Oracle® Application Server

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

10*g* Release 2 (10.1.2)

Part No. B14059-01

November 2004

ORACLE®

Oracle Application Server Adapter for J.D. Edwards OneWorld User's Guide, 10*g* Release 2 (10.1.2)

Part No. B14059-01

Primary Author: Ed Marsh

Contributing Author: Meera Srinivasan

Contributors: Arvind Jain, Jennifer Chua

Contents

Send Us Your Comments

Oracle Application Server Adapter for J.D. Edwards OneWorld User's Guide, 10*g* Release 2 (10.1.2)

Part No. B14059-01

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: appserverdocs_us@oracle.com
- FAX: (650) 506-7375. Attn: Oracle Application Server Documentation Manager
- Postal service:

  Oracle Corporation
  Oracle Application Server Documentation
  500 Oracle Parkway, Mailstop 1op6
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This guide explains how to use the Oracle Application Server Adapter for J.D. Edwards OneWorld to access J.D. Edwards OneWorld Business Components and Business Services. In this guide you will learn how to define a delivery channel for J.D. Edwards OneWorld and add an interaction to generate native events, which are XML instances defined by XSD (XML payload defined by an XML Schema Definition instance). In this guide you will also find a chapter describing the datatype mapping between J.D. Edwards OneWorld and XSD.

This preface contains these topics:

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. 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 additional information, visit the Oracle Accessibility Program Web site at

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

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, 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, JAWS 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.

Intended Audience

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

- Create delivery channels and interactions with a J.D. Edwards OneWorld system
- Maintain applications

To use this document, you need some knowledge of J.D. Edwards OneWorld Business Components and Business Services.

Organization

This document contains:

Chapter 1, "Introduction to Oracle Application Server Adapter for J.D. Edwards OneWorld"

This chapter describes the Oracle Application Server Adapter for J.D. Edwards OneWorld.

Chapter 2, "Adapter Configuration Using Oracle Application Explorer"

This chapter provides instructions for starting Application Explorer, for creating projects, establishing a connection to J.D. Edwards OneWorld, and creating schemas and Web services. It also explains how to configure the Event Adapter.

Chapter 3, "Deployment and Integration"

This chapter describes Oracle Containers for J2EE (OC4J) deployment and integration with OracleAS Integration InterConnect.

Chapter 4, "Examples"

This chapter contains examples.

Chapter 5, "Troubleshooting and Error Messages"

This chapter describes how to troubleshoot and interpret error messages.

Chapter 6, "Advanced Topics"

This chapter includes advanced topics for expert users.

Appendix A, "Configuring J.D. Edwards OneWorld for Outbound Transaction Processing"

This appendix describes how to enable outbound transaction processing in OneWorld and how to modify the `jde.ini` file for XML support.

Appendix B, "Sample Files"

Related Documentation

For more information, see these Oracle resources:

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

- *Oracle Application Server Administrator's Guide*
- *Oracle Application Server Concepts*
- *Oracle Application Server Containers for J2EE User's Guide*
- *Oracle Application Server Integration InterConnect User's Guide*

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com`

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

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

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

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

Conventions

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

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

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.

Convention	Meaning	Example
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run *`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

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

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

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

Convention	Meaning	Example
`[ ]`	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
`{ }`	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
`|`	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`
`...`	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	`CREATE TABLE ... AS `*`subquery`*`;` `SELECT `*`col1`*`, `*`col2`*`, ... , `*`coln`*` FROM employees;`

Convention	Meaning	Example
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*
UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

Conventions for Windows Operating Systems

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

Convention	Meaning	Example
Choose Start >	How to start a program.	To start the Database Configuration Assistant, choose Start > Programs > Oracle - *HOME_NAME* > Configuration and Migration Tools > Database Configuration Assistant.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the file name begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32

Convention	Meaning	Example
C:\>	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	C:\oracle\oradata>
Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	C:\>exp scott/tiger TABLES=emp QUERY=\"WHERE job='SALESMAN' and sal<1600\" C:\>imp SYSTEM/*password* FROMUSER=scott TABLES=(emp, dept)
HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	C:\> net start Oracle*HOME_NAME*TNSListener
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle8i release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory. For Windows, the default location was C:\orant. This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is C:\oracle. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is *C:*\oracle\orann, where *nn* is the latest release number. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle Database Platform Guide for Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the ORACLE_BASE\ORACLE_HOME\rdbms\admin directory.

1

Introduction to Oracle Application Server Adapter for J.D. Edwards OneWorld

The Oracle Application Server Adapter for J.D. Edwards OneWorld provides connectivity and executes interactions on a J.D. Edwards OneWorld system. This chapter provides information about the OracleAS Adapter for J.D. Edwards OneWorld to help you accomplish your integration projects.

This chapter discusses the following topics:

Adapter Features

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

The OracleAS Adapter for J.D. Edwards OneWorld can be deployed as a JCA 1.0 resource adapter. This deployment is referred to as the OracleAS JCA adapter. It can also be deployed as a Web services servlet and as such is referred to as the Oracle Application Server Adapter Business Services Engine (OracleAS Adapter BSE).

The adapter uses XML messages to enable non-J.D. Edwards applications to communicate and exchange transactions with J.D. Edwards using services and events.

- Services: Applications use this capability to initiate a J.D. Edwards business event.
- Events: Applications use this capability to access J.D. Edwards data only when a J.D. Edwards business event occurs.

To support event functionality, two features are implemented:

- **port**

  A port associates a particular business object exposed by the adapter with a particular disposition. A disposition is a URL that defines the protocol and location of the event data. The port defines the end point of the event consumption. For example, you can use the MSMQ protocol to route the result of a Purchase Order update in the J.D. Edwards system to a queue hosted by your application server.

The port is the Oracle adapter component that pushes the event received from the EIS to the adapter client. The only port supported in this release is Remote Method Invocation (RMI). It is used for integration with OracleAS Integration InterConnect.

- Channel

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

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

The OracleAS Adapter for J.D. Edwards OneWorld provides:

- XML schemas for the JCA 1.0 resource adapter.
- Web services for the Business Services Engine.

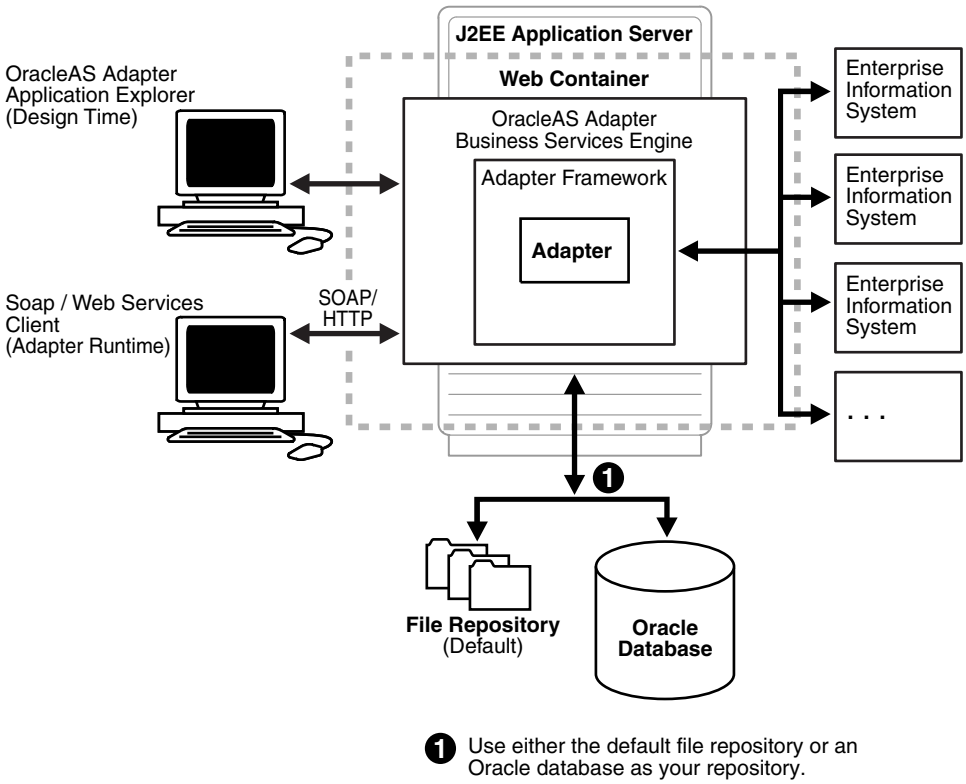
See Also: *Oracle Application Server Adapter Concepts*

Resource Adapters

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

Figure 1–1 shows the generic architecture for the Oracle Web service adapter for packaged applications. The adapter works in conjunction with the Oracle Application Server Adapter Business Services Engine (BSE), as deployed to a Web container in a J2EE application server.

Figure 1–1 Oracle Application Server Adapter Business Services Engine Architecture



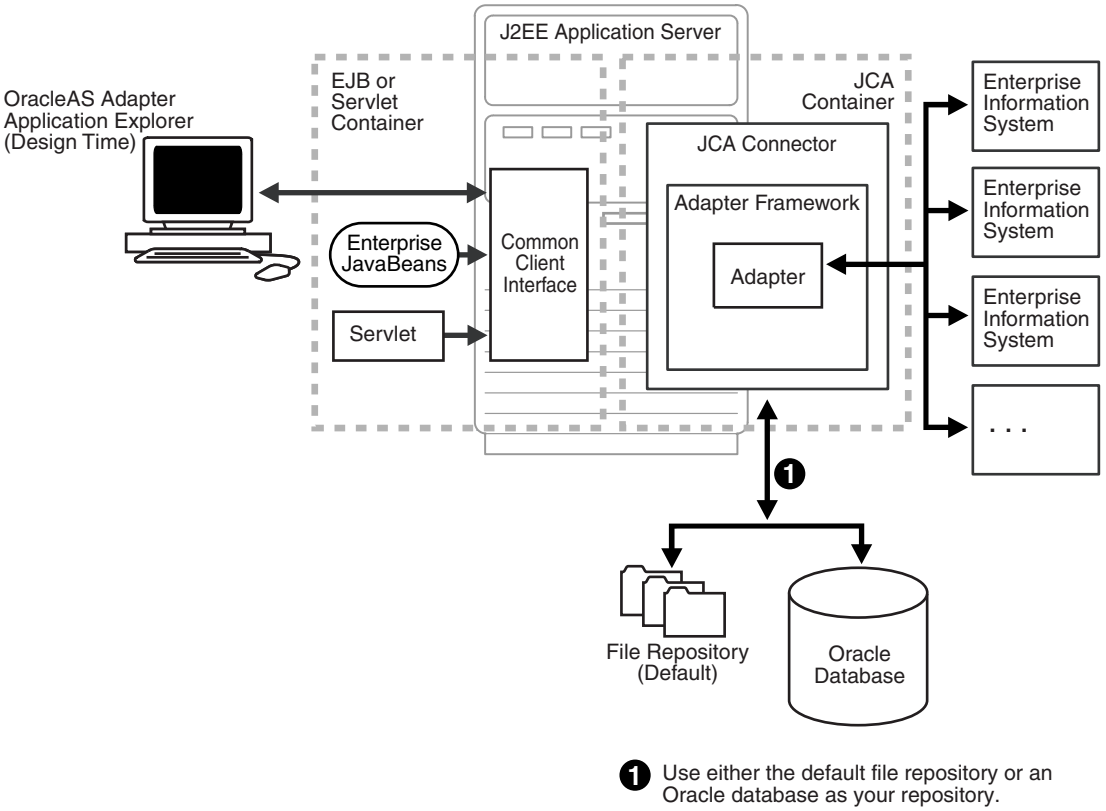
Oracle Application Server Adapter Application Explorer (Application Explorer), a design-time tool deployed along with BSE, is used to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. Metadata created while you perform these operations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

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

BSE receives the adapter response, wraps the response XML in a SOAP envelope, and returns it to the adapter bridge. The bridge then strips the SOAP envelope, strips the namespace prefix, if present, and passes the DTD-compliant XML to the IC Adapter agent.

Figure 1–2 shows the generic architecture for the Oracle JCA adapter for packaged applications.The JCA connector is deployed to a standard JCA Container and serves as host container to the adapters. The connector is configured with a repository.

Figure 1–2 Oracle Application Server Adapter Generic JCA Architecture



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

Processing Business Functions

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

External applications that access OneWorld through the OracleAS Adapter for J.D. Edwards OneWorld use either XML schemas or Web services to pass data between the external application and the adapter. The following topics describe 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.

J.D. Edwards OneWorld Concepts

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

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

Integration with J.D. Edwards OneWorld

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

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

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

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

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

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

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

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

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

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

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

Propagating External Listeners Into J.D. Edwards OneWorld

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

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

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

Propagating Internal Listeners out of J.D. Edwards OneWorld

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

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

J.D. Edwards OneWorld Interoperability Framework

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

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

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

Figure 1–3 illustrates the inbound processing framework.

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

Figure 1–3 J.D. Edwards OneWorld Inbound Processing

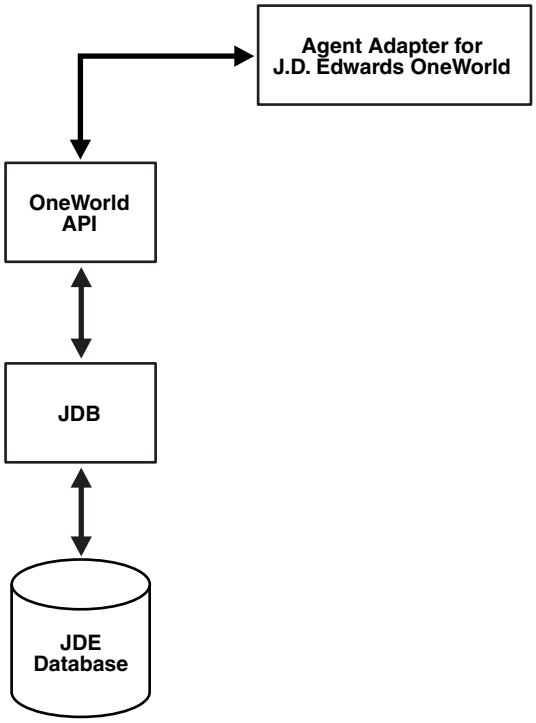
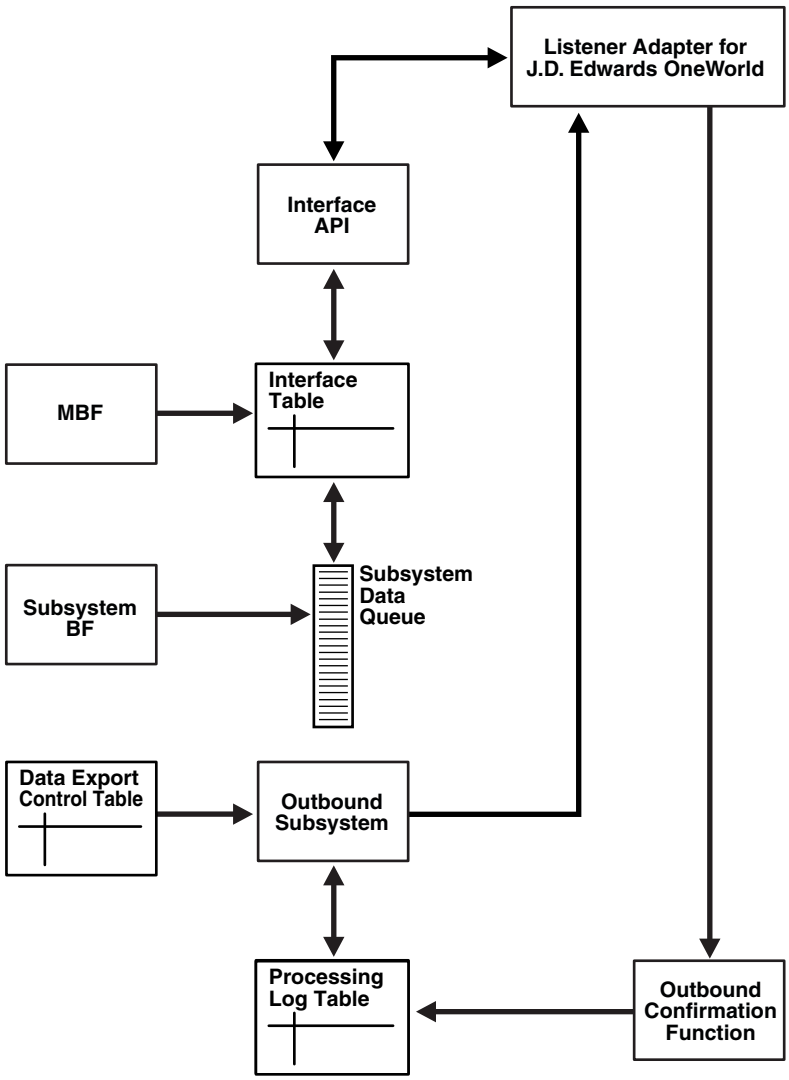


Figure 1–4 illustrates the outbound processing framework.

Figure 1–4 J.D. Edwards OneWorld Outbound Processing



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

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

Adapter Deployment

The OracleAS Adapter for J.D. Edwards OneWorld works in conjunction with Application Explorer and the Business Services Engine (BSE) or the Enterprise Connector for J2EE Connector Architecture (JCA).

Application Explorer is used to configure database connections and create Web services and events. It can be configured to work in a Web services environment in conjunction with the Business Services Engine or with the Enterprise Connector for J2EE Connector Architecture (JCA). When working in a JCA 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 JCA are deployed to an application server with the Application Explorer and the adapters.

Application Explorer

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

See Also:

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

2

Adapter Configuration Using Oracle Application Explorer

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

This chapter discusses the following topics:

Starting Application Explorer

The server must be started where Application Explorer is deployed.

To start Application Explorer:

1. Ensure the server is started where Application Explorer is deployed.
2. On Windows, invoke the iaexplorer script, `iaexplorer.exe`, found under *`OracleAS_home`*`\adapters\application\tools` or on UNIX invoke the iwae script, `iwae.sh`, found under *`OracleAS_home`*`/adapters/application/tools`.

OracleAS_home
Is the directory where the Oracle Application Server is installed.

Application Explorer opens. You are ready to define new targets to your Enterprise Information System (EIS).

Configuring Settings for BSE or JCA

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

Configuring the OracleAS Adapter Business Services Engine

After the OracleAS Adapter Business Services Engine (BSE) is deployed to Oracle Application Server, you can configure it through the BSE configuration page.

To configure BSE:

1. Open the following page in your browser:

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

   hostname
   Is the hostname of the Oracle Application Server.

   port
   Is the HTTP port for the Oracle Application Server.

   For example,

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

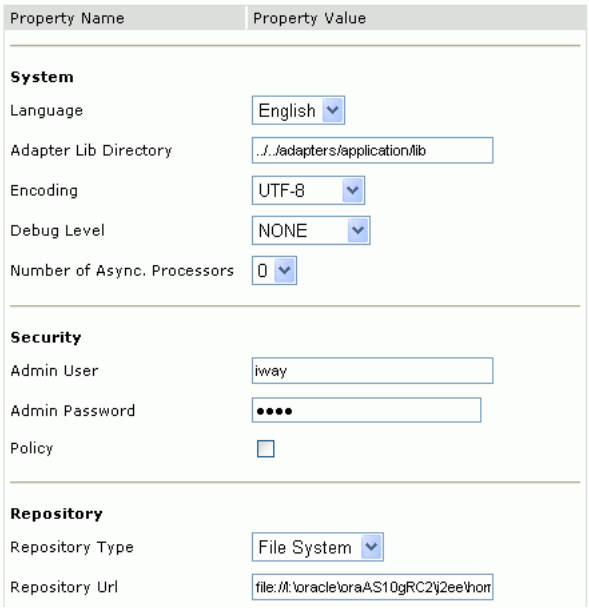
   Note: The first time you access this page, it may take time to load.

2. When prompted, log on.

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

   - User name: iway
   - Password: iway

The BSE configuration page opens.



The image shows the BSE Settings page.

****

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

OracleAS_home\adapters\application\lib

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

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

> **Note:** The **Repository URL** field specifies where the file system repository is located. To use a database repository, you must enter the repository connection information. For the initial verification, use a file system repository. See "Configuring an Oracle Repository" on page 2-5 for information on switching to a database repository.

5. Click **Save.**

Configuring BSE System Settings

To configure BSE system settings:

1. Open the BSE configuration page by entering the following URL:

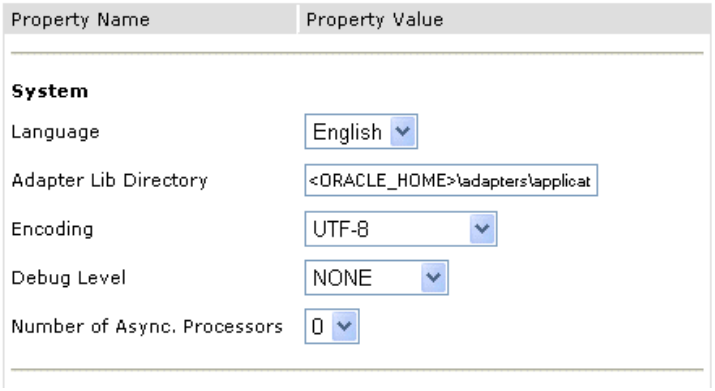
   http://*hostname*:*port*/ibse/IBSEConfig

hostname
Is the machine where BSE is installed.

port
Is the port number on which BSE is listening.

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

The BSE configuration page opens.



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

Parameter	Description
Language	Specify your required language.
Adapter Lib Directory	Type the full path to the directory where the adapter jar files reside
Encoding	Specify the default encoding from one of the following options: UTF-8 EBCDIC-CP-US ISO-88859-1 Shift JIS UNICODE
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.

Figure 2–1 shows the Security pane of the BSE configuration page.

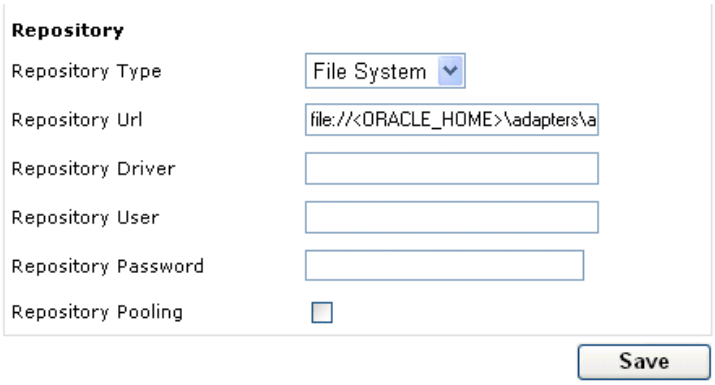
Figure 2–1 BSE Security Pane

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

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

Figure 2–2 shows the Repository pane of the BSE configuration page.

Figure 2–2 BSE Repository Pane



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

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

Parameter	Description
Repository Type	Select one of the following repositories from the list: Oracle File
Repository URL	Type the URL to use when opening a connection to the database.
Repository Driver	Provide the driver class to use when opening a connection to the database (optional).
Repository User	Type the user ID to use when opening a connection to the database.
Repository Password	Type the password associated with the user ID.
Repository Pooling	Select the check box to enable pooling.

5. Click **Save.**

Configuring a File System Repository

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

The default location for the repository on Windows is:

`OracleAS_home\config\base\ibserepo.xml`
On other platforms, use the corresponding location.

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

Configuring an Oracle Repository

To configure an Oracle repository:

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

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

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

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

For other platforms, see the corresponding location.

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

```
iwse.ora
```

3. Type the following command:

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

Creating a Repository Project

Before you use Application Explorer with the OracleAS Adapter for J.D. Edwards OneWorld, you must create a repository project. You can create two kinds of repository projects, Web services and JCA, depending on the container to which the adapter is deployed.

At design time, the repository is used to store metadata created when using Application Explorer to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. The information in the repository is also referenced at runtime.

