Oracle® Application Server
Adapter for J.D. Edwards OneWorld User's Guide
10g Release 3 (10.1.3.4.0)
E14197-01

January 2009
Preface
Audience
Documentation Accessibility
Related Documents
Conventions
Help Us to Serve You Better

1 Introduction
Adapter Features
J.D. Edwards OneWorld Platforms, Products, and Releases Supported
J.D. Edwards OneWorld Inbound WSDL Generation on UNIX Platforms
J.D. Edwards OneWorld Unit Of Work (UOW)
J.D. Edwards OneWorld Versions and Library Files
J.D. Edwards OneWorld Concepts
Integration with J.D. Edwards OneWorld
Adapter Architecture
BSE Versus OracleAS Adapter J2CA Deployment

2 Configuring OracleAS Adapter for J.D. Edwards One World
Starting Application Explorer
Configuring Settings for BSE or J2CA
Configuring BSE
Configuring J2CA
Creating a Repository Configuration
Creating a Configuration for J.D. Edwards OneWorld
Creating a Configuration for J2CA
Connecting to a BSE or J2CA Configuration
Establishing a Connection (Target) for J.D. Edwards OneWorld
Defining a Target to J.D. Edwards OneWorld
Creating an XML Schema
Creating a Request and a Response Schema
Using GenJava to Generate a Schema
Generating WSDL (J2CA Configurations Only)
Creating and Testing a Web Service (BSE Configurations Only)
Configuring an Event Adapter
### 3 OC4J Deployment and Integration

- Adapter Integration with OC4J: 3-1
- Deployment of Adapter: 3-1
- Updating Adapter Configuration: 3-2
- How to Write a Java Application Client Using the CCI API: 3-4

### 4 Integration with Oracle BPEL Process Manager

- Overview of Adapter Integration with Oracle BPEL Process Manager: 4-1
- Deployment of Adapter: 4-2
- Design Time: 4-2
  - Namespace Requirements: 4-2
  - Design a BPEL Process for Request-Response Service (Outbound): 4-3
- J.D. Edwards OneWorld Service Integration: 4-9
  - Design-Time Configuration: 4-9
- Invoking Adapter Request-Response Service from Oracle BPEL Process Manager: 4-18
- J.D. Edwards OneWorld Event Integration: 4-21
  - Design-Time Configuration: 4-21
  - Runtime Configuration: 4-30

### 5 ESB Integration Examples

- Configuring an ESB Outbound Process: 5-2
- Configuring an ESB Inbound Process: 5-32

### 6 Troubleshooting and Error Messages

- Troubleshooting: 6-1
- BSE Error Messages: 6-4
  - General Error Handling in BSE: 6-4
  - Adapter-Specific Error Handling: 6-5

### 7 Advanced User Tools

- Web Services Policy-Based Security: 7-1
  - Configuring Web Services Policy-Based Security: 7-2
- Migrating Repositories: 7-8

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

- Using the GenJava Development Tool (Outbound Processing): A-1
- Specifying Inbound Functionality for a Business Function: A-2
- Configuring J.D. Edwards OneWorld Inbound Processing: A-2
**Sample Files**

Issuing a Single-Function Request ........................................................................................................ B-1
Issuing a Multiple-Function Request .................................................................................................... B-2
Sample Sales Order Request ................................................................................................................ B-11
Sample Sales Order Response ................................................................................................................ B-13

**Glossary**

**Index**
This Preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions
- Help Us to Serve You Better

**Audience**

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

- Install applications
- Maintain applications

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

**Documentation Accessibility**

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

**Accessibility of Code Examples in Documentation**

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.
Accessibility of Links to External Web Sites in Documentation
This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services
To reach AT&T Customer Assistants, dial 711 or 1.800.855.2880. An AT&T Customer Assistant will relay information between the customer and Oracle Support Services at 1.800.223.1711. Complete instructions for using the AT&T relay services are available at http://www.consumer.att.com/relay/tty/standard2.html. After the AT&T Customer Assistant contacts Oracle Support Services, an Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process.

Related Documents
For more information, refer to these Oracle resources:

■ Oracle Application Server Adapter Concepts
■ Oracle Application Server Adapters Installation Guide

Printed documentation is available for sale in the Oracle Store at http://oraclestore.oracle.com/

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 user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at http://www.oracle.com/technology/documentation/

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

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

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

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.</td>
<td>When you specify this clause, you create an index-organized table.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Italics</em></td>
<td>Italic typeface indicates book titles or emphasis.</td>
<td><em>Oracle Database Concepts</em></td>
</tr>
<tr>
<td><strong>UPPERCASE</strong></td>
<td>Uppercase monospace typeface indicates elements supplied by the system.</td>
<td>Ensure that the recovery catalog and target database do <em>not</em> reside on the same disk.</td>
</tr>
<tr>
<td><em>lowercase</em></td>
<td>Lowercase monospace typeface indicates executable programs, filenames, directory names, and sample user-supplied elements.</td>
<td>You can specify this clause only for a <em>NUMBER</em> column.</td>
</tr>
<tr>
<td><strong>lowercase</strong></td>
<td>Lowercase italic monospace font represents placeholders or variables.</td>
<td>You can back up the database by using the <em>BACKUP</em> command.</td>
</tr>
<tr>
<td><strong>italic</strong></td>
<td>Lowercase italic monospace font represents placeholders or variables.</td>
<td>Query the <em>TABLE_NAME</em> column in the <em>USER_TABLES</em> data dictionary view.</td>
</tr>
<tr>
<td>Other symbols</td>
<td>You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (</td>
<td>), and ellipsis points (...) exactly as shown.</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[ ]</code></td>
<td>Anything enclosed in brackets is optional.</td>
<td><code>DECIMAL (digits [, precision])</code></td>
</tr>
<tr>
<td><code>{ }</code></td>
<td>Braces are used for grouping items.</td>
<td>`{ENABLE</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
<td>A vertical bar represents a choice of two options.</td>
</tr>
<tr>
<td><code>...</code></td>
<td>Ellipsis points mean repetition in syntax descriptions.</td>
<td><code>CREATE TABLE ... AS subquery;</code></td>
</tr>
<tr>
<td>Other symbols</td>
<td>You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (</td>
<td>), and ellipsis points (...) exactly as shown.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>acctbal NUMBER(11,2);</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>acct    CONSTANT NUMBER(4) := 3;</code></td>
</tr>
</tbody>
</table>
### Conventions for Windows Operating Systems

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

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
</table>
| **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 to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. Because these terms are not case sensitive, you can use them in either UPPERCASE or lowercase. | SELECT last_name, employee_id FROM employees;  
SELECT * FROM USER_TABLES;  
DROP TABLE hr.employees; |
| **lowercase** | Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files. | CREATE USER mjones IDENTIFIED BY ty3MU9; |
| **Note:** | Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown. | |

### Conventions for Windows Operating Systems

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

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click <strong>Start</strong>, and then choose the <strong>menu item</strong></td>
<td>How to start a program.</td>
<td>To start the Database Configuration Assistant, click <strong>Start</strong>, and choose <strong>Programs</strong>. In the <strong>Programs</strong> menu, choose <strong>Oracle - HOME_NAME</strong> and then click <strong>Configuration and Migration Tools</strong>. Choose <strong>Database Configuration Assistant</strong>.</td>
</tr>
<tr>
<td><strong>File and directory names</strong></td>
<td>File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (&lt;=), right angle bracket (&gt;=), colon (:), double quotation marks (&quot;), slash (/), pipe (</td>
<td>), and dash (-). The special character backslash () is treated as an element separator, even when it appears in quotes. If the filename begins with \, then Windows assumes it uses the Universal Naming Convention.</td>
</tr>
<tr>
<td><strong>C:&gt;</strong></td>
<td>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.</td>
<td>C:\oracle\oradata&gt;</td>
</tr>
</tbody>
</table>
| **Special characters** | The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters. | C:\>exp HR/HR TABLES=employees  
QUERY="WHERE job_id='SA_REP' and salary<8000" |
Help Us to Serve You Better

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

The following list includes the specifications our consultants require.

- **Platform:**
- **Operating System:**
- **Operating System Version:**
- **Product List:**
- **Adapters:**
- **Adapter Deployment:**
  - For example, J2CA or Business Services Engine (BSE)
- **Container Version:**

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

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOME_NAME</td>
<td>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.</td>
<td>C:&gt; net start OracleHOME_NAME TNSListener</td>
</tr>
<tr>
<td>ORACLE_HOME and ORACLE_BASE</td>
<td>In Oracle8i release 8.1.3 and lower, when you installed Oracle components, all subdirectories were located under a top level ORACLE_HOME directory. This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level ORACLE_HOME directory. There is a top level directory called ORACLE_BASE that by default is C:\oracle\product\10.1.0. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is C:\oracle\product\10.1.0\db_n, where n is the latest Oracle home number. The Oracle home directory is located directly under ORACLE_BASE. All directory path examples in this guide follow OFA conventions. Refer to Oracle Database Installation Guide for Windows for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.</td>
<td>Change to the ORACLE_BASE\ORACLE_HOME\rdbms\admin directory.</td>
</tr>
</tbody>
</table>

---

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

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapter</td>
<td></td>
</tr>
<tr>
<td>EIS (DBMS/APP)</td>
<td></td>
</tr>
<tr>
<td>HOTFIX/Service Pack</td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

- XML schema
- XML instances
- Other input documents (transformation)
- Error screen shots
- Error output files
- Trace and log files
- Log transaction
OracleAS Adapter for J.D. Edwards OneWorld provides connectivity and carries out interactions on a J.D. Edwards OneWorld system. This chapter provides information about OracleAS Adapter for J.D. Edwards OneWorld to help you accomplish your integration projects.

This chapter discusses the following topics:

- Adapter Features
- J.D. Edwards OneWorld Platforms, Products, and Releases Supported
- J.D. Edwards OneWorld Concepts
- Integration with J.D. Edwards OneWorld
- Adapter Architecture
- BSE Versus OracleAS Adapter J2CA Deployment

**Adapter Features**

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

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

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

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

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

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

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

**See Also:** Oracle Application Server Adapter Concepts

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

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

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

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

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

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

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

1. iWay recommends the customer to generate individual business functions of J.D. Edwards OneWorld and then group them together.

2. Generating the individual business functions and grouping them together is completely based on the experience and knowledge related to the business functions of J.D. Edwards OneWorld. There is no documentation on generating the UOW.

3. To create the XML Schema Document (XSD) files that can be used for UOW, perform the following steps:
   a. Trigger the appropriate event from the J.D. Edwards OneWorld GUI and generate an XML output file based on the event.
   b. Using the XML file, create an XSD file using an XML editor, such as XMLSPY.
   c. When creating the XSD, make sure that the XSD satisfies the SOA 10.1.3.1 namespace requirements. Manually add the namespace, target namespace, and other items that are required for SOA 10.1.3.1.
4. Once generated, copy the XSD files for the UOW into the repository folder. This repository folder is automatically configured on your file system when a J.D. Edwards OneWorld target is created using Application Explorer.

**J.D. Edwards OneWorld Versions and Library Files**

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

<table>
<thead>
<tr>
<th>J.D. Edwards OneWorld Version</th>
<th>Required Library Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>XE (B7333)</td>
<td>Connector.jar and Kernel.jar</td>
</tr>
<tr>
<td>ERP 8.0 (B7334)</td>
<td>Connector.jar and Kernel.jar</td>
</tr>
<tr>
<td>EnterpriseOne 8.9 (B9)</td>
<td>Connector.jar, Kernel.jar, jdeutil.jar, and log4j.jar</td>
</tr>
<tr>
<td>EnterpriseOne 8.10</td>
<td>Connector.jar, Kernel.jar, jdeutil.jar, and log4j.jar</td>
</tr>
<tr>
<td>EnterpriseOne 8.11 (SP1 and Tools Release 8.95)</td>
<td>Connector.jar, Kernel.jar, jdeutil.jar, and log4j.jar</td>
</tr>
</tbody>
</table>

**J.D. Edwards OneWorld Concepts**

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

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

**Integration with J.D. Edwards OneWorld**

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

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

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

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

- Receives requests from a legacy system, another EIS, or a non-EIS client.
- Transforms the XML request document into the EIS-specific format.
  - The request document conforms to a request XML schema.
  - The schema is based on metadata in the EIS.
- Calls the underlying function in the EIS and waits for its response.
- Transforms the response from the EIS-specific data format to an XML document.
  - The response document conforms to a response XML schema that is generated by the adapter.
  - The schema is based on metadata in the EIS.

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

Channels are consumers of EIS-specific messages and may or may not provide a response. A channel performs the following functions:
- Receives messages from an EIS client
- Transforms the EIS-specific message format into an XML format.

Propagating Internal Listeners out of J.D. Edwards OneWorld

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

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

J.D. Edwards OneWorld Interoperability Framework

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

OracleAS Adapter for J.D. Edwards OneWorld supports the following integration access methods:
- J.D. Edwards OneWorld ThinNet API
- J.D. Edwards OneWorld XML
- J.D. Edwards unedited transaction tables (Z tables)

Figure 1–1 illustrates the outbound processing framework.

