Enterprise Manager Console

Perform the following steps to invoke the input XML document in the Oracle Enterprise Manager console.

1. Log in to the Oracle Enterprise Manager console by using the following URL:

   `http://localhost:7001/em`

2. Expand your domain in the left pane followed by the **SOA** folder.

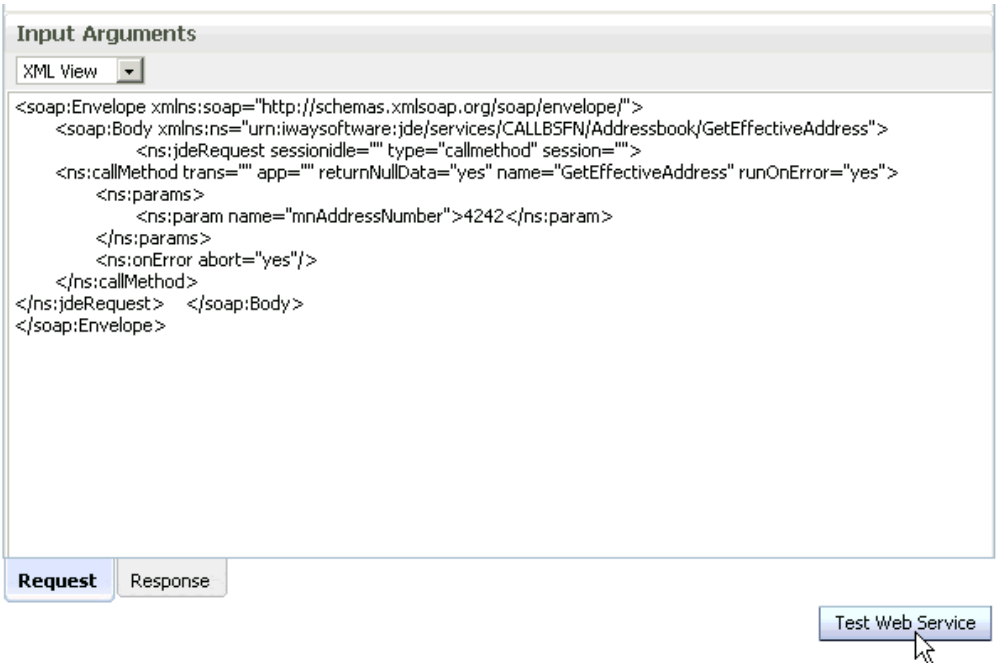


3. Select an available project (for example, JDE_Outbound_GetEffectiveAddress_Mediator).

4. Click **Test** in the right pane.

   The Test Web Service page is displayed.

5. Click the **Request** tab.
6. Scroll down to the Input Arguments section.



7. Select **XML View** from the list in the upper-left corner.
8. Provide an appropriate input XML document in the Input Arguments area.