A default JCA repository is created for the default ManagedConnectinFactory. The name of this project is jca_sample.

For more information, see "Adapter Features" on page 1-1.

Creating a Repository Project for BSE

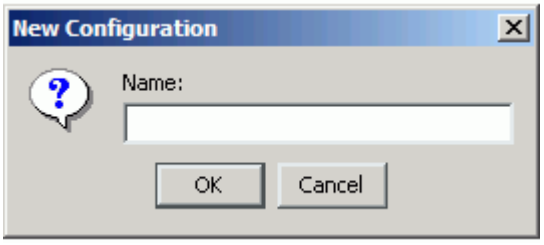
To create a repository project for BSE using Application Explorer, you must first define a new configuration.

Defining a New Configuration for BSE

To define a new configuration for BSE:

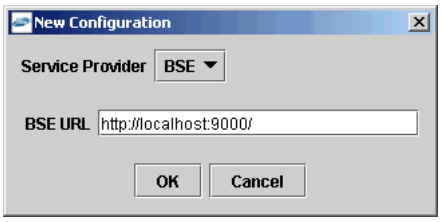
1. Right-click **Configurations** and select **New.**

   The New Configuration dialog box opens.



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

   The New Configuration dialog box opens.



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

   `http://hostname:port/`

hostname
Is the machine where your application server resides.

port
Is the port number where the application server is listening.

5. Click **OK.**

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

Creating a Repository Project for JCA

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

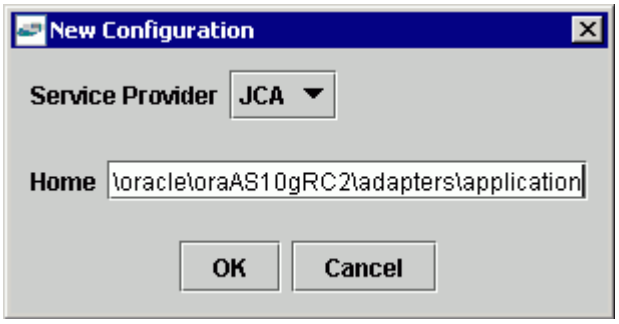
Defining a New Configuration for JCA

To define a new configuration for JCA:

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

   The New Configuration dialog box opens.

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



3. From the **Service Provider** list, select **JCA.**
4. In the **Home** field, enter a path to your JCA configuration directory where the repository, schemas, and other information is stored, for example:

```
OracleAS_home\adapters\application
```

5. Click **OK.**

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

Connecting to a New Configuration

To connect to a new 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).

Use the Adapters folder to create inbound interaction with J.D. Edwards. For example, you use the J.D. Edwards node in the Adapters folder to configure a service that updates J.D. Edwards.

Use the Event Adapters folder to configure listeners that listen for events in J.D. Edwards OneWorld.

Use the Business Services folder 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 are now ready to define new targets to J.D. Edwards OneWorld.

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

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

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 automatically is saved. You must connect to the system every time you start Application Explorer or after you disconnect.

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

Defining a Target to J.D. Edwards OneWorld

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

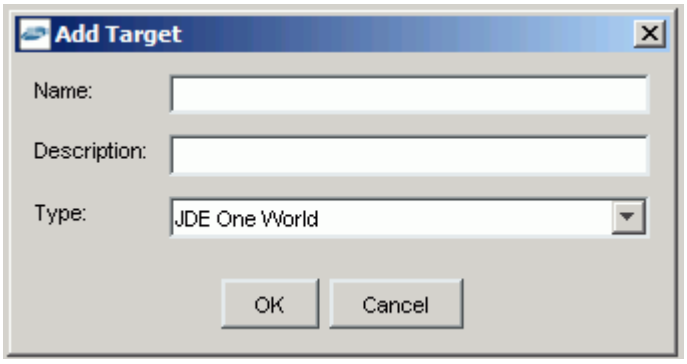
To define a target:

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

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

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

   The Add Target dialog box opens.



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

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

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

3. Click **OK.**

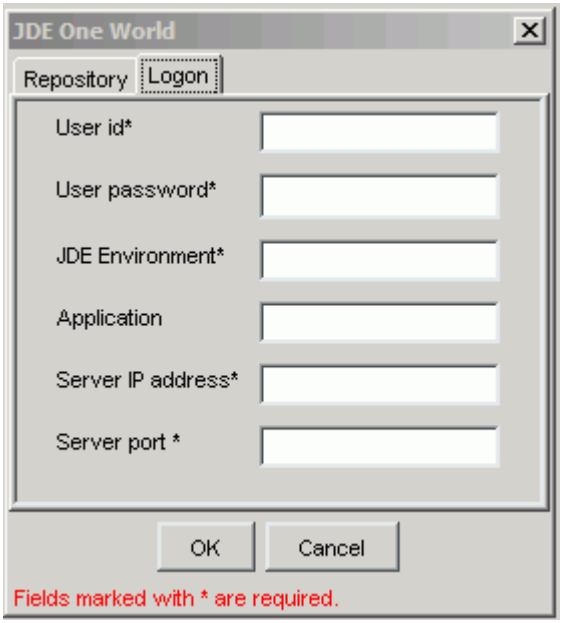
   The JDE One World dialog box appears.

   a. On the **Repository** tab, type the path to the GenJava repository.

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

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

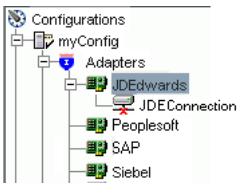
   b. Click the **Logon** tab and type the appropriate information for your target type based on the information in the following table. Fields with an asterisk are required.



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 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.

4. Click **OK.**

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



   For information on how to create schemas for the adapter, see "Creating an XML Schema" on page 2-12.

Connecting to a Defined J.D. Edwards OneWorld Target

To connect to a target:

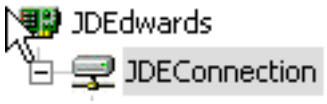
1. Expand the **Service Adapters** node.
2. Expand the **JDEdwards** node.

3. Click the target name (for example, JDEConnection) under the JDEdwards node.

   The Connection dialog box opens, populated with values you entered for the connection parameters.

4. Verify your connection parameters. If required, provide the password.
5. 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.**

   Disconnecting from JDEdwards drops the connection with JDEdwards, but the node remains.

   The x icon appears, indicating that the node is disconnected, as shown in Figure 2–3.

Figure 2–3 JDEdwards Node Disconnected

Editing a Target

To edit a target:

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

   The Edit pane opens on the right.

3. Modify the target information.
4. Click **OK.**

Deleting a Target to J.D. Edwards OneWorld

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

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.

Viewing Application System Objects

See Also: For more information, see the *J.D .Edwards Interoperability Guide Release OneWorld XE.*

Creating an XML Schema

To execute a Master Business Function (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. The Application Explorer creates both the XML request schema and the XML response schema.

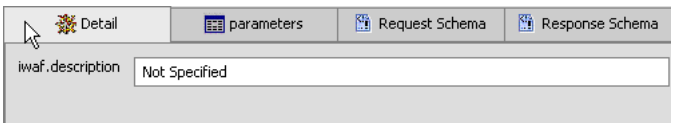
Creating a Request and a Response Schema

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

1. Connect to a J.D. Edwards OneWorld target as described in "Connecting to a Defined J.D. Edwards OneWorld Target" on page 2-10.
2. Expand the **Services** node.
3. Expand the node of the Master Business Function (MBF) for which you want to create the schema.
4. Expand and then select the node beneath the MBF.

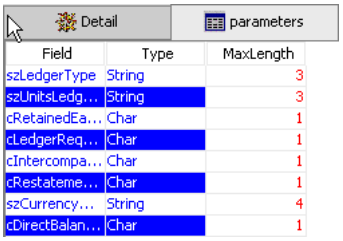
   Figure 2–4shows the tabs that appear on the right.

Figure 2–4 Services Node Tabs

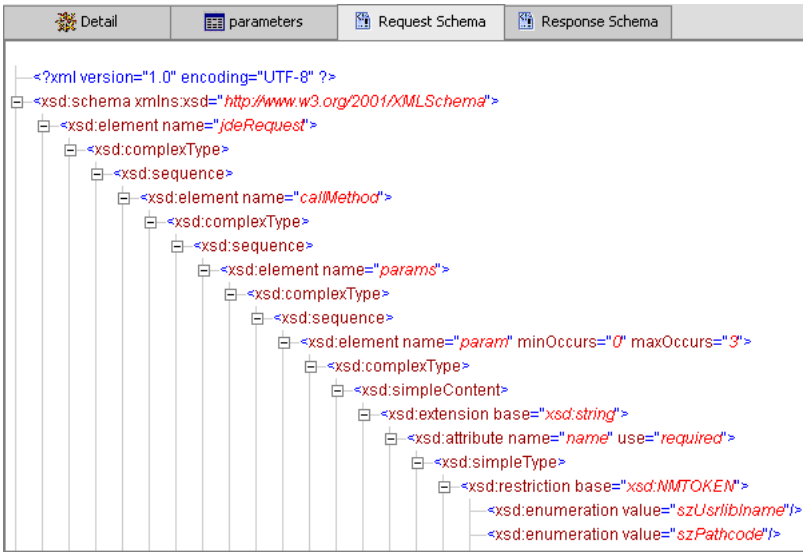


5. Click the **parameters** tab to view the parameter information, as shown in Figure 2–5.

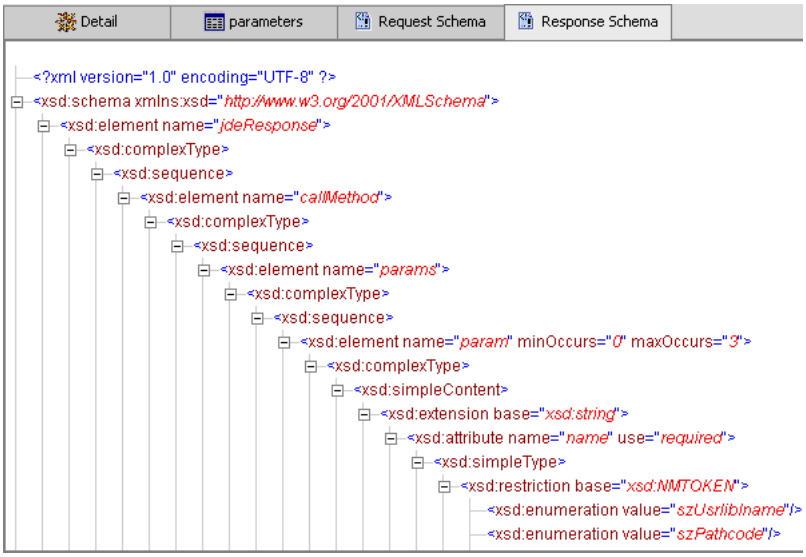
Figure 2–5 Parameters Tab



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



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



Using GenJava to Generate a Schema

To create schemas for the adapter, you must use GenJava wrappers. You create the GenJava wrappers using the OneWorld utility called GenJava. You use the Application Explorer to generate schemas against OneWorld GenJava wrappers. GenJava is supplied as a command line process with several run-time options. For more information on GenJava, see the *J.D. Edwards Interoperability Guide for OneWorld Xe.*

Creating a Web Service or Business Service

You can generate a Web service (also known as a **business service**). 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 (JCA) implementation, Web services are not available. When the adapters are deployed to use JCA, the Common Client Interface (CCI) provides integration services.

To create a Web service for a business function:

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

   The Create Business Service dialog box opens.

   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. 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. Type a description for the service (optional).

   d. Select one of the available licenses.
4. Click **Next.**

   The license and method dialog box opens.

   a. In the **License** field, select one or more license codes to assign to the Web service. To select more than one, hold down the **Ctrl** key and click the licenses.

   b. In the **Method Name** field, type a descriptive name for the method.

   c. In the **Description** field, type a brief description of the method.
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 (business) service:

1. If you are not on the Business Services node of Application Explorer, click the node to access Web services.
2. If it is not expanded, expand the list of Web services under **Business Services.**
3. Expand the **Services** node.
4. Select the name of the business service you want to test.

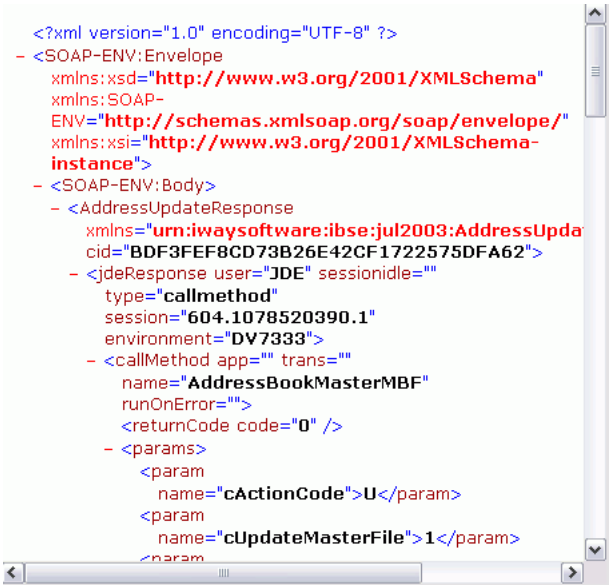
   The business service name appears as a link in the right pane.
5. In the right pane, click the named business services link.

The test option appears in the right pane. If you are testing a Web service that requires XML input, an input field appears.

6. Enter the appropriate input.
7. Click **Invoke.**

   Application Explorer displays the results. Figure 2–6 shows the XML for the results.

Figure 2–6 XML Test Results



Configuring an Event Adapter

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

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

- Port

A port associates a particular business object exposed by the Adapter with a particular disposition. A disposition is a URL that defines the protocol and location of the event data. The port defines the end point of the event consumption. For example, you can use the MSMQ protocol to route the result of a Purchase Order update in the J.D. Edwards OneWorld system to a queue hosted by your application server. See "Creating an Event Port" on page 2-16 for more information.

- Channel

A channel represents configured connections to particular instances of back-end systems. A channel binds one or more event ports to a particular listener managed by the adapter. See "Creating a Channel Using Application Explorer" on page 2-17 for more information.

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

Creating an Event Port

The Application Explorer enables you to create event ports from the Adapters node or from the Events node.

Creating an Event Port from the Adapters Node

You can bypass the Events node and create an event port directly from the Adapters node.

To create an event port from the Adapters node:

1. Select the J.D. Edwards object for which you want to create an event port.
2. Right-click the node and select **Add Port.**

   The Add Port dialog box opens.

   a. Type a name for the event port and provide a brief description.

   b. From the list, select the required disposition, for example, File.

   c. Type the disposition url.
3. Click **OK.**

See "Creating an Event Port From the Events Node" on page 2-16 for information on configuring port dispositions.

Creating an Event Port From the Events Node

The following procedure describes how to create an event port from the Events node for a disposition using Application Explorer. You can switch between a BSE and a JCA deployment by choosing one or the other from the menu in the upper right of Application Explorer.

You also can create an event port directly from the Adapters node. See "Creating an Event Port from the Adapters Node" for more information.

Creating an Event Port for RMI

To create a specific event port for RMI:

1. Expand the **Events** node.
2. Expand the **JDEdwards** node.
3. Right-click the **Ports** node and select **Add Port.**

   The Add Port dialog box opens.

   a. Type a name for the event port and provide a brief description.

   b. From the **Disposition Protocol** list, select **RMI.**

   c. In the **URL** field, specify a destination file to which the event data is written.

      When pointing Application Explorer to a JCA deployment, provide the full path to the directory.

   d. From the **Disposition protocol** list, select **RMI.**

The following table defines the parameters for the disposition.

Parameter	Description
location	Destination and file name of the document where event data is written, for example, `ifile://D:\in\x.txt;errorTo=ifile://D:\error`
errorTo	Predefined port name or another disposition URL to which error logs are sent.

4. Click **OK.**

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

   You are ready to associate the event port with a channel. For more information, see "Creating a Channel Using Application Explorer" on page 2-17.

Editing an Event Port

To edit an event port using Application Explorer:

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

   The Edit Port pane opens.
4. Make the required changes and click **OK.**

Deleting an Event Port

To delete an event port using Application Explorer:

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

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

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

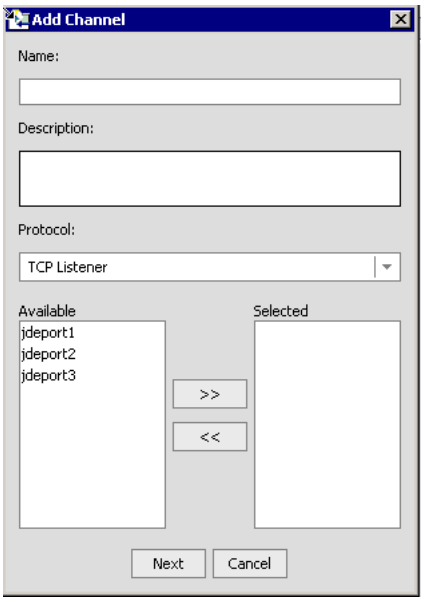
Creating a Channel Using Application Explorer

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

1. Click the **Event Adapters** 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 box opens.



a. Type a name for the channel, for example, NewChannel.

b. Type a brief description.

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

d. Select an event port from the list of available ports. To select more than one, hold down the **Ctrl** key and click the ports.

e. Click the **double right (>>)** arrow button to transfer the port(s) to the list of selected ports.

4. Click **Next.**

The TCP Listener dialog box opens with the Basic tab active.

a. Enter the parameters that are specific to your J.D. Edwards environment.

b. Click the **preparser** tab.

c. Enter the required parameters.

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

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

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

Is Length Prefix	For J.D. Edwards OneWorld events that send data back that is not in XML format. The TCP/IP event application must prefix the data with a 4-byte binary length field when writing the data to the TCP/IP port.
Is XML	For J.D. Edwards OneWorld events that send data back in XML format. No preparser is required.

Parameter	Description
Is Keep Alive	Maintains continuous communication between the event transaction and the channel.
User id*	A valid user ID for J.D. Edwards OneWorld.
User password*	The password associated with the user ID.
JDE Environment*	The J.D. Edwards OneWorld environment, for example, DU7333. For more information about this parameter, see your J.D. Edwards OneWorld documentation or ask your OneWorld system administrator.
Application	XMLInterop or the application name in J.D. Edwards OneWorld. Optional.
Server IP address*	Name of the server on which J.D. Edwards OneWorld is running. This can be the name of the server, for example, JDEOW, or its IP address, for example, 123.45.67.89.
Server port*	Port number on which the server is listening, for example, 6009.

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

5. Click **OK.**

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

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

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

   The channel becomes active.

   The X that was over the icon in the left pane disappears.

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

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

Editing a Channel

To edit a channel:

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

   The Edit channels pane opens.

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

Deleting a Channel

To delete a channel:

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

   A confirmation dialog box opens.

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

   The channel disappears from the list in the left pane.

The OneWorld Event Listener

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

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

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

Configuring the OneWorld Event Listener

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

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

The OneWorld Event listener includes the following components:

- The listener exit `(IWOEvent)`.

  The file extension you use depends on your operating system, for example, for Windows, the exit is `IWOEvent.dll`.
- The listener configuration file `(iwoevent.cfg)`.
- The outbound agent `(XDJdeOutboundAgent)`.

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

Creating the iwoevent.cfg File

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

To create the `iwoevent.cfg` file:

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

2. Create an environment variable, *IWOEVENT_HOME*, to point to the directory containing the `iwoevent.cfg` file.
   - On Windows: Add *IWOEVENT_HOME* to the system environment variables.
   - On UNIX: Add the following command to your start-up script:

     ```
     export IWOEVENT_HOME =/directory_name
     ```

Adding Connection Information

The OneWorld Event listener requires connection information for the associated adapter to initiate events properly. This information is contained in the `iwoevent.cfg` file. You must create this file and add the connection information to it.

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

A sample `iwoevent.cfg` file is installed on the J.D. Edwards server and is in the root path. The `iwoevent.cfg` file has three distinct sections:

- Common
- Alias
- Trans

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

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

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

aliasname
Is the symbolic name given to the connection.

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

port
Is the port defined for the Adapter for J.D. Edwards OneWorld (required).

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

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`*

jdeTransactionName
Is the JDE-defined name for the outbound transaction.

alias1,alias2,aliasn
Is the list of aliases to which the transactions are sent.

Adding Connection Information to iwoevent.cfg

To add connection information to the iwoevent.cfg file:

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

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

on
Sets the tracing to on.

off
Sets the tracing to off. Off is the default value.

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

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

3

Deployment and Integration

This chapter describes Oracle Containers for J2EE (OC4J) deployment and integration with OracleAS Integration InterConnect.

This chapter discusses the following topics:

- Oracle OC4J Integration
- OracleAS Adapter BSE Integration with OracleAS Integration InterConnect

See Also:

- *Oracle Application Server Integration InterConnect User's Guide*
- *Oracle Application Server Containers for J2EE User's Guide*

Oracle OC4J Integration

The following topic shows the basic commands for using CCI with packaged application adapters.

See Also:

- "OC4J Containers" in *Oracle Application Server Adapter Concepts*
- "Deployment and Integration through J2CA" in *Oracle Application Server Adapter Concepts*

Application Development Using the CCI API

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

Step 1. Obtain the Connection Factory

The connection factory is obtained by JNDI lookup.

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

Step 2. Obtaining a Connection for the Adapter

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

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

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

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

The iWAFConnectionSpec can be made to invoke an interaction with J.D. Edwards OneWorld by specifying the adapter name and configuration parameters in the ConnectionSpec. For example,

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

In this snippet, `ADAPTER` and `TARGET` refer to the adapter being invoked, in this case J.D. Edwards OneWorld, and the name of a target defined in Application Explorer. See"Complete Code Sample" on page 3-4 for more information.