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

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

![Diagram of J.D. Edwards OneWorld Outbound Processing]

*Figure 1–2 illustrates the inbound processing framework.*

*Figure 1–2  J.D. Edwards OneWorld Inbound Processing*

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

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

**Adapter Architecture**

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

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

**OracleAS Adapter Application Explorer (Application Explorer)**

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

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

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

**Resource Adapters**

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

**Oracle Application Server Adapter Business Services Engine (BSE) Architecture**

Figure 1–3 shows the generic architecture for the Oracle Web service adapter for packaged applications. The adapter works with BSE, as deployed to a Web container in a J2EE application server.
Application Explorer, a design-time tool deployed along with BSE, is used to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. Metadata created while you perform these operations are stored in the repository by BSE.

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

**Oracle Application Server Adapter Generic J2CA Architecture**

*Figure 1–4* shows the generic architecture for Oracle J2CA adapter for packaged applications. The J2CA connector is deployed to a standard J2CA Container and serves as host container to the adapters. The connector is configured with a repository.
Application Explorer, a design tool that works with the connector, is used to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. Metadata created during these operations is stored in the repository by the connector. The repository can be a file system or an Oracle database. It is deployed as a RAR file and has an associated deployment descriptor called ra.xml. You can create multiple connector factories by editing the OC4J deployment descriptor oc4j-ra.xml. See Chapter 3, "OC4J Deployment and Integration" for more information on OC4J deployment.

Processing Business Functions

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

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

See Also:

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

BSE Versus OracleAS Adapter J2CA Deployment

If you are using OracleAS Adapter for J.D. Edwards OneWorld with BPEL Process Manager, please note that:
- Only OracleAS Adapter J2CA deployment supports inbound integration (event notification) with BPEL Process Manager.
- Both OracleAS Adapter J2CA and BSE deployments support outbound integration (request-response service) with BPEL Process Manager.

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

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

2. OracleAS Adapter J2CA provides slightly better performance.

   OracleAS Adapter J2CA does provide slightly better performance than BSE. However, the difference decreases as the transaction rate increases.

3. OracleAS Adapter J2CA and the BSE option both provide identity propagation at run-time.

   The BSE option provides the capability to pass identity using the SOAP header. For OracleAS Adapter J2CA, user name and password can be passed using the connection specification of the CCI.
This chapter describes how to use OracleAS Adapter Application Explorer (Application Explorer) to define a target to connect to a J.D. Edwards OneWorld system, view system objects, and create XML schemas and Web services. This chapter also explains how to configure an event adapter.

This chapter discusses the following topics:

- Starting Application Explorer
- Configuring Settings for BSE or J2CA
- Creating a Repository Configuration
- Establishing a Connection (Target) for J.D. Edwards OneWorld
- Creating an XML Schema
- Generating WSDL (J2CA Configurations Only)
- Creating and Testing a Web Service (BSE Configurations Only)
- Configuring an Event Adapter

Starting Application Explorer

To start Application Explorer:

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

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

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

Application Explorer starts. You can now define new targets to your Enterprise Information System (EIS).
Configuring Settings for BSE or J2CA

Before a repository configuration can be created, you must configure OracleAS Adapter Business Services Engine (BSE). You need not configure the Connector for J2CA because the `oc4j_ra.xml` file is configured automatically during installation.

Configuring BSE

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

To configure BSE:

1. Display the following page in your browser:
   
   ```
   http://host_name:port/ibse
   ```

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

   For example,
   
   ```
   http://localhost:7777/ibse
   ```

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

2. Log on when prompted.

   When first installed, the user ID and the password are:
   
   - User name: iway
   - Password: iway

   The BSE configuration page is displayed.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Property Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>English</td>
</tr>
<tr>
<td>Adapter Lib Directory</td>
<td><code>OracleAS_home\adapters\application\lib</code></td>
</tr>
<tr>
<td>Encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>Debug Level</td>
<td>DEBUG</td>
</tr>
<tr>
<td>Number of Async. Processors</td>
<td>0</td>
</tr>
</tbody>
</table>
   
   | Security            |                |
   | Admin User          | iway           |
   | Admin Password      | ********       |
   |
   | Repository          | File System    |
   | Repository Type     |                |
   | Repository URL      | `http://localhost:7777/ibse` |

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

   ```
   OracleAS_home\adapters\application\lib
   ```
After you specify the path, adapters in the lib directory are available to BSE.

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

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

5. Click Save.

**Configuring BSE System Settings**

To configure BSE system settings:

1. Display the BSE configuration page by entering the following URL in a browser:
   
   `http://host name:port/ibse/IBSEConfig`

   Where *host name* is the system where BSE is installed and *port* is the port number on which BSE is listening.

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

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

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Property Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System</strong></td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>English</td>
</tr>
<tr>
<td>Adapter Lib Directory</td>
<td>/Adapters/ApplicationLib</td>
</tr>
<tr>
<td>Encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>Debug Level</td>
<td>DEBUG</td>
</tr>
<tr>
<td>Number of Async. Processors</td>
<td>0</td>
</tr>
</tbody>
</table>

2. Configure the system settings according to the information in the following table.
The following image shows the Security pane of the BSE configuration page.

<table>
<thead>
<tr>
<th>Security</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin User</td>
<td>admin</td>
</tr>
<tr>
<td>Admin Password</td>
<td>●●●●</td>
</tr>
<tr>
<td>Policy</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin User</td>
<td>Provide a BSE administrator ID.</td>
</tr>
<tr>
<td>Admin Password</td>
<td>Enter the password associated with the BSE administrator ID.</td>
</tr>
<tr>
<td>Policy</td>
<td>Select the check box to enable policy security.</td>
</tr>
</tbody>
</table>

The following image shows the Repository pane of the BSE configuration page.
BSE requires a repository to store transactions and metadata required for the delivery of Web services. For more information, see “Configuring a File System Repository” on page 2-5 and “Configuring an Oracle Repository” on page 2-6.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository Type</td>
<td>Select one of the following repositories from the list:</td>
</tr>
<tr>
<td></td>
<td>- Oracle.</td>
</tr>
<tr>
<td></td>
<td>- File. (Do not use a file repository for BSE in production environments.)</td>
</tr>
<tr>
<td>Repository URL</td>
<td>Enter the URL to use when opening a connection to the database. For example, the following repository URL format is used when connecting to Oracle:</td>
</tr>
<tr>
<td></td>
<td>jdbc:oracle:thin:@host name:port;SID</td>
</tr>
<tr>
<td>Repository Driver</td>
<td>Provide the driver class to use when opening a connection to the database (optional). For example, the following repository driver format is used when connecting to Oracle:</td>
</tr>
<tr>
<td></td>
<td>oracle.jdbc.driver.OracleDriver</td>
</tr>
<tr>
<td>Repository User</td>
<td>Enter a valid user ID to use when opening a connection to the database.</td>
</tr>
<tr>
<td>Repository Password</td>
<td>Enter a valid password that is associated with the user ID.</td>
</tr>
<tr>
<td>Repository Pooling</td>
<td>If selected, repository pooling will be used. This option is disabled by default.</td>
</tr>
</tbody>
</table>

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 system. However, a file system repository is less secure and efficient than a database repository. When BSE is first installed, it is automatically configured to use a file system repository.
Configuring Settings for BSE or J2CA

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, and the ability to create and execute stored procedures.

2. Open a command prompt and navigate to the setup directory. Its default location on Windows is:
   
   ```
   OracleAS_home\iWay55\etc\setup
   ```

   For other platforms, navigate to the corresponding location.

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

   `iwse.ora`

   **Note:** If Oracle Database is not on the same system as the Oracle Application Server, copy the `iwse.ora` file to the Oracle system. Then, from a command prompt on the Oracle system, navigate to the directory containing the `iwse.ora` file.

3. Enter the following command:

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

### Configuring J2CA

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

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

### Configuring a Database Repository for J2CA

To configure a database repository for J2CA:

1. Execute the `iwse.ora` SQL statement on the system where the database is installed.
2. Create the jcatransport.properties file and save it in the following directory:

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

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

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

4. Open the oc4j-ra.xml file in a text editor.

5. Provide the JDBC connection information as a value for the IWAYRepo_URL property.

6. Provide a valid user name for the IWAYRepo_User property.

7. Provide a valid password for the IWAYRepo_Password property.

8. Save your changes to the oc4j-ra.xml file.

9. Copy the Oracle database JDBC drivers (Classes12.jar, Classes12.zip, andnls_charset12.jar) to the following directory:

```
OracleAS_home\adapters\application\lib
```

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

**Password Encryption**

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

**Configuring Password Encryption**

To encrypt a password:

1. Open Application Explorer.

2. Click Help and select Encryption.

   The Encryption dialog box opens.

3. Type a password in the Password field and click OK.

   An encrypted version of the password displays in the Encryption field.

4. Copy the password.

5. In the jcatransport.properties file, which is used during design time, replace the existing password with the encrypted value.

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

```
iwafjca.log.level=DEBUG
iwafjca.repo.url=jdbc:oracle:thin:@172.30.166.100:1521:orcl
iwafjca.repo.user=scott
iwafjca.repo.password=ENCR (318931973183297321831293164323332123227)
```

6. In the oc4j-ra.xml file, which is used during run time, replace the existing password with the encrypted value for the IWayRepoPassword element.
7. Restart the Oracle Application Server.

Creating a Repository Configuration

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

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

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

Creating a Configuration for BSE

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

Defining a New Configuration for BSE

To define a new configuration for BSE:

1. Right-click Configurations and select New.
   
   The New Configuration dialog box is displayed.

2. Enter a name for the new configuration (for example, `myConfig`) and click OK.
   
   The New Configuration dialog box is displayed.

3. From the Service Provider list, select `iBSE`.

4. In the iBSE URL field, accept the default URL or replace it with a different URL using the following format:

   `http://host name:port/ibse/IBSEServlet`
Where host name is the system where your application server resides and port is the HTTP port number where the application server is running.

5. Click OK.

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

Creating a Configuration for J2CA

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

Defining a New Configuration for J2CA

To define a new configuration for J2CA:

1. Right-click Configurations and select New.

   The New Configuration dialog box is displayed.

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

3. From the Service Provider list, select JCA.

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

   OracleAS_home\adapters\application

5. Click OK.

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

HTTP Repository Connection

J2CA users can create an HTTP repository connection, which enables them to generate and store WSDL documents remotely. Perform the following steps to create an HTTP repository connection in Application Explorer. To use the HTTP repository, make sure that the iwjcaivp test tool(jca-app-adapter-test) is successfully deployed and running.
1. Start the Application Explorer.
2. Right-click the Configurations node in the left pane and select New.
   The New Configuration dialog box opens.
3. Type a name for the configuration and click OK.
4. Select JCA from the Service Provider list box and enter an HTTP target value in the Home field.
   Use the following format for the HTTP target value:
   http://host name:port/iwafjca/JCAServlet
   For example:
   http://iwserv14:7777/iwafjca/JCAServlet
5. Click OK.
   The new HTTP repository connection is added to the Configurations node.
   Once you connect to the remote server, you can create new Adapter targets, generate WSDL documents, and store them in the remote server.
   **Note:** When you configure an Adapter target with the J2CA HTTP repository, you are not required to restart the Oracle Application Server for run time purposes.

**Connecting to a BSE or J2CA Configuration**

To connect to a BSE or J2CA configuration:

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

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

The following is an example of a BSE configuration named myConfig:

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

You can now define new targets to J.D. Edwards OneWorld.
Establishing a Connection (Target) for J.D. Edwards OneWorld

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

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

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

Defining a Target to J.D. Edwards OneWorld

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

To define a target:

1. In the left pane, expand the Adapters node.
   The applications systems supported by Application Explorer appear as nodes based on the adapters that are installed.

2. Right-click the JDEdwards node and select Add Target.
   The Add Target dialog box is displayed.

   Perform the following steps:
   a. In the Name field, enter a descriptive name, for example, JDEConnection.
   b. In the Description field, enter a description for the target (optional).
   c. From the Type list, select JDE One World.

3. Click OK.
   The JDE One World dialog box appears.
Establishing a Connection (Target) for J.D. Edwards OneWorld

2-12 Oracle Application Server Adapter for J.D. Edwards OneWorld User's Guide

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

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

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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User id*</td>
<td>A valid user ID for J.D. Edwards OneWorld.</td>
</tr>
<tr>
<td>User password*</td>
<td></td>
</tr>
<tr>
<td>JDE Environment*</td>
<td></td>
</tr>
<tr>
<td>Server IP address*</td>
<td></td>
</tr>
<tr>
<td>Server port *</td>
<td></td>
</tr>
<tr>
<td>User role</td>
<td>^ALL</td>
</tr>
</tbody>
</table>
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 2-14.

**Connecting to a Defined J.D. Edwards OneWorld Target**

To connect to a target:

1. Expand the Service Adapters node.
2. Expand the JDEdwards One World node.

   The disconnected target is displayed.

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

   The Logon tab displays the values you entered for connection parameters.

5. Verify your connection parameters.
6. Provide the correct password.
7. Right-click the target name and select Connect.