   For example:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body
xmlns:ns="urn:iwaysoftware:jde/services/CALLBSFN/Addressbook/GetEffectiveAddres
s">
        <ns:jdeRequest sessionidle="" type="callmethod" session="">
<ns:callMethod trans="" app="" returnNullData="yes" name="GetEffectiveAddress"
runOnError="yes">
<ns:params>
<ns:param name="mnAddressNumber">4242</ns:param>
</ns:params>
<ns:onError abort="yes"/>
</ns:callMethod>
</ns:jdeRequest>    </soap:Body>
</soap:Envelope>
```

9. Click **Test Web Service**.

   The output response is received in the Response tab of the Oracle Enterprise Manager console.

5.3 Configuring a Mediator Inbound Process

The following example describes how to configure a Mediator inbound process to your J.D. Edwards OneWorld system, using a Mediator project in Oracle JDeveloper.

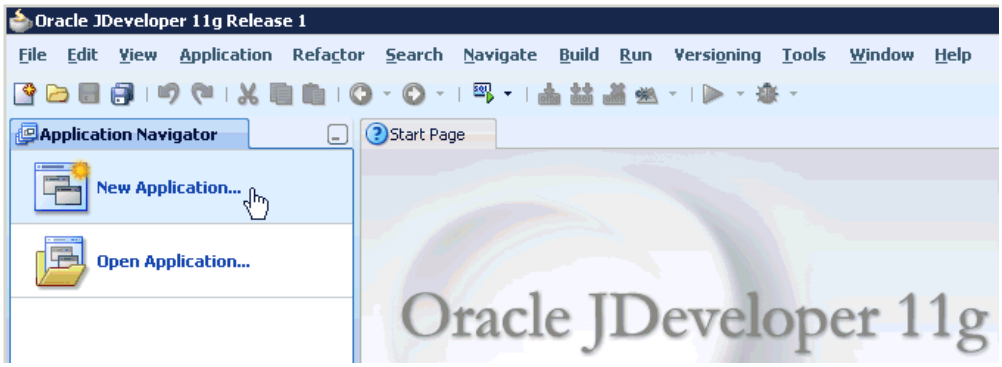
Prerequisites

Before you design a Mediator inbound process, you must generate the respective WSDL file using Application Explorer. See "Generating WSDL for Event Integration" on page 4-34 for more information.

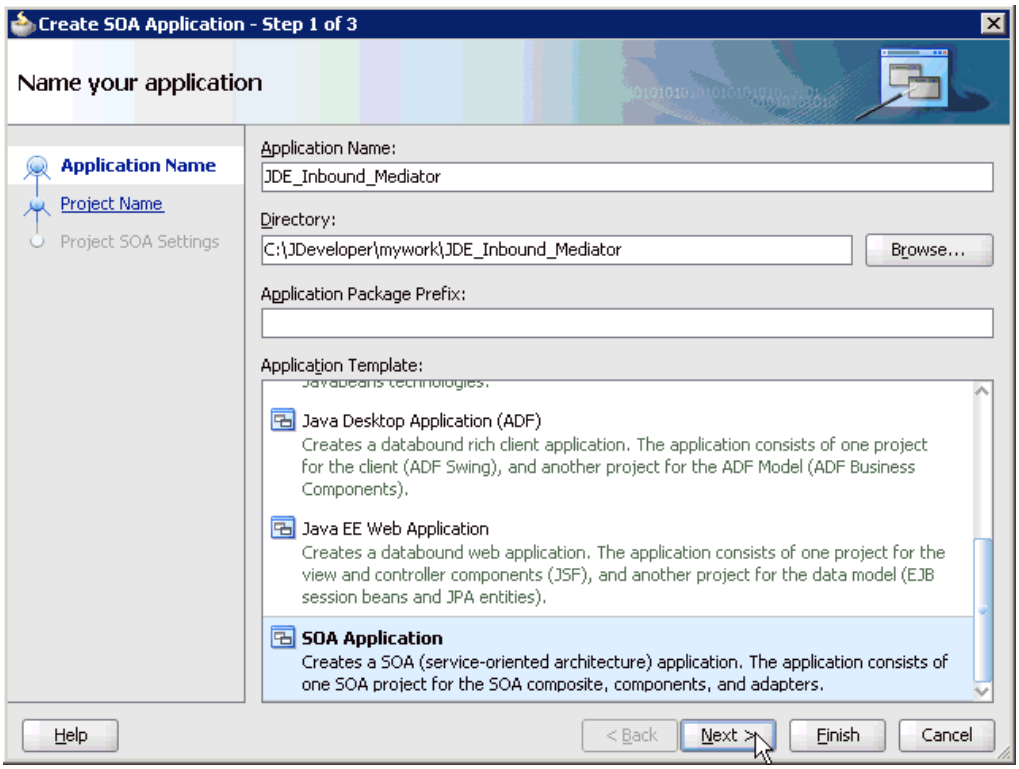
5.3.1 Creating a New SOA Application for the Inbound Mediator Process

Perform the following steps to create a new SOA application for the inbound Mediator process:

1. Open Oracle JDeveloper on your system.
2. In the Application Navigator tab, click **New Application**.

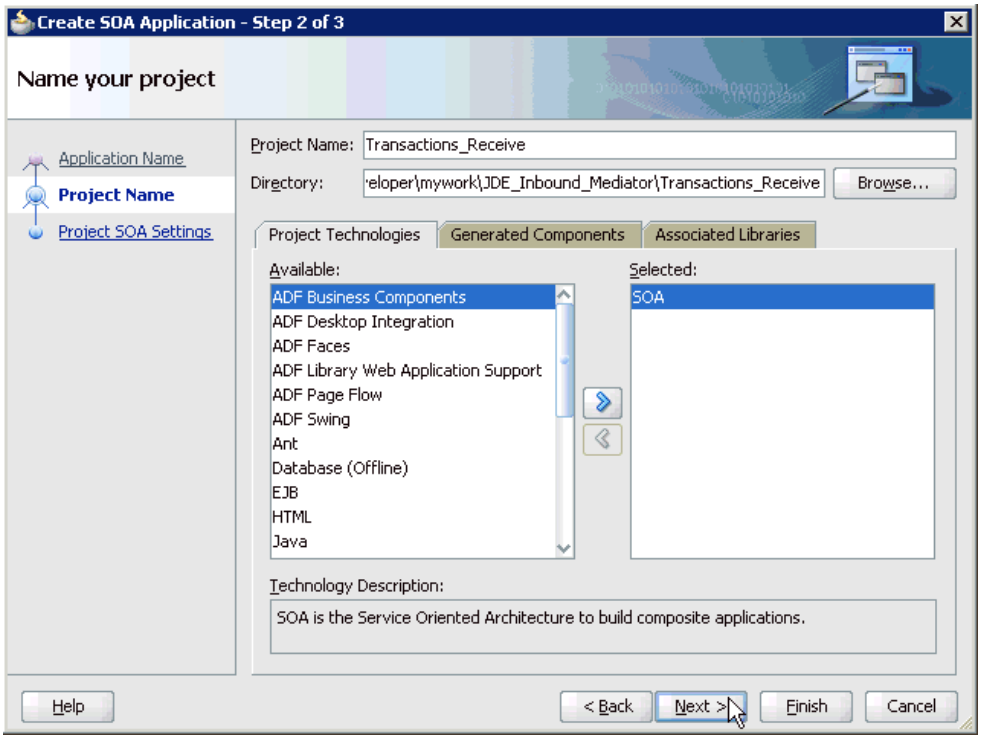


The Create SOA Application wizard is displayed.



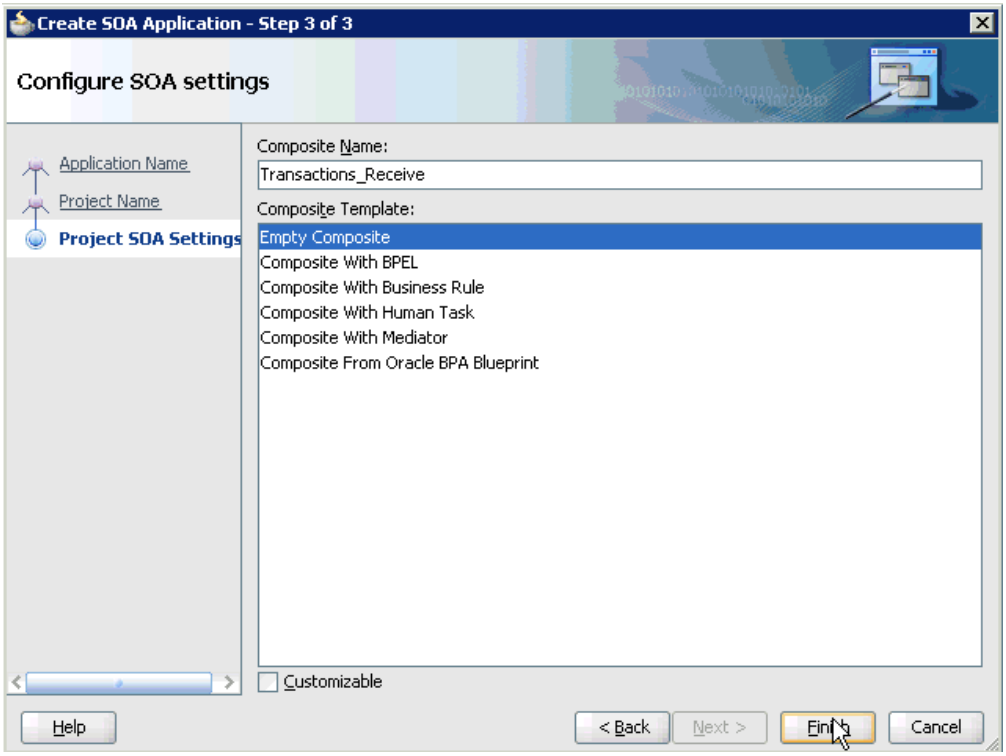
3. From the Application Template list, click **SOA Application**.
4. Enter name for the new SOA application (for example, JDE_Inbound_Mediator) and click **Next**.

   The Name your project page is displayed.

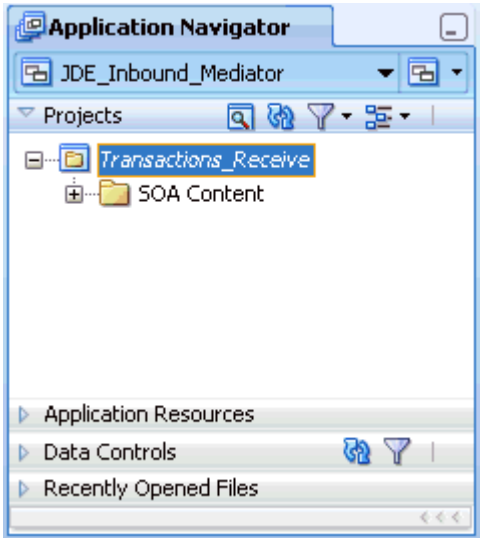


5. Enter a project name (for example, Transactions_Receive) and click **Next**.

The Configure SOA settings page is displayed.



6. From the Composite Template list, select **Empty Composite** and click **Finish**.



The new SOA application (JDE_Inbound_Mediator) and associated project (Transactions_Receive) are added to the Application Navigator tab in the left pane.

5.3.2 Defining a Mediator Inbound Process

This section describes how to define a Mediator inbound process, which consists of the following stages:

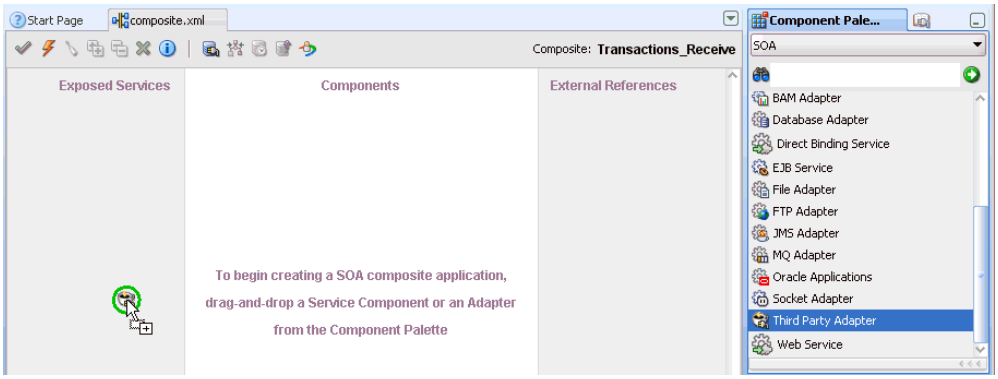
1. Configuring a Third Party Adapter Service Component

2. Configuring an Inbound Mediator Process Component With a File Adapter