Step 3. Create interaction with interactionSpec for runtime

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

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

Step 4. Create Input Record and Execute Interaction

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

A standard JCA Indexed Record is used in this example:

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

A special record is supported in this example:

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

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

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<jdeRequest type="callmethod" user="JDE" pwd="JDE" environment="DV7333" session=""
sessionidle="">
<callMethod name="AddressBookMasterMBF" app="" runOnError="" trans="">
<params>
  <param name="cActionCode">A</param>
  <param name="cUpdateMasterFile">1</param>
  <param name="cProcessEdits">1</param>
  <param name="cSuppressErrorMessages"/>
  <param name="szErrorMessageID" />
  <param name="szVersion">ZJDE0001</param>
  <param name="mnSameAsExcept" />
  <param name="mnAddressBookNumber" id="1" />
  <param name="szLongAddressNumber" />
  <param name="szTaxId" />
  <param name="szSearchType">C</param>
  <param name="szAlphaName">Ed Marsh</param>
  <param name="szSecondaryAlphaName">Ed Marsh</param>
  <param name="szMailingName">Ed Marsh</param>
  <param name="szSecondaryMailingName">Ed Marsh</param>
  <param name="szDescriptionCompressed" />
  <param name="szBusinessUnit"/>
  <param name="szAddressLine1">1 Main St.</param>
  <param name="szAddressLine2">Apt 101</param>
  <param name="szAddressLine3"/>
  <param name="szAddressLine4"/>
  <param name="szPostalCode">75000</param>
  <param name="szCity">AnyTown</param>
  <param name="szCounty"/>
  <param name="szState">TX</param>
  <param name="szCountry">US</param>
  <param name="szCarrierRoute" />
  <param name="szBulkMailingCenter" />
  <param name="szPrefix1" />
  <param name="szPhoneNumber1">917-339-6491</param>
  <param name="szPhoneNumberType1" />
  <param name="szPhoneAreaCode2" />
  <param name="szPhoneNumber2" />
  <param name="szPhoneNumberType2" />
```

```
  <param name="cPayablesYNM">Y</param>
  <param name="cReceivablesYN">N</param>
  <param name="cEmployeeYN">N</param>
  <param name="cUserCode" />
  <param name="cARAPNettingY">N</param>
  <param name="cPersonCorporationCode" />
  <param name="szCertificate" />
  <param name="szAddlIndTaxID" />
  <param name="szCreditMessage" />
  <param name="szLanguage" />
  <param name="szIndustryClassification" />
  <param name="cEMail" />
  <param name="szCategoryCode01" />
  <param name="szRemark"/>
  <param name="szUserReservedCode"/>
  <param name="jdUserReservedDate"/>
  <param name="mnUserReservedAmount"/>
  <param name="mnUserReservedNumber"/>
  <param name="szUserReservedReference"/>
  <param name="jdDateEffective" />
  <param name="szRemark1" /> </params>
<onError abort=""/>
</callMethod>
</jdeRequest>
```

Complete Code Sample

The following is a sample of the complete code:

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

/**
 * The purpose of this sample is to illustrate how to use the IWAF Universal
 * JCA connector.
 *
 * Author: Marcelo Borges
 * Date: August, 2004
 */
public class IWAFJCASimple {

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

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

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

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

// 1. Getting the Connection factory through JNDI lookup
// -----------------------------------------------------
  InitialContext context = new InitialContext();
  ConnectionFactory cf = (ConnectionFactory)context.lookup(iwayJndi)
  // 2. Getting a connection for a particular adapter target, in this case JDE
// -----------------------------------------------------
  IWAFConnectionSpec cs = new IWAFConnectionSpec();
```

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

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

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

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

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

  } // main()

}
```

Creating a Managed Connection Factory

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

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

The parameters are defined in the following table:

Parameter Name	Description
IWayHome	The base installation directory for the OracleAS packaged application adapter.

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

Creating Multiple Managed Connection Factories

To establish Multiple Managed Connection Factories, you must edit the `OC4J-ra.xml` file to add the required information. The file can contain more than one `<connector-factory>` element. By adding more `<connector-factory>` elements, you can create Multiple Managed connection factories. For example, the default jca_sample configuration in Application Explorer is represented in the `OC4J-ra.xml` file as follows:

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

To create Multiple Managed Connection Factories, you must add new <connector-factory> nodes in the file. For example:

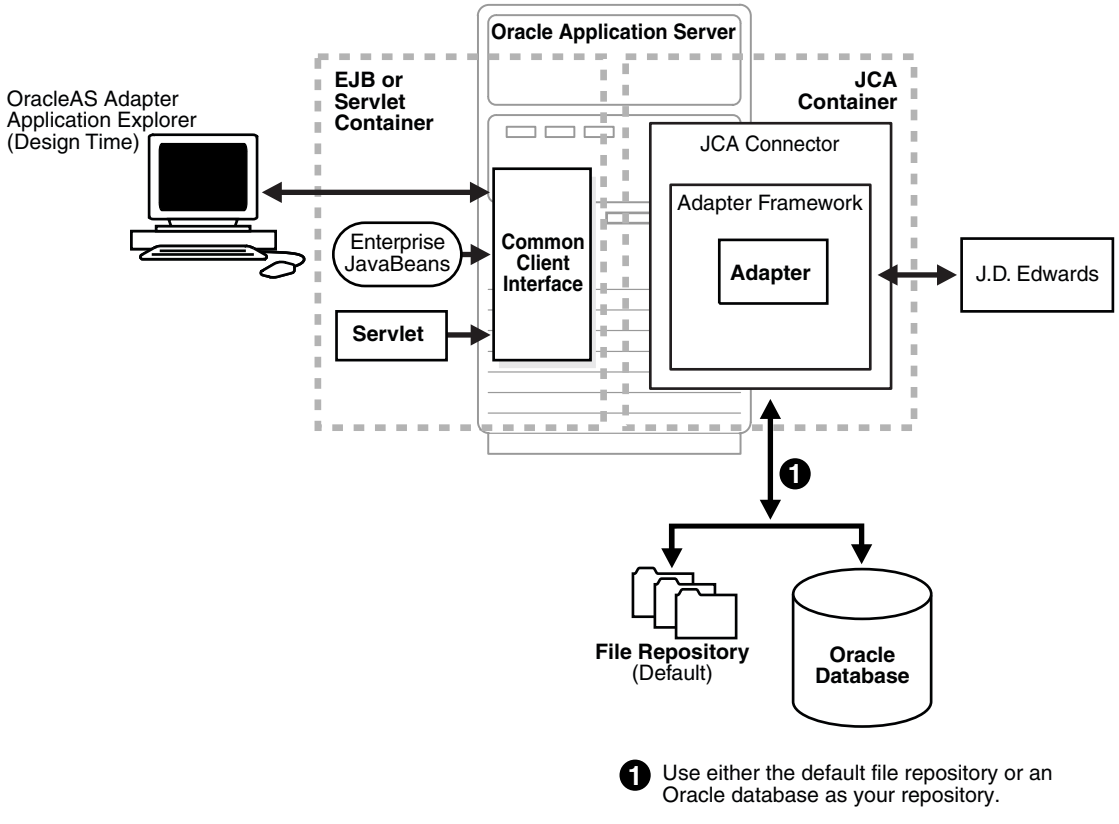
```
<?xml version="1.0"?>
<!DOCTYPE oc4j-connector-factories PUBLIC "-//Oracle//DTD Oracle Connector
9.04//EN"
"http://xmlns.oracle.com/ias/dtds/oc4j-connector-factories-9_04.dtd">
<oc4j-connector-factories>
   <connector-factory location="eis/OracleJCAAdapter/DefaultConnection1"
connector-name="IWAFJCA10">
```

```
   <config-property name="IWayHome" value="../../adapters/application"/>
   <config-property name="IWayConfig" value="jca_sample"/>
   <config-property name="IWayRepoURL" value=""/>
   <config-property name="IWayRepoUser" value=""/>
   <config-property name="IWayRepoPassword" value=""/>
   <config-property name="logLevel" value="debug"/>
   </connector-factory>
<connector-factory location="eis/OracleJCAAdapter/DefaultConnection2"
connector-name="IWAFJCA10">
   <config-property name="IWayHome" value="../../adapters/application"/>
   <config-property name="IWayConfig" value="jca_sample2"/>
   <config-property name="IWayRepoURL" value=""/>
   <config-property name="IWayRepoUser" value=""/>
   <config-property name="IWayRepoPassword" value=""/>
   <config-property name="logLevel" value="debug"/>
   </connector-factory>
</oc4j-connector-factories>
```

Oracle Application Server Adapter JCA Architecture

Figure 3–1 shows deployment of the Connector to the Oracle Application Server. In a runtime service scenario, an Enterprise Java Bean (EJB), Servlet, or Java program client makes CCI calls to JCA 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 JCA Architecture



OracleAS Adapter BSE Integration with OracleAS Integration InterConnect

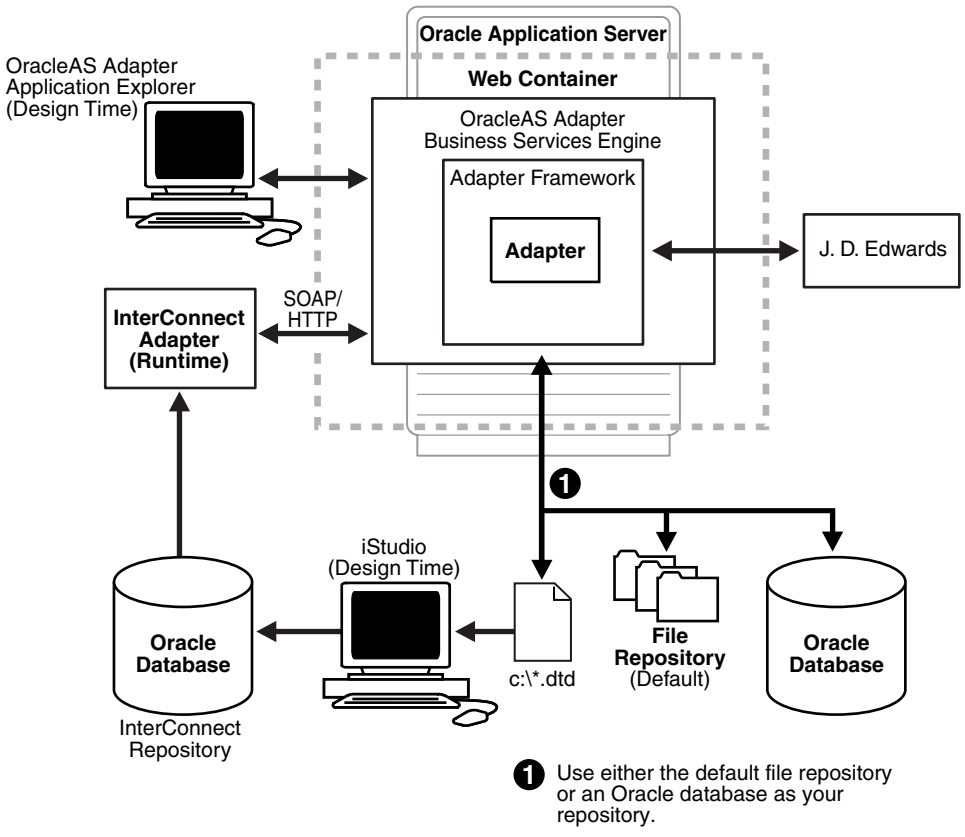
See Also: "Deployment and Integration through OracleAS Web Services" in *Oracle Application Server Adapter Concepts*

BSE Architecture as Deployed to Oracle Application Server

Figure 3–2 shows adapter framework deployment with BSE to OracleAS Integration InterConnect. In a runtime service scenario, the OracleAS Integration InterConnect EIS Adapter Plugin (EIS Adapter Plugin) receives DTD-compliant XML from the **agent** component of the EIS Adapter Plugin. The EIS Adapter Plugin strips runtime information from the XML, wraps the XML in a SOAP envelope, and sends the result to BSE, including the runtime information in the SOAP request. BSE receives the request, removes the envelope, retrieves Web service method metadata, including adapter and connection information from the repository, and makes the adapter request.

BSE receives the adapter response, wraps the response XML in a SOAP envelope, and returns it to the EIS Adapter Plugin. The EIS Adapter Plugin then strips the SOAP envelope, strips the namespace prefix, if present, and passes the DTD-compliant XML to the agent component of the EIS Adapter Plugin.

Figure 3–2 BSE Architecture as Deployed to Oracle Application Server



Upon installation of the Oracle Web Services Adapter, an `adapter.ini` file is created. The file consists of all the initialization parameters that the adapter reads at startup. Some of the parameters in this file are configurable.

See Also: *Oracle Application Server Adapters Installation Guide*

4

Examples

This chapter contains the following examples:

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

Prerequisites

The following components must be configured:

- OracleAS Adapter for J.D. Edwards OneWorld installed on the Oracle Application Server.
- OracleAS Integration InterConnect Adapter Plugin for EIS.

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

Configuration Steps

The examples present all the configuration steps necessary for demonstrating service and event integration with J.D. Edwards. The following cross references are given to identify where more information can be obtained.

1. Configure the OracleAS Adapter for J.D. Edwards OneWorld for services and events. See Chapter 2, "Adapter Configuration Using Oracle Application Explorer" for more information.
2. Configure OracleAS Integration InterConnect iStudio for service and event interactions. For more information, see the following service and event steps.

J.D. Edwards OneWorld Service Integration

This topic illustrates J.D. Edwards event integration. The procedures describe design time and runtime.

OracleAS Integration InterConnect Design Time

The following procedures describe how to start the repository and create a common view and then, publish and subscribe an event.

Starting the Repository

To start the repository, double-click the `start.bat` file located in the following directory:

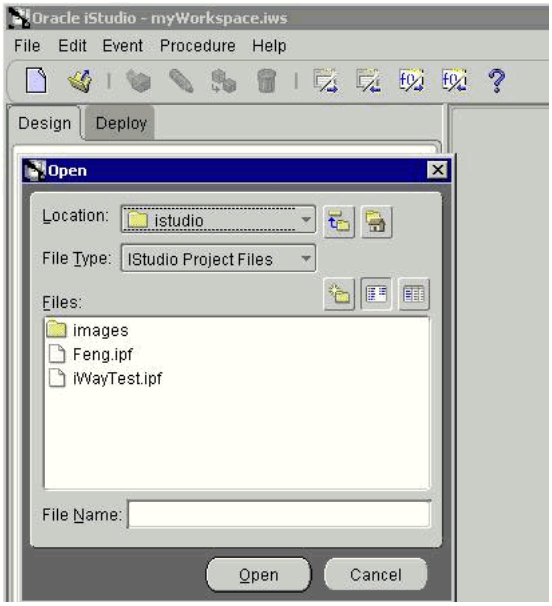
OracleAS_home`\repository\start.bat`

OracleAS_home
Is the directory where the Oracle Application Server is installed.

Creating a Common View

To create a Common View:

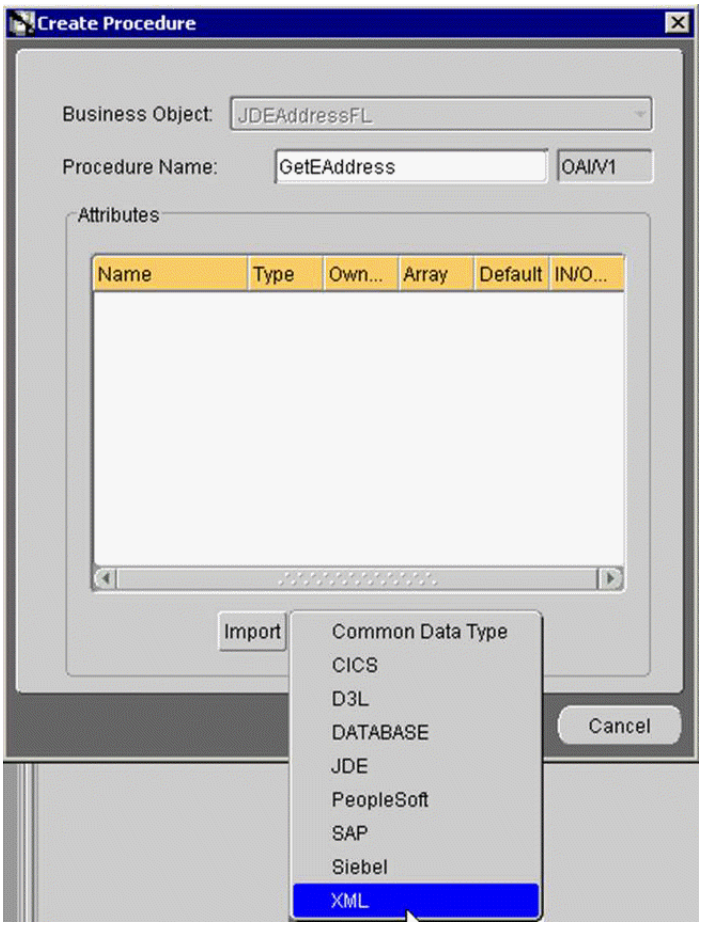
1. Start Oracle iStudio.



2. Open a new project.
3. Open **Common Views** and **Business Objects.**
4. Create a Business Object called **JDEAddressFL** and a new procedure under **GetEAddress.**

> **Note:** The procedure name must be the root element of the DTD generated from Application Explorer. In this example, the root element in the DTD is `GetEAddress`.

The Create Procedure window opens.

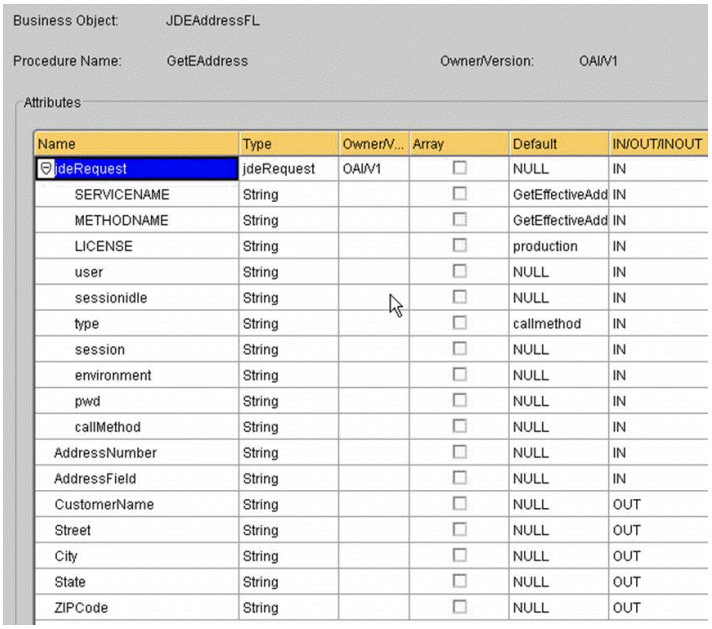


5. Click **Import** and select **XML** from the list.
6. Open the DTD generated from Application Explorer.

   Click **OK.**

   The **Choose Import Type** dialog box opens.
7. Choose the **IN arguments** radio button, and click **OK.**
8. Select **jdeRequest,** and ensure your parameters are similar to Figure 4–1 for both the In and Out parameters.

Figure 4–1 In and Out Parameters



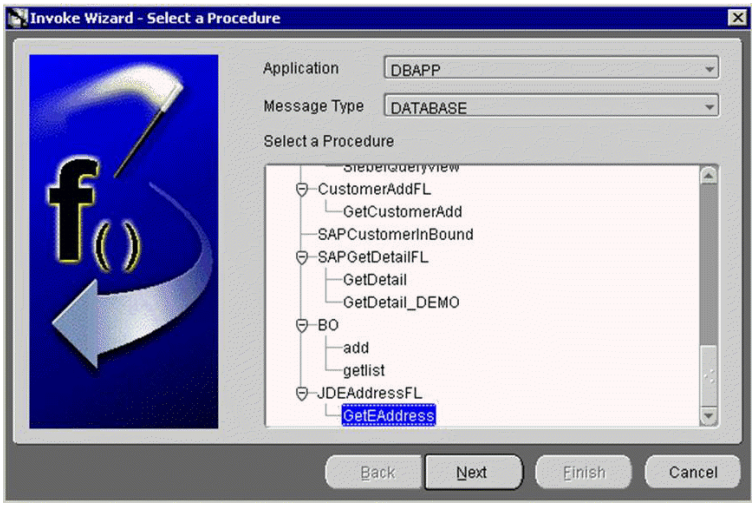
9. Click **Save.**

Invoking a Procedure

To invoke a procedure:

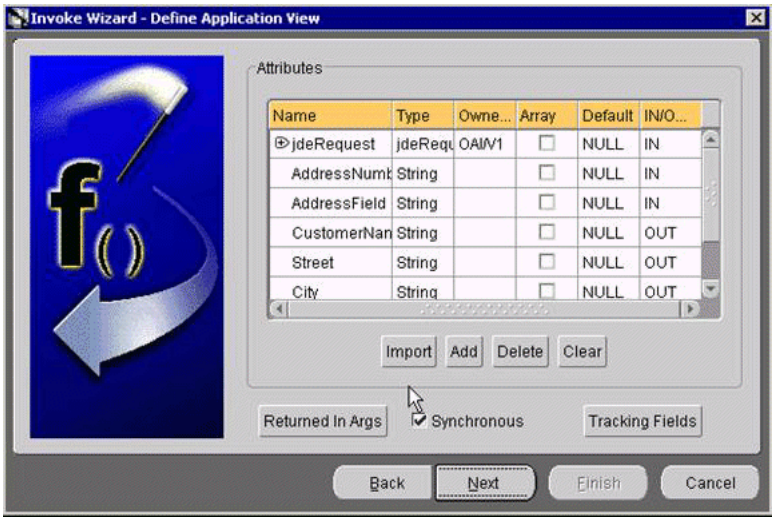
1. Create a new application called **DBAPP.**
2. Right-click **Invoked Procedures** and select **New.**

   The Invoke Wizard - Select a Procedure dialog box opens.

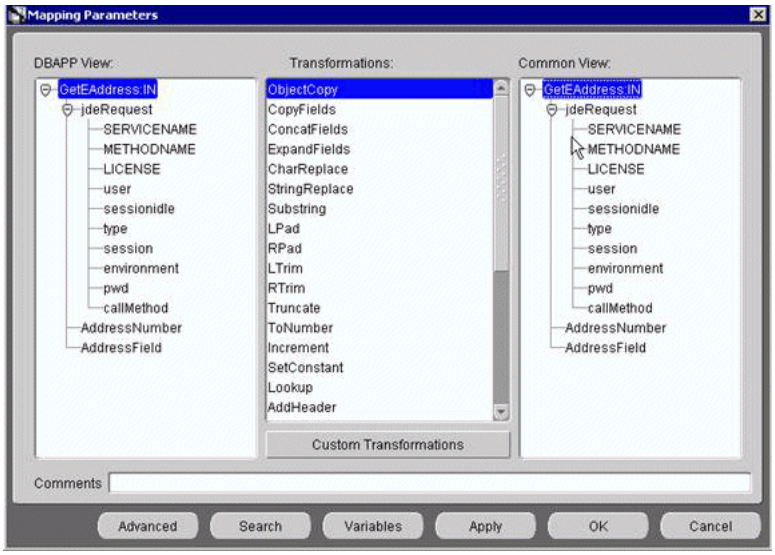


3. From the **Message Type** list, select **DATABASE.**
4. As the procedure, choose **GetEAddress** under **JDEAddressFL.**
5. Click **Import** and select **Common View.**

   The structure is loaded as follows. Because this is a request and response, ensure that **Synchronous** is selected.



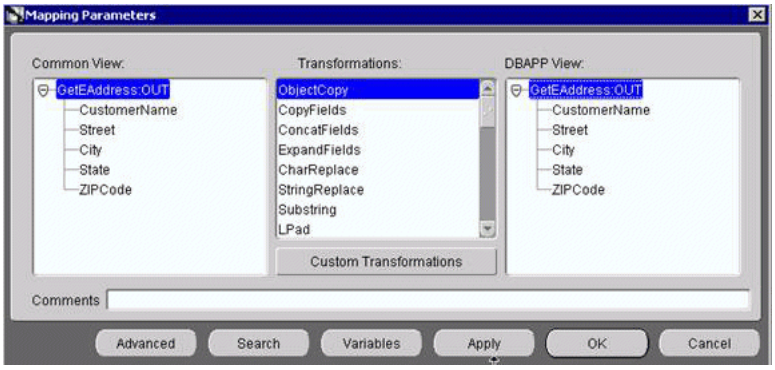
6. Click **Next**, and then **New** to create a mapping between the Common View and the Application View for the In parameters.



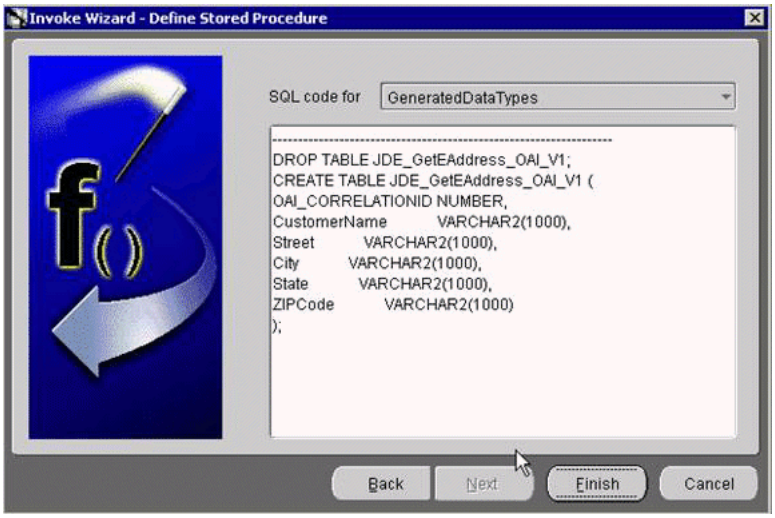
In this case, the Application View and the Common View have the same structure and can be mapped using the ObjectCopy transformation.

7. Click **Apply** and then **OK.**

The second Mapping Parameters dialog box opens.



8. Click **Apply** and then **OK.**
9. The Define Stored Procedures dialog box opens.



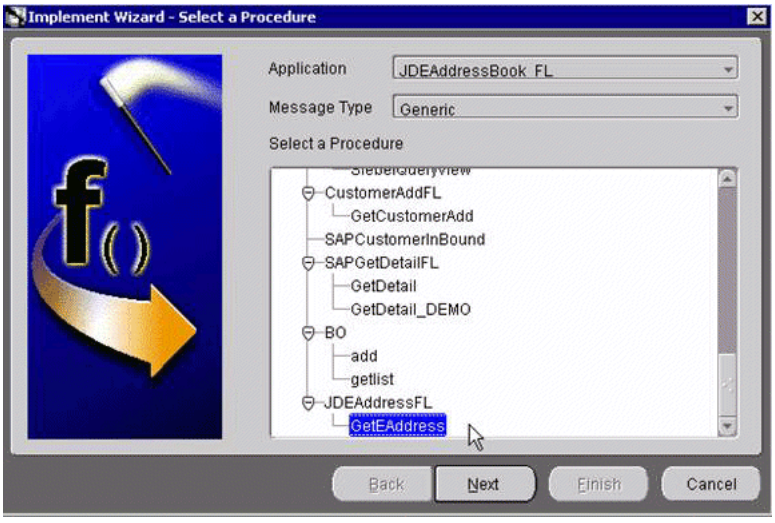
Some SQL code is automatically generated.

10. Click **Finish.**

Implementing a Procedure

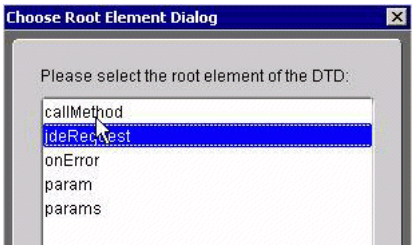
In this implemented procedure, a new application called `JDEAddressBook` is created.

1. Create a new application named `JDEAddressBook`.
2. Expand the application and right-click **Implemented Procedure.**
3. To create an implemented procedure, select **New.**



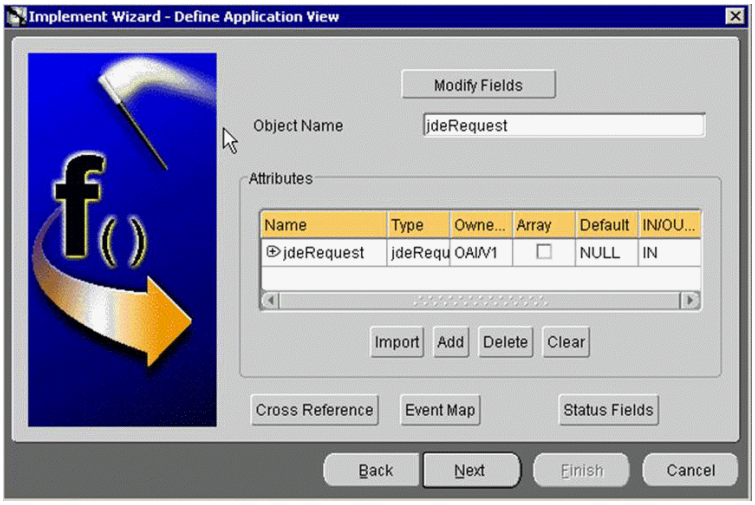
4. Select **Generic** as the message type.
5. Expand **JDEAddressFL** and select **GetEAddress.**
6. Click **Next.**
7. Click **Import** and select **XML.**

8. Navigate to the location of the request and response DTDs generated by Application Explorer and import both into iStudio.



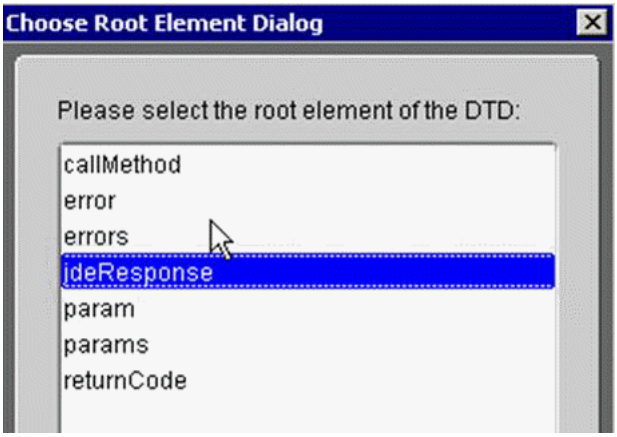
9. Select **jdeRequest** as the root element of the DTD.
10. Choose the **IN arguments** option button, and click **OK.**

    The Implement Wizard - Define Application View window opens.



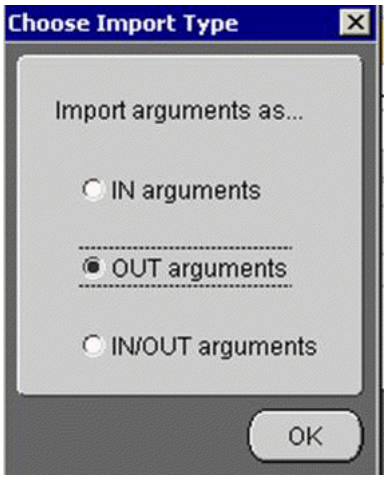
11. Click **Import,** then click **Next.**

    The Choose Root Element Dialog box opens.



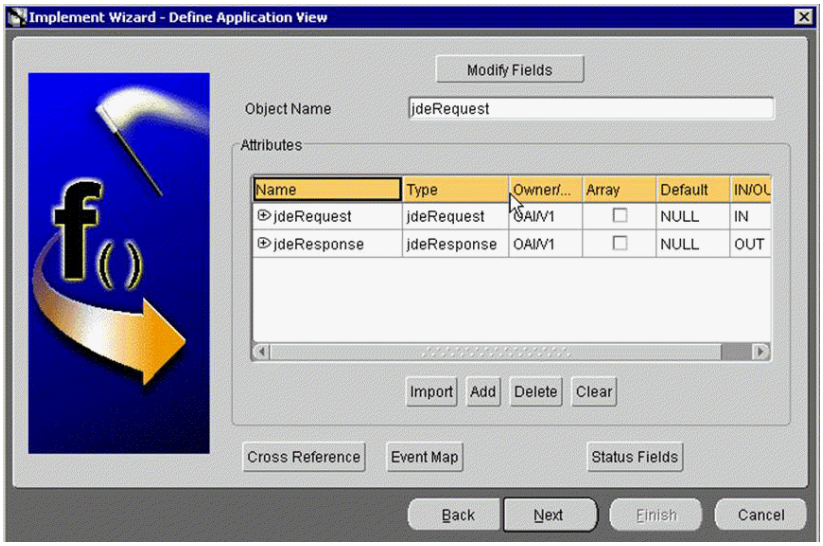
12. Choose **jdeResponse** and click **Next.**

    The Choose Import Type dialog box opens.



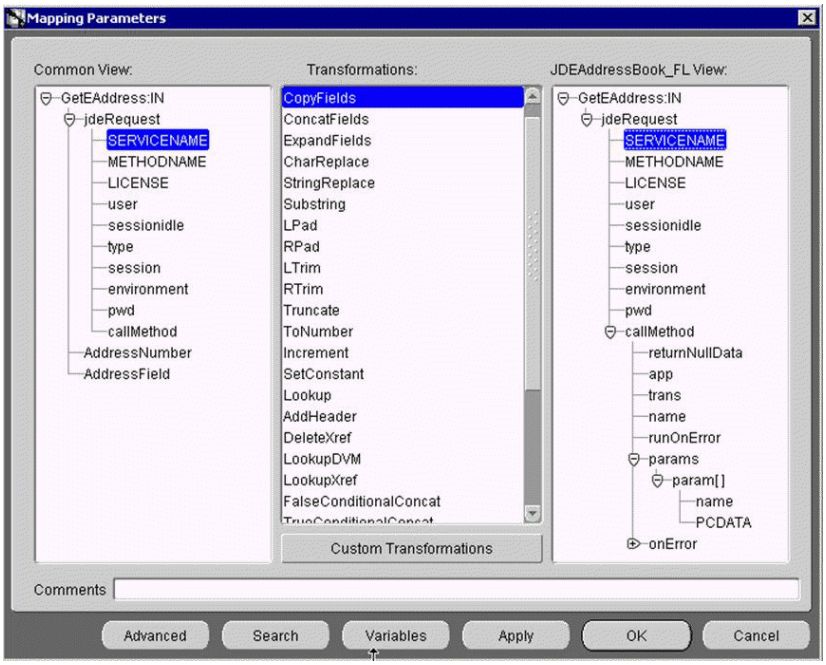
13. Select **OUT arguments**, and click **Next.**

    The Define Application View window opens.



14. Click **Next.**

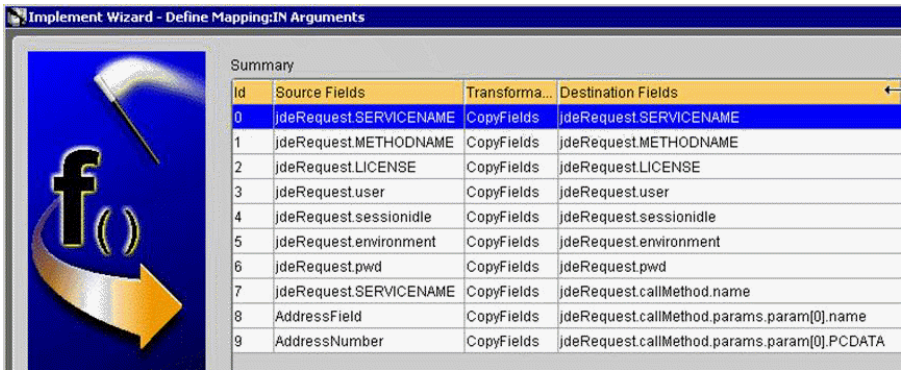
    The Mapping Parameters window opens.



In this example, the Application View and Common View have the same structure. All the attributes can be mapped by using the ObjectCopy transformation.

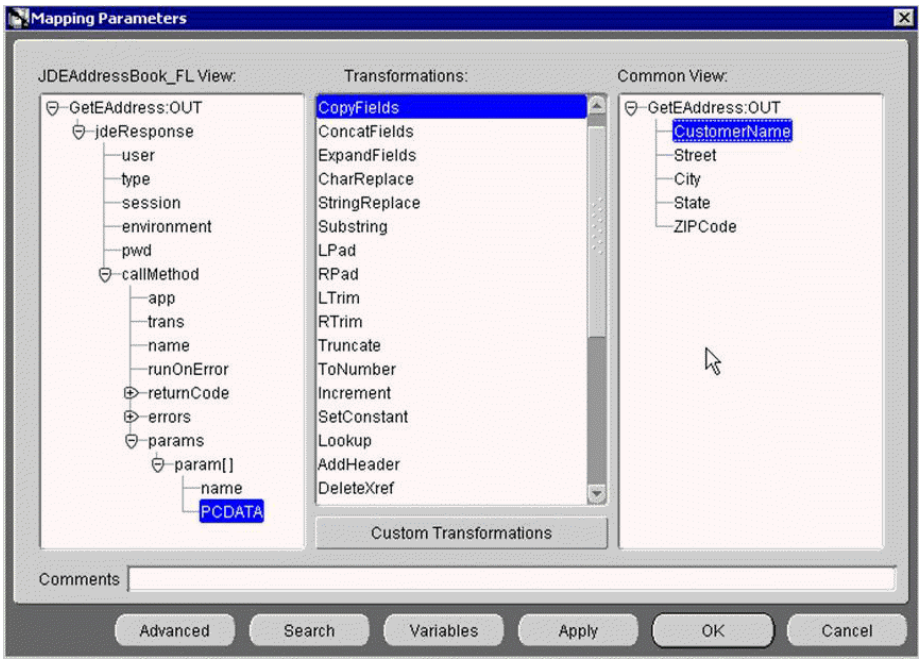
15. Select **Copy Fields** in the **Transformations** field, and click **OK.**

    The Define Mapping:IN Arguments window opens.



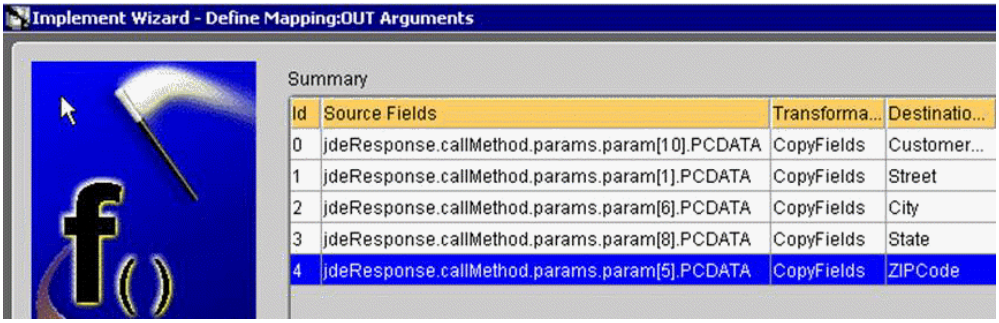
16. Click **Next.**

    The Mapping Parameters window opens for the out parameters.



17. Ensure **Copy Fields** is selected in the **Transformation** field, then click **OK**.

    The Define Mapping:OUT arguments window opens.



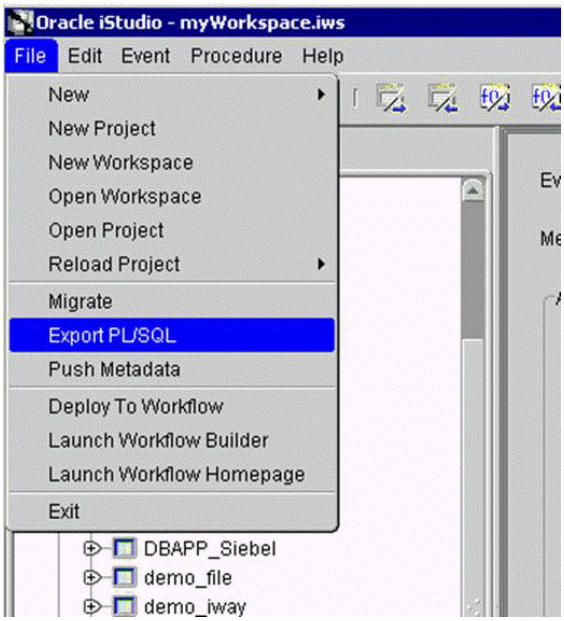
18. Click **Next,** then **Finish.**

    The application definition for an invoked procedure is now complete.

Exporting PL/SQL Code from iStudio

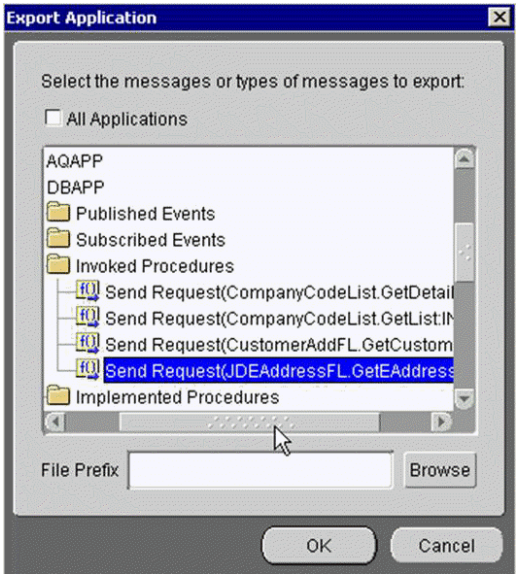
You must export the PL/SQL code created in "Invoking a Procedure" on page 4-4 and execute it against the appropriate schema. In this example, the schema used is DBAPP_JDEAddressFL.

To export PL/SQL code from iStudio:



1. In iStudio, click **File** and **Export PL/SQL.**

   The Export Application dialog box opens.



   a. Select the application from which to export PL/SQL.

   b. Type or browse to the file prefix (path to the application).

2. Click **OK.**

   In this example, two SQL scripts are created:

   - `DBAPP_JDEAddressFLTYPES.sql`
   - `DBAPP_JDEAddressFL.sql`

3. Log on to the database with the appropriate privileges (in this example, DBAPP_JDEAddress) and execute the following in the order given:

   - `DBAPP_JDEAddressFLTypes.sql`

- DBAPP_JDEAddressFL.sql

4. Create another stored procedure, JDEADDRESSFL_EXE, in the same schema. It executes at runtime to create the database message that is sent to the hub.