   The x icon disappears, indicating that the node is connected.
Disconnecting from J.D. Edwards OneWorld

To disconnect from a target:

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

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

Editing a Target

To edit a target:

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

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

Deleting a Target

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

To delete a target:

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

   The node disappears from the list of available connections.

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

Creating an XML Schema

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

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

1. Connect to a J.D. Edwards OneWorld target as described in "Connecting to a Defined J.D. Edwards OneWorld Target" on page 2-13.

2. Expand the Services node.

3. Expand the node of the MBF for which you want to create the schema.

4. Expand and then select the node beneath the MBF.

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

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

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

7. Click Response Schema to view the response schema information.
Generating WSDL (J2CA Configurations Only)

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

Generating WSDL for Outbound Interaction

To generate a WSDL file for request-response service:

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

2. Expand Services, JDEJAVA_CFIN, and then B0100033. Select GetEffectiveAddress.

3. Right-click GetEffectiveAddress.

   The following menu is displayed.

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

   The Export WSDL dialog box is displayed.

Using GenJava to Generate a Schema

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

See Also: J.D. Edwards Interoperability Guide for OneWorld Xe
5. Accept the default name for the file and make sure qualified is selected. The .wsdl file extension is added automatically. By default, the names of WSDL files generated for request-response services end with _invoke, while those generated for event notification end with _receive.

---

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

6. Click OK.

   The WSDL file is saved in the specified location.

**Generating WSDL for Inbound Interaction**

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

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

   **Important:** Do not restart the BPEL PM Server or Oracle Application Server after the channel is started.

3. Send an inbound message from J.D. Edwards OneWorld.
4. Capture the inbound message payload in the log file located under `OracleAS_home\adapters\application\config\jca_sample\log\iwaf_jca15.log`

   Alternatively, you can create a port with the File protocol under Event in Application Explorer to dispose the event message to the file system.

5. Use a third party tool such as XMLSpy to create the XSD schema using the XML payload captured in the previous step.
6. Verify that `orabpel-adapters.jar` is in your classpath.
7. Open a command prompt and change directory to: `OracleAS_home\adapters\application\tools`
8. Run `obadapter.bat` to set the environment.
9. Enter the following command to generate WSDL:

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

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

For example:

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

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

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

---

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

10. Stop the channel in Application Explorer.

---

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

The WSDL you generate using Application Explorer is used in a BPEL process as a Partner Link.

### Creating and Testing a Web Service (BSE Configurations Only)

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

---

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

---

**Creating a Web Service**

To create a Web service for a business function:

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

The Create Web Service dialog box is displayed.

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

   a. From the Existing Service Names list, select either <new service> or an existing service.

   b. In the **Service Name** field, specify a service name if you are creating a new service. This name identifies the Web service in the list of services under the Business Services node.

   c. Enter a description for the service (optional).

4. Click **Next**.

Perform the following steps:

   a. In the License Name field, select one or more license codes to assign to the Web service.

   b. In the Method Name field, enter a descriptive name for the method or accept the default name.

   c. In the Description field, enter a brief description of the method (optional).

   d. In the **DTD Directory** field, specify a location where the Web service will be saved. If you want to select a location different than the default, click **Browse** and navigate to the desired location.

5. Click **OK**.

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

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

To test a Web service:

1. Click the **Business Services** node to access your Web services.
2. Expand the **Services** node.
3. Select the name of the business service you want to test.
   The business service name appears as a link in the right pane.
4. In the right pane, click the named business services link.
   The test option appears in the right pane. If you are testing a Web service that requires XML input, an input field appears.
5. Enter the appropriate input.
6. Click **Invoke**.

Application Explorer displays the results.

Identity Propagation

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

```xml
<SOAP-ENV:Header>
  <m:ibsinfo xmlns:m="urn:schemas-iwaysoftware-com:iwse">
    <m:service>String</m:service>
    <m:method>String</m:method>
    <m:license>String</m:license>
    <m:disposition>String</m:disposition>
    <m:Username>Strings</m:Username>
    <m:Password>Strings</m:Password>
    <m:language>Strings</m:language>
  </m:ibsinfo>
</SOAP-ENV:Header>
```
You can remove the `<m:disposition>` and `<m:language>` tags from the SOAP header, since they are not required.

**Configuring an Event Adapter**

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

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

---

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

---

A **channel** represents configured connections to particular instances of back-end systems. A channel binds one or more event ports to a particular listener managed by the adapter. See "Creating and Editing a Channel" on page 2-21 for more information.

---

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

---

**Creating and Editing a Channel**

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

---

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

---

**Creating a Channel**

To create a channel:

1. Click **Event Adapters**.
2. Expand the **JDEdwards One World** node.
   The ports and channels nodes appear in the left pane.
3. Right-click **Channels** and select **Add Channel**.
   The Add Channel dialog box is displayed.
Perform the following steps:

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

4. Click Next.

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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port Number*</td>
<td>Port on which the Host database is listening.</td>
</tr>
<tr>
<td>Host/IP Binding</td>
<td>Name or URL of the system where the database resides.</td>
</tr>
<tr>
<td>Synchronization Type</td>
<td>Possible values are: REQUEST_RESPONSE REQUEST_ACK REQUEST</td>
</tr>
<tr>
<td>Is Length Prefix</td>
<td>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.</td>
</tr>
<tr>
<td>Is XML</td>
<td>For J.D. Edwards OneWorld events that send data back in XML format. No preparser is required.</td>
</tr>
<tr>
<td>Is Keep Alive</td>
<td>Maintains continuous communication between the event transaction and the channel.</td>
</tr>
</tbody>
</table>
Configuring an Event Adapter

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

5. Click **OK**.
   The channel appears under the channels node in the left pane.

   ![Channels](image)

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

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

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

   ![Channels](image)

   The X over the icon disappears.

7. To stop the channel, right-click the connected channel node and select **Stop**.
   The channel becomes inactive and an X appears over the icon.

**Editing a Channel**
To edit a channel:

1. In the left pane, locate the channel you want to edit.
2. Right-click the channel and select **Edit**.
   The Edit channels pane is displayed.
3. Make the required changes to the channel configuration and click **Finish**.

**Deleting a Channel**
To delete a channel:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User id*</td>
<td>A valid user ID for J.D. Edwards OneWorld.</td>
</tr>
<tr>
<td>User password*</td>
<td>The password associated with the user ID.</td>
</tr>
<tr>
<td>JDE Environment*</td>
<td>The J.D. Edwards OneWorld environment, for example, DU7333. For more</td>
</tr>
<tr>
<td></td>
<td>information about this parameter, see your J.D. Edwards OneWorld documentation or ask your OneWorld system administrator.</td>
</tr>
<tr>
<td>Application</td>
<td>XMLInterop or the application name in J.D. Edwards OneWorld. Optional.</td>
</tr>
<tr>
<td>Server IP address*</td>
<td>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.</td>
</tr>
<tr>
<td>Server port*</td>
<td>Port number on which the server is listening, for example, 6009.</td>
</tr>
</tbody>
</table>
1. In the left pane, locate the channel you want to delete.
2. Right-click the channel and select **Delete**.
   A confirmation dialog box is displayed.
3. To delete the channel you selected, click **OK**.
   The channel disappears from the list in the left pane.

### The OneWorld Event Listener

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

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

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

### Configuring the OneWorld Event Listener

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

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

The OneWorld Event Listener includes the following components:

- The listener exit (IWOEvent), located under `adapters_home\etc\jde`, where `adapters_home` is `OracleAS_home\adapters\application\`. For example: `C:\OraHome_1\adapters\application\etc\jde\Iwoevent.dll`

  The file extension varies depending on your operating system:
  - For **Windows**, the exit is `Iwoevent.dll`.
  - For **Sun Solaris**, the exit is `libiwoevent.so`.
  - For **HP-UX**, the exit is `libiwoevent.sl`.
  - For **AS/400**, the exit is `iwaysav.sav`.
  - For **IBM AIX**, the exit is `libiwoevent.so`.

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

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

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

Creating the iwoevent.cfg File

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

To create the iwoevent.cfg file:

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

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

Adding Connection Information

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

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

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

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

  Where:

  - `aliasname` is the symbolic name given to the connection.
  - `ipaddress|dsn` is the IP address or DSN name for the server containing OracleAS Adapter for J.D. Edwards OneWorld (required).
  - `port` is the port defined for OracleAS Adapter for J.D. Edwards OneWorld (required).
  - `trace=(on|off)` sets the tracing to on for the particular alias.
Trans

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

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

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

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

Adding Connection Information to iwoevent.cfg

To add connection information to the `iwoevent.cfg` file:

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

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

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

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

```
common.trace=on
alias.edamcs1=172.1.1.1:3694
alias.edamcs1t=172.1.1.1:3694, trace=on
alias.edamcs2=222.2.2.2:1234
trans.JDESONW=edamcs1t,edamcs2
trans.JDEPOOUT=edamcs1
```
OC4J Deployment and Integration

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

This chapter discusses the following topics:

- Adapter Integration with OC4J
- Deployment of Adapter
- Updating Adapter Configuration
- How to Write a Java Application Client Using the CCI API

See Also:

- Oracle Application Server Adapter Concepts

Adapter Integration with OC4J

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

See Also:

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

Deployment of Adapter

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

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

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

Creating a Managed Connector Factory Object

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

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

See Also:
- Oracle Application Server Adapter Concepts
Updating Adapter Configuration

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

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

Creating Multiple Managed Connector Factory Objects

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

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

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

```xml
<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>
```
How to Write a Java Application Client Using the CCI API

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

**Note:** The OracleAS Adapter for J.D. Edwards OneWorld does not support invocation through the CCI API. Only invocation using Oracle BPEL Process Manager and ESB is supported. The following section is only provided for reference purposes.

---

### Step 1. Obtain the Connection Factory

The connection factory is obtained by JNDI lookup.

```java
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 run-time service adapter connection. The ConnectionSpec has seven parameters. Connection Pooling is fully supported and established based on these parameters, except log level.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the packaged application adapter.</td>
</tr>
<tr>
<td>config -</td>
<td>Adapter configuration name. NOT REQUIRED FOR IWAEAdapter.</td>
</tr>
<tr>
<td>language</td>
<td>Default is en.</td>
</tr>
<tr>
<td>country</td>
<td>Default is us.</td>
</tr>
</tbody>
</table>
The iWAFConnectionSpec can initiate an interaction with J.D. Edwards OneWorld if the adapter name and configuration parameters are specified in the ConnectionSpec. For example,

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

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

### Step 3. Create interaction with interactionSpec for runtime

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

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

### Step 4. Create Input Record and Execute Interaction

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

A standard J2CA Indexed Record is used in this example:

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

A special record is supported in this example:
//IWAFRecord rIn = new IWAFRecord("input");
//rIn.setRootXML(msg_run);
//IWAFRecord response = executeRunInteraction(c, rIn);
//IWAFRecord rOut = (IWAFRecord)i.execute(is, rIn);
//System.out.println(rOut.getRootXML());

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