3. Configuring the Routing Rules

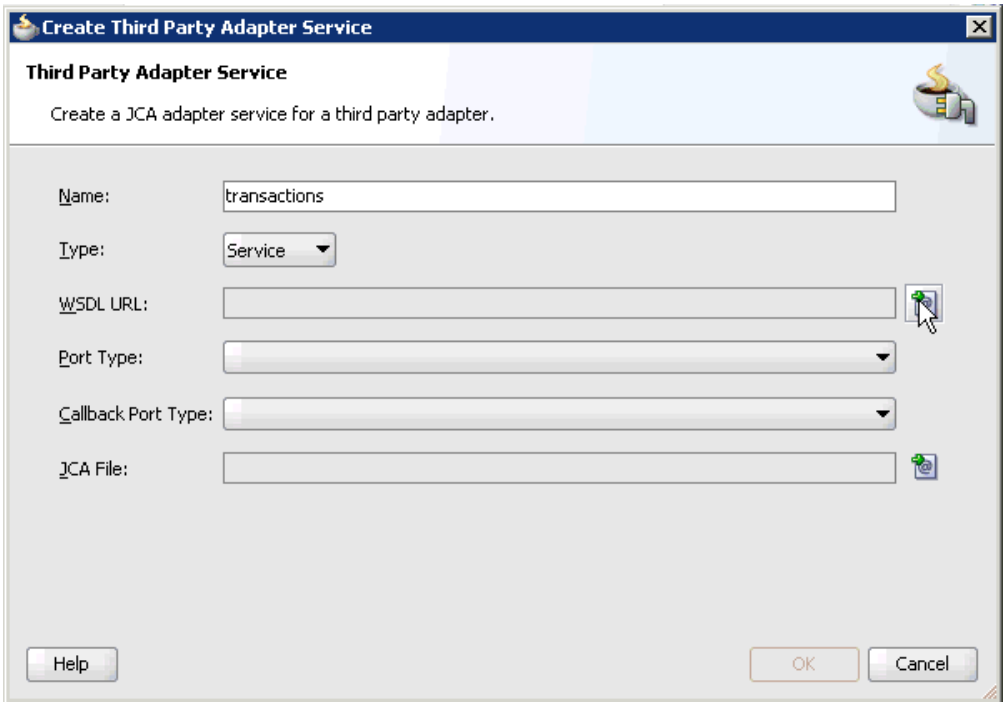
Configuring a Third Party Adapter Service Component

Perform the following steps to create a third party adapter service component:

1. Drag and drop the **Third Party Adapter** component from the Component Palette tab (Service Adapters section) to the Exposed Services pane.

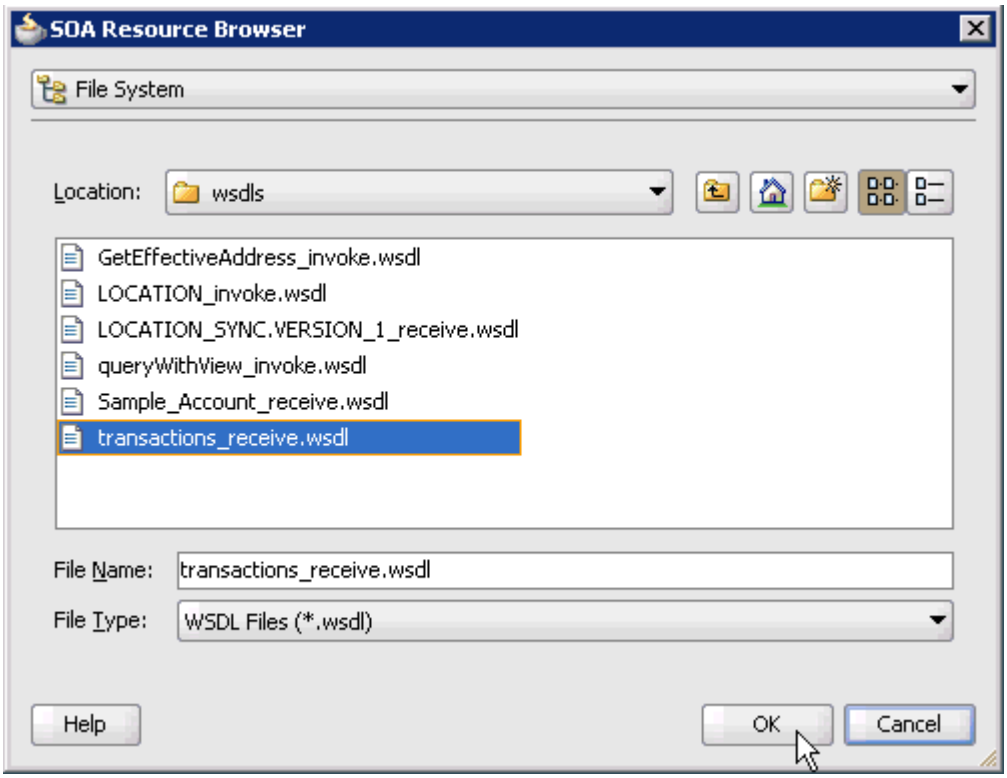


The Create Third Party Adapter Service dialog is displayed.



2. Enter a name for the third party adapter service.
3. Ensure that **Service** is selected from the Type list (default).
4. Click the **Find existing WSDLs** icon, which is located to the right of the WSDL URL field.

   The SOA Resource Browser dialog is displayed.

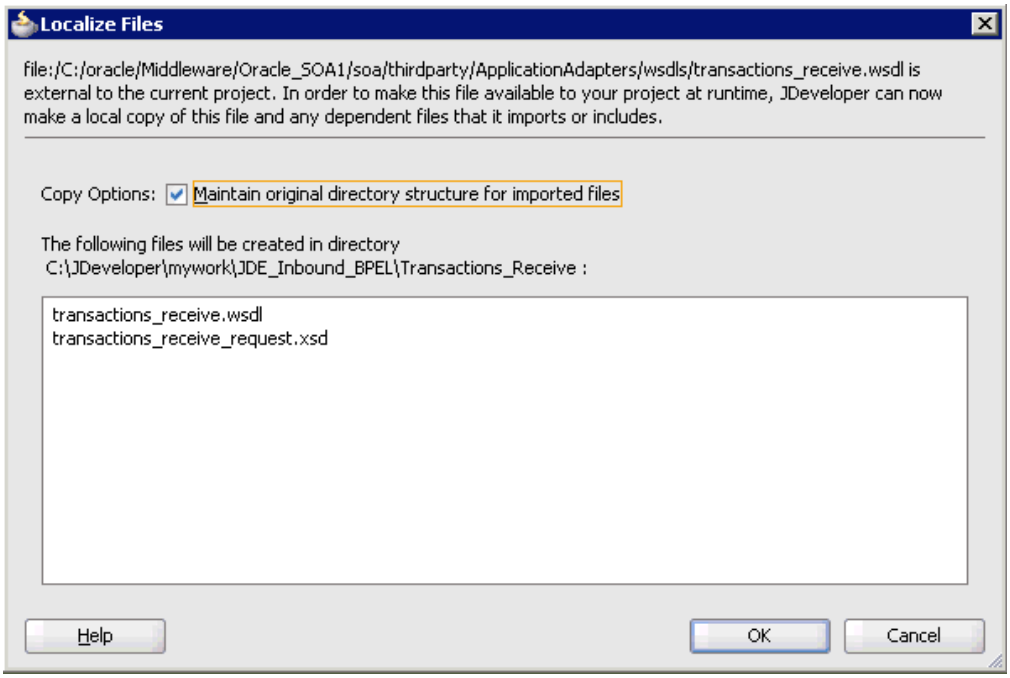


5. Browse and select an inbound WSDL file from the following directory:

   ```
   C:\oracle\Middleware\home_GA\Oracle_
   SOA1\soa\thirdparty\ApplicationAdapters\wsdls
   ```

6. Click **OK**.

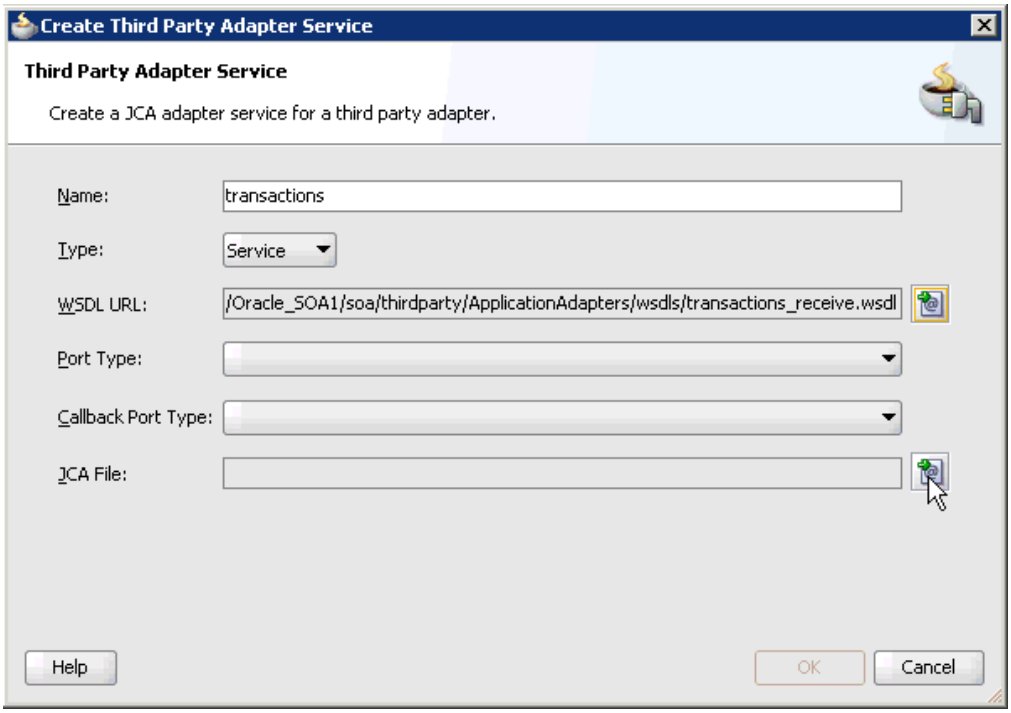
   The Localize Files dialog is displayed.



7. Click **OK**.

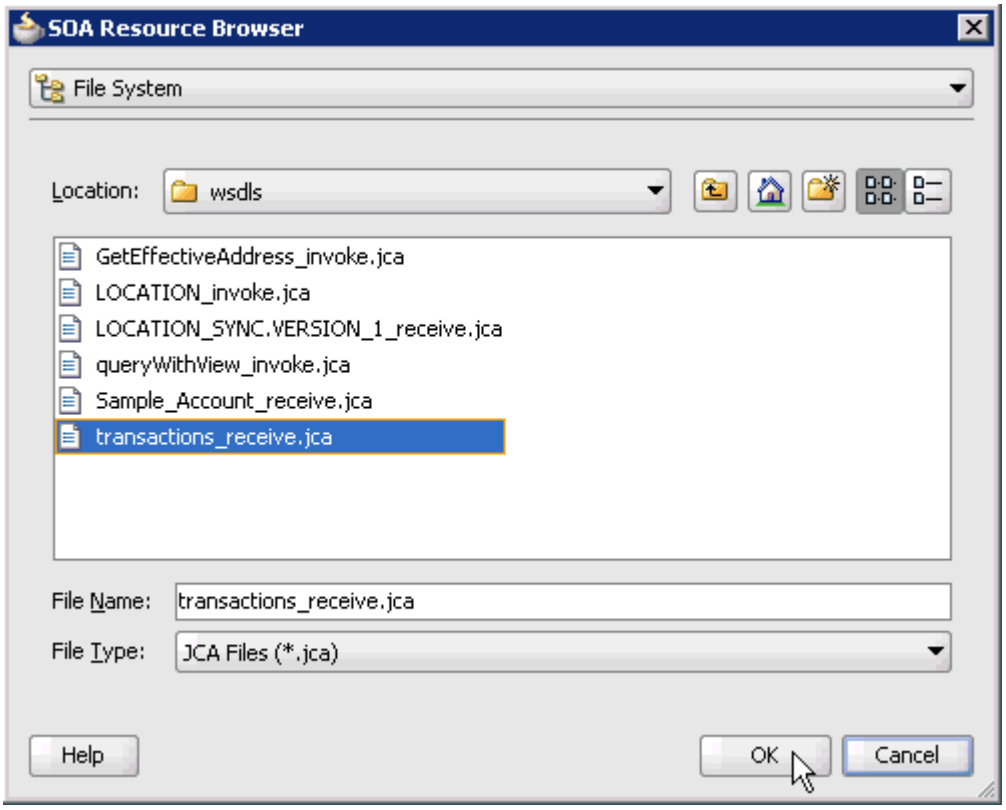
   The inbound WSDL file and associated receive/request XML schema file (.xsd) are imported to the project folder that has been created.

   You are returned to the Create Third Party Adapter Service dialog.



8. Click the **Find JCA Files** icon, which is located to the right of the JCA File field.

   The SOA Resource Browser dialog is displayed.

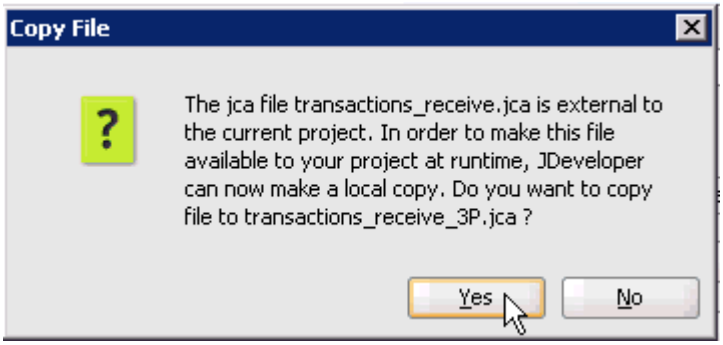


9. Browse and select the JCA properties file from the following directory:

   ```
   C:\oracle\Middleware\home_GA\Oracle_
   SOA1\soa\thirdparty\ApplicationAdapters\wsdls
   ```

10. Click **OK**.

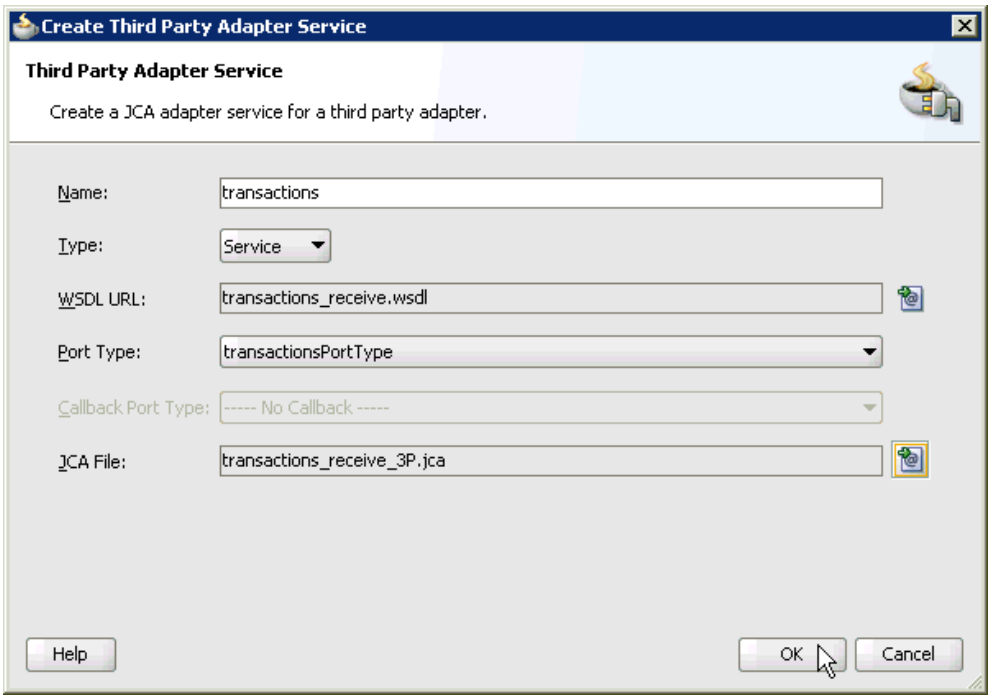
    The following message is displayed.



11. Click **Yes**.

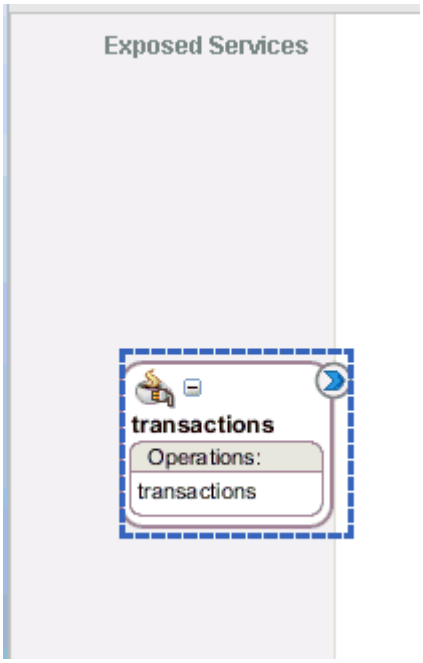
    A copy of the JCA properties file is made in the project folder.

    You are returned to the Create Third Party Adapter Service dialog.



12. Click **OK**.

    The third party adapter service component (transactions) is created in the Exposed Services pane, as shown in the following image.

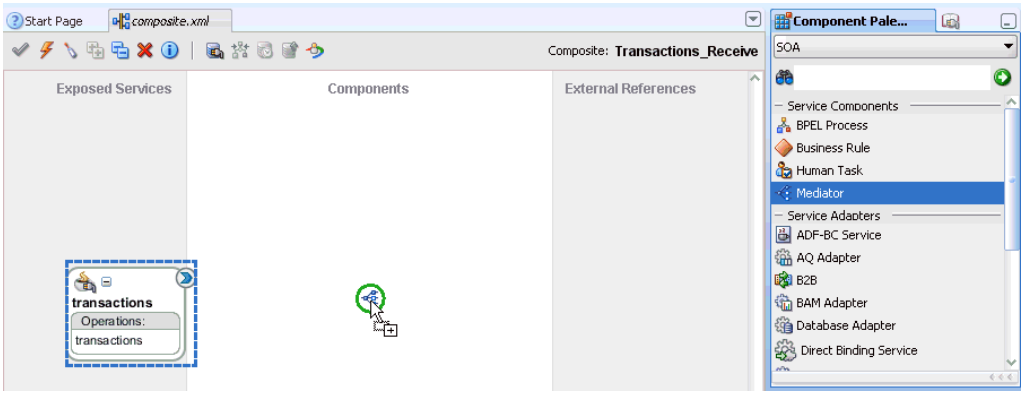


You are now ready to configure an inbound Mediator process component.

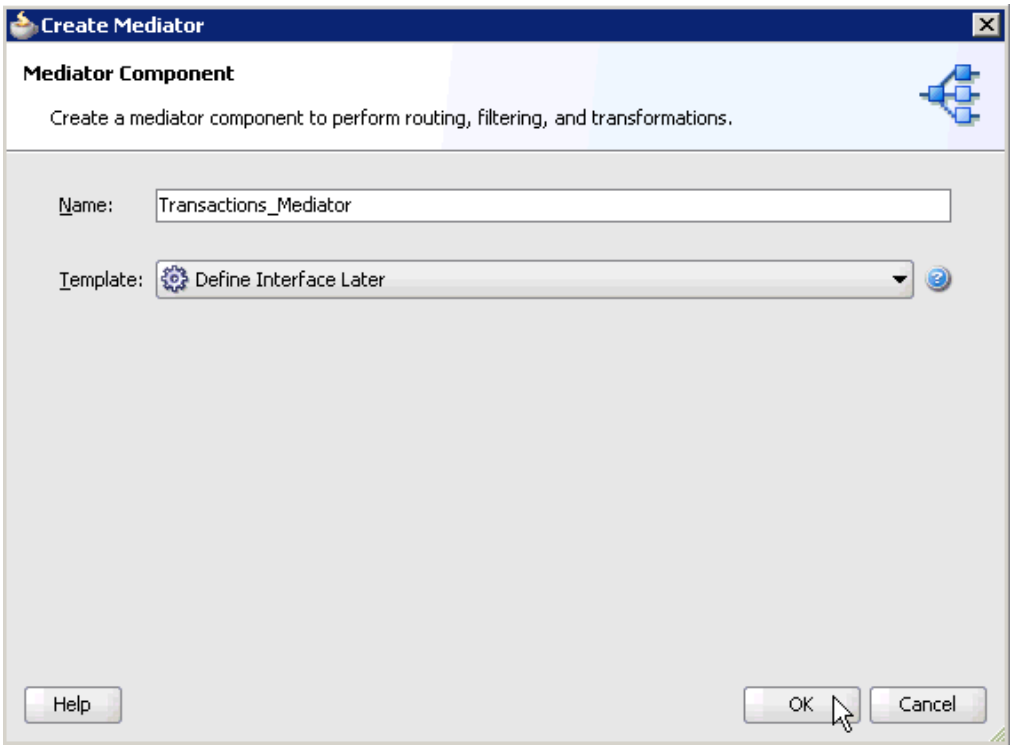
Configuring an Inbound Mediator Process Component With a File Adapter