```
CREATE OR REPLACE  PROCEDURE "DBAPP"."JDEADDRESSFL_EXE" (
servicename LONG,
methodname  LONG,
license     LONG,
customerid  LONG
)
AS
  moid NUMBER;
  aoid NUMBER;
  coid NUMBER;
  businessname LONG;
  address LONG;
  city LONG;
  state LONG;
  phone LONG;
  country LONG;
  detailid NUMBER;
BEGIN
  JDEAddressFL.crMsg_GetEAdress_OAI_V1(moid, aoid);
  jdeid := JDEAddressFL.cr_jdeRequest_jdeRequest
(servicename,methodname,license,username,sessionidle, calltype, sessionid,
environment, pwd, callmethod, moid,aoid);
  coid := JDEAddressFL.inv_GetEAddress_OAI_
V1(moid,'DBAPP','',customername,street,city,state,zipcode);
COMMIT;
END;
```

Edit Adapter.ini

Add the following two lines to adapter.ini for the Oracle Application Server Adapter for J.D. Edwards OneWorld:

```
//Bridge Class
bridge_class=com.iwaysoftware.iwbridge.IWBridge

//IBSE URL
ibse_url=http://hostname:port/ibse/IBSEServlet/XDSOAPRouter
```

hostname
Is the URL of the server.

port
Is the port number.

OracleAS Integration InterConnect Runtime

The following topic describes how to verify service integration using the OracleAS Adapter for J.D. Edwards OneWorld.

Verifying Service Integration

To verify service integration:

1. Start the Oracle Application Server or ensure that the server is running.
2. Restart OC4J, if required, by executing the following command:

```
\OracleAS_home\opmn\bin\opmnctl  stopproc process-type=home
\OracleAS_home\opmn\bin\opmnctl  startproc process-type=home
```

3. Check the status of OC4J by executing the following command:

```
\OracleAS_home\opmn\bin\opmnctl  status
```

The expected output is a list of the processes in the instance, as in the following:

iAS-component	Process-type	PID	Status
DSA	DSA	N/A	Down
HTTP_Server	HTTP_Server	1592	Alive
LogLoader	logloaderd	N/A	Down
dcm-daemon	dcm-daemon	3016	Alive
OC4J	home	3496	Alive
WebCache	WebCache	1800	Alive
WebCache	WebCacheAdmin	1804	Alive

4. Invoke and implement the adapter by executing the following commands:

```
\InterConnect_HOME\oai\9.0.4\adapters\JDEAddressBook_FL\start.bat
\InterConnect_HOME\oai\9.0.4\adapters\DBAPP\start.bat
```

5. Log on to SQL*Plus with `DBAPP` and execute the following command:

```
exec
jdeaddressfl_exe
('GetEffectiveAddress','GetEffectiveAddress','test',’JDE’,"'callmethod","’DV733
3’,’JDE’,'GetEffectiveAddress',’4242’,’mnAddressNumber’);
```

Figure 4–2 shows the `JDEAddress_FL` example. It receives a reply from J.D. Edwards OneWorld and returns the reply to the hub.

Figure 4–2 JDEAddress_FL Test Results

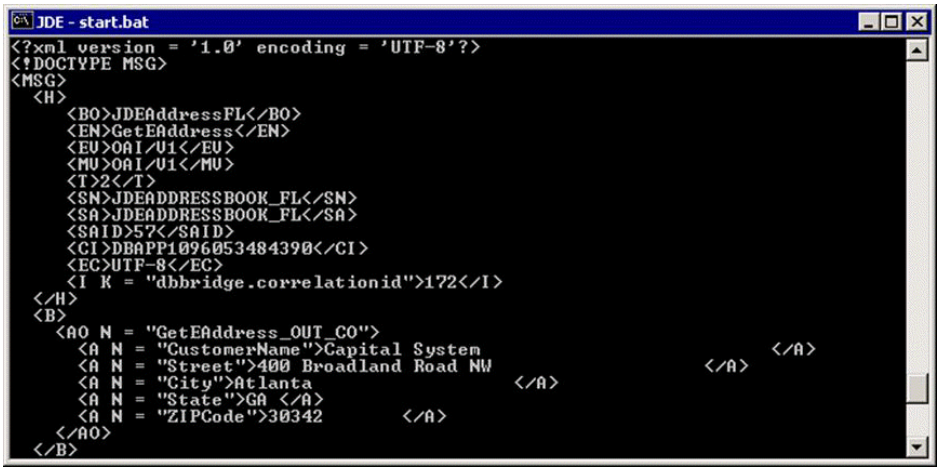
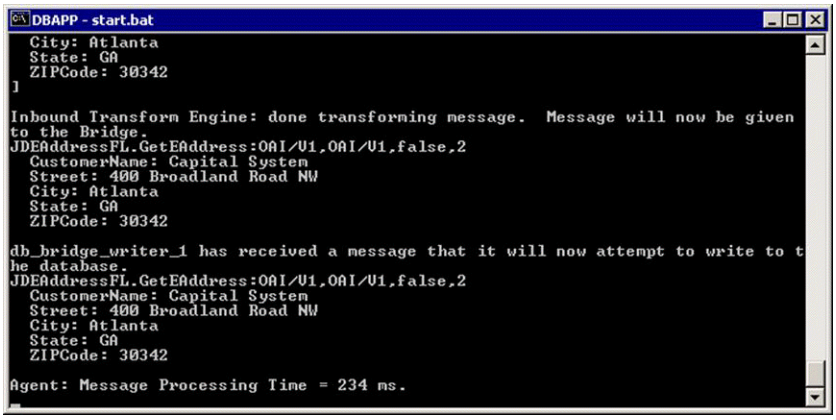


Figure 4–3 shows the `DBAPP` example. It receives a reply from the hub and writes the data to the database table.

Figure 4–3 DBAPP Test Results



J.D. Edwards OneWorld Event Integration

This topic and the example illustrate how the OracleAS Adapter for J.D. Edwards OneWorld integrates with J.D. Edwards OneWorld to receive event data. The procedures describe design time and runtime. In the example, a JDE event occurs as a result of a sales order event in the J.D. Edwards OneWorld system. The adapter receives the J.D. Edwards OneWorld event customer data and disposes the data to an RMI event port. The RMI server resides on the OracleAS Integration InterConnect Hub. An OracleAS Database Adapter on the OracleAS Integration InterConnect Hub subscribed to this event receives the customer data, transforms the event data, and then inserts the data into a database table. The design time and runtime procedures are outlined in the following sections.

J.D. Edwards Transaction Sales Order

To create a DTD for a J.D. Edwards event, you must:

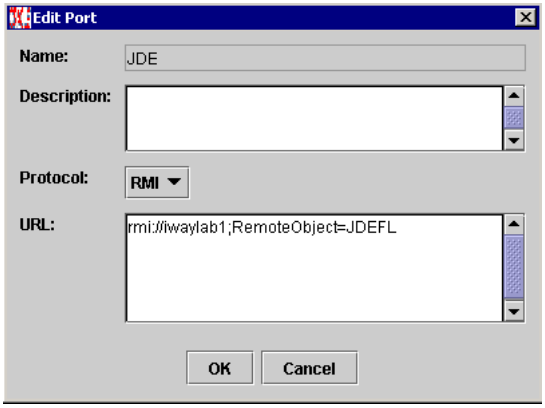
- Create a port in Application Explorer. See "Creating a Port in Application Explorer".
- Create a channel in Application Explorer. See "Creating a Channel in Application Explorer".
- Trigger an event from the J.D. Edwards system.
- Capture the XML event payload in the BSE log.
- Create a DTD based on the J.D. Edwards XML message using third party tools, such as XML Spy.

Creating a Port in Application Explorer

To create a port:

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

   The Edit Port dialog box opens.



3. Enter a description in the **Description** field (optional).
4. Select **RMI** from the **Protocol** list.
5. Enter the URL for the server in the **URL** field, and click **OK**.

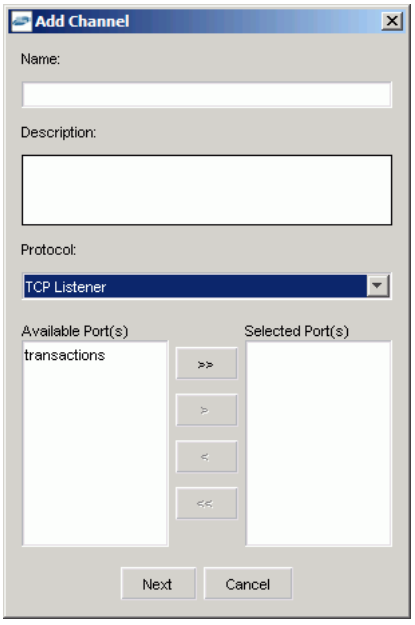
   The port is created, and shows under the Ports node.

Creating a Channel in Application Explorer

To create a channel:

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

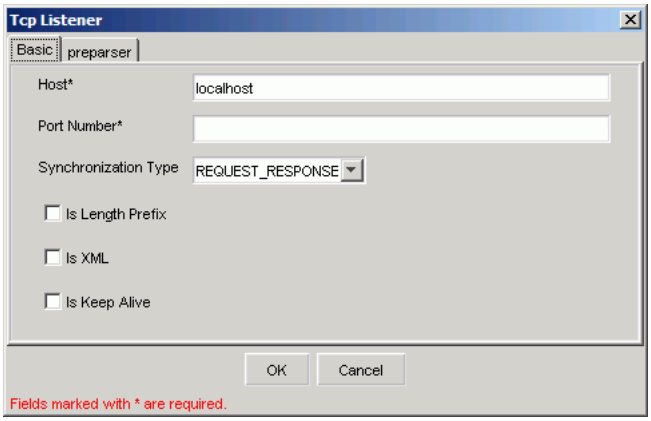
   The Add Channel dialog box opens.



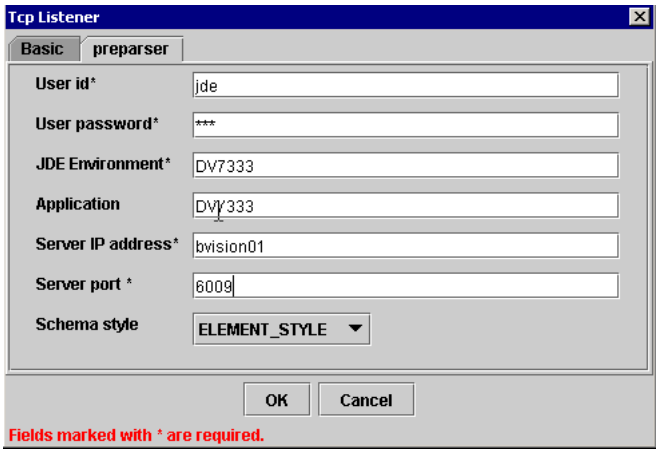
3. In the **Name** field, enter a descriptive name for the channel.
4. Enter a description in the **Description** field (optional).
5. Choose a protocol for your channel from the **Protocol** list.
6. In the Available Port(s) field, select the port or ports you wish to associate with the channel, and click the right arrow (>) button. To add all the ports, click the double right arrow button (>>).

7. Click **Next**.

   The dialog box opens for the selected listener.



8. Enter the location of the server in the **Host** field.
9. Enter the port number of the channel in the **Port Number** field.
10. Select the Synchronization type from the **Synchronization Type** list.
11. 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.
12. Select **Is XML** for events that send data back in XML format. No preparser is required.
13. Select **Is Keep Alive** to maintain a continuous communication between the event transaction and the channel.
14. 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.

Parameter	Description
JDE environment*	The J.D. Edwards OneWorld environment, for example, DU7333. For more information about this parameter, see your J.D. Edwards OneWorld documentation or ask your OneWorld system administrator.
Application	XMLInterop or the application name in J.D. Edwards OneWorld. Optional.
Server IP address*	The name of the server on which J.D. Edwards OneWorld is running. This can be the name of the server, for example, JDEOW, or its IP address, for example, 123.45.67.89.
Server Port*	The port number on which the server is listening, for example, 6009.
Schema style	Choose a style from the list.

Click **OK.**

The channel is created, and shows under the Channels node.

15. Start the channel:
    1. Right-click the channel's node and select **Start.**

       The channel you created becomes active.
16. This triggers an event from the J.D. Edwards system.
17. Check the BSE log located at:

    `OracleAS_HOME\j2ee\home\applications\ws-app-adapter\ibse\ibselogs`
    for the XML event message.

Starting the Repository

To start the repository, double-click the `start.bat` file located in the following directory:

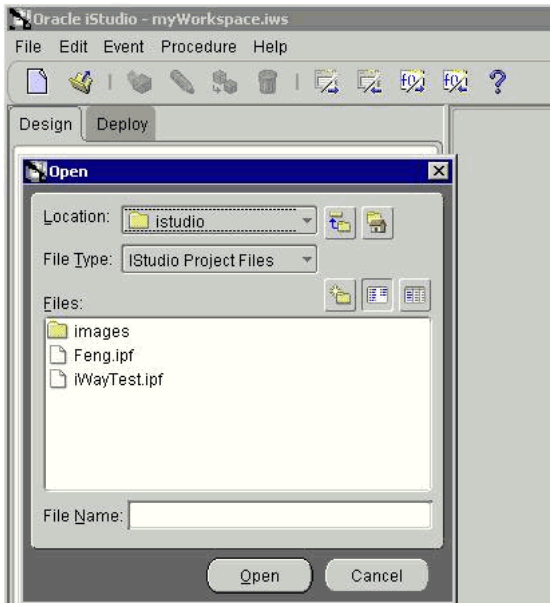
`OracleAS_home``\repository\start.bat`

OracleAS_home
Is the directory where the Oracle Application Server is installed.

Creating a Common View

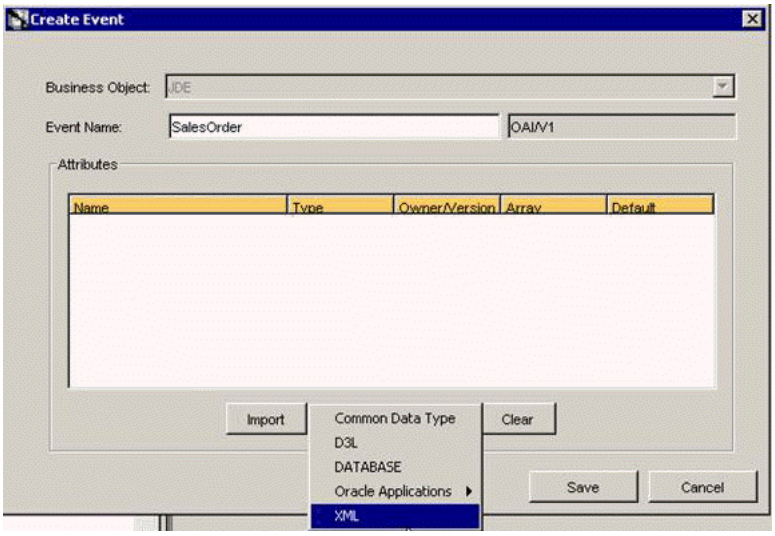
To create a Common View:

1. Start Oracle iStudio.



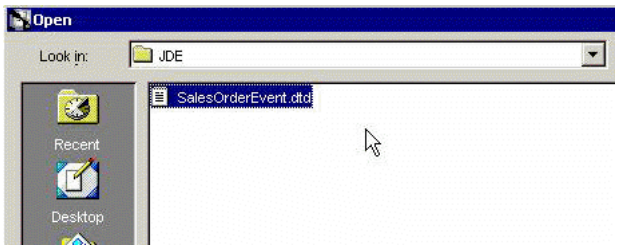
2. Open a new project.
3. Open **Common Views** and **Business Objects.**
4. Create a Business Object called **JDE** and a new event under **SalesOrder**.

   The **Create Event** dialog box opens.



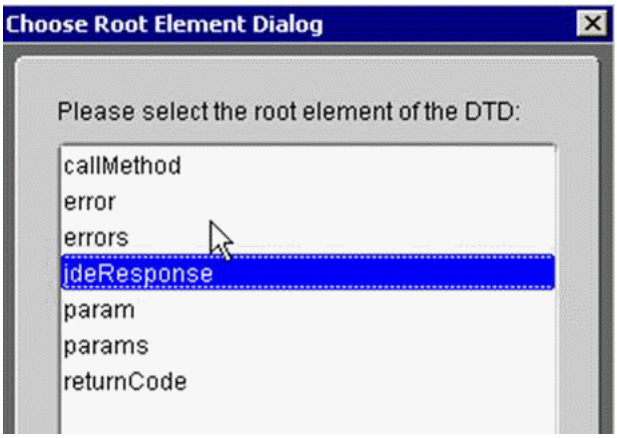
5. Click **Import**, then select **XML** as the import type.

   The **Open** dialog box opens.



6. Select the DTD generated from Application Explorer, and click **OK**.

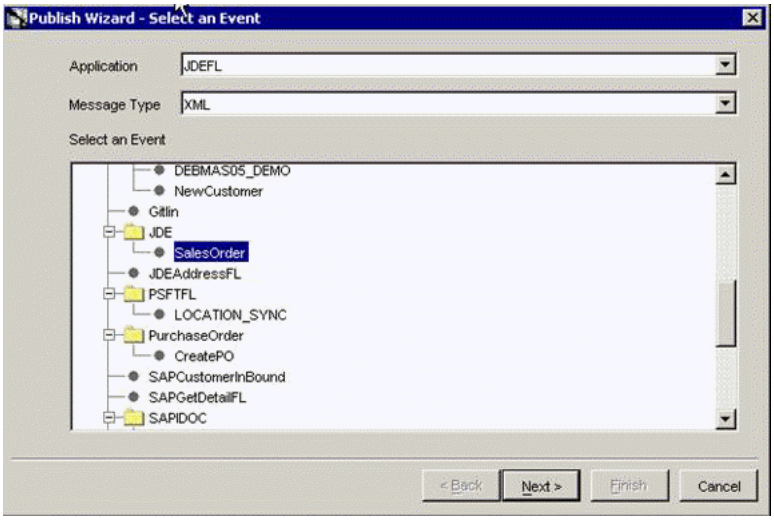
   The Choose Root Element Dialog dialog box opens.



7. Choose the Root element of the importing DTD, and click **OK**.
8. Click **Save.**

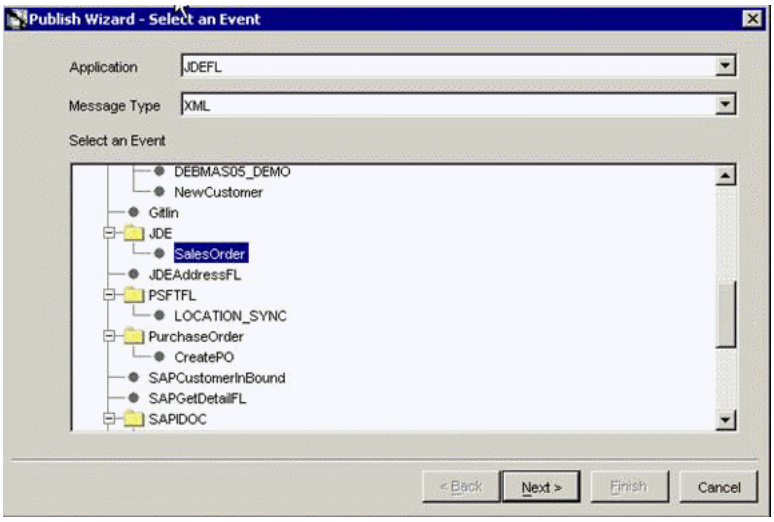
Publishing an Event Using the J.D. Edwards Adapter

To publish an event:



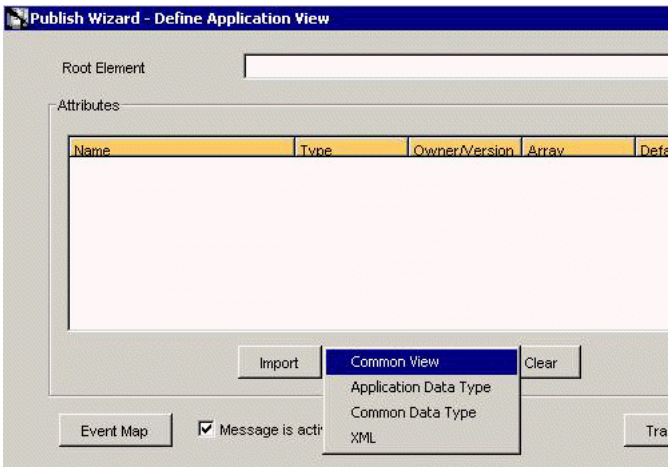
1. Create a new application named JDEFL. The application name must be uppercase.
2. Expand the new JDEFL node.
3. Right-click **Publish Events** and select **New** to create a Publish Event.

   The Publish Wizard - Select an Event dialog box opens.



4. From the Message Type list, select **XML**.
5. In the **Select an Event** field, expand the JDE node.
6. Select **SalesOrder** as the Event.
7. Click **Next**.

   The Publish Wizard - Define Application View dialog box opens



8. Click **Import**, and select **Common View**.

> **Note:** If the application message structure is different from the Common View structure, select **XML** to load a Application specific schema.

9. In the Root Element field, enter the root element of the XML message, jdeResponse in this example.
10. Click **Next**.

    The Mapping Parameters dialog box opens.

11. Click **New** to create a mapping between the Common View and Application View. In this example, the Application and Common View have the same structure. All attributes can be mapped using the **ObjectCopy** Transformation.
12. Click **OK**.
13. Click **Finish**.

The Application definition for the Publishing Event is now complete.

Runtime

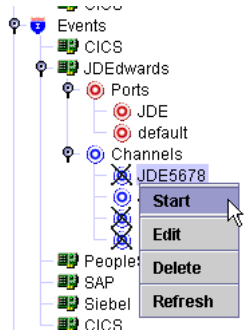
1. Start the Oracle Application Server or ensure that the server is running.
2. Restart OC4J, if required, by executing the following command:

```
\OracleAS_home\opmn\bin\opmnctl  stopproc process-type=home
\OracleAS_home\opmn\bin\opmnctl  startproc process-type=home
```

3. Check the status of OC4J by executing the following command:

```
\OracleAS_home\opmn\bin\opmnctl  status
```

4. In Application Explorer, expand the **JDEdwards** node.



5. Expand the **Channels** node.
6. Right-click the channel you wish to use, and select **Start**.
7. Invoke and implement the adapter by executing the following commands:

```
OracleAS_home\integration\interconnect\adapters\AQAPP\start.bat
OracleAS_home\integration\interconnect\adapters\JDEFL\start.bat
```

Verifying Results

The following topic describes how to verify event integration using the OracleAS Adapter for J.D. Edwards OneWorld.

Publishing Adapter (JDEFL) Log File

```
Tue Nov 16 18:42:05 GMT-05:00 2004: The message was sent to topic(s) {oai_hub_
queue=[AQAPP, DBAPP]}. Processing Time = 25,796 ms.

<?xml version = '1.0' encoding = 'UTF-8'?>
<!DOCTYPE MSG>
<MSG>
  <H>
     <BO>JDE</BO>
     <EN>SalesOrder</EN>
     <EV>OAI/V1</EV>
     <MV>OAI/V1</MV>
```

```
    <T>0</T>
    <SN>JDEFL</SN>
    <SA>JDEFL</SA>
    <SAID>21</SAID>
    <CI>JDEFL1100648500828</CI>
</H>
<B>
  <AO N = "SalesOrder_CO">
    <AO N = "jdeResponse">
      <A N = "type">trans</A>
      <A N = "user">jde</A>
      <A N = "xmlns">urn:Schemas-jdedwards-com:trans.response.JDESOOUT</A>
      <A N = "session">212.1100644166.9</A>
      <A N = "environment">DV7333</A>
      <AO N = "transaction">
        <A N = "type">JDESOOUT</A>
        <A N = "action">transactionInfo</A>
        <AO N = "returnCode">
          <A N = "code">0</A>
          <A N = "PCDATA">XML Request OK</A>
        </AO>
        <AO N = "key">
          <AO N = "column">
            <A N = "name">EdiUserId</A>
            <A N = "PCDATA">JDE</A>
          </AO>
          <AO N = "column">
            <A N = "name">EdiBatchNumber</A>
            <A N = "PCDATA">13484</A>
          </AO>
          <AO N = "column">
            <A N = "name">EdiTransactNumber</A>
            <A N = "PCDATA">104336</A>
          </AO>
        </AO>
        <AO N = "table">
          <A N = "name">F4201Z1</A>
          <A N = "type">Header</A>
          <AO N = "column">
            <A N = "name">EdiUserId</A>
            <A N = "PCDATA">JDE</A>
          </AO>
          <AO N = "column">
            <A N = "name">EdiBatchNumber</A>
            <A N = "PCDATA">13484</A>
          </AO>
          <AO N = "column">
            <A N = "name">EdiTransactNumber</A>
            <A N = "PCDATA">104336</A>
          </AO>
          <AO N = "column">
            <A N = "name">EdiLineNumber</A>
            <A N = "PCDATA">1.000</A>
          </AO>
          <AO N = "column">
            <A N = "name">EdiDocumentType</A>
            <A N = "PCDATA">SO</A>
          </AO>
          <AO N = "column">
            <A N = "name">TypeTransaction</A>
```

```
              <A N = "PCDATA">JDESOOUT</A>
            </AO>
            <AO N = "column">
              <A N = "name">EdiTranslationFormat</A>
...
Tue Nov 16 18:42:33 GMT-05:00 2004: AQ Adapter: received the message from the
Agent and will now write it to AQ.
Tue Nov 16 18:42:33 GMT-05:00 2004: AQ Adapter: created a writer for queue xml_
raw_q1.
Tue Nov 16 18:42:34 GMT-05:00 2004: AQ Adapter: successfully converted the OAI
message to XML
<?xml version = '1.0' encoding = 'UTF-8' standalone = 'yes'?>
<jdeResponse type="trans" user="jde"
xmlns="urn:Schemas-jdedwards-com:trans.response.JDESOOUT"
session="212.1100644166.9" environment="DV7333">
   <transaction type="JDESOOUT" action="transactionInfo">
      <returnCode code="0">XML Request OK</returnCode>
      <key>
         <column name="EdiUserId">JDE</column>
         <column name="EdiBatchNumber">13484</column>
         <column name="EdiTransactNumber">104336</column>
      </key>
      <table name="F4201Z1" type="Header">
         <column name="EdiUserId">JDE</column>
         <column name="EdiBatchNumber">13484</column>
         <column name="EdiTransactNumber">104336</column>
         <column name="EdiLineNumber">1.000</column>
         <column name="EdiDocumentType">SO</column>
         <column name="TypeTransaction">JDESOOUT</column>
         <column name="EdiTranslationFormat"></column>
         <column name="EdiTransmissionDate"></column>
         <column name="DirectionIndicator">2</column>
         <column name="EdiDetailLinesProcess">0</column>
         <column name="EdiSuccessfullyProcess">Y</column>
         <column name="TradingPartnerId"></column>
         <column name="TransactionAction">A</column>
         <column name="CompanyKeyOrderNo">00200</column>
         <column name="DocumentOrderInvoiceE">3146</column>
         <column name="OrderType">SO</column>
         <column name="OrderSuffix">000</column>
         <column name="CostCenter">         M30</column>
         <column name="Company">00200</column>
         <column name="CompanyKeyOriginal"></column>
         <column name="OriginalPoSoNumber"></column>
         <column name="OriginalOrderType"></column>
         <column name="CompanyKeyRelated"></column>
         <column name="RelatedPoSoNumber"></column>
         <column name="RelatedOrderType"></column>
         <column name="AddressNumber">4242</column>
         <column name="AddressNumberShipTo">4242</column>
        ...
```

5

Troubleshooting and Error Messages

This chapter explains the limitations and workarounds when connecting to J.D. Edwards OneWorld. The following topics are discussed:

- Troubleshooting
- BSE Error Messages

The adapter-specific errors listed in this chapter can arise whether using the adapter with an OracleAS Adapter JCA or with a BSE configuration.

Troubleshooting

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

- Application Explorer
- J.D. Edwards OneWorld
- OracleAS Adapter JCA
- BSE

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

- The OracleAS Adapter JCA trace information can be found under the *OracleAS_home*\opmn\logs directory.
- BSE trace information can be found under the *OracleAS_home*\j2ee\home\applications\ws-app-adapter\ibse\ibselogs directory.
- The log file for Application Explorer can be found under the *OracleAS_home*\adapters\application\tools directory.

Application Explorer

To use Application Explorer on Windows for debugging or testing purposes, invoke the ae batch script, ae.bat, found under *OracleAS_home*\adapters\application\tools or on UNIX invoke the ae script, ae.sh, found under *OracleAS_home*/adapters/application/tools.

Error	Solution
Cannot connect to the OracleAS Adapter for J.D. Edwards OneWorld from Application Explorer.	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.

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 data base problems in your production library. Contact your data processing department.
Item Branch record does not exist.	An Item Branch record (F4102) does not exist for this item in the Branch/Plant specified.	Correct the Branch or enter an Item Branch record for this item in Branch Plant Item Information (P41026).
&1 does not match any of the valid values.	The &1 does not match any of the valid values specified in the Data Dictionary for this field.	Enter a valid value.
Date out of range.	The Last Service Date and the Inspection Date must be within the range of the effective dates of the Service Contract.	Change the date to be greater than or equal to the beginning effective date and less than or equal to the ending effective date of the Service Contract.

OracleAS Adapter JCA

Error	Solution
In Application Explorer, the following error message appears when you attempt to connect to an OracleAS Adapter JCA configuration: `Could not initialize JCA`	In the Details tab in the right pane, ensure that the directory specified in the Home field points to the correct directory, for example, *OracleAS_home*\adapters\application

BSE Error Messages

This topic discusses the different types of errors that can occur when processing Web services through Business Services Engine (BSE).

General Error Handling in BSE

The Business Services Engine (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 execution time, various conditions can cause errors in BSE when Web services that use adapters are running. Some of these conditions and resulting errors are exposed the same way, regardless of the specific adapter; others are exposed differently, based on the adapter being used. This topic explains what you can expect when you encounter some of the more common error conditions on an adapter-specific basis.

Usually the SOAP gateway (agent) inside BSE passes a SOAP request message to the adapter required for the Web service. If an error occurs, how it is exposed depends on the adapter and the API or interfaces that the adapter uses. A few scenarios cause the SOAP gateway to generate a SOAP fault. In general, anytime the SOAP agent inside BSE receives an invalid SOAP request, a SOAP fault element is generated in the SOAP response. The SOAP fault element contains fault string and fault code elements. The fault code contains a description of the SOAP agent error.

The following SOAP response document results when BSE receives an invalid SOAP request:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

 <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>SOAP-ENV:Client</faultcode>
         <faultstring>Parameter node is missing</faultstring>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In this example, BSE did not receive an element in the SOAP request message that is mandatory for the WSDL for this Web service.

Adapter-Specific Error Handling

When an adapter raises an exception during execution, the SOAP agent in BSE produces a SOAP fault element in the generated SOAP response. The SOAP fault element contains fault code and fault string elements. The fault string contains the native error description from the adapter target system. Since adapters use the target system interfaces and APIs, whether or not an exception is raised depends on how the target systems interface or API treats the error condition. If a SOAP request message is passed to an adapter by the SOAP agent in BSE, and that request is invalid based on the WSDL for that service, the adapter may raise an exception yielding a SOAP fault.

While it is almost impossible to anticipate every error condition that an adapter may encounter, the following is a description of how adapters handle common error conditions and how they are then exposed to the Web services consumer application.

Invalid SOAP Request

When the Oracle Application Server Adapter receives a SOAP request message that does not conform to the WSDL for the Web services being executed, the following SOAP response is generated.