```xml
<?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="szErrorMessageID" />
  <param name="szVersion">ZJDE0001</param>
  <param name="mmSameAsExcept" />
  <param name="mmAddressBookNumber" id="1" />
  <param name="szLongAddressNumber" />
  <param name="szTaxId" />
  <param name="szSearchType">C</param>
  <param name="szAlphaName">John J. Smith</param>
  <param name="szSecondaryAlphaName">John J. Smith</param>
  <param name="szMailingName">John J. Smith</param>
  <param name="szSecondaryMailingName">John J. Smith</param>
  <param name="szDescriptionCompressed" />
  <param name="szBusinessUnit" />
  <param name="szAddressLine1">1 Main St.</param>
  <param name="szAddressLine2">Apt 101</param>
  <param name="szAddressLine3" />
  <param name="szAddressLine4" />
  <param name="szPostalCode">75000</param>
  <param name="szCity">AnyTown</param>
  <param name="szCounty" />
  <param name="szState">TX</param>
  <param name="szCountry">US</param>
  <param name="szCarrierRoute" />
  <param name="szBulkMailingCenter" />
  <param name="szPrefix1" />
  <param name="szPhoneNumber1">917-123-4567</param>
  <param name="szPhoneNumberType1" />
  <param name="szPhoneAreaCode2" />
  <param name="szPhoneNumber2" />
  <param name="szPhoneNumberType2" />
  <param name="cPayablesYNM">Y</param>
  <param name="cReceivablesYN">N</param>
  <param name="cEmployeeYN">N</param>
  <param name="cUserCode" />
  <param name="cARAPNettingY">N</param>
  <param name="cPersonCorporationCode" />
  <param name="szCertificate" />
</params>
</callMethod>
</jdeRequest>
```
Complete Code Sample
The following is a sample of the complete code:

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

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

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

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

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

// 1. Getting the Connection factory through JNDI lookup
// ---------------------------------------------------------
InitialContext context = new InitialContext();
ConnectionFactory cf = (ConnectionFactory)context.lookup(iwayJndi)
// 2. Getting a connection for a particular adapter target, in this case JDE
// ---------------------------------------------------------
IWAFConnectionSpec cs = new IWAFConnectionSpec();
cs.setAdapterName(ADAPTER);
cs.setConfig(TARGET);
cs.setLogLevel(LOG_LEVEL); // Adapter layer log level
Connection c = cf.getConnection(cs);// where cf is the connection factory
// 3. Create interaction with interactionSpec for RUNTIME
// ---------------------------------------------------------
Interaction i = c.createInteraction();
IWAFInteractionSpec is = new IWAFInteractionSpec();
is.setFunctionName("PROCESS");
}
// 4. Create input Record and execute interaction
// ---------------------------------------------------------

// 4.1 Using JCA standard Indexed Record
// Use JCA IndexRecord, named 'input' for run-time 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()
Integration with Oracle BPEL Process Manager


This chapter includes the following topics:

- Overview of Adapter Integration with Oracle BPEL Process Manager
- Deployment of Adapter
- Design Time
- J.D. Edwards OneWorld Service Integration
- Invoking Adapter Request-Response Service from Oracle BPEL Process Manager
- J.D. Edwards OneWorld Event Integration

Overview of Adapter Integration with Oracle BPEL Process Manager

To integrate with Oracle BPEL Process Manager, OracleAS Adapter for J.D. Edwards OneWorld must be deployed in the same OC4J container as Oracle BPEL Process Manager. The underlying adapter services must be exposed as WSDL files, which are generated during design time in Oracle Application Server Adapter Application Explorer (Application Explorer) for both request-response (outbound) and event notification (inbound) services of the adapter. See "Generating WSDL (J2CA Configurations Only)" on page 2-16 for more information.

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

When using the adapter with Oracle BPEL Process Manager installed on OracleAS BPEL PM home directory is OC4J_SOA, located as follows:

OracleAS_home\j2ee\OC4J_SOA
Deployment of Adapter

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

See Also: Oracle Application Server Adapter Concepts

Design Time

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

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

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

Before you design a BPEL process, you must generate WSDL using Application Explorer. See "Generating WSDL (J2CA Configurations Only)" on page 2-16 for more information. The WSDL generated in Application Explorer is used in the BPEL process as Partner Links.

Namespace Requirements

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

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

1. Input XML for BPEL based on unqualified namespaces:

   ```xml
   <jdeRequest type="callmethod" user="PSFT" pwd="PSFT" environment="DV810"
   session="">
   <callMethod name="GetEffectiveAddress">
   ```
Design a BPEL Process for Request-Response Service (Outbound)

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

Create a New Application Server Connection

To create a new Application Server connection:

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

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

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

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

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

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

5. Specify a valid user name and password for the Application Server you want to connect to.


7. Click **Next**.

The Create Application Server Connection - Step 3 of 4: Connection dialog box is displayed.
8. Select the Single Instance connection option.
9. Enter localhost as the host name and 6003 for the OPMN port.
10. Enter home as the OC4J instance name.
11. Click Next.

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

12. Click Test Connection.

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

Your newly created Application Server connection is displayed in the Connections Navigator tab under the Application Server node.
Create a New Integration Server Connection

To create a new Integration Server connection:

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

2. Right-click Integration Server and select New Integration Server Connection.
   
The Create Integration Server Connection - Welcome dialog box is displayed.

3. Click Next.
   
The Create Integration Server Connection - Step 1 of 3: Name dialog box is displayed.
4. Specify a unique name and click Next.

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

5. Select an Application Server connection, which is already created.

6. Enter localhost as the host name and 8888 for the port number.

7. Select Add host name to the list of proxy exceptions and click Next.

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

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

9. Click Finish.

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

Testing Outbound BPEL and ESB Processes

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

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

The ESB data flows can be tested using the Enterprise Manager console. When creating an ESB data flow and interactions, the Web services are created and registered with the Oracle Application Server. For more information on creating an ESB outbound process, see Chapter 5, "ESB Integration Examples".

J.D. Edwards OneWorld Service Integration

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

Design-Time Configuration

Generating WSDL for Request/Response Service

To generate WSDL for outbound interaction:

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

   4. Select Create Outbound JCA Service (Request/Response).
      
      The Export WSDL dialog box is displayed.

   5. Accept the default name for the file.
      
      The .wsdl file extension is added automatically.

   6. Ensure that qualified is selected as the element form, which is the default.

   7. Click OK.
      
      The WSDL file is created.
Creating a BPEL PM Server Connection
Before you design an outbound BPEL process, you must configure a new Application Server and Integration Server connection in Oracle JDeveloper. For more information, see Chapter 4, “Integration with Oracle BPEL Process Manager”.

Creating a BPEL Project for a Synchronous BPEL Process
To create a BPEL Project for a synchronous BPEL process:

1. At the top of the upper left pane, click the Applications Navigator tab and select an application. If an application does not exist, then you must create an application.

2. Right-click the application and select New Project.
The New Gallery window is displayed.

3. From the Items list, select BPEL Process Project and click OK.
The BPEL Project Creation Wizard - Project Settings dialog box is displayed.

4. Perform the following steps:
   a. Specify a name for the BPEL project, for example, JDE_Service.
      The Namespace field is updated automatically.
   b. From the Template list, select Synchronous BPEL Process.

5. Click OK.

**Designing the BPEL Process for GetEffectiveAddress (Outbound Service)**

To design the BPEL Process:

1. From the Services pane on the right, drag and drop a PartnerLink to the visual editor.
   The Create Partner Link dialog box is displayed.
2. Click the Service Explorer icon (second icon from the left preceding the WSDL File field).

   The Service Explorer dialog box is displayed.

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

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

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

Perform the following steps:

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

b. From the Partner Role list, select the default value GetEffectiveAddressRole. This is the role of the BPEL process.
5. Click OK.
   
The new PartnerLink appears in the visual editor.

6. Select Save from the File menu.

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

   The new PartnerLink appears in the visual editor.

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

   ![Diagram View of Invoke Process Activity]

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

   The Edit Invoke dialog box is displayed.
Perform the following steps:

a. Provide a name for the Invoke activity, for example, JDE_GetEffectiveAddress.

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

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

9. Click OK.

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

The following image shows the new Assign activity in JDeveloper visual editor.
11. Double-click the **Assign** activity icon.

The Assign dialog box is displayed.

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

The Create Copy Operation dialog box is displayed.

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

b. In the **To** pane, expand **Variables**, then **JDE_GetEffectiveAddress_GetEffectiveAddress_InputVariable**, and then highlight **input_GetEffectiveAddress**.
Your Create Copy Operation dialog box should look as follows:

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

Verify that you have mapped all variables as follows:

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

You have completed the design of your BPEL process.

Deploying the BPEL Process for GetEffectiveAddress (Outbound Service)
JDeveloper deploys BPEL processes directly to Oracle BPEL Console.
To deploy your BPEL process in JDeveloper:
1. Right-click your project in the Applications Navigator tab.

2. Select Deploy, then Your BPEL PM Server connection, and then Deploy to default domain.

   The deployment process starts automatically.

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

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

   ![Message Log](image)

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

**Invoking Adapter Request-Response Service from Oracle BPEL Process Manager**

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

**Manage the Deployed Outbound Process in Oracle BPEL Console**

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

To invoke adapter request-response service:

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

   `http://host:port/BPELConsole`

2. Provide a valid user name and password.

   The Oracle BPEL Console main page is displayed. All deployed BPEL processes are listed in the Dashboard tab.
3. Click the **BPEL Processes** tab.
   This tab provides a more detailed view of each deployed process.

4. Click the J.D. Edwards outbound process link.
   The Manage window provides options for managing this BPEL process. Do not change any of the following default settings.

5. Click the **Initiate** tab.
   The Initiate tab enables you to test your BPEL process.
Perform the following steps:

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

b. Select the Java through delivery API link in the right side of the test area.

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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jdeRequest type="callmethod">
  <callMethod name="GetEffectiveAddress" runOnError="no">
  </callMethod>
</jdeRequest>
```

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

**See Also:** *Oracle Application Server Adapter Concepts*

**J.D. Edwards OneWorld Event Integration**

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

**Design-Time Configuration**

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

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

**Creating a Channel in Application Explorer**

To create a channel:

1. In Application Explorer, expand the JDEdwards node.
2. Right-click the Channels node, and select Add Channels.
   
   The Add Channel dialog box is displayed.

   ![Add Channel Dialog Box](image)

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

...
The dialog box is displayed for the selected listener.

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

Enter values based on the table.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User id*</td>
<td>A valid user ID for J.D. Edwards OneWorld.</td>
</tr>
<tr>
<td>User password*</td>
<td>The password associated with the user ID.</td>
</tr>
</tbody>
</table>
Click OK.

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

---

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

---

### Generating WSDL for Event Notification (Command Prompt Only)

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

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

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

1. Create a channel in Application Explorer under the J.D. Edwards events node. See "Creating a Channel in Application Explorer" on page 4-21.
2. Start the channel.
   
   **Important:** Do not restart the BPEL PM Server or Oracle Application Server after the channel is started.
3. Send an inbound message from J.D. Edwards OneWorld.
4. Capture the inbound message payload in the log file located under
   
   `OracleAS_home\adapters\application\config\jca_sample\log\iwaf_jca15.log`
   
   Alternatively, you can create a port with the File protocol under Event in Application Explorer to dispose the event message to the file system.
5. Use a third party tool such as XMLSpy to create the XSD schema using the XML payload captured in the previous step.
6. Verify that `orabpel-adapters.jar` is in your classpath.
7. Open a command prompt and change directory to:
   
   `OracleAS_home\adapters\application\tools`
8. Run `obadapter.bat` to set the environment. On UNIX platforms, run `Jbadapter.bat`.

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDE environment*</td>
<td>Your J.D. Edwards OneWorld environment. For more information about this parameter, see your J.D. Edwards OneWorld documentation or ask your OneWorld system administrator.</td>
</tr>
<tr>
<td>Application</td>
<td>XMLInterop or the application name in J.D. Edwards OneWorld. Optional.</td>
</tr>
<tr>
<td>Server IP address*</td>
<td>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.</td>
</tr>
<tr>
<td>Server Port*</td>
<td>The port number on which the server is listening, for example, 6009.</td>
</tr>
<tr>
<td>User Role</td>
<td>Define a user role according to your requirements.</td>
</tr>
<tr>
<td>Schema style</td>
<td>Choose a style from the list.</td>
</tr>
</tbody>
</table>

---
9. Enter the following command to generate WSDL:

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

Where:

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

For example:

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

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

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

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

10. Stop the channel in Application Explorer.

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

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

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

**Designing the BPEL Project for the SalesOrder Inbound Service**
To design a BPEL project for inbound interaction:

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

2. Right-click the application and select **New Project**.
   The New Gallery window is displayed.
3. From the Items list, select **BPEL Process Project** and click **OK**.
   The BPEL Project Creation Wizard - Project Settings dialog box is displayed.

![BPEL Project Creation Wizard](image)

4. Perform the following steps:
   a. Specify a name for the BPEL project.
      The Namespace field is updated automatically.
   b. From the Template list, select **Empty BPEL Process**.
   c. Click **OK**.

5. From the Services pane on the right, drag and drop a **PartnerLink** to the visual editor.
   The Create Partner Link dialog box is displayed.

![Create Partner Link](image)
6. Click the **Service Explorer** icon (second icon from the left preceding the WSDL File field).

   The Service Explorer dialog box is displayed.

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

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

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

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

Perform the following steps:

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

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

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

The new PartnerLink appears in the visual editor.

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

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

The Edit Receive dialog box is displayed.
Perform the following steps:

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

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

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

12. Click Apply.

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

13. Click OK.

14. Select Save from the File menu.

**Deploying the BPEL Project for the Inbound Service**

1. Right-click your project in the Applications Navigator tab.

2. Select Deploy, then Your BPEL PM Server connection, and then Deploy to default domain.

   The deployment process starts automatically.
Runtime Configuration

Events are generated by activity in a database or in an application system. You can use events to trigger an action in your application. See "Configuring an Event Adapter" on page 2-21 for more information on event configuration.

**Triggering an Event in J.D. Edwards OneWorld**

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

1. Log in to your J.D. Edwards OneWorld system.

2. In the Fast Path field of the J.D. Edwards OneWorld Explorer window, type G4211 and press Enter.

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

4. Select Prompt for, and then Values.

The Processing Options dialog box is displayed.
Perform the following steps:

a. Click the Interop tab.

b. In the Transaction Type field, type JDESOOUT.

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

5. Click OK.

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

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

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

   To move to a different field, use the Tab key on your keyboard.
8. Enter a value for **Quantity Ordered** and **Item Number**.
   For example:

   ![Example Image]

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

   ![Processed Image]

   10. Click **OK**.
       An event is triggered in the J.D. Edwards OneWorld system.
Verifying the Results
To verify your results:

1. Log in to Oracle BPEL Console at
   http://host:port/BPELConsole

2. Provide a valid user name and password.

3. Click the Instances tab.

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

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

This chapter contains the following examples:

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

The scenarios shown in this chapter require the following prerequisites.

**Prerequisites**
The following are installation and configuration requirements:

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

*See Also:* Oracle Application Server Adapters Installation Guide

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

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

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

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

1. **Design Time:** OracleAS Adapter for J.D. Edwards OneWorld is configured in Application Explorer for services and events, as described in Chapter 2,
"Configuring OracleAS Adapter for J.D. Edwards One World". Integration logic is modeled in iStudio. Metadata are stored in repositories.