Perform the following steps to configure an inbound Mediator process component with a File adapter.

1. Drag and drop the **Mediator** component from the Component Palette tab (Service Components section) to the Components pane.

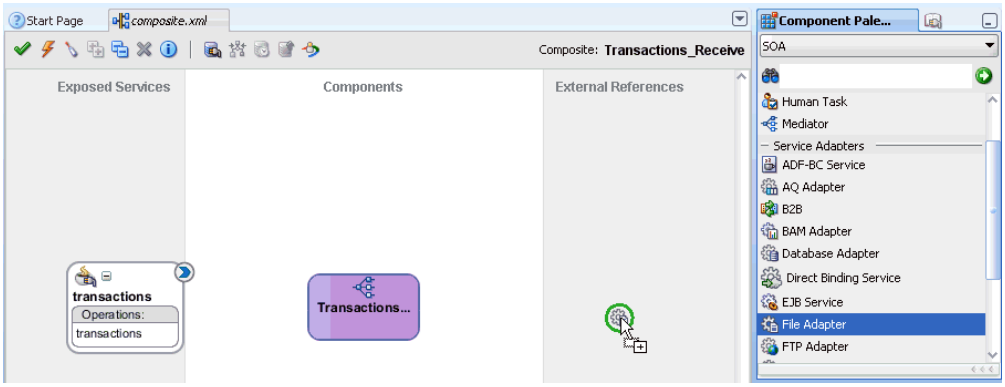


The Create Mediator dialog is displayed.



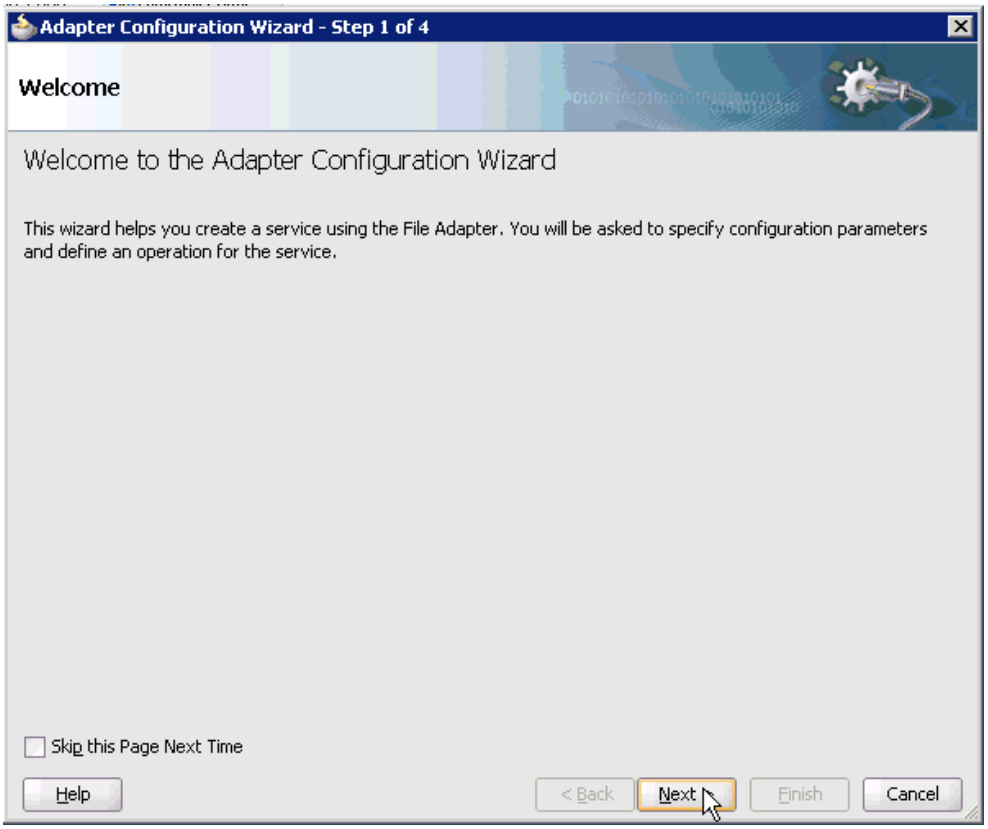
2. In the Name field, enter a name to identify the new inbound Mediator process component (for example, Transactions_Mediator).
3. From the Template list, select **Define Interface Later**.
4. Click the **OK**.

   The new Mediator process component is added to the Components pane, as shown in the following image.



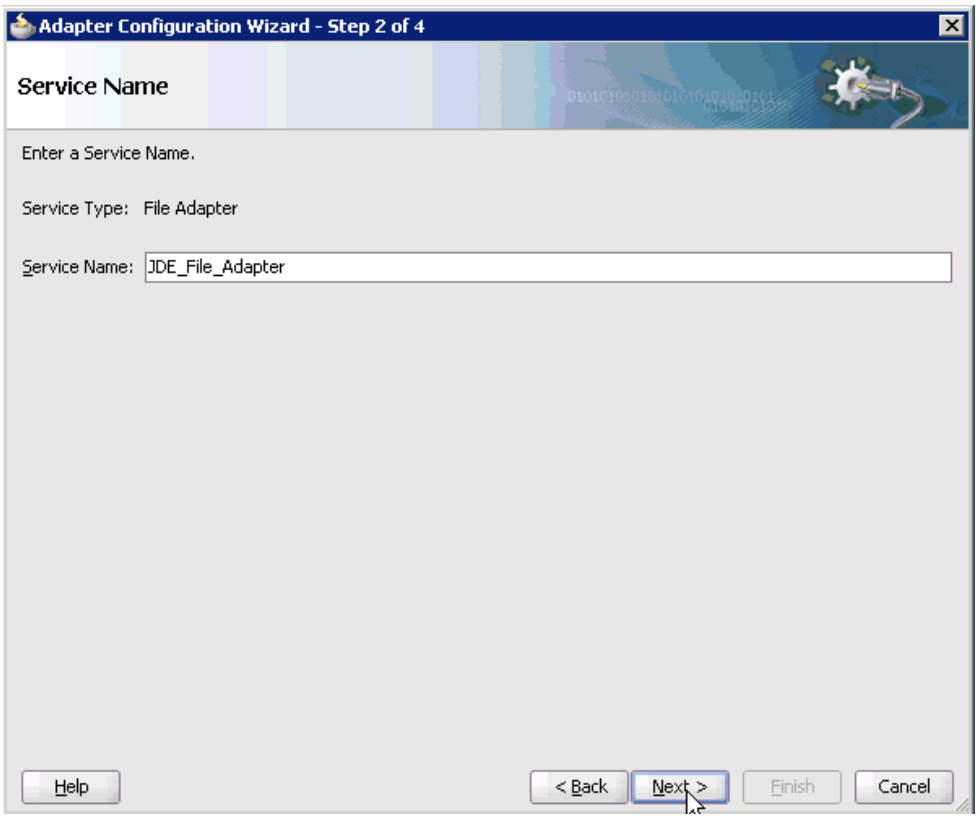
5. Drag and drop the **File Adapter** component from the Service Adapters pane to the External References pane.

   The Adapter Configuration Wizard is displayed, showing the Welcome page.

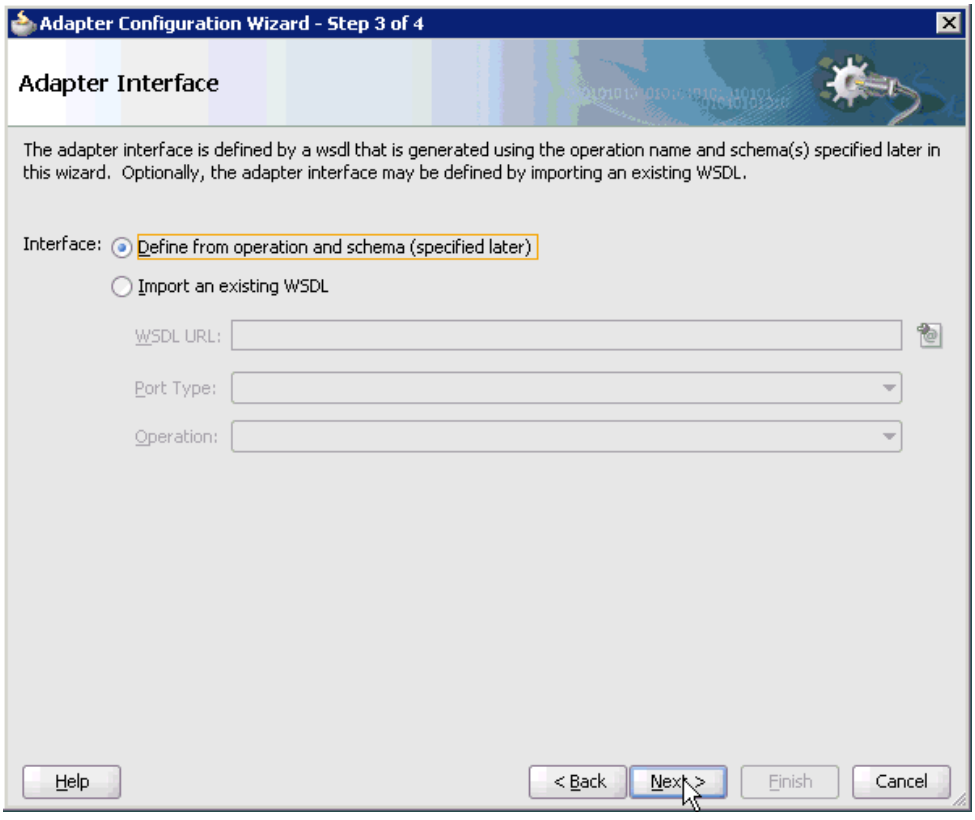


6. Click **Next**.

   The Service Name page is displayed.

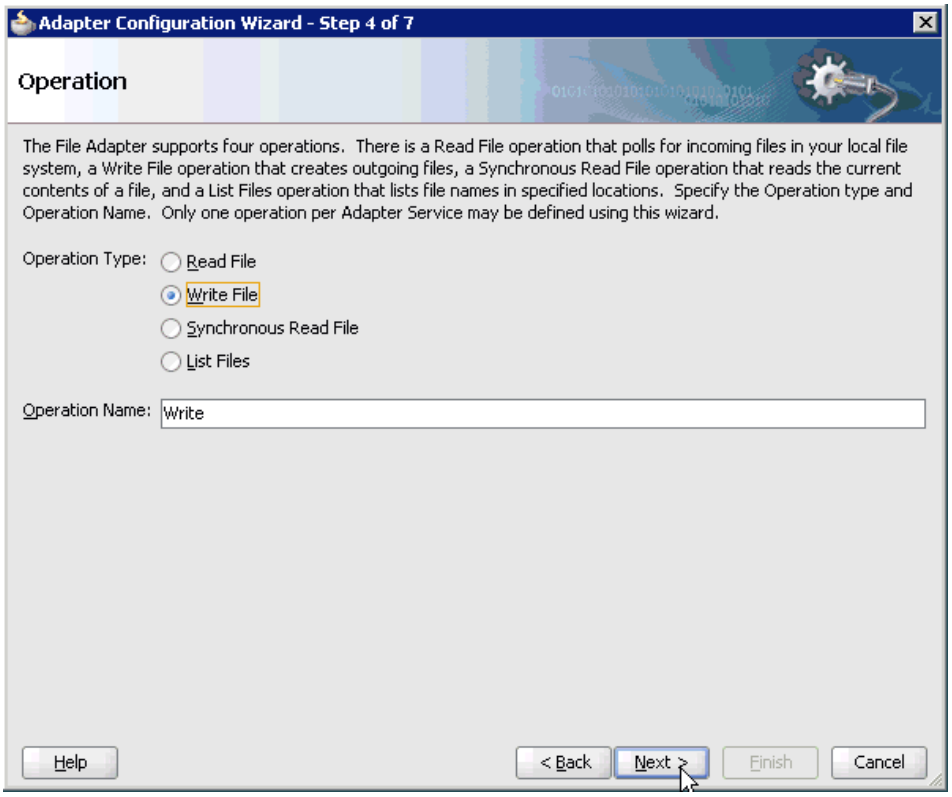


7. Type a name for the new File adapter in the Service Name field and click **Next**. The Adapter Interface page is displayed.



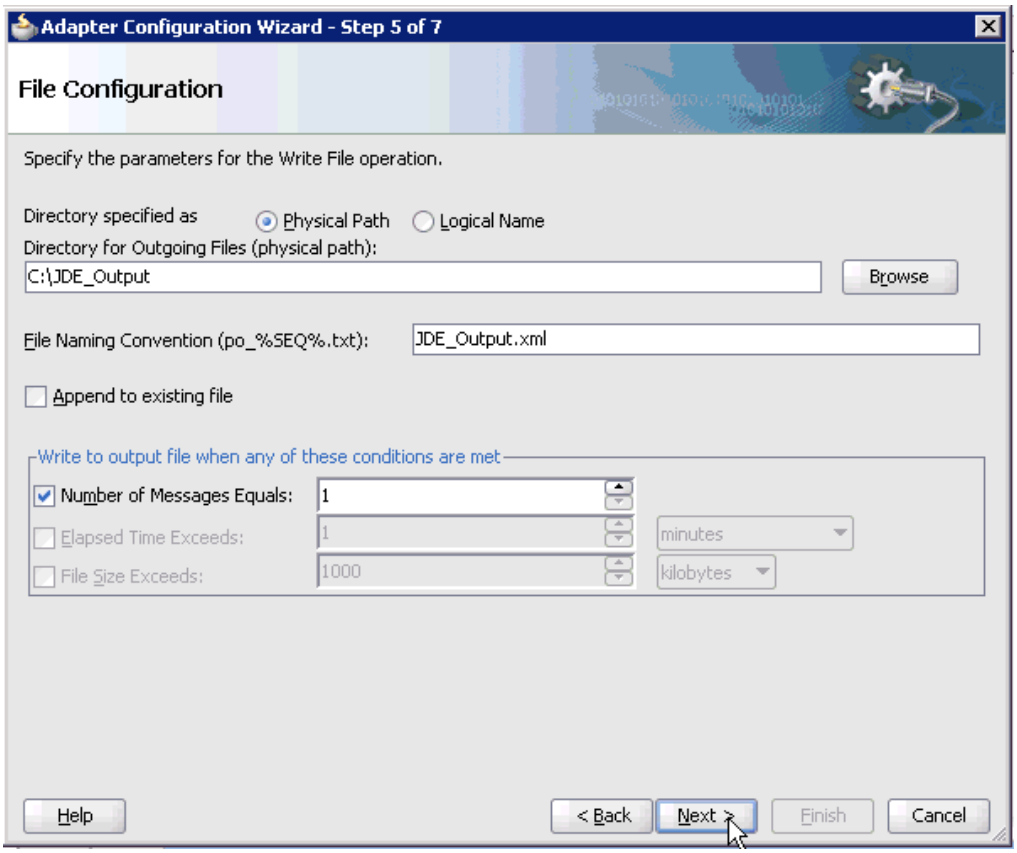
8. Ensure that the **Define from operation and schema (specified later)** option is selected.
9. Click **Next**.

   The Operation page is displayed.



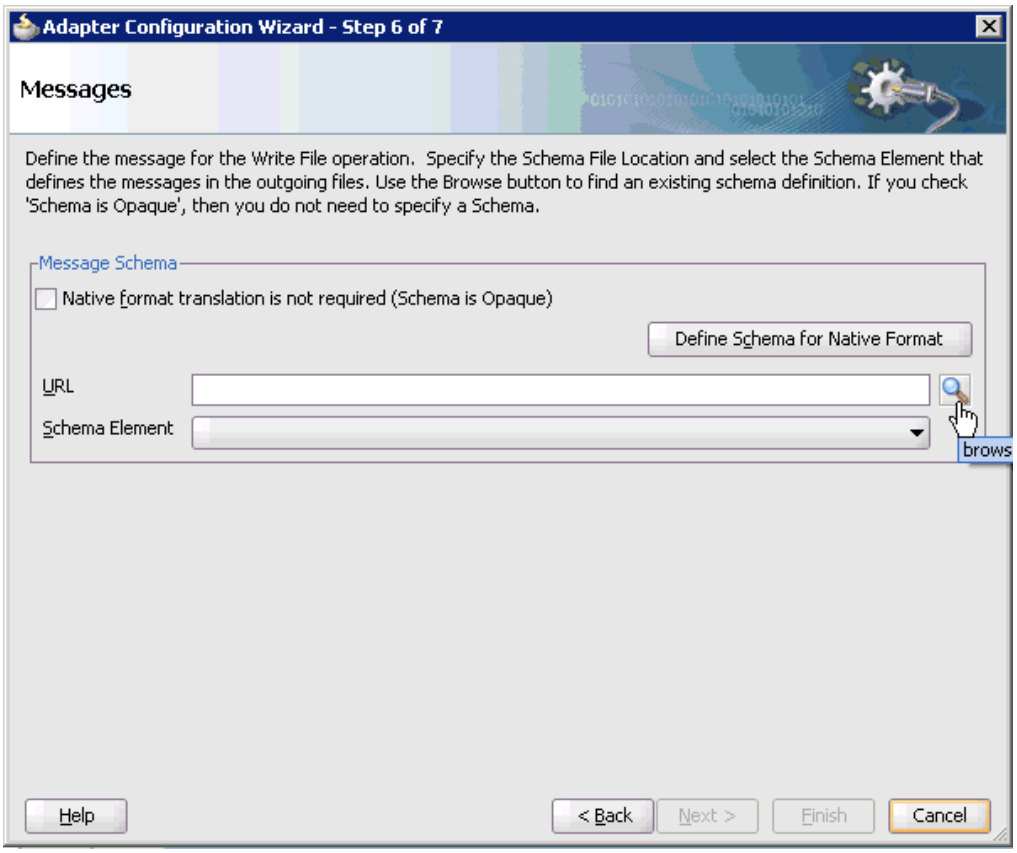
10. Select **Write File** from the list of Operation Type options and specify an Operation Name (for example, Write).
11. Click **Next**.

    The File Configuration page is displayed.



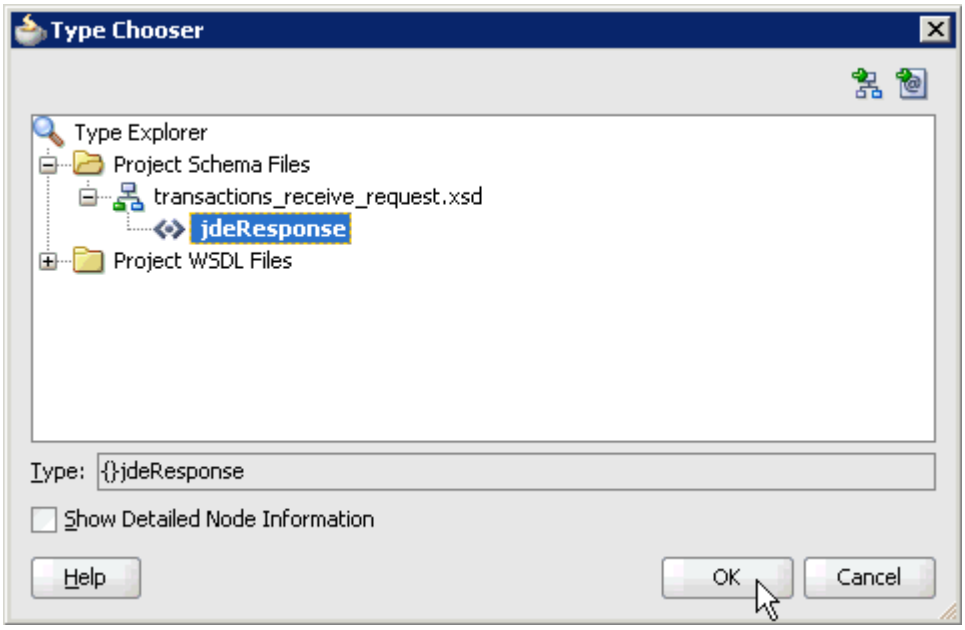
12. Specify a location on your file system where the output file is written.
13. In the File Naming Convention field, specify a name for the output file.
14. Click **Next**.

    The Messages page is displayed.



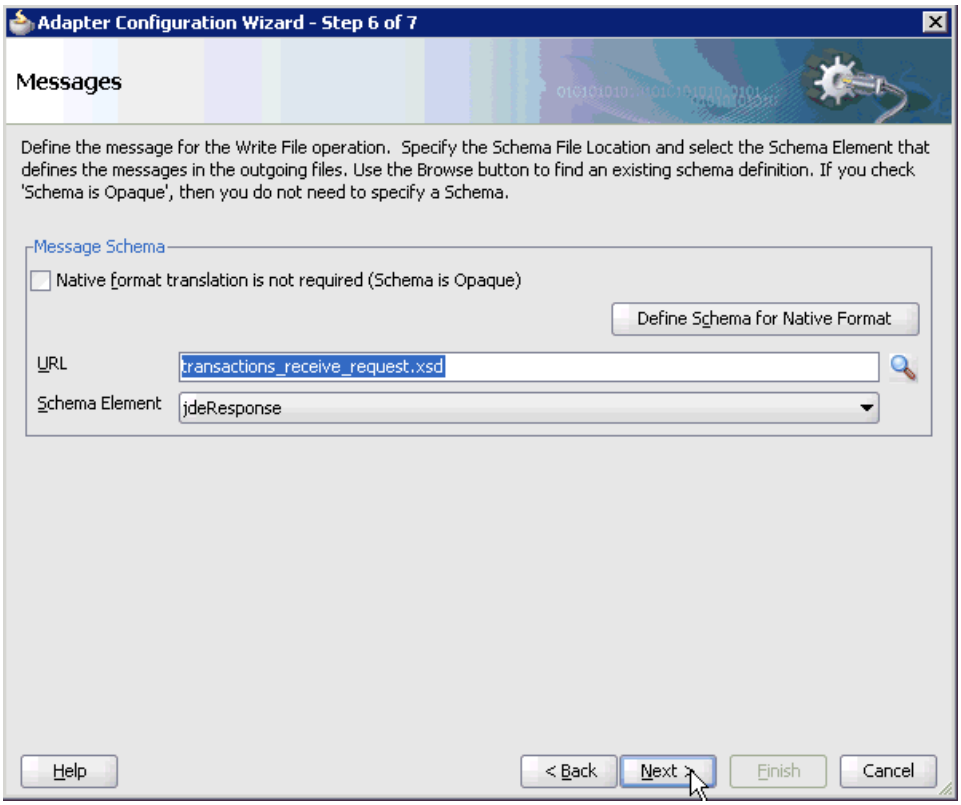
15. Click **browse for schema file**, which is located to the right of the URL field.

    The Type Chooser dialog is displayed.



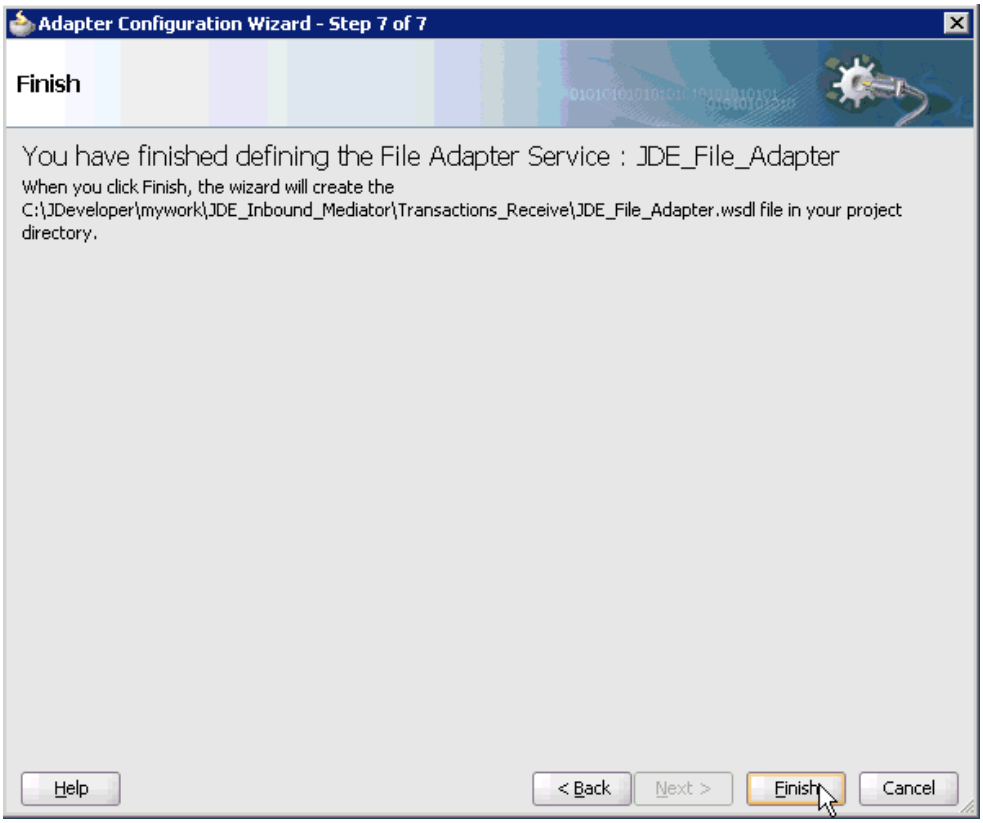
16. Select the available schema.
17. Click **OK**.

    You are returned to the Messages page.

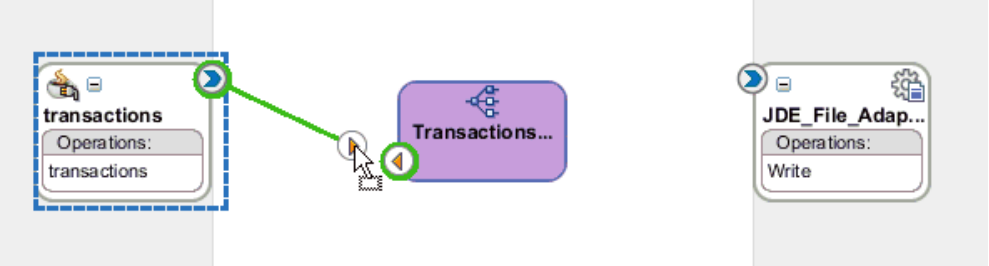


18. Click **Next**.

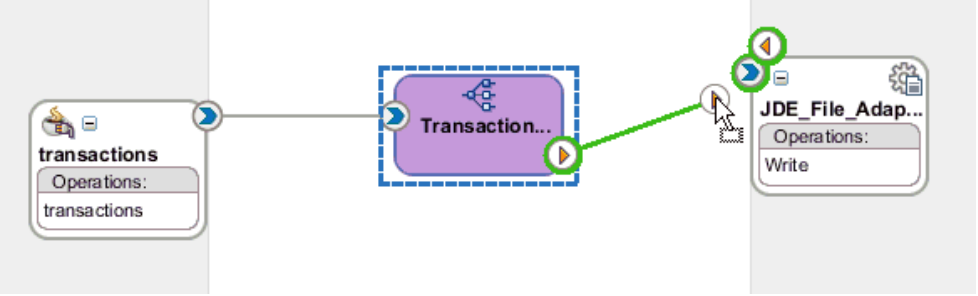
The Finish page is displayed.



19. Click **Finish**.
20. Create a connection between the inbound Mediator process component and the third party adapter service component.