```
<?xml version="1.0" encoding="ISO-8859-1"
 ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
 <SOAP-ENV:Fault>
  <faultcode>SOAP-ENV:Server</faultcode>
  <faultstring>RPC server connection failed: Connection refused:
connect</faultstring>
 </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Empty Result From Oracle Application Server Adapter Request

When the Oracle Application Server Adapter executes a SOAP request using input parameters passed that do not match records in the target system, 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

When the Oracle Application Server Adapter executes an invalid SOAP log in request, the following SOAP response is generated.

```
[2004-07-19T16:28:56:718Z] DEBUG (SOAP1) W.SOAP1.2: POST received
[2004-07-19T16:28:56:718Z] DEBUG (SOAP1) W.SOAP1.2: in XDSOAPHTTPWorker agentName
is [XDSOAPRouter]
[2004-07-19T16:28:56:718Z] DEBUG (SOAP1) W.SOAP1.2: before parse:
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Header>
<m:ibsinfo xmlns:m="urn:schemas-iwaysoftware-...[861]
[2004-07-19T16:28:56:718Z] ERROR (SOAP1) W.SOAP1.2: Attempting string, no encoding
recognized in document
[2004-07-19T16:28:56:734Z] DEEP (SOAP1) W.SOAP1.2: parse complete in 16 msecs
[2004-07-19T16:28:56:859Z] DEEP (SOAP1) W.SOAP1.2: ST_NODICT
[2004-07-19T16:28:56:859Z] DEEP (SOAP1) W.SOAP1.2: ST_FINISH
[2004-07-19T16:28:56:859Z] DEBUG (SOAP1) extractControl - edaDoc: false
[2004-07-19T16:28:56:859Z] DEBUG (SOAP1) now: 2004-07-19T16:28:56Z expires:
2004-07-20T16:28:56Z
[2004-07-19T16:28:56:859Z] DEBUG (SOAP1) W.SOAP1.2: checking for cached agent
[2004-07-19T16:28:56:859Z] DEBUG (SOAP1) W.SOAP1.2: pushagent: adding agent
com.ibi.iwse.XDSOAPRouter
[2004-07-19T16:28:56:875Z] DEBUG (SOAP1) W.SOAP1.2: inside worker the soap Action
```

```
is [B0100033.GetEffectiveAddressRequest#test##]
[2004-07-19T16:28:56:875Z] DEBUG (SOAP1) W.SOAP1.2: precedence: 1
[2004-07-19T16:28:56:875Z] DEBUG (SOAP1) W.SOAP1.2: precedence: 1
[2004-07-19T16:28:56:875Z] DEBUG (SOAP1) W.SOAP1.2: numagents: 1
[2004-07-19T16:28:56:890Z] DEBUG (SOAP1) W.SOAP1.2: running agent 1 name
com.ibi.iwse.XDSOAPRouter document 1
[2004-07-19T16:28:56:890Z] INFO  (manager) MGR00X01: Adding active worker:
W.SOAP1.2
[2004-07-19T16:28:56:890Z] DEBUG (SOAP1) W.SOAP1.2: <?xml version="1.0"
encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <SOAP-ENV:Header>
      <m:ibsinfo xmlns:m="urn:schemas-iwaysoftware-com:iwse">
         <m:service>B0100033</m:service>
         <m:method>GetEffectiveAddress</m:method>
         <m:license>test</m:license>
         <m:Username>user</m:Username>
         <m:Password>password</m:Password>
      </m:ibsinfo>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <m:GetEffectiveAddress
xmlns:m="urn:iwaysoftware:ibse:jul2003:GetEffectiveAddress">
         <m:jdeRequest type="callmethod">
            <m:callMethod name="GetEffectiveAddress">
               <m:params>
                  <m:param name="mnAddressNumber">12345</m:param>
               </m:params>
               <m:onError/>
            </m:callMethod>
         </m:jdeRequest>
      </m:GetEffectiveAddress>
   </SOAP-ENV:Body>
   <SOAPAction agentName="XDSOAPRouter"
cid="1FF3D44E0B0AFB2A4E9538ED42B71437">B0100033.GetEffectiveAddressRequest#test##<
/SOAPAction>
</SOAP-ENV:Envelope>
[2004-07-19T16:28:56:890Z] DEBUG (SOAP1) W.SOAP1.2: business method:
m:GetEffectiveAddress
[2004-07-19T16:28:56:906Z] DEBUG (SOAP1) W.SOAP1.2: input:
[2004-07-19T16:28:56:906Z] DEBUG (SOAP1) W.SOAP1.2: <?xml version="1.0"
encoding="UTF-8" ?><jdeRequest xmlns:xsd="http://www.w3.org/2001/XMLSchema"
type="callmethod" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><callMethod
name="GetEffectiveAddress"><params><param name="mnAddressNumber">12345</param>
      </params><onError/></callMethod></jdeRequest>
[2004-07-19T16:28:58:234Z] DEBUG (SOAP1) W.SOAP1.2: Agent returned success
[2004-07-19T16:28:58:234Z] INFO  (manager) MGR00X02: Removing active worker:
W.SOAP1.2
[2004-07-19T16:28:58:234Z] DEBUG (SOAP1) W.SOAP1.2: doing docTran, docVal,
listTran for agent(1)
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: sendToAll reply to XDReply:
[protocol=http */null]
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: preemitters from doc: null
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: no preemitters, emitting
contents of doc, usestream=false encoding=UTF-8
```

```
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeEntity, len: 670 data:
<?xml version="1.0" encoding="UTF-8" ?><SOAP-ENV:Envelope
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><SOAP-ENV:Body><GetEffective
AddressResponse xmlns="urn:iwaysoftware:ibse:jul2003:GetEffectiveAddress:response"
cid="1FF3D44E0B0AFB2A4E9538ED42B71437"><jdeResponse user="USER" type="callmethod"
session="" environment="DV7333"><returnCode code="12">Environment
&apos;DV7333&apos; could not be initialized for user, check user, pwd and
environment attribute
values</returnCode></jdeResponse></GetEffectiveAddressResponse></SOAP-ENV:Body></S
OAP-ENV:Envelope>
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: HTTP/1.0
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: 200
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: OK
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: Content-Type:
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: text/xml
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: Content-Length:
[2004-07-19T16:28:58:265Z] DEBUG (SOAP1) W.SOAP1.2: writeString: 670
[2004-07-19T16:28:58:265Z] INFO  (SOAP1) W.SOAP1.2: W0000X13: Ended message
processing, rc=0
[2004-07-19T16:28:58:265Z] DEEP (SOAP1) W.SOAP1.2: storing used socket
[2004-07-19T16:28:58:265Z] DEBUG (SOAP1) W.SOAP1.2: entering waitforDocument
[2004-07-19T16:29:03:875Z] DEEP (SOAP1) W.SOAP1.2: cleanup: closing sockets(0)
```

Empty Result From Oracle Application Server Adapter Request

When the Oracle Application Server Adapter executes a SOAP request using input parameters passed that do not match records in the target system, 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 the Oracle Application Server Adapter, 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)
```

6

Advanced Topics

This chapter includes the following topics:

Using Web Services Policy-Based Security

Application Explorer provides a security model called Web services policy-based security. The following topics describe how the feature works and how to configure it.

Web Services Policy-Based Security

Web services provide a layer of abstraction between the back-end business logic they invoke, and the user or application running the Web service. This enables easy application integration but raises the issue of controlling the use and execution of critical and sensitive business logic that is run as a Web service.

Application Explorer controls the use of Web services that use adapters, using a feature called policy-based security. This feature enables an administrator to apply "policies" to Business Services (Web services) to deny or permit their execution.

A policy is a set of privileges dealing with the execution of a Business Service (BS) that can be applied to an existing or new BS. When you set specific rights or privileges inside a policy, you do not have to re-create privileges for every BS that has security concerns in common with other Business Services. Instead, you reuse a policy on multiple Business Services.

The goal of the feature is to secure requests at both the transport and the SOAP request level transmitted on the wire. Some of the policies do not deal with security issues directly, but do effect the runtime behavior of the Web services to which they have been applied.

The Business Services administrator creates an "instance" of a policy type, names it, associates individual users or groups (a collection of users), and then applies that policy to one or more Business Services.

You can assign a policy to a Business Service, or to a method within a Business Service. If a policy is only applied to a method, other methods in that Business Service will not be governed by it. However, if a policy is applied to the Business Service, all methods are governed by it. At runtime, the user ID and password that are sent to BSE in the SOAP request message are checked against the list of users for all policies applied to that specific Business Service. The policy type that is supported is Resource Execution, which dictates who can or cannot execute the Business Service.

When a policy is not applied, the default value for a Business Service is to "grant all". For example, anybody can execute the Business Service, until the Resource Execution policy is associated to the Business Service. At that time, only those granted execution permissions, or users not part of the group that has been denied execution permissions, have access to the Business Service.

Configuring Web Services Policy-Based Security

The following procedures describe how to configure Web services policy-based security.

Creating and Associating a User with a Policy

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

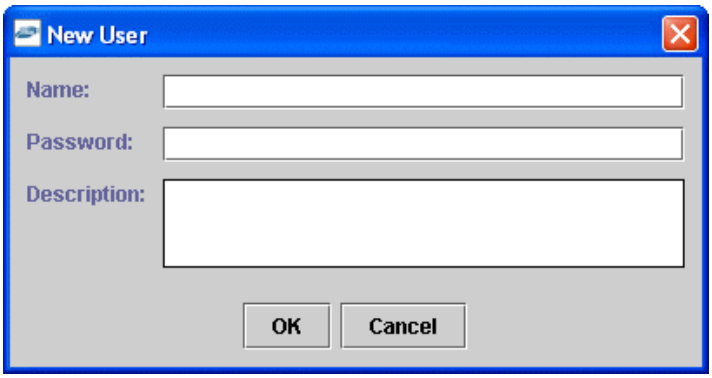
1. Open Application Explorer.
2. Right-click the configuration to which you want to connect, for example, `SampleConfig`. See Chapter 2, "Adapter Configuration Using Oracle Application Explorer" for information on creating a new configuration.
3. Select **Connect.**

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

   a. Expand the **Business Services** node by clicking the plus (+) symbol.
   b. Expand the **Configuration** node by clicking the plus (+) symbol.
   c. Expand the **Security** node by clicking the plus (+) symbol.
   d. Expand the **Users and Groups** node by clicking the plus (+) symbol.

4. Right-click **Users** and click **New User.**

   The New User dialog box opens.



a. In the **Name** field, type a user ID.

b. In the **Password** field, type the password associated with the user ID.

c. In the **Description** field, type 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. Open Application Explorer.
2. Right-click the configuration to which you want to connect, for example, SampleConfig. See Chapter 2, "Adapter Configuration Using Oracle Application Explorer" for information on creating a new configuration.
3. Select **Connect.**

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

a. Expand the **Business Services** node by clicking the plus (+) symbol.

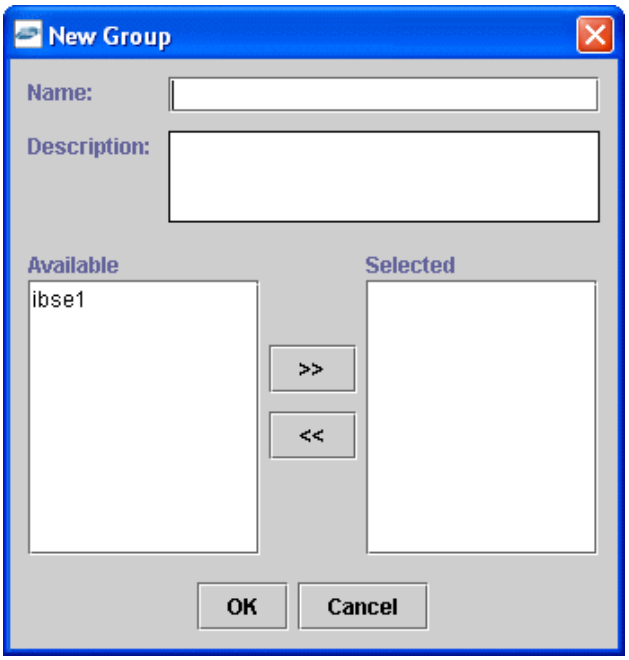
b. Expand the **Configuration** node by clicking the plus (+) symbol.

c. Expand the **Security** node by clicking the plus (+) symbol.

d. Expand the **Users and Groups** node by clicking the plus (+) symbol.

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

   The New Group dialog box opens.



   a. In the **Name** field, type a a name for the group.

   b. In the **Description** field, type 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 Group node.

Creating an Execution Policy

An execution policy governs who can execute the Business Services to which the policy is applied.

To create an execution policy:

1. Open Application Explorer.

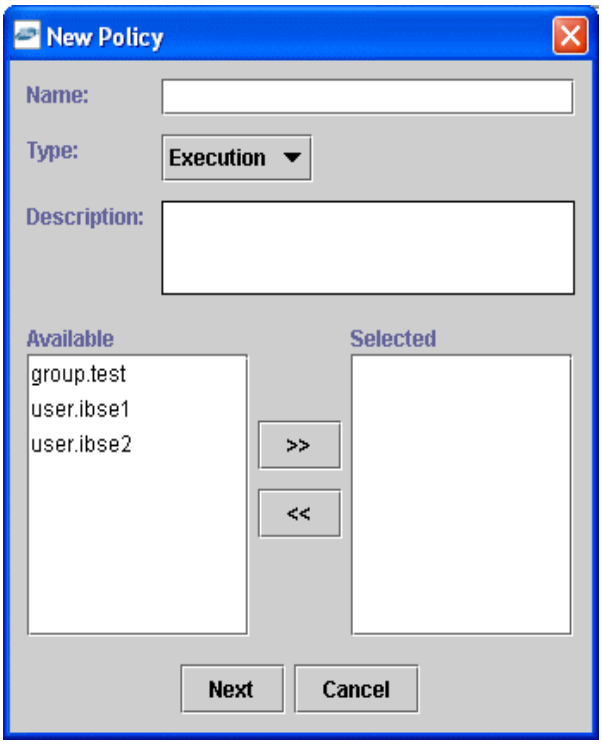
2. Right-click the configuration to which you want to connect, for example, SampleConfig. See Chapter 2, "Adapter Configuration Using Oracle Application Explorer" for information on creating a new configuration.
3. Select **Connect.**

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

   a. Expand the **Business Services** node by clicking the plus (+) symbol.
   b. Expand the **Configuration** node by clicking the plus (+) symbol.
   c. Expand the **Security** node by clicking the plus (+) symbol.
   d. Expand the **Policies** node by clicking the plus (+) symbol.

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

   The New Policy dialog box opens.



   a. In the **Name** field, type a a name for the policy.

b. From the **Type** list, select **Execution.**

c. In the **Description** field, type a description for the policy (optional).

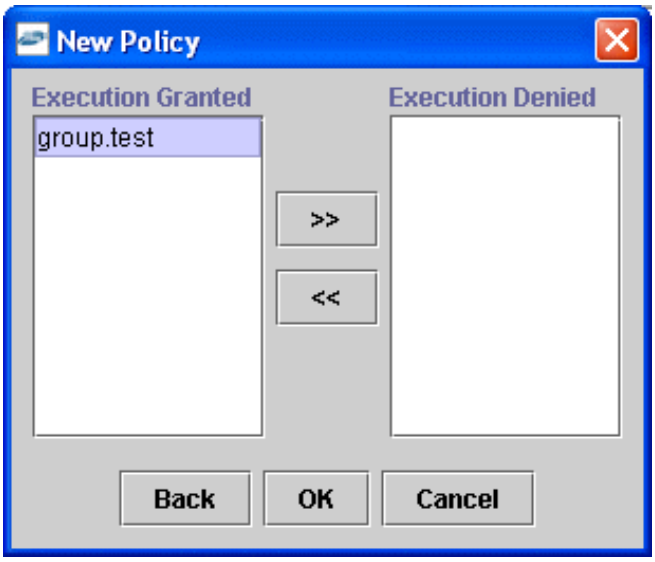
d. From the available list of users in the left pane, select one or more users and add them to the **Selected** list by clicking the double right facing arrow.

Note: This user ID is checked against the value in the user ID element of the SOAP header sent to BSE in a SOAP request.

---

5. When you have selected at least one user selected, click **OK.**
6. Click **Next.**

The New Policy permissions dialog box opens.



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.

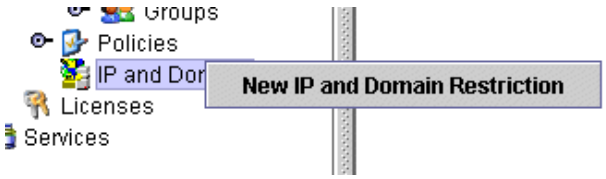
1. Open Application Explorer.
2. Right-click the configuration to which you want to connect, for example, SampleConfig. See Chapter 2, "Adapter Configuration Using Oracle Application Explorer" for information on creating a new configuration.
3. Select **Connect.**

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

   a. Expand the **Business Services** node by clicking the plus (+) symbol.

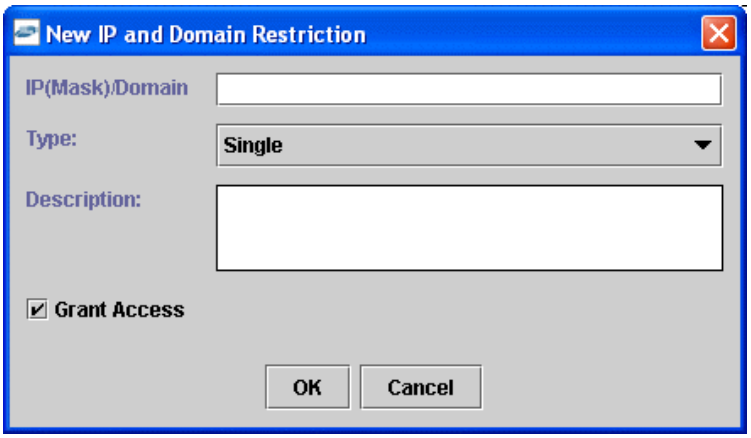
   b. Expand the **Configuration** node by clicking the plus (+) symbol.

   c. Expand the **Security** node by clicking the plus (+) symbol.



4. Right-click **IP and Domain** and select **New IP and Domain Restriction.**

   The New IP and Domain Restriction dialog box opens.



   a. In the **IP(Mask)/Domain** field, type 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, for example, yahoo.com.

   b. From the **Type** list, select the type of restriction.

c. In the **Description** field, type a description (optional).

d. To grant access, select the **Grant Access** check box.

5. Click **OK.**

The new domain is added under the IP and Domain node.

The following pane summarizes your configuration.

- **IP Address(Mask)/Domain** www.yahoo.com
- **Type** Domain
- **Access** Denied
- **Description**

Migrating Repositories

During design time, the Oracle repository is used to store metadata created when using Application Explorer to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. The information in the repository is also referenced at runtime. For management purposes, you can migrate BSE and JCA repositories that are configured for Oracle to new destinations without affecting your existing configuration. For example, you may want to migrate a repository from a test environment to a production environment.

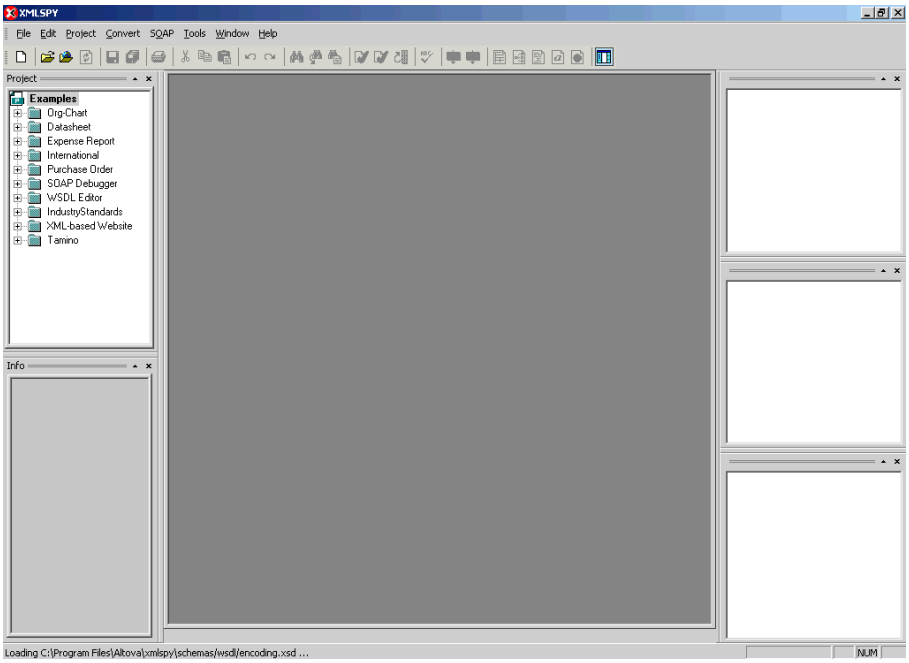
Migrating a BSE Repository

To migrate a BSE repository:

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

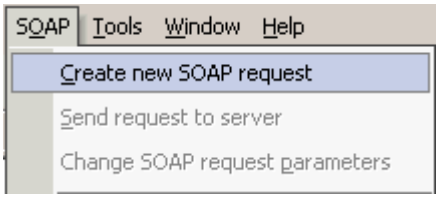
   ```
   http://localhost:7777/ibse/IBSEServlet/admin/iwcontrol.ibs
   ```

2. Open a third party XML editor, for example, XMLSPY.



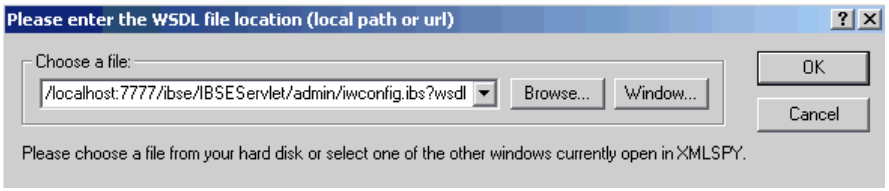
3. In the menu bar, click **SOAP.**

   A list of options appears.



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

   The WSDL file location dialog box opens.



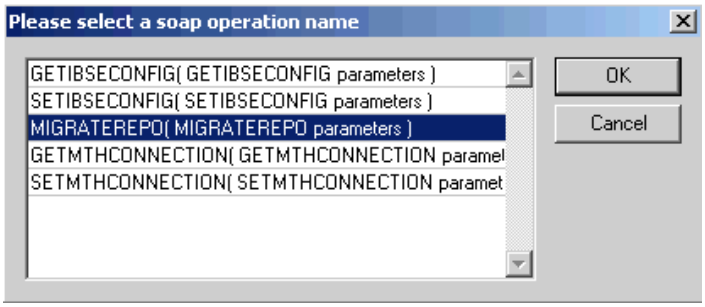
   a. In the **Choose a file** field, paste the BSE control service URL.

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

      `http://localhost:7777/ibse/IBSEServlet/admin/iwcontrol.ibs?wsdl`

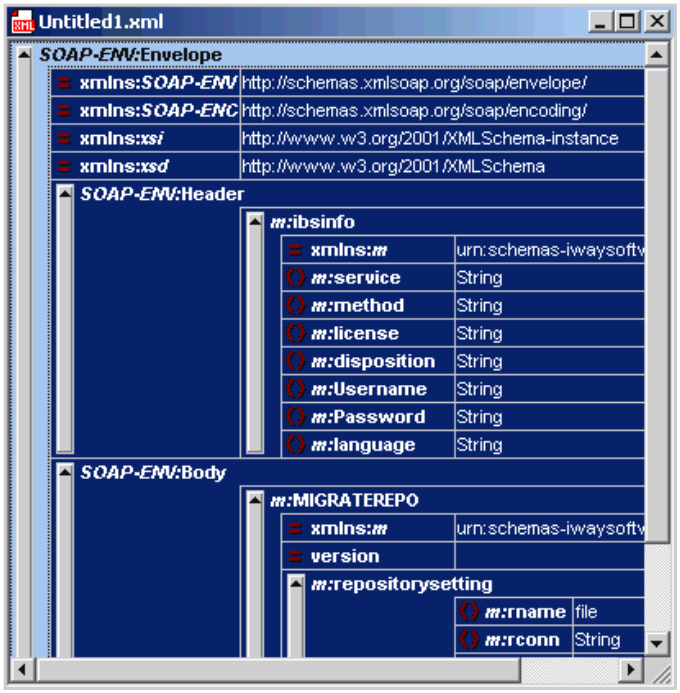
5. Click **OK.**

   The soap operation name dialog box opens and lists the available control methods.



6. Select the **MIGRATEREPO(MIGRATEREPO parameters)** control method and click **OK.**

   The window opens, which 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>
```

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.