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

**Namespace Requirements**

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

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

1. **Input XML for ESB based on unqualified namespaces:**

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

2. **Input XML for ESB based on qualified namespaces:**

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

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

**Configuring an ESB Outbound Process**

The following example describes how to configure an ESB outbound process to your J.D. Edwards OneWorld system, using an ESB project in Oracle JDeveloper.
Prerequisites
Before you proceed, you must create an outbound WSDL file for the adapter by using the following steps:

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

Creating an Outbound ESB Project and Assigning an Outbound WSDL File
1. At the top of the upper left pane, click the Applications Navigator tab.

2. Right-click an application node that you created and select New Project.
The New Gallery window is displayed.
3. From the Items list, select **ESB Project** and click **OK**.

The Create ESB Project dialog box is displayed.

4. Perform the following steps:
   a. Specify a name for the ESB project.
      
      The Directory Name field and Diagram Name fields are updated automatically.
   b. Click **OK**.
      
      The ESB project is added at the top of the upper left pane.
5. Right-click the ESB project in the middle pane, select Create ESB Service followed by Custom Adapter.

**Note:** Users who want to create an ESB process for a Web service, must create the ESB process using the SOAP Service. Right-click the work area, select Create ESB Service from the context menu, and click SOAP Service.

The Create Adapter Service dialog box is displayed.

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

   The Service Explorer dialog box is displayed.
7. Expand your new connection under Adapter Services, followed by adapters, and then applications.

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

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

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

The new ESB project appears in the visual editor.
Creating a Read Process Operation Using the File Adapter

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

2. Enter a name for the File adapter and click the Configure adapter service wsdl icon next to the WSDL File field.
   The Adapter Configuration Wizard - Welcome window is displayed.
3. Click Next.
   The Adapter Configuration Wizard - Step 1 of 6: Service Name window is displayed.

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

5. Click Read File as the Operation Type and click Next.
   The Adapter Configuration Wizard - Step 3 of 6: File Directories window is displayed.
6. Enter the path of the input directory where you are placing the incoming XML file and click Next.

The Adapter Configuration Wizard - Step 4 of 6: File Filtering window is displayed.

7. Enter the input file extension, for example *.xml, and click Next.

The Adapter Configuration Wizard - Step 5 of 6: File Polling window is displayed.
8. Change the Polling Frequency to seconds and click Next.
   The Adapter Configuration Wizard - Step 6 of 6: Messages window is displayed.

9. Click Browse to select the WSDL.
   The Type Chooser window is displayed.
10. Click the Import WSDL File icon on the upper right corner of the dialog box. The Import WSDL File dialog box is displayed.

11. Select the WSDL file and click OK. The Imported WSDL Files folder is added.
12. Expand the Imported WSDL Files folder, select an Inline Schema, for example, jdeRequest, and click OK.

You are returned to the Adapter Configuration Wizard - Step 6 of 6: Messages window.

13. Click Next.

The Adapter Configuration Wizard - Finish window is displayed.
14. Click Finish.

You are returned to the Create File Adapter Service dialog box.

15. Click OK.

The Read operation with a routing service is added to the ESB outbound project view.
Providing a Routing Service for the Read Operation

1. Double-click the routing service.

   The Routing Service window is displayed.
2. Expand the Routing Rules.

3. Click the green plus sign icon, which represents the option to Create a new Routing Rule.

   The Browse Target Service Operation window is displayed.

4. Expand Services in project, Default System, your adapter service node, for example, JDE_ESB_Outbound, and select the service name, for example, GetEffectiveAddress.

5. Click OK.

   You are returned to the Routing Rules window.
6. Click the icon next to the <<Transformation Map>> field (Select an existing mapper file or create a new one).

The Request Transformation Map dialog box is displayed.

7. Select the Create New Mapper File option, specify the file name, and click OK.

The following mapping window is displayed.
8. Select the WSDL file and map it to the Write operation.
   Once you map the WSDL file, the Auto Map Preferences dialog box is displayed.

9. Click OK.
   The mapping is completed as shown in the following window.
10. Double-click the ESB outbound project file in the left pane, for example, ESB_Outbound.esb.

Notice that the Routing service is now created for the Read operation.
Creating a Write Process Operation Using the File Adapter

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

The Create File Adapter Service dialog box is displayed.

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

   The Adapter Configuration Wizard - Welcome window is displayed.
3. Click Next.
   The Adapter Configuration Wizard - Step 1 of 4: Service Name window is displayed.

4. Click Next.
   The Adapter Configuration Wizard - Step 2 of 4: Operation window is displayed.

5. Click Write File as the Operation Type and click Next.
   The Adapter Configuration Wizard - Step 3 of 4: File Configuration window is displayed.
6. Enter the path of the output directory and name of the output file and click Next. The Adapter Configuration Wizard - Step 4 of 4: Messages window is displayed.

7. Click Browse to select the WSDL. The Type Chooser window is displayed.
8. Expand the Project WSDL Files folder, select an Inline Schema and click OK. You are returned to the Adapter Configuration Wizard - Step 4 of 4: Messages window.

9. Click Next. The Adapter Configuration Wizard - Finish window is displayed.
10. Click Finish.

You are returned to the Create File Adapter Service dialog box.

11. Click OK.

The Write operation is added to the ESB outbound project view.
Providing a Routing Service for the Write Operation

1. Double-click the routing service.

   The Routing Service window is displayed.

2. Expand the Routing Rules.
3. Click the icon next to the <<Target Operation>> field (Browse for target service operations).

The Browse Target Service Operation window is displayed.

4. Expand Services in project, Default System, your adapter service node, for example, JDE_ESB_Outbound, and select the service name, for example, Write.

5. Click OK.

You are returned to the Routing Rules window.

6. Click the icon next to the <<Transformation Map>> field (Select an existing mapper file or create a new one).
The Reply Transformation Map dialog box is displayed.

7. Select the Create New Mapper File option, specify the file name, and click OK.

The following mapping window is displayed.

8. Select the WSDL file and map it to the Write operation.

Once you map the WSDL file, the Auto Map Preferences dialog box is displayed.
9. Click OK.

The mapping is completed as shown in the following window.

10. Double-click the ESB outbound project file in the left pane, for example, ESB_Outbound.esb.

Notice that the Routing service is now created for the Write operation.
Deploying the Project

1. Right-click the created project, for example, ESB_Outbound, select Register with ESB, and the server connection, for example, ServerConnection1.
After successful deployment, the Registration of services Successful message is displayed.

![ESB Registration Summary]

2. Logon to the ESB Control console to check whether the project has been successfully deployed.

![Login dialog]

The deployed process is listed under the Default System node.

3. Place the XML file in the folder that you specified during the creation of the Read operation.
4. Check whether you are receiving the response in the output folder, which you have specified during the creation of the write operation and also the corresponding instance in the ESB Control console.

5. If the response is not received in the output folder, check the instance and the logs for the corresponding errors in the ESB Control console.
Configuring an ESB Inbound Process

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

**Prerequisites**

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

1. Create a target using Application Explorer.
2. Create a channel.
3. Create a WSDL file with the noport option.
4. Restart the Oracle Application Server.

**Creating an Inbound ESB Project and Assigning an Inbound WSDL File**

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

2. Right-click the application node you created and select **New Project**. The New Gallery window is displayed.
3. From the Items list, select **ESB Project** and click **OK**.

   The Create ESB Project dialog box is displayed.

4. Perform the following steps:
   
a. Specify a name for the ESB project.
      The Directory Name field and Diagram Name fields are updated automatically.
   
b. Click **OK**.
      The ESB project is added at the top of the upper left pane.
5. Right-click the ESB project in the middle pane, select Create ESB Service followed by Custom Adapter.

The Create Adapter Service dialog box is displayed.

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

The Service Explorer dialog box is displayed.
7. Expand your new connection under Adapter Services, followed by adapters, and then applications.

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

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

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

The new ESB project appears in the visual editor.

Creating a Write Process Operation Using the File Adapter

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

The Create File Adapter Service dialog box is displayed.
2. Enter a name for the File adapter and click the Configure adapter service wsdl icon next to the WSDL File field.

The Adapter Configuration Wizard - Welcome window is displayed.

3. Click Next.

The Adapter Configuration Wizard - Step 1 of 4: Service Name window is displayed.

4. Click Next.

The Adapter Configuration Wizard - Step 2 of 4: Operation window is displayed.
5. Click Write File as the Operation Type and click Next.
   The Adapter Configuration Wizard - Step 3 of 4: File Configuration window is displayed.

6. Enter the path of the output directory and name of the output file and click Next.
   The Adapter Configuration Wizard - Step 4 of 4: Messages window is displayed.
7. Click Browse to select the WSDL.
   The Type Chooser window is displayed.

8. Click the Import WSDL File icon on the upper right corner of the dialog box.
   The Import WSDL File dialog box is displayed.
9. Select the WSDL file and click OK.
   The Imported WSDL Files folder is added.

10. Expand the Imported WSDL Files folder, select an Inline Schema, for example, LOCATION_SYNC, and click OK.
    You are returned to the Adapter Configuration Wizard - Step 4 of 4: Messages window.
11. Click Next.

The Adapter Configuration Wizard - Finish window is displayed.

12. Click Finish.

You are returned to the Create File Adapter Service dialog box.
13. Click OK.

The Write operation with a routing service is added to the ESB inbound project view.

Providing a Routing Service for the Write Operation
1. Double-click the routing service.
The Routing Service window is displayed.

2. Expand the Routing Rules.

3. Click the green plus sign icon, which represents the option to Create a new Routing Rule.
The Browse Target Service Operation window is displayed.

4. Expand Services in project, Default System, your adapter service node, for example, JDE_ESB_Inbound_RS, and select the service name, for example, Write.

5. Click OK.

You are returned to the Routing Rules window.

6. Click the icon next to the <<Transformation Map>> field (Select an existing mapper file or create a new one).

The Request Transformation Map dialog box is displayed.
7. Select the Create New Mapper File option, specify the file name, and click OK. The following mapping window is displayed.

8. Select the WSDL file and map it to the Write operation. Once you map the WSDL file, the Auto Map Preferences dialog box is displayed.
9. Click OK.

The mapping is completed as shown in the following window.

10. Double-click the ESB inbound project file in the left pane, for example, ESB_Inbound.esb.

Notice that the Routing service is now created for the Write operation in the middle pane.
Deploying the Project

1. Right-click the created project, for example, ESB_Outbound, select Register with ESB, and the server connection, for example, ServerConnection1.

After successful deployment, the Registration of services Successful message is displayed.

2. Logon to the ESB Control console to check whether the project has been successfully deployed.
The deployed process is listed under the Default System node.

3. Trigger the event.
4. Check whether you are receiving the response in the output folder, which you have specified during the creation of the write operation.
5. If the response is not received in the output folder, check the instance and the logs for the corresponding errors in the ESB Control console.
Troubleshooting and Error Messages

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

- Troubleshooting
- BSE Error Messages

The adapter-specific errors listed in this chapter can arise whether using the adapter with an OracleAS Adapter J2CA or with a OracleAS Adapter Business Services Engine (BSE) configuration.

Troubleshooting

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

- OracleAS Adapter Application Explorer (Application Explorer)
- J.D. Edwards OneWorld
- OracleAS Adapter J2CA
- BSE

---

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

- OracleAS Adapter J2CA trace information can be found under the `OracleAS_home\opmn\logs` directory.
- BSE trace information can be found under the `OracleAS_home\j2ee\home\applications\ws-app-adapter\ibse\ibselogs` directory.
- The log file for Application Explorer can be found under the `OracleAS_home\adapters\application\tools` directory.