21. Create a connection between the inbound Mediator process component and the File adapter component.



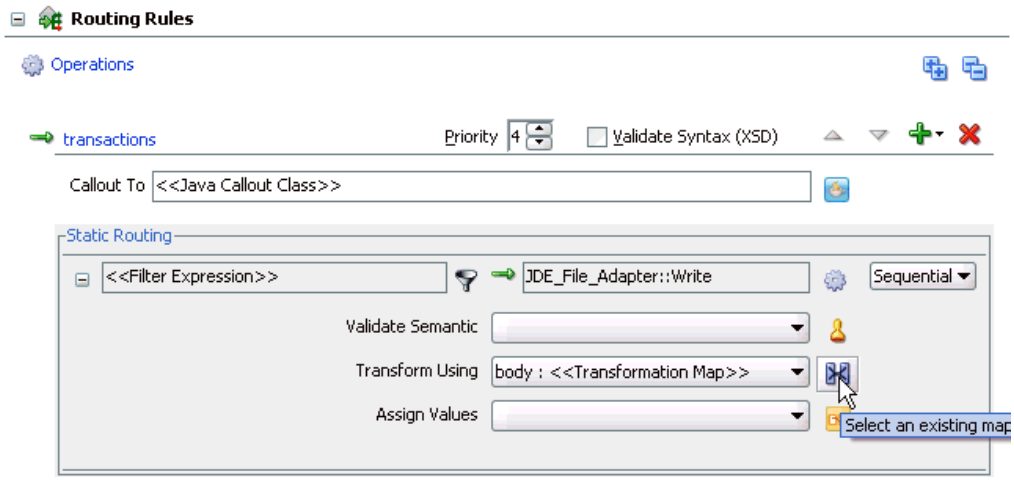
You are now ready to configure the routing rules.

Configuring the Routing Rules

Perform the following steps to configure routing rules for the Mediator inbound process component:

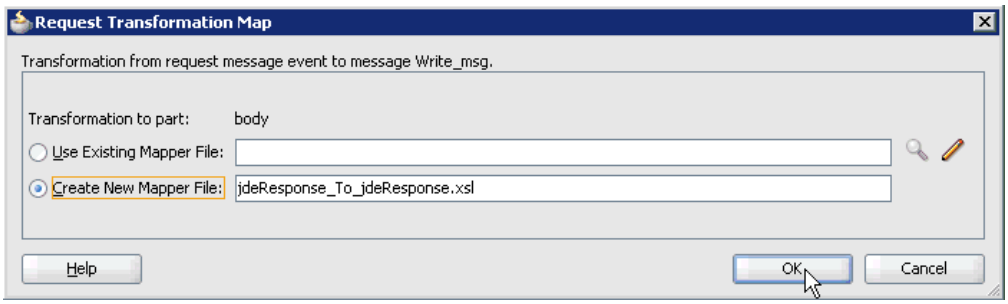
1. Double-click the inbound Mediator process component in the Components pane.

The Routing Rules dialog is displayed.



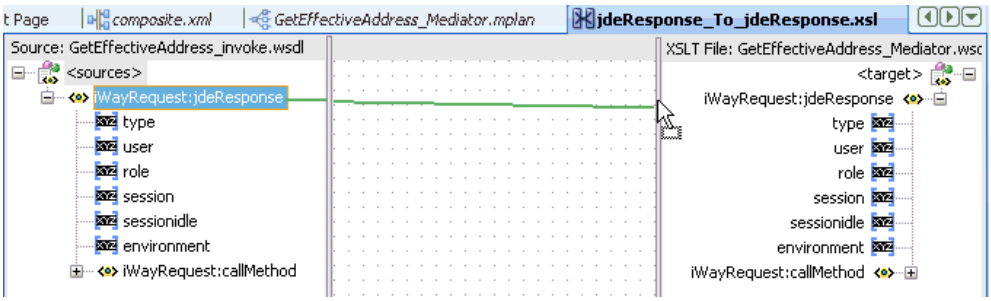
2. In the <<Filter Expression>> area, click the **Select an existing mapper file or create a new one** icon to the right of the Transform Using field.

   The Request Transformation Map dialog is displayed.



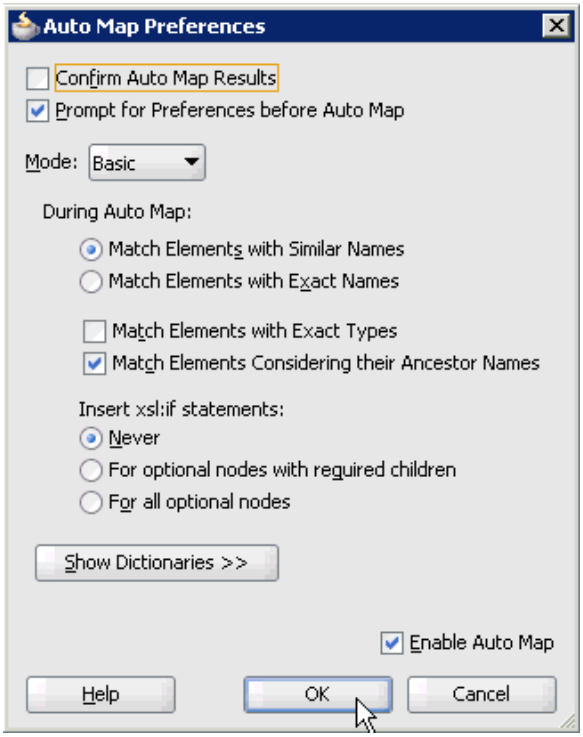
3. Select the **Create New Mapper File** option and click **OK**.

   The jdeResponse_To_jdeResponse.xsl tab is displayed.



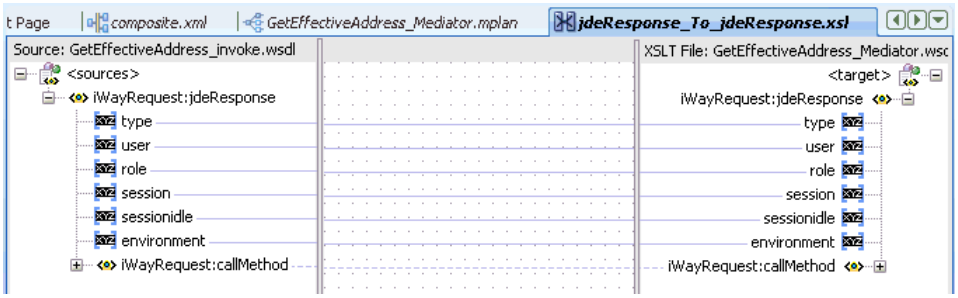
4. Map the **iWayRequest:jdeResponse** source element to the **iWayRequest:jdeResponse** target element.

   The Auto Map Preferences dialog is displayed.

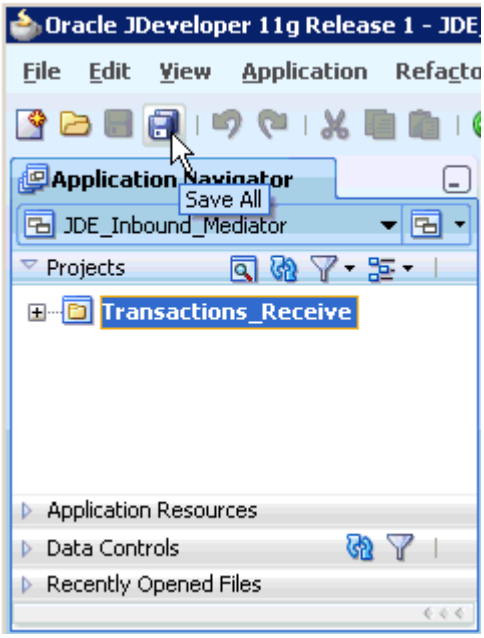


5. Retain the default values and click **OK**.

   The mapping is complete, as shown in the following image.



6. Click the **Save All** icon in the menu bar to save the new inbound Mediator process component that was configured.



You are now ready to deploy the Mediator inbound process. You can follow the same procedure that is described in "Deploying the BPEL Inbound Process" on page 4-50.

Once event messages are triggered through J.D. Edwards OneWorld, output XML is received in the location that was specified for the File adapter component. For more information on triggering events in J.D. Edwards OneWorld, see "Triggering an Event in J.D. Edwards OneWorld" on page 4-53.

6

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 Oracle Adapter J2CA or with a Oracle Adapter Business Services Engine (BSE) configuration.

6.1 Troubleshooting

This topic provides troubleshooting information for J.D. Edwards OneWorld, separated into four categories:

- Oracle Adapter Application Explorer (Application Explorer)
- J.D. Edwards OneWorld
- Oracle Adapter J2CA
- Oracle Adapter Business Services Engine (BSE)

Note: Log file information that can be relevant in troubleshooting can be found in the following locations:

- The Oracle Adapter J2CA trace information can be found under the following directory:

  ```
  C:\oracle\Middleware\Oracle_
  SOA1\soa\thirdparty\ApplicationAdapters\config\config_name\log
  ```

- BSE trace information can be found under the following directory:

  ```
  C:\oracle\Middleware\ user_projects\domains\base_
  domain\servers\soa_server1\stage\ibse\ibse.war\ibselogs
  ```

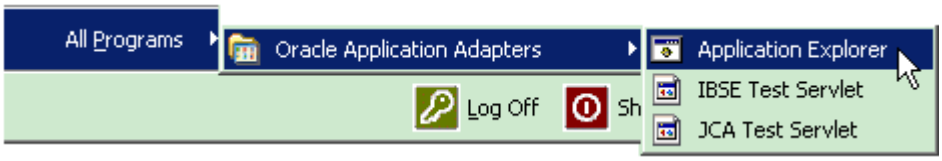
- The log file for Application Explorer can be found under the following directory:

  ```
  C:\oracle\Middleware\Oracle_
  SOA1\soa\thirdparty\ApplicationAdapters\tools\iwae\bin
  ```

Application Explorer

To use Application Explorer on **Windows** for debugging or testing purposes:

1. Ensure that Oracle WebLogic Server is started, which is where Application Explorer is deployed.
2. Start Application Explorer by clicking the Windows **Start** menu, selecting **All Programs**, **Oracle Application Adapters**, and clicking **Application Explorer**.



You can also start Application Explorer by executing the **ae.bat** file, which is located in the following directory:

```
C:\oracle\Middleware\home_0309\Oracle_
SOA1\soa\thirdparty\ApplicationAdapters\tools\iwae\bin\ae.bat
```

It is a good practice to create a shortcut for the **ae.bat** file on your desktop.

If you are using a UNIX or Linux platform you can start Application Explorer by executing the **iwae.sh** file.

Error	Solution
Cannot connect to Oracle Application Adapter for J.D. Edwards OneWorld from Application Explorer: `Problem activating adapter. (Failed to connect to J.D.Edwards OneWorld, check system availability and 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 run-time.	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 are 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`

Error	Solution
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 system.

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.

Error	Cause	Solution
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`.

Oracle Adapter J2CA

Error	Solution
In Application Explorer, the following error message appears when you attempt to connect to an Oracle 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: `C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\tools\iwae\bin\..\..\..\`

6.2 BSE Error Messages

This topic discusses the different types of errors that can occur when processing Web services through BSE.

6.2.1 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 run-time, various conditions can cause errors in BSE when Web services that use adapters run. 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.

6.2.2 Adapter-Specific Error Handling

When an adapter raises an exception during run-time, 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 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 WebLogic 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 WebLogic Server Adapter Request

If Oracle WebLogic 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 WebLogic 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
&amp;apos;DV7333&amp;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 WebLogic Server Adapter Request

If Oracle WebLogic 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 WebLogic 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)
```

7

Advanced User Tools

This chapter includes the following topics:

7.1 Web Services Policy-Based Security

Oracle 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. Easy application integration is enabled, but the issue of controlling the use and implementation of critical and sensitive business logic that is run as a Web service is raised.