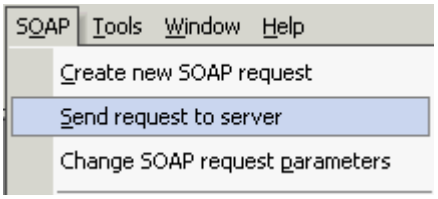
- If you want to migrate a single Web service from the current BSE repository, enter the Web service name in the `<m:servicename>` tag, for example:

```
<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. In 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 JCA Repository

To migrate a JCA repository:

1. Navigate to the location of your JCA configuration directory where the repository schemas and other information is stored, for example:

```
OracleAS_home\adapters\application
```

2. Locate and copy the `repository.xml` file.

3. Place this file in a new JCA configuration directory to migrate the existing repository.

Your JCA repository is migrated to the new JCA configuration directory.

A

Configuring J.D. Edwards OneWorld for Outbound Transaction Processing

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

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

- Specifying Outbound Functionality for a Business Function
- Modifying the OneWorld jde.ini File

Specifying Outbound Functionality for a Business Function

You can specify outbound functionality for business functions and manage the flow of data. You enable outbound transaction processing using a processing option that controls how a transaction is written.

Outbound Transaction Processing

To process outbound data, you use the:

- Data Export Control table
- Processing Log table

The Data Export Control table manages the flow of the outbound data to third-party applications. The Processing Log table contains all the information about the OneWorld event.

For more information on configuring J.D. Edwards OneWorld for outbound processing, see "Detailed Tasks for OneWorld Operations" in the *J.D. Edwards Interoperability Guide for OneWorld XE.*

Enabling Outbound Transaction Processing

To enable outbound transaction processing:

1. Right-click the application that contains the processing options for the Master Business Functions of the transaction.

   For a list of these options, see Appendix B of the*J.D. Edwards Interoperability Guide for OneWorld XE.*

2. From the shortcut menu, select **Prompt for Values.** Click either the **Outbound** tab or the **Interop** tab.
3. Enter the transaction type.

The OneWorld Event listener processes only the after image for the business function.

You are not required to set the before image function.

The Data Export Control Table and the Processing Log Table

The Data Export Control table manages the flow of the outbound data to third-party applications. OneWorld allows for the subscription of multiple vendor-specific objects for an interoperability transaction.

The records in the Data Export Control table are used to determine the vendor-specific objects to call from the Outbound Subsystem batch process (R00460) or the Outbound Scheduler batch process (R00461).

The Processing Log table contains all the information about the OneWorld event including the transaction type, order type, and sequence number from the Data Export Control table.

Using the Export Controls

To use the data export controls:

1. On the Work With Data Export Controls pane, click **Add.**
2. Type values in the Transaction Type and Order Type fields.
3. For each detail row, enter either a batch process name or version or a function name and the library.
4. To launch the vendor-specific object for an add or insert, type **1.**
5. For the update, delete, and inquiry actions, type **1.**
6. In the Launch Immediately column, type **1.**
7. Click **OK.**

Modifying the OneWorld jde.ini File

Because the OracleAS Adapter for J.D. Edwards OneWorld uses XML for the transfer of information to and from J.D. Edwards OneWorld, you must configure the OneWorld environment to support XML. You can do this easily by modifying the OneWorld `jde.ini` file.

Modifying a jde.ini File for XML Support

The following is an example of how to modify a `jde.ini` file to implement XML support.

1. Add the following blocks:

```
[JDENET_KERNEL_DEF6]
;krnlName=CALL OBJECT KERNEL
;dispatchDLLName=jdekrnl.dll
;dispatchDLLFunction=_JDEK_DispatchCallObjectMessage@28
;maxNumberOfProcesses=10
;numberOfAutoStartProcesses=0
krnlName=CALL OBJECT KERNEL
dispatchDLLName=XMLCallObj.dll
dispatchDLLFunction=_XMLCallObjectDispatch@28
maxNumberOfProcesses=10
numberOfAutoStartProcesses=0
[JDENET_KERNEL_DEF15]
```

```
krnlName=XML TRANSACTION KERNEL
dispatchDLLName=XMLTransactions.dll
dispatchDLLFunction=_XMLTransactionDispatch@28
maxNumberOfProcesses=1
numberOfAutoStartProcesses=1
```

The parameters containing an underscore (_) and @28 are for Windows operating systems only. For other operating systems, replace the parameters with the values in the following table.

Operating System	Call Object dispatch DLLName	XML Trans dispatch DLLName
AS400	XMLCALLOBJ	XMLTRANS
HP9000B	libxmlcallojb.sl	libxmltransactions .lo
Sun or RS6000	libxmlcallojb.so	libxmltransactions .so

2. Change the following block:

```
[JDENET]
serviceNameListen=6009
serviceNameConnect=6009
maxNetProcesses=5
maxNetConnections=400
maxKernelProcesses=50
maxKernelRanges=15
netTrace=1
ServiceControlRefresh=5
MonitorOption=0 0 0 0 0 0 0 0
```

Change maxKernelRanges to 15.

For more information on establishing your J.D. Edwards OneWorld environment for XML support, see "Setting the jde.ini File for XML" in the *J.D. Edwards Interoperability Guide for OneWorld XE.*

B

Sample Files

The Adapter for J.D. Edwards OneWorld supports the jdeRequest and jdeResponse XML structures for executing business functions within OneWorld. Using J.D. Edwards OneWorld XML, you can:

- Aggregate business function calls into a single object.
- Use the J.D. Edwards OneWorld ThinNet API.
- Access both Z files and business functions.

The following topics provide examples of the jdeRequest and jdeResponse XML structures for executing business functions within OneWorld:

Issuing a Single-Function Request

The following example, GetEffectiveAddress, is a single-function call to J.D. Edwards OneWorld, and the result of this request is a standard jdeResponse document. In a single-function request, only one callMethod within the XML object is specified.

Executing a Business Function with a Single-Function Call

The following is a sample GetEffectiveAddress jdeRequest.

```
<jdeRequest type="callmethod" user="JDE" pwd="JDE" environment="DV7333"
session="">
<callMethod name="GetEffectiveAddress" app="BSE" runOnError="no">
<params>
   <param name="mnAddressNumber">1001</param>
   <param name="jdDateBeginningEffective"></param>
   <param name="cEffectiveDateExistence10"></param>
   <param name="szAddressLine1"></param>
   <param name="szAddressLine2"></param>
   <param name="szAddressLine3"></param>
   <param name="szAddressLine4"></param>
   <param name="szZipCodePostal"></param>
   <param name="szCity"></param>
   <param name="szCountyAddress"></param>
   <param name="szState"></param>
   <param name="szCountry"></param>
   <param name="szUserid"></param>
```

```
   <param name="szProgramid"></param>
   <param name="jdDateupdated"></param>
   <param name="szWorkstationid"></param>
   <param name="mnTimelastupdated"></param>
   <param name="szNamealpha"></param>
</params>
<onError abort="yes"></onError>
</callMethod>
</jdeRequest>
```

The following is a sample GetEffectiveAddress jdeResponse.

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

<!DOCTYPE jdeResponse>
<jdeResponse environment="DV7333"
             pwd="JDE"
             session="516.1029417972.68"
             type="callmethod"
             user="JDE">
  <callMethod app="BSE"
              name="GetEffectiveAddress"
              runOnError="no">
    <returnCode code="0"/>
    <params>
      <param name="mnAddressNumber">1001</param>
      <param name="jdDateBeginningEffective"/>
      <param name="cEffectiveDateExistence10"/>
       <param name="szAddressLine1">8055 Tufts Avenue, Suite 1331
</param>
      <param name="szAddressLine2">
</param>
      <param name="szAddressLine3">
</param>
      <param name="szAddressLine4">
</param>
      <param name="szZipCodePostal">80237       </param>
      <param name="szCity">Denver                   </param>
      <param name="szCountyAddress">                          </param>
      <param name="szState">CO</param>
      <param name="szCountry"/>
      <param name="szUserid"/>
      <param name="szProgramid"/>
      <param name="jdDateupdated"/>
      <param name="szWorkstationid"/>
      <param name="mnTimelastupdated">0</param>
      <param name="szNamealpha">J.D. Edwards & Company                 </param>
    </params>
  </callMethod>
</jdeResponse>
```

Issuing a Multiple-Function Request

The following example, GetEffectiveAddress, is a multiple-function call to J.D. Edwards OneWorld, and the result of this request is a standard jdeResponse document with multiple sections. In a multiple-function request, more than one callMethod within the XML object is specified.

Executing a Business Function with a Multiple-Function Call

The following is a sample Purchase Order in the jdeRequest format. The XML contains return parameter specifications as well as file cleanup logic.

```
<?xml version='1.0' encoding='utf-8' ?>
<jdeRequest pwd='password' type='callmethod' user='user' session=''
environment='DV7333' sessionidle=''>
   <callMethod app='XMLTest' name='GetLocalComputerId'
      runOnError='no'>
   <params>
      <param name='szMachineKey' id='machineKey'></param>
   </params>
   <onError abort='yes'>
   </onError>
   </callMethod>
   <callMethod app='XMLTest' name='F4311InitializeCaching'
      runOnError='no'>
   <params>
      <param name='cUseWorkFiles'>2</param>
   </params>
   </callMethod>
   <callMethod app='XMLTest' name='F4311FSBeginDoc' runOnError='no'
      returnNullData='yes'>
   <params>
      <param name='mnJobNumber' id='jobNumber'></param>
      <param name='szComputerID' idref='machineKey'></param>
      <param name='cHeaderActionCode'>A</param>
      <param name='cProcessEdits'>1</param>
      <param name='cUpdateOrWriteToWorkFile'>2</param>
      <param name='cRecordWrittenToWorkFile'>0</param>
      <param name='szOrderCOmpany' id='orderCompany'>00200</param>
      <param name='szOrderType'>OP</param>
      <param name='szOrderSuffix'>000</param>
      <param name='szBranchPlant'>         M30</param>
      <param name='mnSupplierNumber'
         id='supplierNumber'>4343</param>
      <param name='mnShipToNumber'>0.0</param>
      <param name='jdOrderDate'>2000/03/02</param>
      <param name='cEvaluatedReceiptsFlag'>N</param>
      <param name='cCurrencyMode'>D</param>
      <param name='szTransactionCurrencyCode'>USD</param>
      <param name='mnCurrencyExchangeRate'>0.0</param>
      <param name='szOrderedPlacedBy'>SUBSTITUTE</param>
      <param name='szProgramID'>EP4310</param>
      <param name='szPurchaseOrderPrOptVersion'
         id='Version'>ZJDE0001</param>
      <param name='szUserID'>SUBSTITUTE</param>
      <param name='mnProcessID' id='processID'></param>
      <param name='mnTransactionID' id='transactionID'></param>
   </params>
   <onError abort='yes'>
   <callMethod app='XMLTest' name='F4311ClearWorkFiles'
      runOnError='yes' returnNullData='yes'>
   <params>
      <param name='szComputerID' idref='jobNumber'></param>
      <param name='mnJobNumber' idref='machineKey'></param>
      <param name='cClearHeaderFile'>1</param>
      <param name='cClearDetailFile'>1</param>
      <param name='mnLineNumber'>0</param>
      <param name='cUseWorkFiles'>2</param>
```

```
   <param name='mnProcessID' idref='processID'></param>
   <param name='mnTransactionID' idref='transactionID'></param>
</params>
</callMethod>
</onError>
</callMethod>
<!-- This is the first EditLine entry -->
<callMethod app='XMLTest' name='F4311EditLine' runOnError='yes'
   returnNullData='no'>
<params>
   <param name='mnJobNumber' idref='jobNumber'></param>
   <param name='szComputerID' idref='machineKey'></param>
   <param name='cDetailActionCode'>A</param>
   <param name='cProcessEdits'>1</param>
   <param name='cUpdateOrWriteWorkFile'>2</param>
   <param name='cCurrencyProcessingFlag'>Y</param>
   <param name='szPurchaseOrderPrOptVersion'
   idref='version'></param>
   <param name='szOrderCompany' idref='orderCompany'></param>
   <param name='szOrderType'>OP</param>
   <param name='szOrderSuffix'>000</param>
   <param name='szBranchPlant'>         M30</param>
   <param name='mnSupplierNumber'
      idref='supplierNumber'></param>
   <param name='mnShipToNumber'>0.0</param>
   <param name='jdRequestedDate'>2000/03/02</param>
   <param name='jdTransactionDate'>2000/03/02</param>
   <param name='jdPromisedDate'>2000/03/02</param>
   <param name='jdGLDate'>2000/03/02</param>
   <param name='szUnformattedItemNumber'>1001</param>
   <param name='mnQuantityOrdered'>1</param>
   <param name='szDetailLineBranchPlant'>         M30</param>
   <param name='szLastStatus'>220</param>
   <param name='szNextStatus'>230</param>
   <param name='cEvaluatedReceipts'>N</param>
   <param name='szTransactionCurrencyCode'>USD</param>
   <param name='cSourceRequestingPOGeneration'>0</param>
   <param name='szProgramID'>XMLTest</param>
   <param name='szUserID'>SUBSTITUTE</param>
   <param name='szAgreementNumber'></param>
   <param name='mnAgreementSupplement'>0</param>
   <param name='jdEffectiveDate'></param>
   <param name='szPurchasingCostCenter'></param>
   <param name='szObjectAccount'></param>
   <param name='szSubsidiary'></param>
   <param name='mnProcessID' idref='processID'></param>
   <param name='mnTransactionID' idref='transactionID'></param>
</params>
</callMethod>
<!-- This is the second EditLine entry -->
<callMethod app='XMLTest' name='F4311EditLine' runOnError='yes'
   returnNullData='no'>
<params>
   <param name='mnJobNumber' idref='jobNumber'></param>
   <param name='szComputerID' idref='machineKey'></param>
   <param name='cDetailActionCode'>A</param>
   <param name='cProcessEdits'>1</param>
   <param name='cUpdateOrWriteWorkFile'>2</param>
   <param name='cCurrencyProcessingFlag'>Y</param>
   <param name='szPurchaseOrderPrOptVersion'
```

```
      idref='version'></param>
   <param name='szOrderCompany' idref='orderCompany'></param>
   <param name='szOrderType'>OP</param>
   <param name='szOrderSuffix'>000</param>
   <param name='szBranchPlant'>         M30</param>
   <param name='mnSupplierNumber'
      idref='supplierNumber'></param>
   <param name='mnShipToNumber'>0.0</param>
   <param name='jdRequestedDate'>2000/03/02</param>
   <param name='jdTransactionDate'>2000/03/02</param>
   <param name='jdPromisedDate'>2000/03/02</param>
   <param name='jdGLDate'>2000/03/02</param>
   <param name='szUnformattedItemNumber'>2001</param>
   <param name='mnQuantityOrdered'>3</param>
   <param name='szDetailLineBranchPlant'>         M30</param>
   <param name='szLastStatus'>220</param>
   <param name='szNextStatus'>230</param>
   <param name='cEvaluatedReceipts'>N</param>
   <param name='szTransactionCurrencyCode'>USD</param>
   <param name='cSourceRequestingPOGeneration'>0</param>
   <param name='szProgramID'>XMLTest</param>
   <param name='szUserID'>SUBSTITUTE</param>
   <param name='szAgreementNumber'></param>
   <param name='mnAgreementSupplement'>0</param>
   <param name='jdEffectiveDate'></param>
   <param name='szPurchasingCostCenter'></param>
   <param name='szObjectAccount'></param>
   <param name='szSubsidiary'></param>
   <param name='mnProcessID' idref='processID'></param>
   <param name='mnTransactionID' idref='transactionID'></param>
</params>
</callMethod>
<callMethod app='XMLTest' name='F4311EditDoc' runOnError='no'
   returnNullData='no'>
<params>
   <param name='szOrderSuffix'>000</param>
   <param name='szComputerID' idref='machineKey'></param>
   <param name='mnJobnumber' idref='jobNumber'></param>
   <param name='mnAddressNumber' idref='supplierNumber'></param>
   <param name='szOrderType'>OP</param>
   <param name='szOrderCompany' idref='orderCompany'></param>
   <param name='szVersionProcOption' idref='version'></param>
   <param name='cActionCode'>A</param>
   <param name='mnProcessID' idref='processID'></param>
   <param name='mnTransactionID' idref='transactionID'></param>
</params>
</callMethod>
<callMethod app='XMLTest' name='F4311EndDoc' runOnError='no'
   returnNullData='no'>
<params>
   <param name='szComputerID' idref='machineKey'></param>
   <param name='mnJobNumber' idref='jobNumber'></param>
   <param name='szCallingApplicationName'>XMLTest</param>
   <param name='szVersion' idref='version'></param>
   <param name='szUserID'>SUBSTITUTE</param>
   <param name='mnOrderNumberAssigned'
      id='orderNumberAssigned'></param>
   <param name='cUseWorkFiles'>2</param>
   <param name='cConsolidateLines'>0</param>
   <param name='mnProcessID' idref='processID'></param>
```

```
      <param name='mnTransactionID' idref='transactionID'></param>
   </params>
   </callMethod>
   <returnParams runOnError='yes' returnNullData='no'>
      <param name='JobNumber' idref='machineKey'></param>
      <param name='ComputerID' idref='jobNumber'></param>
      <param name='OrderNumberAssigned'
         idref='orderNumberAssigned'></param>
   </returnParams>
   <!-- This is a default error catch for the entire document-->
   <onError abort='yes'>
   <callMethod app='XMLTest' name='F4311ClearWorkFiles'
      runOnError='yes' returnNullData='no'>
   <params>
      <param name='szComputerID' idref='jobNumber'></param>
      <param name='mnJobNumber' idref='machineKey'></param>
      <param name='cClearHeaderFile'>1</param>
      <param name='cClearDetailFile'>1</param>
      <param name='mnLineNumber'>0</param>
      <param name='cUseWorkFiles'>2</param>
      <param name='mnProcessID' idref='processID'></param>
      <param name='mnTransactionID' idref='transactionID'></param>
   </params>
   </callMethod>
   </onError>
</jdeRequest>
```