---

**Application Explorer**

To use Application Explorer on **Windows** for debugging or testing purposes, load the batch script `ae.bat`, found under:

`OracleAS_home\adapters\application\tools`

On **UNIX**, load the shell script `iwae.sh`, found under:

`OracleAS_home/adapters/application/tools`
Troubleshooting

<table>
<thead>
<tr>
<th>Error</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cannot connect to OracleAS Adapter for J.D. Edwards OneWorld from Application Explorer:</td>
<td>Ensure that:</td>
</tr>
<tr>
<td>Problem activating adapter. (Failed to connect to J.D.Edwards, check system availability and configuration parameters:...)</td>
<td>- J.D. Edwards OneWorld is running.</td>
</tr>
<tr>
<td>Check logs for more information.</td>
<td>- The J.D. Edwards OneWorld user ID and password is correct.</td>
</tr>
<tr>
<td>The following error message appears:</td>
<td>- The port number is correct.</td>
</tr>
<tr>
<td>java.lang.IllegalArgumentException:</td>
<td>You have provided invalid connection information for J.D. Edwards OneWorld or the wrong JAR file is in the lib directory.</td>
</tr>
<tr>
<td>java.lang.Exception: Error Logon to J.D. Edwards OneWorld System</td>
<td></td>
</tr>
<tr>
<td>J.D. Edwards OneWorld does not appear in the Application Explorer Adapter node list.</td>
<td>Ensure that the J.D. Edwards OneWorld JAR files, are added to the lib directory.</td>
</tr>
<tr>
<td>Logon failure error at run-time.</td>
<td>If the password for connecting to your J.D. Edwards OneWorld system is not specified when creating a target or with the Edit option in Application Explorer, you will be unable to connect to J.D. Edwards OneWorld. The connection password is not saved in repository.xml. Update the password using the Edit option in Application Explorer, then restart the application server.</td>
</tr>
<tr>
<td>The following exception occurs when you start Application Explorer by activating ae.bat (not iaexplorer.exe):</td>
<td>This is a benign exception. It does not affect adapter functionality. Download BouncyCastle files from:</td>
</tr>
<tr>
<td>org.bouncycastle.jce.provider.BouncyCastleProvider</td>
<td></td>
</tr>
<tr>
<td>Unable to start Application Explorer in a Solaris environment. The following exception is thrown in the console:</td>
<td>JAVACMD is not set on the user system. Before starting Application Explorer, export JAVACMD as follows:</td>
</tr>
<tr>
<td>java.lang.ClassNotFoundException:</td>
<td>JAVAHOME=/&lt;jdk_home&gt;/bin/java, where &lt;jdk_home&gt; is the directory where JDK is installed on your system.</td>
</tr>
<tr>
<td>IWAFManagedConnectionFactory: License violation. at</td>
<td></td>
</tr>
<tr>
<td>com.ibi.afjca.spi.IWAFManagedConnectionFactory.licenseViolationFactory(IWAFManagedConnectionFactory.java:98)</td>
<td></td>
</tr>
<tr>
<td>com.iwayssoftware.iwae.common.JCATransport.getConnectionFactory(JCATransport.java:133)</td>
<td></td>
</tr>
<tr>
<td>com.iwayssoftware.iwae.common.JCATransport.initJCA(JCATransport.java:69)</td>
<td></td>
</tr>
<tr>
<td>com.iwayssoftware.iwae.common.JCATransport.&lt;init&gt;(JCATransport.java:62)</td>
<td></td>
</tr>
<tr>
<td>com.iwayssoftware.iwae.common.AdapterClient.&lt;init&gt;(AdapterClient.java:85)</td>
<td></td>
</tr>
<tr>
<td>com.ibi.bse.ConfigWorker.run(ConfigWorker.java:41)</td>
<td></td>
</tr>
<tr>
<td>java.lang.Thread.run(Thread.java:534)</td>
<td></td>
</tr>
<tr>
<td>Could not create the connection factory.</td>
<td></td>
</tr>
</tbody>
</table>
## J.D. Edwards OneWorld

<table>
<thead>
<tr>
<th>Error</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action code invalid</td>
<td>In the Sales Order request, the Action code appears as &quot;H,&quot; an invalid action code.</td>
<td>Use:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;I&quot; for inquiry.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;C&quot; for change.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;D&quot; for delete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- &quot;A&quot; to add a new record.</td>
</tr>
<tr>
<td>Invalid address number</td>
<td>The address number does not exist in the Address Book Master file (F0101).</td>
<td>Enter an address number using the Address Book Revisions program (PO1051). Ensure that the number entered is correct.</td>
</tr>
<tr>
<td>Record invalid</td>
<td>The record being processed either already exists for an ADD function or does not exist for an INQUIRY, CHANGE, or DELETE function.</td>
<td>If you are attempting to inquire, change, or delete a record you added previously, there could be database problems in your production library. Contact your data processing department.</td>
</tr>
<tr>
<td>Item Branch record does not exist.</td>
<td>An Item Branch record (F4102) does not exist for this item in the Branch/Plant specified.</td>
<td>Correct the Branch or enter an Item Branch record for this item in Branch Plant Item Information (P41026).</td>
</tr>
<tr>
<td>&amp;1 does not match any of the valid values.</td>
<td>The &amp;1 does not match any of the valid values specified in the Data Dictionary for this field.</td>
<td>Enter a valid value.</td>
</tr>
<tr>
<td>Date out of range.</td>
<td>The Last Service Date and the Inspection Date must be within the range of the effective dates of the Service Contract.</td>
<td>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.</td>
</tr>
<tr>
<td>Jde.net timeout exception</td>
<td>Net timeout is set to a wrong value</td>
<td>Verify that net timeout is set to 180 at jde.ini of [NETWORK QUEUE SETTINGS], for example JDENETTimeout=180</td>
</tr>
<tr>
<td>Cannot connect to EnterpriseOne Version 8.10</td>
<td>Missing required library files Kernel.jar and Connector.jar are required for version B7333.</td>
<td>jdeutil.jar and log4j.jar are required for EnterpriseOne Version 8.10, in addition to Kernel.jar and Connector.jar.</td>
</tr>
</tbody>
</table>

### OracleAS Adapter J2CA

<table>
<thead>
<tr>
<th>Error</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>In Application Explorer, the following error message appears when you attempt to connect to an OracleAS Adapter J2CA configuration:</td>
<td>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</td>
</tr>
<tr>
<td>Could not initialize JCA</td>
<td></td>
</tr>
</tbody>
</table>
BSE Error Messages

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

General Error Handling in BSE

BSE serves as both a SOAP gateway into the adapter framework and as the engine for some of the adapters. In both design time and run-time, various conditions can cause errors in BSE when Web services that use adapters run. Some of these conditions and resulting errors are exposed the same way, regardless of the specific adapter; others are

<table>
<thead>
<tr>
<th>Error</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint activation error on deployment of J.D.Edwards event handling project (inbound) in JDeveloper</td>
<td>Verify that the channel used for this inbound J2CA service is stopped in Application Explorer. If you have started this channel for testing or debugging purposes, you must stop it before starting BPEL PM Server. Endpoint activation is managed by BPEL Process Manager.</td>
</tr>
<tr>
<td>The following error message appears in BPEL PM Server Console:</td>
<td>Verify that the specified WSDL file exists at that URL and that the file is valid. Workaround: Change the WSDL location to localhost:7777. The default is 127.0.0.1:7777. Alternative workaround: Add the IP address to the Dhttp.nonProxyHosts list found in obsetenv.bat (Windows) or obsetenv.sh (UNIX)</td>
</tr>
</tbody>
</table>

BSE Error Messages
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:

```xml
  <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 run-time, the SOAP agent in BSE
produces a SOAP fault element in the generated SOAP response. The SOAP fault
element contains fault code and fault string elements. The fault string contains the
native error description from the adapter target system. Since adapters use the target
system interfaces and APIs, whether an exception is raised depends on how the target
systems interface or API treats the error condition. If a SOAP request message is
passed to an adapter by the SOAP agent in BSE, and that request is invalid based on
the WSDL for that service, the adapter may raise an exception yielding a SOAP fault.

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

**Invalid SOAP Request**

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

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>RPC server connection failed: Connection refused: connect</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Empty Result From Oracle Application Server Adapter Request

If Oracle Application Server Adapter executes a SOAP request using input parameters passed that do not match records in the target system, then the following SOAP response is generated.

```
<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="2A3CB42703EB2023F91951B89F3C5AF">
   <RunDBQueryResult run="1" />
  </m:RunDBQueryResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Note:** The condition for this adapter does not yield a SOAP fault.

Error Logging In

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

```
[2004-07-19T16:28:56:718Z] DEBUG (SOAP1) W.SOAP1.2: in XDSOAPHTTPWorker agentName is [XDSOAProxy]
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 <SOAP-ENV:Header>
  <m:ibsinfo xmlns:m="urn:schemas-iwaysoftware-com:iwse" xmlns="urn:schemas-iwaysoftware-com:iwse"
   cid="2A3CB42703EB2023F91951B89F3C5AF">
   <RunDBQueryResult run="1" />
  </m:RunDBQueryResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Troubleshooting and Error Messages  6-7
Empty Result From Oracle Application Server Adapter Request

If Oracle Application Server Adapter executes a SOAP request using input parameters passed that do not match records in the target system, then the following SOAP response is generated.

**Note:** The condition for this adapter does not yield a SOAP fault.

Invalid Call Method

If an invalid call is made to Oracle Application Server Adapter, then the following SOAP response is generated.

```
[2004-07-19T16:24:34:859Z] DEBUG (SOAP1) W.SOAP1.2: POST received
[2004-07-19T16:24:34:859Z] DEBUG (SOAP1) W.SOAP1.2: in XDSOAPHTTPWorker agentName is [XDSOAPRouter]
[2004-07-19T16:24:34:859Z] DEBUG (SOAP1) W.SOAP1.2: before parse:
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
<SOAP-ENV:Body>
<m:GetEffectiveAddress xmlns:m="urn:iwaysoftwar...[581]
```

```
[2004-07-19T16:24:34:859Z] DEEP (SOAP1) W.SOAP1.2: parse complete in 0 msecs
[2004-07-19T16:24:34:859Z] DEBUG (SOAP1) W.SOAP1.2: inside isAsync() the soap Action is ["B0100033.GetEffectiveAddressRequest#test##"]
[2004-07-19T16:24:34:859Z] DEBUG (SOAP1) W.SOAP1.2: inside isAsync() the soap Action is [B0100033.GetEffectiveAddressRequest#test##]
[2004-07-19T16:24:35:031Z] INFO (manager) MGR00X01: Adding active worker: W.SOAP1.2
<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:.onError/>
      </m:params>
    </m:callMethod>
  </m:jdeRequest>
</m:GetEffectiveAddress>
</SOAP-ENV:Body>
```

```
<SOAPAction agentName="XDSOAPRouter"
```
```
BSE Error Messages

Troubleshooting and Error Messages

6-11
This chapter includes the following topics:

- **Web Services Policy-Based Security**
- **Migrating Repositories**

### Web Services Policy-Based Security

OracleAS Adapter Application Explorer (Application Explorer) provides a security model called Web services policy-based security. This section describes how the feature works and how to configure it.

Web services provide a layer of abstraction between the back-end business logic and the user or application running the Web service. Easy application integration is enabled, but the issue of controlling the use and implementation of critical and sensitive business logic that is run as a Web service is raised.

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

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

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

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

You can assign a policy to a Business Service, or to a method within a Business Service. If a policy is only applied to a method, other methods in that Business Service will not be governed by it. However, if a policy is applied to the Business Service, all methods are governed by it. At run-time, the user ID and password that are sent to OracleAS Adapter Business Services Engine (BSE) in the SOAP request message are verified against the list of users for all policies applied to that specific Business Service. The policy type that is supported is Resource Execution, which dictates who can or cannot execute the Business Service.
When a policy is not applied, the default value for a Business Service is to "grant all". For example, anybody can execute the Business Service, until the Resource Execution policy is associated to the Business Service. At that time, only those granted execution permissions, or users not part of the group that has been denied execution permissions, have access to the Business Service.

**Configuring Web Services Policy-Based Security**

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

**Creating and Associating a User with a Policy**

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

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

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

![Configurations](image)

Perform the following steps:

a. Expand the Business Services node.

b. Expand the Configuration node.

c. Expand the Security node.

d. Expand the Users and Groups node.

![Business Services](image)

4. Right-click Users and click New User.

The New User dialog box is displayed.
Perform the following steps:

a. In the Name field, enter a user ID.

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

c. In the Description field, enter a description of the user (optional).

5. Click OK.

The new user is added under the Users node.

Creating a Group to Use with a Policy

To create a group to use with a policy:

1. Start Application Explorer.

2. Right-click the configuration to which you want to connect, for example, newtest. See Chapter 2, "Configuring OracleAS Adapter for J.D. Edwards One World" for information on creating a new configuration.

3. Select Connect.

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

Perform the following steps:

a. Expand the Business Services node.

b. Expand the Configuration node.

c. Expand the Security node.

d. Expand the Users and Groups node.
4. Right-click Groups and select New Group.
   The New Group dialog box is displayed.

   Perform the following steps:
   a. In the Name field, enter a name for the group.
   b. In the Description field, enter a description for the group (optional).
   c. From the available list of users in the left pane, select one or more users and add them to the Selected list by clicking the double right facing arrow.

5. When you have selected at least one user, click OK.
   The new group is added under the Groups node.

**Creating an Execution Policy**

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

To create an execution policy:

1. Start Application Explorer.
2. Right-click the configuration to which you want to connect, for example, SampleConfig. See Chapter 2, "Configuring OracleAS Adapter for J.D. Edwards OneWorld" for information on creating a new configuration.
3. Select Connect.
Nodes appear for Adapters, Events, and Business Services (also known as Web services).

Perform the following steps:

a. Expand the Business Services node.

b. Expand the Configuration node.

c. Expand the Security node.

d. Expand the Policies node.

4. Right-click Policies and select New Policy.

   The New Policy dialog box is displayed.

   Perform the following steps:

   a. In the Name field, enter a name for the policy.

   b. From the Type list, select Execution.

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

   d. From the available list of users in the left pane, select one or more users and add them to the Selected list by clicking the double right facing arrow.
5. When you have selected at least one user selected, click OK.

6. Click Next.

The New Policy permissions dialog box is displayed.

7. To grant permission to a user or group to execute a Business Service, select the user or group and move them into the Execution Granted list by selecting the double left facing arrow.

8. To deny permission to a user or group to execute a Business Service, select the user or group and move them into the Execution Denied list by selecting the double right facing arrow.