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 run-time 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 are not governed by it. However, if a policy is applied to the Business Service, all methods are governed by it. At run-time, the user ID and password that are sent to Oracle 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.

7.1.1 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

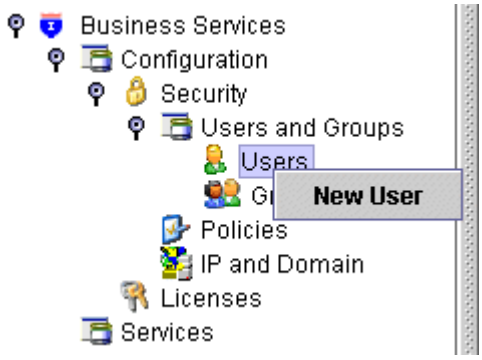
Before you create instances of policies, you must have a minimum of one user or one group to associate to an instance. You can create users and groups using Application Explorer.

1. Start Application Explorer.
2. Right-click the configuration to which you want to connect, for example, **newtest**. See Chapter 2, "Configuring Oracle Application Adapter for J.D. Edwards OneWorld" for information on creating a new configuration.
3. Select **Connect**.

   Nodes appear for Adapters and Business Services (also known as Web services).

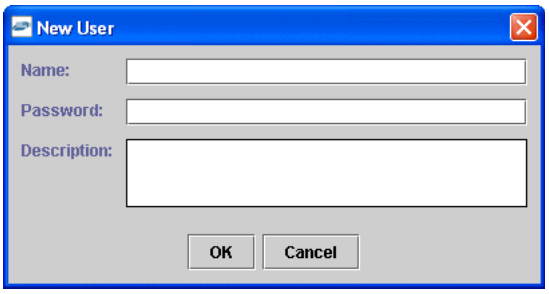
   Perform the following steps:

   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 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 Oracle Application Adapter for J.D. Edwards OneWorld" for information on creating a new configuration.

3. Select **Connect**.

Nodes appear for Adapters and Business Services (also known as Web services).

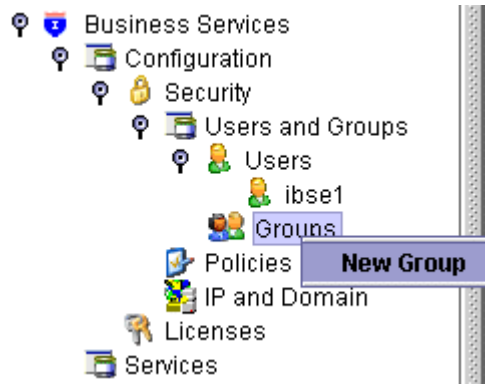
Perform the following steps:

a. Expand the **Business Services** node.

b. Expand the **Configuration** node.

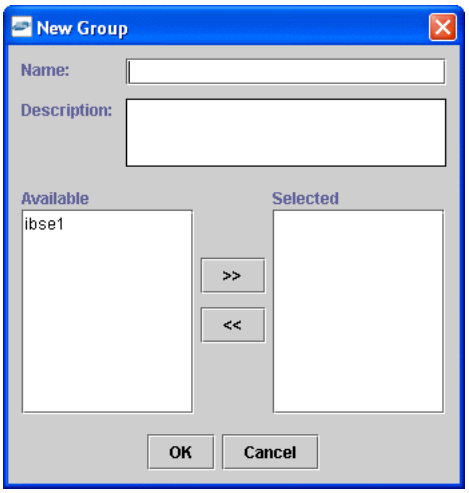
c. Expand the **Security** node.

d. Expand the **Users and Groups** node.



4. Right-click **Groups** and select **New Group**.

   The New Group dialog 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 Oracle Application Adapter for J.D. Edwards OneWorld" for information on creating a new configuration.
3. Select **Connect**.

Nodes appear for Adapters and Business Services (also known as Web services).

Perform the following steps:

a. Expand the **Business Services**node.

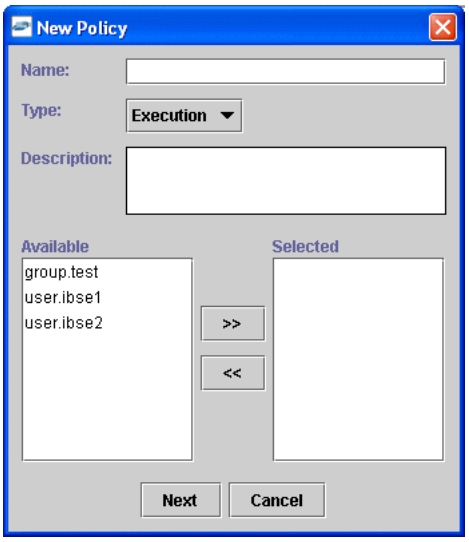
b. Expand the **Configuration** node.

c. Expand the **Security** node.

d. Expand the **Policies** node.

4. Right-click **Policies** and select **New Policy**.

The New Policy dialog 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 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 Oracle Application Adapter for J.D. Edwards OneWorld" for information on creating a new configuration.
3. Select **Connect**.

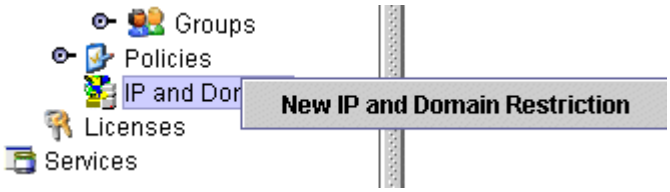
   Nodes appear for Adapters and Business Services (also known as Web services).

   Perform the following steps:

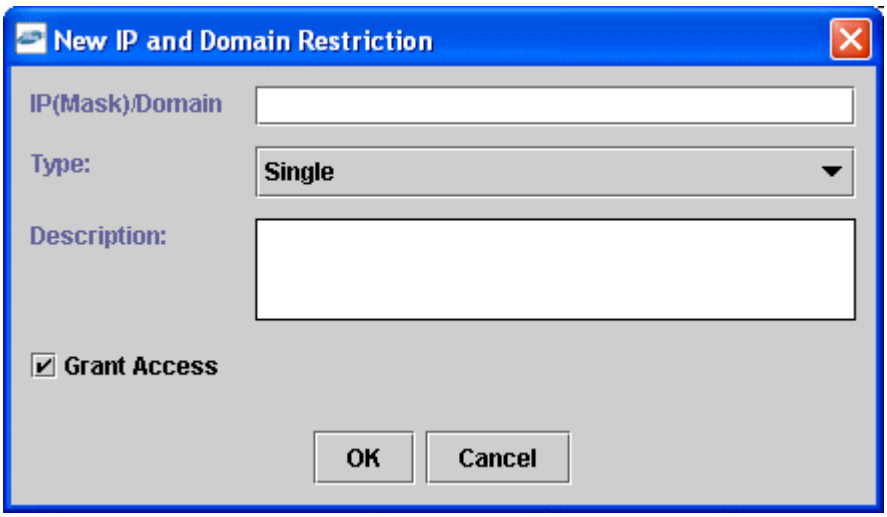
   a. Expand the **Business Services** node.

   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 is displayed.



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**

7.2 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 run-time. 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

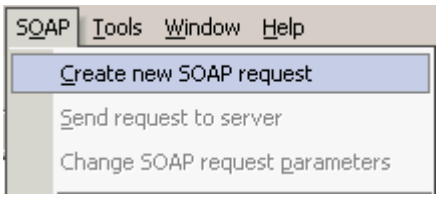
To migrate a BSE repository:

1. Copy the BSE control service URL, for example:

   ```
   http://localhost:8001/ibse/IBSEServlet/admin/iwcontrol.ibs
   ```

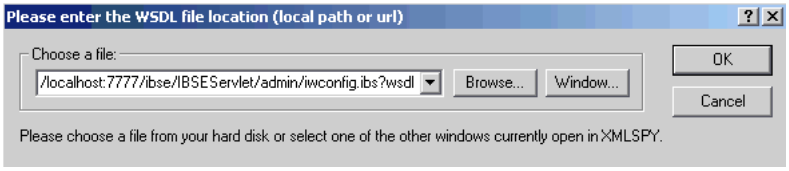
2. Open a third party XML editor, for example, XMLSPY.
3. From the menu bar, click **SOAP**.

   A list of options appears.



4. Select **Create new SOAP request**.

   The WSDL file location dialog is displayed.



   Perform the following steps:

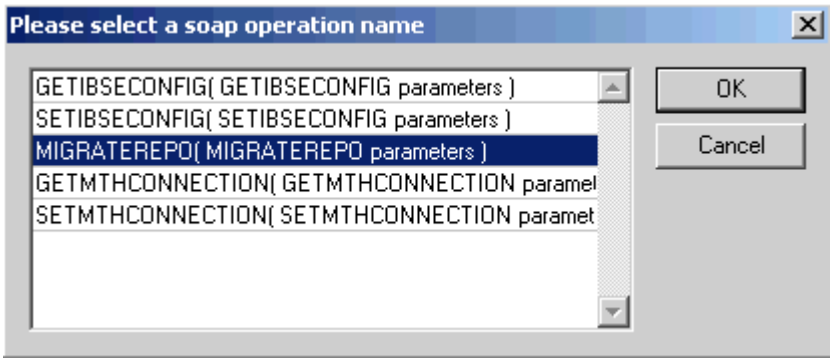
a. In the **Choose a file** field, paste the BSE control service URL.

b. Append **?wsdl** to the URL, for example:

```
http://localhost:8001/ibse/IBSEServlet/admin/iwcontrol.ibs?wsdl
```

5. Click **OK**.

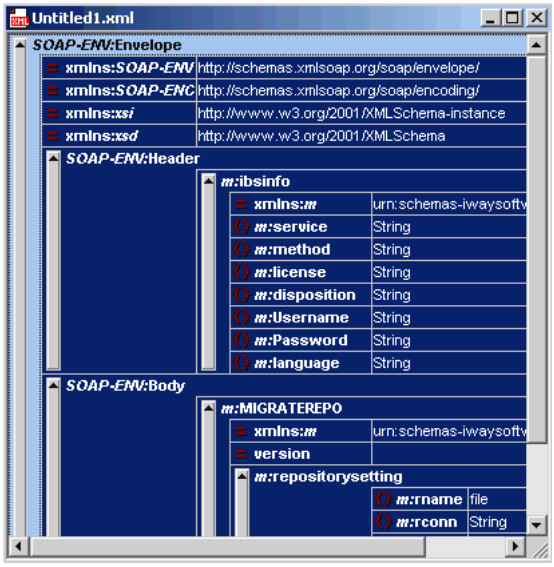
   The soap operation name dialog is displayed and lists the available control methods.



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.



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:

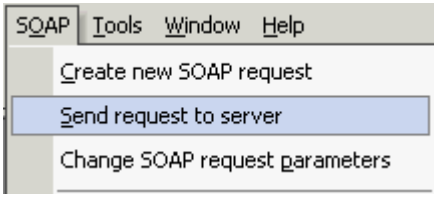
      ```
      <m:servicename>JDEService1</m:servicename>
      ```

    - 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:

   `C:\oracle\Middleware\Oracle_SOA1\soa\thirdparty\ApplicationAdapters\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 and Inbound Processing

J.D. Edwards OneWorld enables you to specify inbound functionality for Master Business Functions (MBF).

The following topics describe how to enable outbound and inbound transaction processing in J.D. Edwards OneWorld and how to modify the `jde.ini` file for XML support.

- Modifying the JDE.INI File for Outbound and Inbound Processing
- Using the GenJava Development Tool (Outbound Processing)
- Triggering J.D. Edwards OneWorld Events

A.1 Modifying the JDE.INI File for Outbound and Inbound Processing

This section describes the settings that are required in the JDE.INI file for the XML call object kernel (outbound and inbound processing).

Open the JDE.INI file and modify the **[JDENET_KERNEL_DEF6]** and **[JDENET_KERNEL_DEF15]** sections as follows:

```
[JDENET_KERNEL_DEF6]
krnlName=CALL OBJECT KERNEL
dispatchDLLName=XMLCallObj.dll
dispatchDLLFunction=_XMLTransactionDispatch@28
maxNumberOfProcesses=1
numberOfAutoStartProcesses=1

[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 NT 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

Operating System	Call Object dispatch DLLName	XML Trans dispatch DLLName
HP9000B	libxmlcallojb.sl	libxmltransactions.lo
Sun or RS6000	libxmlcallojb.so	Libxmltransactions.so

Note: The J.D. Edwards installation for version B7333(XE) does not include **[JDENET_KERNEL_DEF15]**. As a result, if you are using version B7333(XE), you must manually add it to the jde.ini file. For all other J.D. Edwards versions, **[JDENET_KERNEL_DEF15]** is included with the installation.

A.2 Using the GenJava Development Tool (Outbound Processing)

This section describes how to use the GenJava development tool, which is used to create Java wrappers for accessing the J.D. Edwards business functions. The Oracle Application Adapter for J.D. Edwards OneWorld uses these wrappers to call the J.D. Edwards business functions.

J.D. Edwards provides a Java Generation tool called GenJava that you can use to expose J.D. Edwards business functions externally as Java class files. A J.D. Edwards system administrator usually runs the GenJava tool.

During GenJava operation, you must specify a library of business functions, for example CALLBSFN. GenJava creates the associated Java class files for the business functions and related data structures. GenJava also compiles the business functions, generates Java documents, and packages them into two .JAR files. One .JAR file contains Java classes and the second .JAR file contains Java documents.

For example, if the business function library you specified in GenJava is CALLBSFN, the following files are found in the <install>\system\classes directory or any user-specified directory redirected by GenJava:

- JDEJAVA_CALLBSFN.xml
- JDEJAVA_CALLBSFNInterop.jar
- JDEJAVA_CALLBSFNInteropDoc.jar

Once they are generated, these library files must be added to the CLASSPATH.

GenJava also provides access to J.D. Edwards business functions by generating pure Java interfaces for these business functions. GenJava can be generated from a thick client or a deployment server.

Running GenJava

GenJava is located in the <install>\system\bin32 directory. You run GenJava from the command line. There are two GenJava command options that can be used to generate the wrappers.

GenJava Command Option 1

The following command generates Java wrappers for Category 1 (Master Business Functions), Category 2 (Major Business Functions), Category 3 (Minor Business Functions), and Category - (Uncategorized Business Functions) in the CALLBSFN library:

```
GenJava /Cat 1 /Cat 2 /Cat 3 /Cat - CALLBSFN
```

GenJava Command Option 2

The GenJava command can also be run with a JDEScript file and prompts a J.D. Edwards log on window, where you must enter a valid user ID, password, and environment.

1. Using an editor, create a new file called `AddressBook.cmd` and enter the following commands:

```
define library CALLBSFN
login
library CALLBSFN
interface AddressBook
import B0100031
import B0100019
import B0100032
import B0100002
import B0100033
build
logout
```

2. Run the following GenJava command:

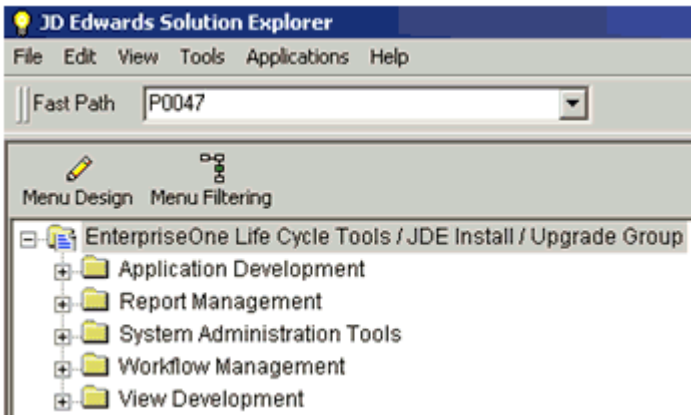
```
GenJava /cmd .\AddressBook.cmd
```

3. GenJava generates the wrappers (CALLBSFNInterop.jar, CALLBSFNInteropDoc.jar, and CALLBSFN.xml) in Java for all business functions that are mentioned in the script file.

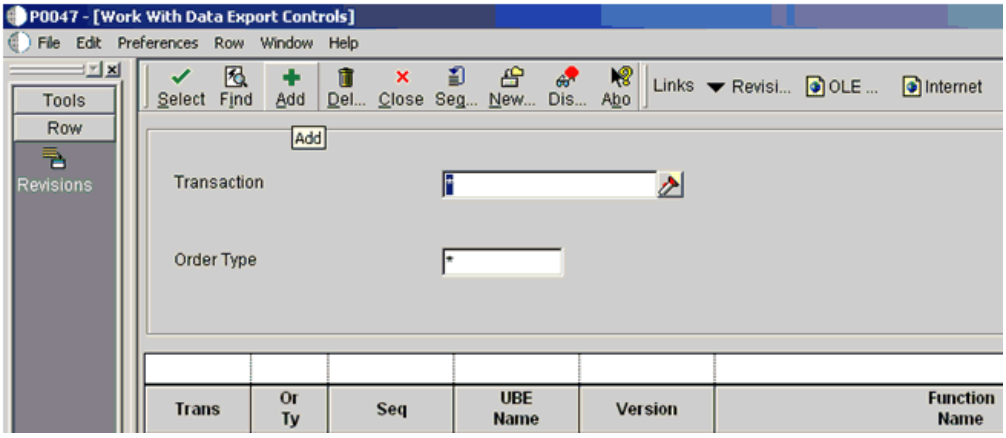
A.3 Triggering J.D. Edwards OneWorld Events

The flow of inbound data to third parties is controlled through the Data Export Controls application. For each transaction type and order type, one or more records can be defined with different function names and libraries.

1. Type **P0047** in the Fast Path field and press **Enter**.

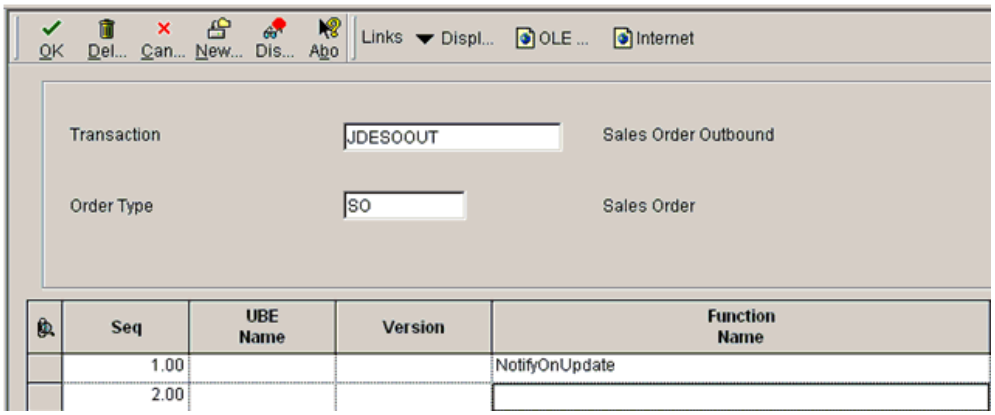


The Work With Data Export Controls window opens.



2. Click **Add**.

   The Data Export Control Revisions window opens. Notice that the sequence (Seq) number automatically increments for each new line.



3. Perform the following steps:

   a. Type **JDESOOUT** in the Transaction field.

   b. Type **SO** in the Order Type field.

   c. Type **NotifyOnUpdate** in the first row of the Function Name column.

   d. Type the absolute path to the location of the iwoevent.dll file in the first row of the Function Library column, for example:

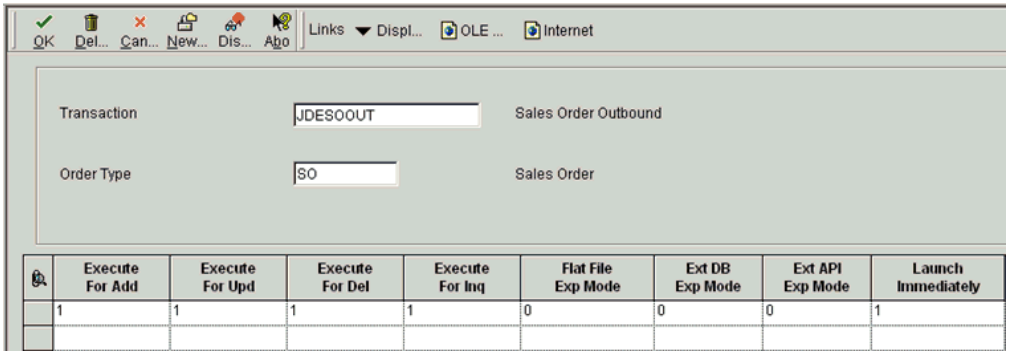
      ```
      D:\JDEdwards\E812\DDP\Outbound\iwoevent.dll
      ```

   e. Type **1** in the first row of the Execute for Add column if you want the notifications for add/insert.

   f. Make the same decision for update, delete, and inquiry and type **1** in the appropriate column.

   g. Type **1** in the Launch Immediately column to launch the Outbound Subsystem batch process (R00460).

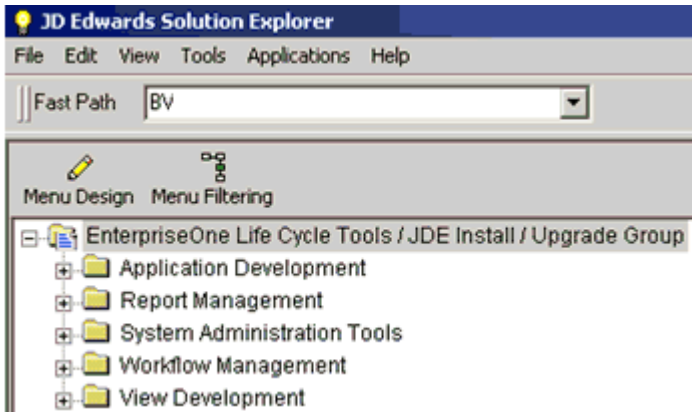
4. Click **OK**.



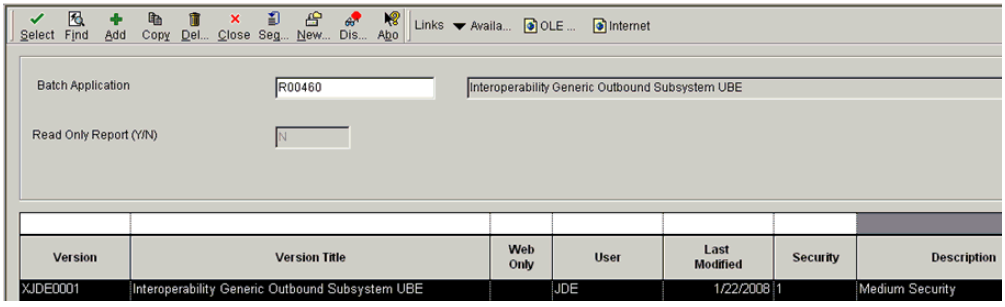
Starting the Outbound Scheduler Subsystem Process (R00460)

Once you have finished defining one or more records for each transaction type and order type, you must manually start the outbound scheduler subsystem process.

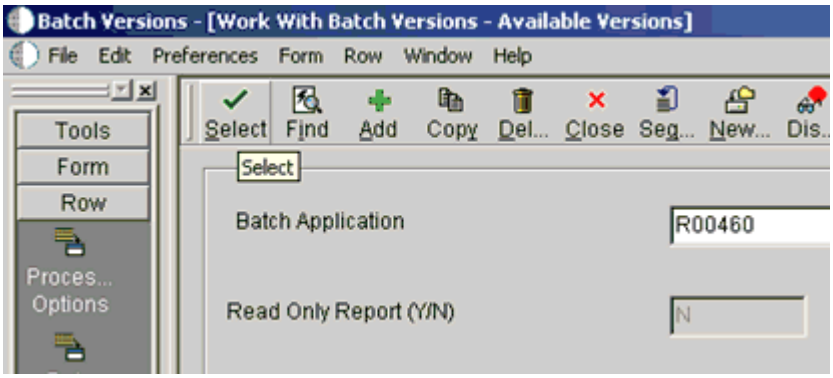
1. Type **BV** in the Fast Path field and press **Enter**.



The Work With Batch Versions - Available Versions window opens.

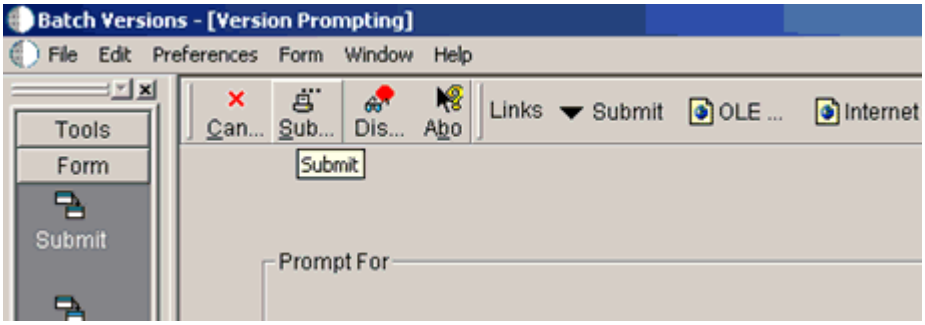


2. Type **R00460** in the Batch Application field and click **Find**.

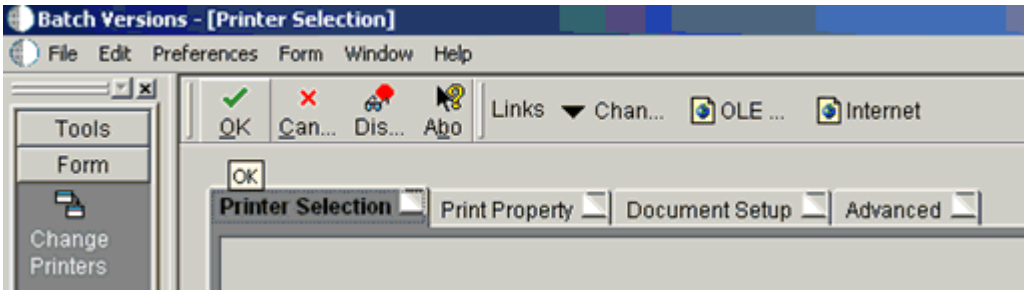


3. Select **Interoperability Generic Outbound Subsystem UBE (XJDE0001)** and click **Select**.

   The Version Prompting window opens.



4. Click **Submit**.

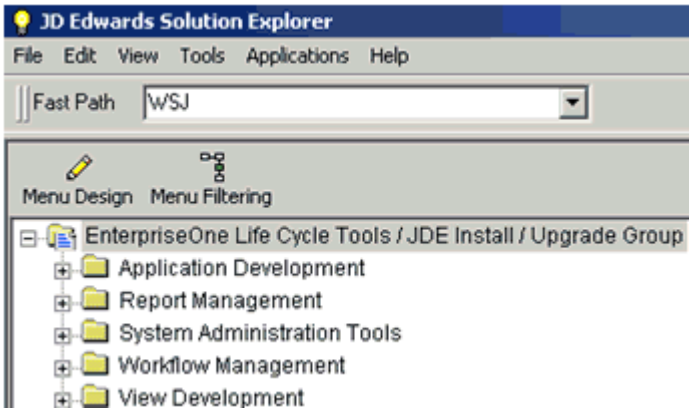


5. Navigate to the last screen and click **OK**.

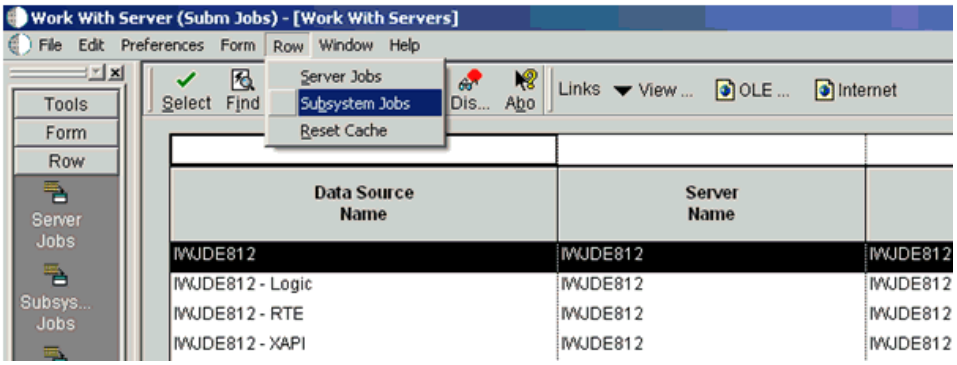
Verifying the Subsystem Process

This section describes how to verify the outbound scheduler subsystem processs.

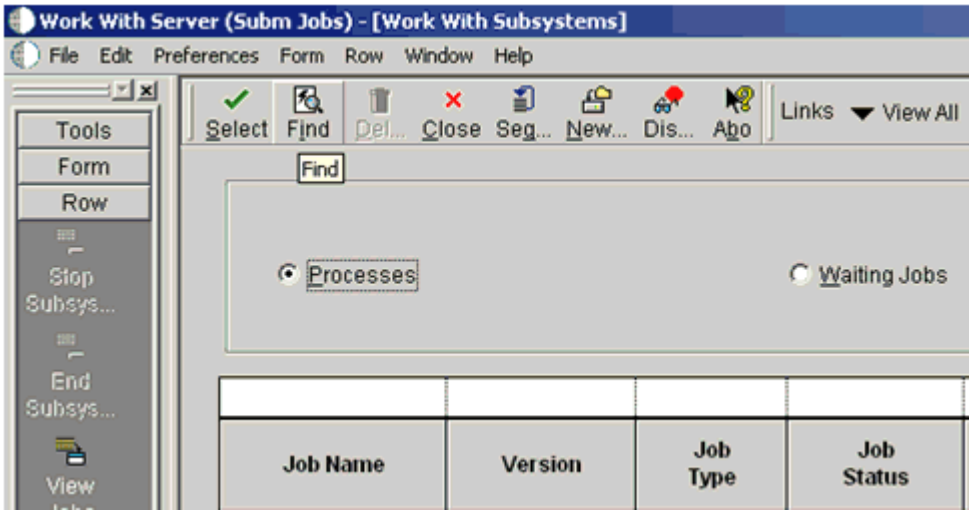
1. Type **WSJ** in the Fast Path field and press **Enter**.



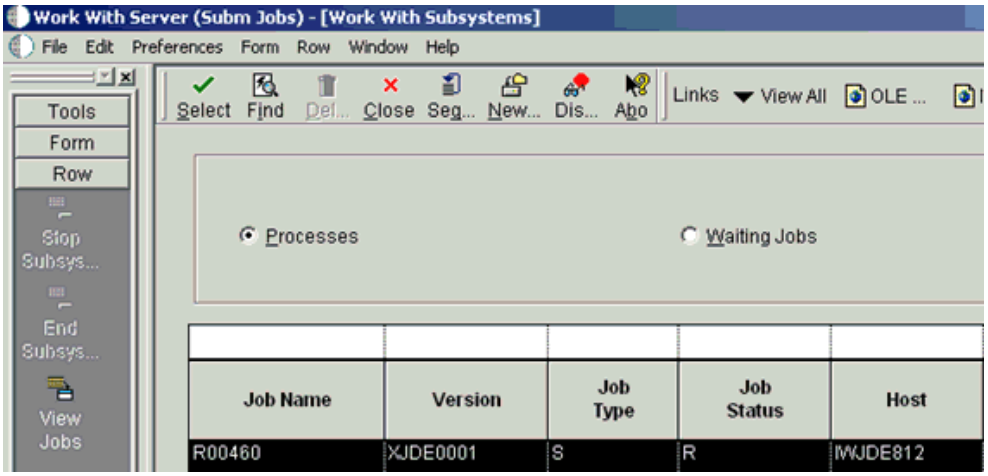
The Work With Server (Subm Jobs) window opens.



2. Select a corresponding server from the table.
3. Click **Row** from the menu bar and select **Subsystem Jobs**.



4. Click **Find**.

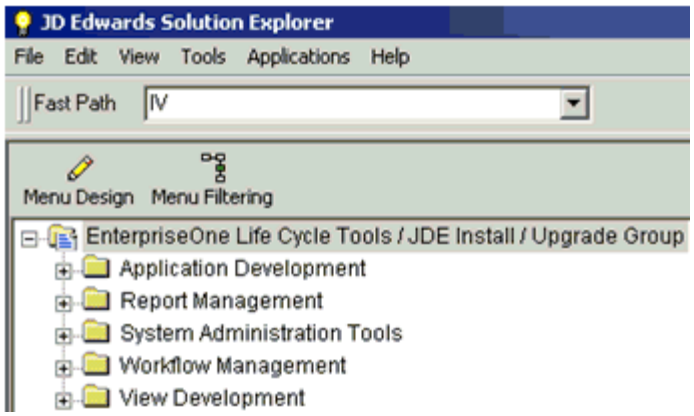


5. Verify that **R** is listed in the Job Status column.

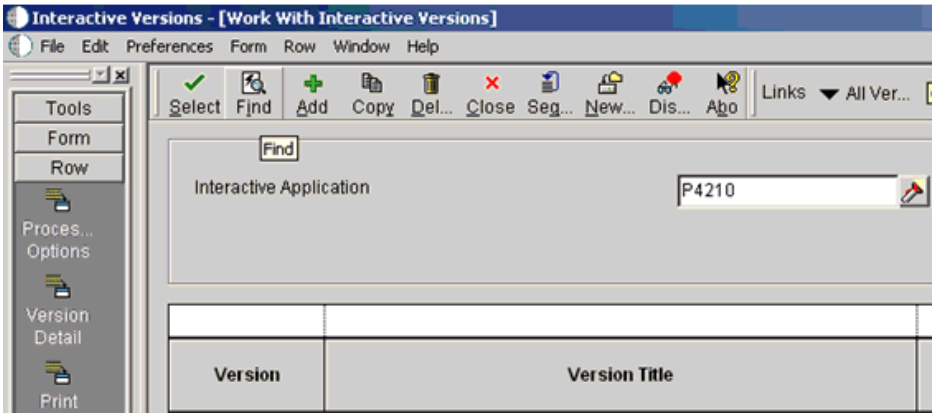
Configuring P4210 (Sales Order) to Trigger an Event

This section describes how to configure a P4210 (Sales Order) to trigger an event.

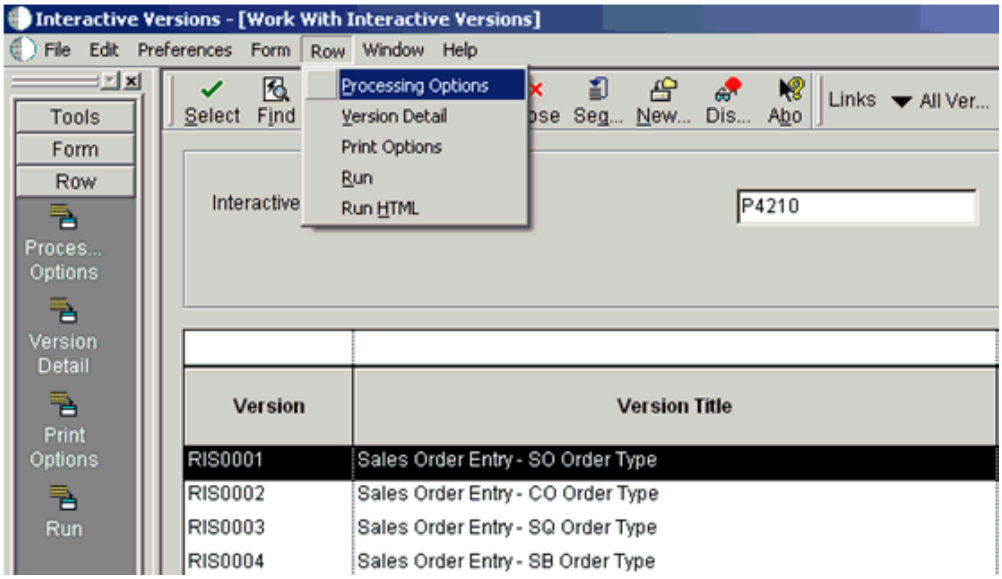
1. Type **IV** in the Fast Path field and press **Enter**.



The Interactive Versions window opens.

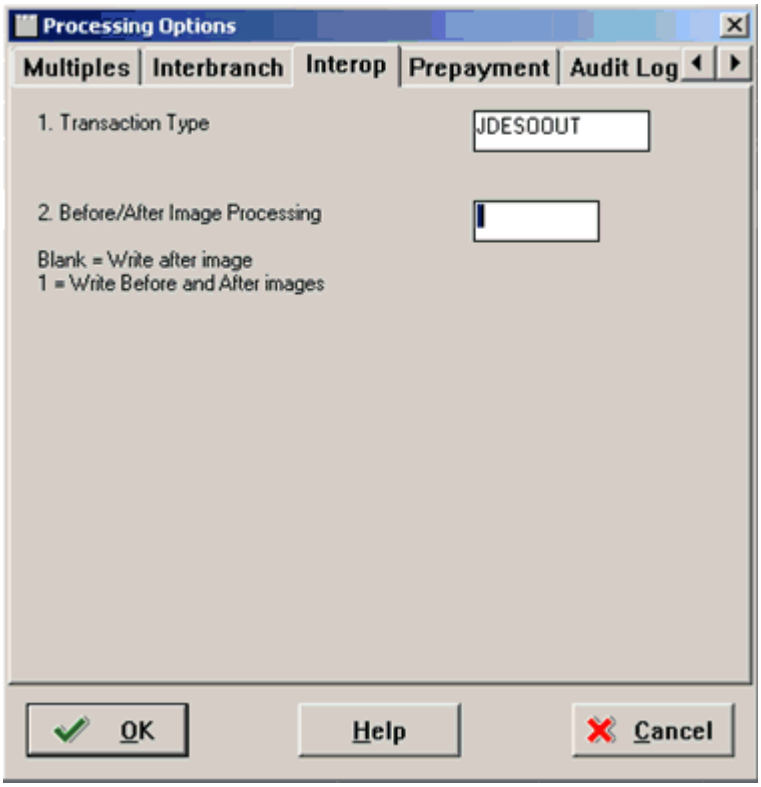


2. Type **P4210** in the Interactive Application field and click **Find**.



3. Select a document version from the table, for example, **RIS0001 - Sales Order Entry - SO Order Type**.
4. Click **Row** from the menu bar and select **Processing Options**.

   The Processing Options dialog opens.

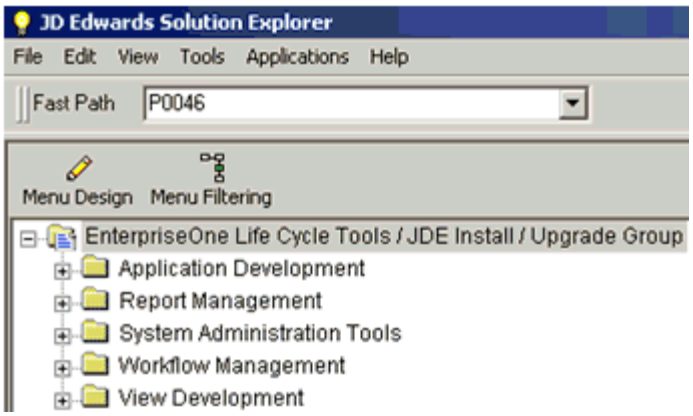


5. Click the **Interop** tab.
6. Type **JDESOOUT** in the Transaction Type field.
7. Add **Sales Order**.

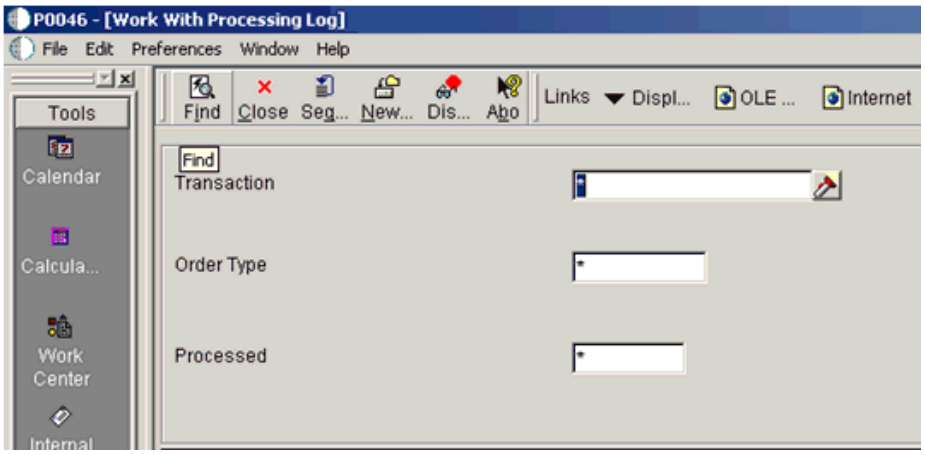
Verifying the Configuration Steps

This section describes how to verify the configuration steps by updating F0046.

1. Type **P0046** in the Fast Path field and press **Enter**.



The P0046 - Work With Processing Log window opens.



2. Click **Find**.

The following data is displayed.

User ID	Batch Number	Transaction Number	Line Number	Trans	Or Ty	Seq	UBE Name	Version	S P
JDE	15147	103322	1.000	JDESOOUT	SO	1.00			N
JDE	15148	103323	1.000	JDESOOUT	SO	1.00			N
JDE	15149	103324	1.000	JDESOOUT	SO	1.00			N
JDE	15150	103325	1.000	JDESOOUT	SO	1.00			N
JDE	15151	103326	1.000	JDESOOUT	SO	1.00			N
JDE	15152	103327	1.000	JDESOOUT	SO	1.00			N
JDE	15153	103328	1.000	JDESOOUT	SO	1.00			N
JDE	15154	103329	1.000	JDESOOUT	SO	1.00			N
JDE	15155	103330	1.000	JDESOOUT	SO	1.00			N
JDE	15156	103331	1.000	JDESOOUT	SO	1.00			N
JDE	15157	103332	1.000	JDESOOUT	SO	1.00			N
JDE	15158	103333	1.000	JDESOOUT	SO	1.00			N
JDE	15159	103334	1.000	JDESOOUT	SO	1.00			N
JDE	15160	103335	1.000	JDESOOUT	SO	1.00			N
JDE	15163	103452	1.000	JDESOOUT	SO	1.00			N
JDE	15164	103453	1.000	JDESOOUT	SO	1.00			N
JDE	15165	103454	1.000	JDESOOUT	SO	1.00			N
JDE	15166	103455	1.000	JDESOOUT	SO	1.00			N

3. Search for the corresponding transaction.

The iwoevnt.log file is created in the following directory:

```
\\iwJDE812\JDEdwards\E812\DDP\system\bin32
```

The iwoevent.log file is created in the outbound folder where the iwoevent.dll and iwoevent.cfg files are located. The following is an example of the event log file:

```
Event call begin...
Server time      : Tue May 27 07:23:55 2008
userId           : JDE
batchNumber      : 15205
transactionNumber: 103494
lineNumber       : 1.000000
transactionType  : JDESOOUT
sequenceNumber   : 1.000000
Request xml:
========================
<? xml version="1.0" encoding="UTF-8"?><jde><request><connection><dsn /><user
/><password /><sp><proc>JDESOOUT</proc><data><ediUserId>JDE
</ediUserId><ediBatchNumber>15205
</ediBatchNumber><ediTransactionNumber>103494
</ediTransactionNumber></data></sp></connection></request></jde>
========================
```

B

Sample Files

The Oracle Application Adapter for J.D. Edwards OneWorld supports the jdeRequest and jdeResponse XML structures for executing business functions within J.D. Edwards 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 J.D. Edwards OneWorld:

B.1 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>
```

B.2 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, and 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>
```

B.3 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>
```

B.4 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

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

W

X

Z