The Purchase Order response document contains individual return codes for each callMethod executed. In addition, this method returns the order number assigned for the Purchase Order.

```
<?xml version="1.0" encoding="utf-8" ?>

<jdeResponse environment="DV7333" user="JDE" type="callmethod" sessionidle=""
session="2612.1026498135.5" pwd="JDE">
   <callMethod name="GetLocalComputerId" runOnError="no" app="XMLTest">
   <returnCode code="0"/>
   <params>
      <param name="szMachineKey" id="machineKey">XEENT</param>
   </params>
   </callMethod>
   <callMethod name="F4311InitializeCaching" runOnError="no" app="XMLTest">
   <returnCode code="0"/>
   <params>
      <param name="cUseWorkFiles">2</param>
   </params>
   </callMethod>
   <callMethod name="F4311FSBeginDoc" returnNullData="yes" runOnError="no"
app="XMLTest">
   <returnCode code="0"/>
   <params>
      <param name="mnJobNumber" id="jobNumber">3</param>
      <param name="szComputerID" idref="machineKey">XEENT</param>
      <param name="cHeaderActionCode">1</param>
      <param name="cProcessEdits">1</param>
      <param name="cUpdateOrWriteToWorkFile">2</param>
      <param name="cRecordWrittenToWorkFile">1</param>
      <param name="cCurrencyProcessingFlag">Z</param>
      <param name="szOrderCOmpany" id="orderCompany">00200</param>
      <param name="mnOrderNumber">0</param>
      <param name="szOrderType">OP</param>
```

```
   <param name="szOrderSuffix">000</param>
   <param name="szBranchPlant">         M30</param>
   <param name="szOriginalOrderCompany"/>
   <param name="szOriginalOrderNumber"/>
   <param name="szOriginalOrderType"/>
   <param name="szRelatedOrderCompany"/>
   <param name="szRelatedOrderNumber"/>
   <param name="szRelatedOrderType"/>
   <param name="mnSupplierNumber" id="supplierNumber">17000</param>
    <param name="mnShipToNumber">6074</param>
    <param name="jdRequestedDate">2002/07/12</param>
    <param name="jdOrderDate">2000/03/02</param>
    <param name="jdPromisedDate">2002/07/12</param>
    <param name="jdCancelDate"/>
    <param name="szReference01"/>
    <param name="szReference02"/>
     <param name="szDeliveryInstructions01">
</param>
      <param name="szDeliveryInstructions02">
</param>
    <param name="szPrintMessage"/>
    <param name="szSupplierPriceGroup"/>
    <param name="szPaymentTerms"/>
    <param name="szTaxExplanationCode"/>
    <param name="szTaxRateArea"/>
    <param name="szTaxCertificate">                    </param>
    <param name="cAssociatedText"/>
    <param name="szHoldCode"/>
    <param name="szFreightHandlingCode"/>
    <param name="mnBuyerNumber">0</param>
    <param name="mnCarrierNumber">0</param>
    <param name="cEvaluatedReceiptsFlag">N</param>
    <param name="cSendMethod"/>
    <param name="szLandedCostRule">   </param>
    <param name="szApprovalRouteCode"/>
    <param name="mnChangeOrderNumber">0</param>
    <param name="cCurrencyMode">D</param>
    <param name="szTransactionCurrencyCode">USD</param>
    <param name="mnCurrencyExchangeRate">0</param>
    <param name="szOrderedPlacedBy">SUBSTITUTE</param>
    <param name="szOrderTakenBy"/>
    <param name="szProgramID">EP4310</param>
    <param name="szApprovalRoutePO"/>
    <param name="szPurchaseOrderPrOptVersion" id="Version">ZJDE0001</param>
    <param name="szBaseCurrencyCode">USD</param>
    <param name="szUserID">SUBSTITUTE</param>
    <param name="cAddNewLineToExistingOrder"/>
    <param name="idInternalVariables">0</param>
    <param name="cSourceOfData"/>
    <param name="mnSODOrderNumber">0</param>
    <param name="szSODOrderType"/>
    <param name="szSODOrderCompany"/>
    <param name="szSODOrderSuffix"/>
    <param name="mnRetainage">0</param>
    <param name="szDescription"/>
   <param name="szRemark"/>
   <param name="jdEffectiveDate"/>
   <param name="jdPhysicalCompletionDate"/>
   <param name="mnTriangulationRateFromCurrenc">0</param>
   <param name="mnTriangulationRateToCurrency">0</param>
```

```
      <param name="cCurrencyConversionMethod"/>
      <param name="szPriceAdjustmentScheduleN"/>
      <param name="cAIADocument"/>
      <param name="mnProcessID" id="processID">2612</param>
      <param name="mnTransactionID" id="transactionID">4</param>
   </params>
   </callMethod>
   <callMethod name="F4311EditLine" returnNullData="no" runOnError="yes"
app="XMLTest">
   <returnCode code="0"/>
   <params>
      <param name="mnJobNumber" idref="jobNumber">3</param>
      <param name="szComputerID" idref="machineKey">XEENT</param>
      <param name="mnOrderLineNumber">1</param>
      <param name="cDetailActionCode">1</param>
      <param name="cProcessEdits">1</param>
      <param name="cUpdateOrWriteWorkFile">2</param>
      <param name="cRecordWrittenToWorkFile">1</param>
      <param name="cCurrencyProcessingFlag">Y</param>
      <param name="szPurchaseOrderPrOptVersion"
         idref="version">ZJDE0001</param>
      <param name="szOrderCompany"
         idref="orderCompany">00200</param>
      <param name="szOrderType">OP</param>
      <param name="szOrderSuffix">000</param>
      <param name="szBranchPlant">         M30</param>
      <param name="mnSupplierNumber" idref="supplierNumber">17000</param>
      <param name="mnShipToNumber">6074</param>
      <param name="jdRequestedDate">2000/03/02</param>
      <param name="jdTransactionDate">2000/03/02</param>
      <param name="jdPromisedDate">2000/03/02</param>
      <param name="jdGLDate">2000/03/02</param>
        <param name="szUnformattedItemNumber">1001
   </param>
      <param name="mnQuantityOrdered">1</param>
      <param name="mnUnitPrice">32,1000</param>
      <param name="mnExtendedPrice">32,1</param>
      <param name="szLineType">S</param>
      <param name="szDescription1">Bike Rack - Trunk Mount</param>
     <param name="szDescription2">                                </param>
     <param name="szDetailLineBranchPlant">         M30</param>
     <param name="szLocation">  .   .             </param>
     <param name="szLotNumber">                              </param>
     <param name="szTransactionUoM">EA</param>
     <param name="szPurchasingUoM">EA</param>
     <param name="szLastStatus">220</param>
     <param name="szNextStatus">230</param>
     <param name="mnDiscountFactor">1</param>
     <param name="szInventoryPriceRule">        </param>
     <param name="szPrintMessage"> </param>
     <param name="cTaxable">Y</param>
     <param name="szGLClassCode">IN30</param>
     <param name="mnBuyerNumber">8444</param>
     <param name="szPurchasingCategoryCode1"> </param>
     <param name="szPurchasingCategoryCode2"> </param>
     <param name="szPurchasingCategoryCode3"> </param>
     <param name="szPurchasingCategoryCode4">240</param>
     <param name="szLandedCostRule"> </param>
     <param name="mnWeight">80</param>
     <param name="szWeightUoM">OZ</param>
```

```
  <param name="mnVolume">2,25</param>
  <param name="szVolumeUoM">FC</param>
  <param name="cEvaluatedReceipts">N</param>
  <param name="cInventoryInterface">Y</param>
  <param name="szTransactionCurrencyCode">USD</param>
  <param name="szBaseCurrencyCode">USD</param>
  <param name="cSourceRequestingPOGeneration">0</param>
  <param name="szProgramID">XMLTest</param>
  <param name="szUserID">SUBSTITUTE</param>
  <param name="szAgreementNumber"/>
  <param name="mnAgreementSupplement">0</param>
  <param name="jdEffectiveDate"/>
  <param name="szPurchasingCostCenter"/>
  <param name="szObjectAccount"/>
  <param name="szSubsidiary"/>
  <param name="cStockingType">P</param>
  <param name="mnProcessID" idref="processID">2612</param>
  <param name="mnTransactionID" idref="transactionID">4</param>
  <param name="mnIdentifierShortItem">60003</param>
</params>
</callMethod>
<callMethod name="F4311EditLine" returnNullData="no"
      runOnError="yes" app="XMLTest">
<returnCode code="0"/>
<params>
  <param name="mnJobNumber" idref="jobNumber">3</param>
  <param name="szComputerID" idref="machineKey">XEENT</param>
  <param name="mnOrderLineNumber">2</param>
  <param name="cDetailActionCode">1</param>
  <param name="cProcessEdits">1</param>
  <param name="cUpdateOrWriteWorkFile">2</param>
  <param name="cRecordWrittenToWorkFile">1</param>
  <param name="cCurrencyProcessingFlag">Y</param>
  <param name="szPurchaseOrderPrOptVersion"
      idref="version">ZJDE0001</param>
  <param name="szOrderCompany"
      idref="orderCompany">00200</param>
  <param name="szOrderType">OP</param>
  <param name="szOrderSuffix">000</param>
  <param name="szBranchPlant">         M30</param>
  <param name="mnSupplierNumber"
   idref="supplierNumber">17000</param>
  <param name="mnShipToNumber">6074</param>
  <param name="jdRequestedDate">2000/03/02</param>
  <param name="jdTransactionDate">2000/03/02</param>
  <param name="jdPromisedDate">2000/03/02</param>
  <param name="jdGLDate">2000/03/02</param>
  <param name="szUnformattedItemNumber">2001
  </param>
  <param name="mnQuantityOrdered">3</param>
  <param name="mnUnitPrice">164,0817</param>
  <param name="mnExtendedPrice">492,2451</param>
  <param name="szLineType">S</param>
  <param name="szDescription1">Cro-Moly Frame, Red           </param>
  <param name="szDescription2">                              </param>
  <param name="szDetailLineBranchPlant">         M30</param>
  <param name="szLocation">  .   .             </param>
  <param name="szLotNumber">                              </param>
  <param name="szTransactionUoM">EA</param>
  <param name="szPurchasingUoM">EA</param>
```

```
      <param name="szLastStatus">220</param>
      <param name="szNextStatus">230</param>
      <param name="mnDiscountFactor">1</param>
      <param name="szInventoryPriceRule">        </param>
      <param name="szPrintMessage"> </param>
      <param name="cTaxable">Y</param>
      <param name="szGLClassCode">IN30</param>
      <param name="szPurchasingCategoryCode1"> </param>
      <param name="szPurchasingCategoryCode2"> </param>
      <param name="szPurchasingCategoryCode3"> </param>
      <param name="szPurchasingCategoryCode4">200</param>
      <param name="szLandedCostRule"> </param>
      <param name="mnWeight">3</param>
      <param name="szWeightUoM">OZ</param>
      <param name="szVolumeUoM">FC</param>
      <param name="cEvaluatedReceipts">N</param>
      <param name="cInventoryInterface">Y</param>
      <param name="szTransactionCurrencyCode">USD</param>
      <param name="szBaseCurrencyCode">USD</param>
      <param name="cSourceRequestingPOGeneration">0</param>
      <param name="szProgramID">XMLTest</param>
      <param name="szUserID">SUBSTITUTE</param>
      <param name="szAgreementNumber"/>
      <param name="mnAgreementSupplement">0</param>
      <param name="jdEffectiveDate"/>
      <param name="szPurchasingCostCenter"/>
      <param name="szObjectAccount"/>
      <param name="szSubsidiary"/>
      <param name="cStockingType">M</param>
      <param name="mnProcessID" idref="processID">2612</param>
      <param name="mnTransactionID" idref="transactionID">4</param>
      <param name="mnIdentifierShortItem">60062</param>
    </params>
    </callMethod>
    <callMethod name="F4311EditDoc" returnNullData="no"
          runOnError="no" app="XMLTest">
    <returnCode code="0"/>
    <params>
      <param name="szOrderSuffix">000</param>
      <param name="szComputerID" idref="machineKey">XEENT</param>
      <param name="mnJobnumber" idref="jobNumber">3</param>
      <param name="mnAddressNumber"
          idref="supplierNumber">17000</param>
      <param name="szOrderType">OP</param>
      <param name="szOrderCompany"
          idref="orderCompany">00200</param>
      <param name="szVersionProcOption"
          idref="version">ZJDE0001</param>
      <param name="cActionCode">A</param>
      <param name="mnProcessID" idref="processID">2612</param>
      <param name="mnTransactionID" idref="transactionID">4</param>
</params>
    </callMethod>
    <callMethod name="F4311EndDoc" returnNullData="no"
          runOnError="no" app="XMLTest">
    <returnCode code="0"/>
    <params>
      <param name="szComputerID" idref="machineKey">XEENT</param>
      <param name="mnJobNumber" idref="jobNumber">3</param>
      <param name="szCallingApplicationName">XMLTest</param>
```

```
      <param name="szVersion" idref="version">ZJDE0001</param>
      <param name="szUserID">SUBSTITUTE</param>
      <param name="mnOrderNumberAssigned"
          id="orderNumberAssigned">4884</param>
      <param name="cUseWorkFiles">2</param>
      <param name="cConsolidateLines">0</param>
      <param name="mnProcessID" idref="processID">2612</param>
      <param name="mnTransactionID" idref="transactionID">4</param>
   </params>
   </callMethod>
   <returnParams>
      <param name="JobNumber" idref="machineKey">XEENT</param>
      <param name="ComputerID" idref="jobNumber">3</param>
      <param name="OrderNumberAssigned" idref="orderNumberAssigned">4884</param>
</returnParams>
</jdeResponse>
```

Sample Sales Order Request

The following is a sample Sales Order request.

Executing a Sales Order Request

The following is an example of a Sales Order request.

```
<?xml version='1.0' encoding='utf-8' ?>
<jdeRequest type='callmethod' user='JDE' pwd='JDE' environment='DV7333'>
   <callMethod name='GetLocalComputerId' app='XMLInterop'
         runOnError='no'>
   <params>
     <param name='szMachineKey' id='2'></param>
   </params>
   <onError abort='yes'>
   </onError>
   </callMethod>
   <callMethod name='F4211FSBeginDoc' app='XMLInterop'
         runOnError='no'>
   <params>
     <param name='mnCMJobNumber' id='1'></param>
     <param name='cCMDocAction'>A</param>
     <param name='cCMProcessEdits'>1</param>
     <param name='szCMComputerID' idref='2'></param>
     <param name='cCMUpdateWriteToWF'>2</param>
     <param name='szCMProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
     <param name='szOrderType'>SO</param>
     <param name='szBusinessUnit'>         M30</param>
     <param name='mnAddressNumber'>4242</param>
     <param name='jdOrderDate'>2000/03/29</param>
     <param name='szReference'>10261</param>
     <param name='cApplyFreightYN'>Y</param>
     <param name='szCurrencyCode'></param>
     <param name='cWKSourceOfData'></param>
     <param name='cWKProcMode'></param>
     <param name='mnWKSuppressProcess'>0</param>
   </params>
   <onError abort='yes'>
   <callMethod name='F4211ClearWorkFile' app='XMLInterop'
      runOnError='yes'>
   <params>
```

```
<param name='mnJobNo' idref='1'></param>
<param name='szComputerID' idref='2'></param>
<param name='mnFromLineNo'>0</param>
<param name='mnThruLineNo'>0</param>
<param name='cClearHeaderWF'>2</param>
<param name='cClearDetailWF'>2</param>
<param name='szProgramID'>XMLInterop</param>
<param name='szCMVersion'>ZJDE0001</param>
   </params>
   </callMethod>
   </onError>
   </callMethod>
   <callMethod name='F4211FSEditLine' app='XMLInterop'
      runOnError='yes'>
   <params>
     <param name='mnCMJobNo' idref='1'></param>
     <param name='cCMLineAction'>A</param>
     <param name='cCMProcessEdits'>1</param>
     <param name='cCMWriteToWFFlag'>2</param>
     <param name='szCMComputerID' idref='2'></param>
<!-- param name='mnLineNo'>10261</param -->
     <param name='szItemNo'>1001</param>
     <param name='mnQtyOrdered'>1</param>
     <param name='cSalesTaxableYN'>N</param>
     <param name='szTransactionUOM'>EA</param>
     <param name='szCMProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
     <param name='cWKSourceOfData'></param>
   </params>
   <onError abort='no'>
   </onError>
   </callMethod>
   <callMethod name='F4211FSEditLine' app='XMLInterop'
      runOnError='yes'>
   <params>
     <param name='mnCMJobNo' idref='1'></param>
     <param name='cCMLineAction'>A</param>
     <param name='cCMProcessEdits'>1</param>
     <param name='cCMWriteToWFFlag'>2</param>
     <param name='szCMComputerID' idref='2'></param>
<!-- param name='mnLineNo'>10262</param -->
     <param name='szItemNo'>1001</param>
     <param name='mnQtyOrdered'>10</param>
     <param name='cSalesTaxableYN'>N</param>
     <param name='szTransactionUOM'>EA</param>
     <param name='szCMProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
     <param name='cWKSourceOfData'></param>
   </params>
   <onError abort='no'>
   </onError>
   </callMethod>
   <callMethod name='F4211FSEndDoc' app='XMLInterop'
      runOnError='no'>
   <params>
     <param name='mnCMJobNo' idref='1'></param>
     <param name='szCMComputerID' idref='2'></param>
     <param name='szCMProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
     <param name='cCMUseWorkFiles'>2</param>
```

```
   </params>
   <onError abort='no'>
   <callMethod name='F4211ClearWorkFile' app='XMLInterop'
      runOnError='yes'>
<params>
     <param name='mnJobNo' idref='1'></param>
     <param name='szComputerID' idref='2'></param>
     <param name='mnFromLineNo'>0</param>
     <param name='mnThruLineNo'>0</param>
     <param name='cClearHeaderWF'>2</param>
     <param name='cClearDetailWF'>2</param>
     <param name='szProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
   </params>
   </callMethod>
   </onError>
   </callMethod>
   <returnParams failureDestination='ERROR.Q'
      successDestination='SUCCESS.Q' runOnError='yes'>
   </returnParams>
   <onError abort='yes'>
   <callMethod name='F4211ClearWorkFile' app='XMLInterop'
      runOnError='yes'>
   <params>
     <param name='mnJobNo' idref='1'></param>
     <param name='szComputerID' idref='2'></param>
     <param name='mnFromLineNo'>0</param>
     <param name='mnThruLineNo'>0</param>
     <param name='cClearHeaderWF'>2</param>
     <param name='cClearDetailWF'>2</param>
     <param name='szProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
   </params>
   </callMethod>
   </onError>
   </jdeRequest>
```

Sample Sales Order Response

This is the corresponding response document for the Sales Order request. There are error messages returned in the document. The error messages can be used within a workflow. For example:

```
<error code="2597">Warning: WARNING: Duplicate Customer Order Number
</error>
<error code="4136">Warning: Pick date is less than todays date</error>
```

Using the Sales Order Response

The following is the jdeResponse document.

```
<?xml version="1.0" encoding="utf-8" ?>
<jdeResponse environment="DV7333" user="JDE" type="callmethod" pwd="JDE">

  <callMethod name="GetLocalComputerId" runOnError="no"
      app="XMLInterop">
   <returnCode code="0"/>
   <params>
      <param name="szMachineKey" id="2">XEENT</param>
   </params>
   </callMethod><callMethod name="F4211FSBeginDoc" runOnError="no"
```

```
     app="XMLInterop">
   <returnCode code="1"/>
   <params>
     <param name="mnCMJobNumber" id="1">3</param>
     <param name="cCMDocAction">A</param>
     <param name="cCMProcessEdits">1</param>
     <param name="szCMComputerID" idref="2">XEENT</param>
     <param name="cCMErrorConditions">1</param>
     <param name="cCMUpdateWriteToWF">2</param>
     <param name="szCMProgramID">XMLInterop</param>
     <param name="szCMVersion">ZJDE0001</param>
     <param name="szOrderCo">00200</param>
     <param name="szOrderType">SO</param>
     <param name="szBusinessUnit">              M30</param>
     <param name="mnAddressNumber">4242</param>
     <param name="mnShipToNo">4242</param>
     <param name="jdRequestedDate">2000/03/29</param>
     <param name="jdOrderDate">2000/03/29</param>
     <param name="jdPromisedDate">2000/03/29</param>
     <param name="szReference">10261</param>
     <param name="szDeliveryInstructions1">              </param>
     <param name="szDeliveryInstructions2">               </param>
     <param name="szPrintMesg">          </param>
     <param name="szPaymentTerm">   </param>
     <param name="cPaymentInstrument"> </param>
     <param name="mnTradeDiscount">,000</param>
     <param name="szTaxExplanationCode">S </param>
     <param name="szTaxArea">DEN       </param>
     <param name="szCertificate">                    </param>
     <param name="szHoldOrdersCode">  </param>
     <param name="cPricePickListYN">Y</param>
     <param name="szRouteCode">   </param>
     <param name="szStopCode">   </param>
     <param name="szZoneNumber">   </param>
     <param name="szFreightHandlingCode">   </param>
     <param name="cApplyFreightYN">Y</param>
     <param name="mnCommissionCode1">6001</param>
     <param name="mnCommissionRate1">5,000</param>
     <param name="mnCommissionRate2">,000</param>
     <param name="szWeightDisplayUOM">  </param>
     <param name="szVolumeDisplayUOM">  </param>
     <param name="cMode">D</param>
     <param name="szCurrencyCode">USD</param>
     <param name="jdDateUpdated">2002/07/12</param>
     <param name="szWKBaseCurrency">USD</param>
     <param name="cWKAdvancedPricingYN">N</param>
     <param name="szWKCreditMesg">  </param>
     <param name="szWKTempCreditMesg">  </param>
     <param name="cWKSourceOfData"/>
     <param name="cWKProcMode"/>
     <param name="mnWKSuppressProcess">0</param>
     <param name="szPricingGroup">PREFER  </param>
     <param name="mnProcessID">2252</param>
     <param name="mnTransactionID">4</param>
   </params><errors><error code="2597">Warning: WARNING: Duplicate
      Customer Order Number</error><error code="4136">Warning: Pick
     date is less than todays date</error></errors>
   </callMethod><callMethod name="F4211FSEditLine" runOnError="yes"
      app="XMLInterop">
 <returnCode code="1"/><params>
```

```
  <param name="mnCMJobNo" idref="1">3</param>
  <param name="cCMLineAction">A</param>
  <param name="cCMProcessEdits">1</param>
  <param name="cCMWriteToWFFlag">2</param>
  <param name="cCMRecdWrittenToWF">1</param>
  <param name="szCMComputerID" idref="2">XEENT</param>
  <param name="cCMErrorConditions">1</param>
  <param name="szOrderCo">00200</param>
   <param name="szOrderType">SO</param>     <param name="szBusinessUnit">
   M30</param>
  <param name="mnShipToNo">4242</param>
  <param name="jdRequestedDate">2000/03/29</param>
  <param name="jdPromisedDate">2000/03/29</param>
  <param name="jdPromisedDlvryDate">2000/03/29</param>
  <param name="szItemNo">1001                    </param>
  <param name="szLocation">  .   .   </param>
  <param name="szDescription1">Bike Rack Trunk Mount </param>
  <param name="szDescription2">                       </param>
  <param name="szLineType">S</param>
  <param name="szLastStatus">900</param>
  <param name="szNextStatus">540</param>
  <param name="mnQtyOrdered">1</param>
  <param name="mnQtyBackordered">1</param>
  <param name="mnUnitPrice">44,99</param>
  <param name="mnUnitCost">32,1000</param>
  <param name="szPrintMesg">          </param>
  <param name="cPaymentInstrument"> </param>
  <param name="cSalesTaxableYN">N</param>
  <param name="cAssociatedText"> </param>
  <param name="szTransactionUOM">EA</param>
  <param name="szPricingUOM">EA</param>
  <param name="mnItemWeight">80</param>
  <param name="szWeightUOM">OZ</param>
  <param name="mnForeignUnitPrice">44,99</param>
  <param name="mnForeignUnitCost">32,1000</param>
  <param name="mnDiscountFactor">1</param>
  <param name="mnCMLineNo">1</param>
  <param name="szCMProgramID">XMLInterop</param>
  <param name="szCMVersion">ZJDE0001</param>
  <param name="mnSupplierNo">4343</param>
  <param name="mnWKOrderTotal">44,99</param>
  <param name="mnWKForeignOrderTotal">44,99</param>
  <param name="mnWKTotalCost">32,1</param>
  <param name="mnWKForeignTotalCost">32,1</param>
  <param name="cWKSourceOfData"/>
  <param name="cWKCheckAvailability">1</param>
  <param name="mnLastLineNoAssigned">1</param>
  <param name="cStockingType">P</param>
  <param name="cParentItmMethdOfPriceCalcn">1</param>
  <param name="mnShortItemNo">60003</param>
  <param name="szSalesOrderFlags">0</param>
  <param name="jdPriceEffectiveDate">2000/03/29</param>
  <param name="jdPromisedShip">2000/03/29</param>
  <param name="mnQuantityAvailable">-34</param>
  <param name="mnItemVolume_ITVL">2,25</param>
  <param name="szVolumeUOM_VLUM">FC</param>
  <param name="szRevenueBusinessUnit">  M30</param>
  <param name="mnProcessID">2252</param>
  <param name="mnTransactionID">4</param>
</params><errors><error code="030B">Warning: Order Quantity
```

```
   Exceeds what&apos;s Available</error></errors>
</callMethod><callMethod name="F4211FSEditLine" runOnError="yes"
   app="XMLInterop"><returnCode code="1"/><params>
  <param name="mnCMJobNo" idref="1">3</param>
  <param name="cCMLineAction">A</param>
  <param name="cCMProcessEdits">1</param>
  <param name="cCMWriteToWFFlag">2</param>
  <param name="cCMRecdWrittenToWF">1</param>
  <param name="szCMComputerID" idref="2">XEENT</param>
  <param name="cCMErrorConditions">1</param>
  <param name="szOrderCo">00200</param>
  <param name="szOrderType">SO</param>
  <param name="szBusinessUnit">         M30</param>
  <param name="mnShipToNo">4242</param>
  <param name="jdRequestedDate">2000/03/29</param>
  <param name="jdPromisedDate">2000/03/29</param>
  <param name="jdPromisedDlvryDate">2000/03/29</param>
  <param name="szItemNo">1001                    </param>
  <param name="szLocation">  .   .   </param>
  <param name="szDescription1">Bike Rack-Trunk Mount   </param>
  <param name="szDescription2">                       </param>
  <param name="szLineType">S</param>
  <param name="szLastStatus">900</param>
  <param name="szNextStatus">540</param>
  <param name="mnQtyOrdered">10</param>
  <param name="mnQtyBackordered">10</param>
  <param name="mnUnitPrice">44,99</param>
  <param name="mnUnitCost">32,1000</param>
  <param name="szPrintMesg">          </param>
  <param name="cPaymentInstrument"> </param>
  <param name="cSalesTaxableYN">N</param>
  <param name="cAssociatedText"> </param>
  <param name="szTransactionUOM">EA</param>
  <param name="szPricingUOM">EA</param>
  <param name="mnItemWeight">800</param>
  <param name="szWeightUOM">OZ</param>
  <param name="mnForeignUnitPrice">44,99</param>
  <param name="mnForeignUnitCost">32,1000</param>
  <param name="mnDiscountFactor">1</param>
  <param name="mnCMLineNo">2</param>
  <param name="szCMProgramID">XMLInterop</param>
  <param name="szCMVersion">ZJDE0001</param>
  <param name="mnSupplierNo">4343</param>
  <param name="mnWKOrderTotal">494,89</param>
  <param name="mnWKForeignOrderTotal">494,89</param>
  <param name="mnWKTotalCost">321</param>
  <param name="mnWKForeignTotalCost">321</param>
  <param name="cWKSourceOfData"/>
  <param name="cWKCheckAvailability">1</param>
  <param name="mnLastLineNoAssigned">2</param>
  <param name="cStockingType">P</param>
  <param name="cParentItmMethdOfPriceCalcn">1</param>
  <param name="mnShortItemNo">60003</param>
  <param name="szSalesOrderFlags">                0   </param>
  <param name="jdPriceEffectiveDate">2000/03/29</param>
  <param name="jdPromisedShip">2000/03/29</param>
  <param name="mnQuantityAvailable">-44</param>
  <param name="mnItemVolume_ITVL">22,5</param>
  <param name="szVolumeUOM_VLUM">FC</param>
  <param name="szRevenueBusinessUnit">         M30</param>
```

```
  <param name="mnProcessID">2252</param>
  <param name="mnTransactionID">4</param>
</params><errors><error code="030B">Warning: Order Quantity
   Exceeds what&amp;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