9. Click OK.

The following pane summarizes your configuration.

- **Name**
  - test
- **Type**
  - Execution
- **Description**
- **User and Group Restrictions**
  - group.test Execution Granted

**Using the IP and Domain Restrictions Policy Type**

You configure the IP and Domain Restriction policy type slightly differently from other policy types. The IP and Domain Restriction policy type controls connection access to BSE and therefore need not be applied to individual Web services. You need not create a policy, however, you must enable the Security Policy option in Application Explorer.

1. Start Application Explorer.
2. Right-click the configuration to which you want to connect, for example, SampleConfig. See Chapter 2, "Configuring OracleAS Adapter for J.D. Edwards One World" for information on creating a new configuration.

3. Select Connect.

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

Perform the following steps:
   a. Expand the Business Services node.
   b. Expand the Configuration node.
   c. Expand the Security node.

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

   The New IP and Domain Restriction dialog box is displayed.

   Perform the following steps:
   a. In the IP(Mask)/Domain field, enter the IP or domain name using the following guidelines.
      If you select Single (Computer) from the Type list, you must provide the IP address for that computer. If you only know the DNS name for the computer, click DNS Lookup to obtain the IP Address based on the DNS name.
      If you select Group (of Computers), you must provide the IP address and subnet mask for the computer group.
      If you select Domain, you must provide the domain name.
   b. From the Type list, select the type of restriction.
   c. In the Description field, enter a description (optional).
Migrating Repositories

During design time, the Oracle repository is used to store metadata created when using Application Explorer to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. The information in the repository is also referenced at run-time. For management purposes, you can migrate BSE and J2CA repositories that are configured for Oracle to new destinations without affecting your existing configuration. For example, you may want to migrate a repository from a test environment to a production environment.

Migrating a BSE Repository

To migrate a BSE repository:

1. Copy the BSE control service URL, for example:
   
   http://localhost:7777/ibse/IBSEServlet/admin/iwcontrol.ibs

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

3. From the menu bar, click SOAP.
   
   A list of options appears.

4. Select Create new SOAP request.
   
   The WSDL file location dialog box is displayed.

Perform the following steps:
Migrating Repositories

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

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

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

5. Click OK.

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

   Please select a soap operation name

   GETIBSECONFIG( GETIBSECONFIG parameters )
   SETIBSECONFIG( SETIBSECONFIG parameters )
   **MIGRATEREPO( MIGRATEREPO parameters )**
   GETMTHCONNECTION( GETMTHCONNECTION parameters )
   SETMTHCONNECTION( SETMTHCONNECTION parameters )

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

   **Note:** The MIGRATEREPO(MIGRATEREPO parameters) control method is available from the BSE administration console. This control method migrates all Web services to the new (empty) repository. You can choose to migrate select Web services only.

   The following window shows the structure of the SOAP envelope.

7. Locate the Text view icon in the toolbar.
8. To display the structure of the SOAP envelope as text, click the Text view icon.

The <SOAP-ENV:Header> tag is not required and can be deleted from the SOAP envelope.

9. Locate the following section:

```xml
<m:MIGRATEREPO xmlns:m="urn:schemas-iwaysoftware-com:jul2003:ibse:config" version=""">
  <m:repositorysetting>
    <m:rname>oracle</m:rname>
    <m:rconn>String</m:rconn>
    <m:rdriver>String</m:rdriver>
    <m:ruser>String</m:ruser>
    <m:rpwd>String</m:rpwd>
  </m:repositorysetting>
  <m:servicename>String</m:servicename>
</m:MIGRATEREPO>
```

Perform the following steps:

a. For the <m:rconn> tag, replace the String placeholder with a repository URL where you want to migrate your existing BSE repository.

The Oracle repository URL has the following format:

```
jdbc:oracle:thin:@[host]:[port]:[sid]
```

b. For the <m:rdriver> tag, replace the String placeholder with the location of your Oracle driver.

c. For the <m:ruser> tag, replace the String placeholder with a valid user name to access the Oracle repository.

d. For the <m:rpwd> tag, replace the String placeholder with a valid password to access the Oracle repository.

10. Perform one of the following migration options.

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

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

- If you want to migrate multiple Web services from the current BSE repository, duplicate the <m:servicename> tag for each Web service, for example:

```
<m:servicename>JDEService1</m:servicename>
<m:servicename>JDEService2</m:servicename>
```

- If you want to migrate all Web services from the current BSE repository, remove the <m:servicename> tag.

11. From the menu bar, click SOAP and select Send request to server.
Your BSE repository and any Web services you specified are now migrated to the new Oracle repository URL you specified.

**Migrating a J2CA Repository**

To migrate a J2CA repository:

1. Navigate to the location of your J2CA configuration directory where the repository schemas and other information is stored, for example:  
   
   ```
   OracleAS_home\adapters\application\config\JCA_CONFIG
   ```
   
   Where JCA_CONFIG is the name of your J2CA configuration.

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

3. Place this file in a new J2CA configuration directory to migrate the existing repository.

   Your J2CA repository is migrated to the new J2CA configuration directory.
Configuring J.D. Edwards OneWorld for Outbound and Inbound Processing

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

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

- Using the GenJava Development Tool (Outbound Processing)
- Specifying Inbound Functionality for a Business Function
- Configuring J.D. Edwards OneWorld Inbound Processing

Using the GenJava Development Tool (Outbound Processing)

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

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

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

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

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

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

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

GenJava is located in the <install>\system\bin32 directory. You run GenJava from the command line. There are several GenJava command options that are available.

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

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

The following command runs with a JDEScript file and prompts a J.D. Edwards OneWorld log on window, where you must enter a valid user ID, password, and environment:

GenJava /cmd .\AddressBook.cmd

The contents of the resulting AddressBook.cmd file are shown as follows:

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

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

Specifying Inbound Functionality for a Business Function

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

Note: Any instances of “outbound” terminology that is associated with J.D. Edwards OneWorld inbound processing also refers to event handling. For the SOA suite, they are inbound. Take note of this as you review the outbound and inbound material in this appendix.

Configuring J.D. Edwards OneWorld Inbound Processing

This section describes how to configure J.D. Edwards OneWorld inbound processing using the J.D. Edwards OneWorld Enterprise Server.

Configuring Custom Functions

To configure custom functions:

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

\ \ JDE\E812\DDP\Outbound

2. Copy the iwoevent.dll and iwoevent.cfg files in the new Outbound folder.
The following is a sample iwoevent.cfg file that is provided as an example:

```plaintext
common.trace=on
alias.iwserv14=172.30.162.48:1236, trace=on
trans.JDESOOUT=iwserv14
```

Currently, the OracleAS Adapter for J.D. Edwards OneWorld supports 100 entries (alias names) in the configuration file.

3. Create a folder using the alias names that are specified in the iwoevent.cfg file under the following directory:
   ```plaintext
   \JDEdwards\E812\DDP\Outbound
   ```

4. Make the following changes to your environment variables:

   a. Create a new variable called JAVA_HOME.
   b. Set this to the location of your JDK installation location, for example, `D:\jdk1.3`.
   c. Add the following to your PATH variable:
      ```plaintext
      %JAVA_HOME%\bin; %JAVA_HOME%\jre\bin; %JAVA_HOME%\jre\bin\classic;
      ```
   d. Create a new variable called IWOEVENT HOME.
      The value should be set to the full path of the Outbound directory, for example:
      ```plaintext
      D:\JDEdwards\E812\DDP\Outbound
      ```

5. Modify `[JDENET_KERNEL_DEF6]` in the JDE.INI file, which is located in the `\system\bin32` directory on the JDE Enterprise Server, by adding the following section:

```ini
[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:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Call Object dispatch DLLName</th>
<th>XML Trans dispatch DLLName</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS400</td>
<td>XMLCALLOBJ</td>
<td>XMLTRANS</td>
</tr>
<tr>
<td>HP9000B</td>
<td>libxmlcallojb.sl</td>
<td>libxmltransactions.lo</td>
</tr>
<tr>
<td>Sun or RS6000</td>
<td>libxmlcallojb.so</td>
<td>Libxmltransactions.so</td>
</tr>
</tbody>
</table>

**Subscribing to Inbound Transactions**

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

1. Type P0047 in the Fast Path field and press Enter.
The Work With Data Export Controls window opens.

2. Click Add.

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

3. Perform the following steps:
   a. Type JDESOOUT in the Transaction field.
   b. Type SO in the Order Type field.
   c. Type NotifyOnUpdate in the first row of the Function Name column.
d. Type the absolute path to the location of the iwoevent.dll file in the first row of
the Function Library column, for example:
D:\JDEdwards\E812\DDP\Outbound\iwoevent.dll
e. Type 1 in the first row of the Execute for Add column if you want the
notifications for add/insert.
f. Make the same decision for update, delete, and inquiry and type 1 in the
appropriate column.
g. Type 1 in the Launch Immediately column to launch the Outbound Subsystem
batch process (R00460).

4. Click OK.

Starting the Outbound Scheduler Subsystem Process (R00460)

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

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

The Work With Batch Versions - Available Versions window opens.
2. Type R00460 in the Batch Application field and click Find.

3. Select Interoperability Generic Outbound Subsystem UBE (XJDE0001) and click Select.
   The Version Prompting window opens.

4. Click Submit.

5. Navigate to the last screen and click OK.

Verifying the Subsystem Process
This section describes how to verify the outbound scheduler subsystem process.

1. Type WSJ in the Fast Path field and press Enter.
Configuring J.D. Edwards OneWorld Inbound Processing

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

The Work With Server (Subm Jobs) window opens.
5. Verify that R is listed in the Job Status column.

**Configuring P4210 (Sales Order) to Trigger an Event**

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

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

The Interactive Versions window opens.

2. Type P4210 in the Interactive Application field and click Find.
3. Select a document version from the table, for example, RIS0001 - Sales Order Entry - SO Order Type.

4. Click Row from the menu bar and select Processing Options.
The Processing Options dialog box opens.

5. Click the Interop tab.

6. Type JDESOOUT in the Transaction Type field.

7. Add Sales Order.
Verifying the Configuration Steps
This section describes how to verify the configuration steps by updating F0046.

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

   The P0046 - Work With Processing Log window opens.

2. Click Find.

   The following data is displayed.

<table>
<thead>
<tr>
<th>User ID</th>
<th>Batch Number</th>
<th>Transaction Number</th>
<th>Line Number</th>
<th>Trans</th>
<th>Or Ty</th>
<th>Seq</th>
<th>UDE Name</th>
<th>Version</th>
<th>S P</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDE</td>
<td>15148</td>
<td>101322</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15148</td>
<td>101324</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15150</td>
<td>101326</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15151</td>
<td>101328</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15152</td>
<td>101330</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15153</td>
<td>101332</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15154</td>
<td>101334</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15155</td>
<td>101336</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15156</td>
<td>101338</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15157</td>
<td>101340</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15158</td>
<td>101342</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15159</td>
<td>101344</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>JDE</td>
<td>15160</td>
<td>101346</td>
<td>10002000UT</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>JDE</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

3. Search for the corresponding transaction.

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

```
Event call begin...
Server time : Tue May 27 07:23:55 2008
userId : JDE
batchNumber : 15205
transactionNumber : 103494
lineNumber : 1.000000
transactionType : JDESOOUT
sequenceNumber : 1.000000
Request xml:
==========================================
<?xml version="1.0" encoding="UTF-8"?><jde><request><connection><dsn /><user />
<password /><sp><proc>JDESOOUT</proc><data><ediUserId>JDE</ediUserId><ediBatchNumber>15205</ediBatchNumber><ediTransactionNumber>103494</ediTransactionNumber></data></sp></connection></request></jde>
==========================================
```
The Adapter for J.D. Edwards OneWorld supports the jdeRequest and jdeResponse XML structures for executing business functions within OneWorld. Using J.D. Edwards OneWorld XML, you can:

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

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

- Issuing a Single-Function Request
- Issuing a Multiple-Function Request
- Sample Sales Order Request
- Sample Sales Order Response

**Issuing a Single-Function Request**

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

**Executing a Business Function with a Single-Function Call**

The following is a sample GetEffectiveAddress jdeRequest.

```xml
<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>
</params>
</callMethod>
```
Issuing a Multiple-Function Request

The following is a sample GetEffectiveAddress jdeResponse.

```xml
<?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 name="szAddressLine3"/>
      <param name="szAddressLine4"/>
      <param name="szZipCodePostal">80237</param>
      <param name="szCity">Denver</param>
      <param name="szCountyAddress"/>
      <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, and file cleanup logic.

```xml
<?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>
  </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>
  </callMethod>
  <callMethod app='XMLTest' name='F4311FClearWorkFiles'
    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>
    </params>
  </callMethod>
</jdeRequest>
```
Issuing a Multiple-Function Request

```xml
<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'>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>
```

B-4  Oracle Application Server Adapter for J.D. Edwards OneWorld User's Guide
Issuing a Multiple-Function Request

<callMethod app='XMLTest' name='F4311EditDoc' runOnError='no' returnNullData='no'>
  <params>
    <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='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='mnAddressNumber' idref='supplierNumber'></param>
    <param name='szOrderCompany' idref='orderCompany'></param>
    <param name='szVersionProcOption' idref='version'></param>
    <param name='cUseWorkFiles'>2</param>
    <param name='cConsolidateLines'>0</param>
    <param name='mnProcessID' idref='processID'></param>
  </params>
</callMethod>
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
<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' id='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'>2</param>
    <param name='szOrderCompany' id='orderCompany'>00200</param>
    <param name='mnOrderNumber'>0</param>
  </params>
</callMethod>
```
<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 name="szDeliveryInstructions02">
<param name="szPrintMessage"/>
<param name="szSupplierPriceGroup"/>
<param name="szPaymentTerms"/>
<param name="szTaxExplanationCode"/>
<param name="szTaxRateArea"/>
<param name="szTaxCertificate"/>
<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 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="mnTriangulationRateFromCurrency">0</param>
<param name="mmTriangulationRateToCurrency">0</param>
<param name="cCurrencyConversionMethod" /></param>
<param name="szPriceAdjustmentScheduleN" /></param>
<param name="cAIAADocument" /></param>
<param name="mmProcessID" id="processID">2612</param>
<param name="mmTransactionID" 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="szPurchaseOrderPrOptFileVersion" 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="mmQuantityOrdered">1</param>
<param name="mnUnitPrice">32,100</param>
<param name="szLineType">S</param>
<param name="szDescription1">Bike Rack - Trunk Mount</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="mmBuyerNumber">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>
Issuing a Multiple-Function Request

<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="cSourceRequestingPOGeneration">0</param>
<param name="cInventoryInterface">Y</param>
<param name="szTransactionCurrencyCode">USD</param>
<param name="szBaseCurrencyCode">USD</param>
<param name="cSourceRequestingPOGeneration">0</param>
<param name="cInventoryInterface">Y</param>
<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="cOrderLineNumber">2</param>
<param name="cDetailActionCode">1</param>
<param name="cProcessEdits">1</param>
<param name="cRecordWrittenToWorkFile">2</param>
<param name="cCurrencyProcessingFlag">Y</param>
<param name="szPurchaseOrderPrOptVersion" idref="version">ZJDE0001</param>
<param name="szOrderCompany" idref="orderCompany">00200</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>
</params>
</callMethod>
<br>
<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>
<param name="mnAgreementSupplement">0</param>
<param name="jdEffectiveDate"></param>
<param name="szPurchasingCostCenter"></param>
<param name="szObjectAccount"></param>
<param name="szSubsidiary"></param>
<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="F4311EndDoc" returnNullData="no" runOnError="no" app="XMLTest">
<returnCode code="0"></returnCode>
<params>
<param name="szComputerID" idref="machineKey">XEENT</param>
<param name="mnJobNumber" idref="jobNumber">3</param>
</params>
</callMethod>
</callMethod>
<callMethod name="F4311EditDoc" returnNullData="no" runOnError="no" app="XMLTest">
<returnCode code="0"></returnCode>
<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>
<callMethod name="F4311EndDoc" returnNullData="no" runOnError="no" app="XMLTest">
<returnCode code="0"></returnCode>
<params>
<param name="szComputerID" idref="machineKey">XEENT</param>
<param name="mnJobNumber" idref="jobNumber">3</param>
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
<?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>
  </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>
  </callMethod>
</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:

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

Using the Sales Order Response

The following is the jdeResponse document.

```xml
<?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>
</jdeResponse>
```
<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="szCMComputerID" idref="2">XERTN</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="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="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="mnSalesOrderFlags">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>
Warning: Order Quantity Exceeds what is Available

<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="cMRecdWrittenToWF">1</param>
  <param name="szCMComputerID" idref="2">XEENT</param>
  <param name="cCMErrorConditions">1</param>
  <param name="szCMOrderCo">00200</param>
  <param name="szCMOrderType">SO</param>
  <param name="szCMBusinessUnit">M30</param>
  <param name="mnCMShipToNo">4242</param>
  <param name="jdRequestedDate">2000/03/29</param>
  <param name="jdPromisedDate">2000/03/29</param>
  <param name="szCMItemNo">1001</param>
  <param name="szCMLocation">. . </param>
  <param name="szCMDescription1">Bike Rack-Trunk Mount</param>
  <param name="szCMDescription2"></param>
  <param name="szCMLineType">S</param>
  <param name="szCMLastStatus">900</param>
  <param name="szCMNextStatus">540</param>
  <param name="mnCMQtyOrdered">10</param>
  <param name="mnCMQtyBackordered">10</param>
  <param name="mnCMUnitPrice">44,99</param>
  <param name="mnCMUnitCost">32,1000</param>
  <param name="szCMPrintMesg"></param>
  <param name="cCMPaymentInstrument"></param>
  <param name="cCMSalesTaxableYN">N</param>
  <param name="cCMAssociatedText"></param>
  <param name="szCMTransactionUOM">EA</param>
  <param name="szCM PricingUOM">EA</param>
  <param name="mnCMItemWeight">800</param>
  <param name="szCMWeightUOM">OZ</param>
  <param name="mnCMForeignUnitPrice">44,99</param>
  <param name="mnCMForeignUnitCost">32,1000</param>
  <param name="mnCMDiscountFactor">1</param>
  <param name="mnCMLineNo">2</param>
  <param name="szCMProgramID">XMLInterop</param>
  <param name="szCMVersion">ZJDE0001</param>
  <param name="mnCMSupplierNo">4343</param>
  <param name="mnCMWKOrderTotal">494,89</param>
  <param name="mnCMWKForeignOrderTotal">494,89</param>
  <param name="mnCMWKTotalCost">321</param>
  <param name="mnCMWKForeignTotalCost">321</param>
  <param name="cCMWKSourceOfData"></param>
  <param name="cCMWKCheckAvailability">1</param>
  <param name="mnCMLastLineNoAssigned">2</param>
  <param name="cCMStockingType">P</param>
  <param name="cCMParentItmMethdOfPriceCalcn">1</param>
  <param name="mnCMShortItemNo">60003</param>
  <param name="szCMSalesOrderFlags">0</param>
  <param name="jdCMPriceEffectiveDate">2000/03/29</param>
  <param name="jdCMPromisedShip">2000/03/29</param>
  <param name="mnCMQuantityAvailable">-44</param>
  <param name="mnCMItemVolume_ITVL">22.5</param>
  <param name="szVolumeUOM_VLUM">FC</param>
</params>
<param name="szRevenueBusinessUnit">M30</param>
<param name="mmProcessID">2252</param>
<param name="mmTransactionID">4</param>
</params><errors><error code="030B">Warning: Order Quantity Exceeds what'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>
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
access methods, 2-24
access rights, 7-2
action codes, 6-3
adapter configuration
   requirements, 5-1
   updating, 3-2
adapter connections, 3-4
adapter exceptions, 6-2 to 6-5
adapter integration
   BPEL Process Manager and, 4-1
      InterConnect and, 5-1
Adapter Lib Directory parameter, 2-2 to 2-4
adapter types, 1-1, 1-6, 3-1
AdapterName parameter, 3-4 to 3-5
adapters, 1-1 to 1-8
   configuring, 3-2
   deploying, 1-6, 3-1 to 3-2, 4-2
   integrating, 5-1
   troubleshooting, 6-1 to 6-11
Add Channel dialog box, 2-21, 4-21
Add Target dialog box, 2-11
Address Book Master file, 6-3
Address Book Revisions program, 6-3
address numbers, 6-3
Admin Password parameter, 2-3 to 2-4
Admin User parameter, 2-4
agents, 1-3
alias section of iwoevent.cfg file, 2-25
aliases, 2-25 to 2-26
application adapter integration
   BPEL Process Manager and, 4-1
      InterConnect and, 5-1
application adapters, 1-1 to 1-8
   configuring, 3-2
   deploying, 1-6, 3-1 to 3-2, 4-2
   integrating, 5-1
   troubleshooting, 6-1 to 6-11
application clients, 3-4
Application Explorer, 1-6, 2-1, 2-21, 4-2
   application systems and, 2-11
   channels and, 2-21, 2-24
   debugging and, 6-1 to 6-2
   event ports and, 2-21
   J2CA configuration and, 3-2
OracleAS Adapter J2CA and, 6-3
   schemas and, 2-14
security and, 7-1, 7-6
testing and, 6-1 to 6-2
troubleshooting, 6-1 to 6-2
WSDL files and, 4-21
Application parameter, 4-23
application systems, 2-21
   Application Explorer and, 2-11
   supported, 2-11
Applications tab, 4-10, 4-25
Applications-Navigator pane, 4-17, 4-18, 4-29
Assign activities, 4-15 to 4-17
Assign Activity dialog box, 4-16
Assign activity icon, 4-16
Available list, 7-4 to 7-5

B
back-end workflows
calling, 4-18
Basic tab, 2-22
batch processes, 1-4
batch script, 6-1
batch.log file, 2-24
BPEL Console, 4-18, 4-33
BPEL Designer, 4-1
BPEL domain
   passwords and, 4-33
BPEL Process Manager
   adapter integration and, 4-1
   adapter request-response services and, 4-18
   OracleAS Adapter for J.D. Edwards OneWorld and, 4-1
BPEL Process Project dialog box, 4-11, 4-26
BPEL processes, 4-3, 4-10, 4-25
   deploying, 4-17, 4-29
   designing, 4-11
   JDeveloper and, 4-18
   managing, 4-19
   monitoring, 4-18
testing, 4-19
BPEL Processes tab, 4-19
BPEL projects, 4-10
   creating, 4-3
BPEL Server, 4-17, 4-27
connecting to, 4-10
Branch Plant Item Information, 6-3
BSE (OracleAS Adapter Business Services Engine), 1-6
configuring, 2-2, 2-8
connection access to, 7-6
troubleshooting, 6-4 to 6-11
BSE configuration page, 2-2 to 2-3
BSE control service URL, 7-8
BSE deployment, 1-6
BSE repositories, 7-8 to 7-10
BSE settings window, 2-3
BSE system settings, 2-3 to 2-5
BSE URL field, 2-8
business events, 1-1
business function calls, B-1
business functions, 1-8, B-1
creating schemas for, 2-15
executing, B-1 to B-3
business services
creating, 2-18 to 2-19
deploying, 7-1
testing, 2-20
Business Services node, 7-2 to 7-5, 7-7

C
CCI (Common Client Interface), 3-1
CCI calls, 3-1
channel configuration, 2-22
channel configuration parameters, 2-22, 4-22
Application, 2-22, 4-23
Host, 2-22, 4-22
Is Keep Alive, 2-22, 4-22
Is Length Prefix, 2-22, 4-22
Is XML, 2-22, 4-22
JDE Environment, 2-22, 4-22
Port Number, 2-22, 4-22
Schema style, 4-23
Server IP address, 2-22, 4-23
Server Port, 2-22, 4-23
Synchronization Type, 2-22
User id, 2-22, 4-22
User password, 2-22, 4-22
channels, 2-21
creating, 2-21, 2-23, 4-21 to 4-23
debugging and, 4-23
deleting, 2-23
editing, 2-23
ports and, 2-21
starting, 2-23, 4-23
stopping, 2-23, 4-23
testing and, 4-23
Channels node, 2-23, 4-21
channels. See also listeners
Common Client Interface (CCI), 3-1
common section of iwoevent.cfg file, 2-25
Compiler tab, 4-18
Configuration node under Business Services, 7-2 to 7-5, 7-7
configuration parameters, 3-3, 3-5
IWayConfig, 3-3
IWayHome, 3-3
IWayRepoPassword, 3-3
IWayRepoURL, 3-3
IWayRepoUser, 3-3
Loglevel, 3-3
configuration requirements
integration and, 5-1
configurations
OracleAS Adapter Business Services Engine (BSE), 2-8 to 2-9
overwriting, 3-3 to 3-5
Configurations node, 2-8 to 2-9
configuring adapters, 3-2
configuring BSE system settings, 2-3 to 2-5
configuring repositories, 2-5 to 2-6
connecting to BPEL Server, 4-10
connecting to J.D. Edwards OneWorld, 2-11 to 2-13, 6-2
connecting to OracleAS Adapter J2CA, 6-3
Connection dialog box, 2-13
connection factories, 3-4
collection information, 2-25 to 2-26, 6-2
connection parameters, 4-3, 6-2
AdapterName, 3-4
Application, 2-13
Config, 3-4
Country, 3-4
Hostname, 2-1 to 2-3, 2-9
JDE environment, 2-13
Language, 3-4
Loglevel, 3-4
Password, 2-2 to 2-3, 3-4 to 3-5
Port, 2-2 to 2-3
Server IP address, 2-13
Server Port, 2-13
User id, 2-12, 2-13
User password, 2-13, 4-22
UserName, 3-4
connection pooling, 3-4
collections
creating, 2-11 to 2-13
troubleshooting, 6-2 to 6-3
ConnectionSpec, 3-4 to 3-5
Connector
deploying to Oracle Application Server, 3-1
connector factories, 3-2
connector factory objects, 3-2
multiple, 3-3
control methods, 7-9
copy rules
creating, 4-17
Copy Rules tab, 4-16
Country parameter, 3-4
Create Copy Rule dialog box, 4-16 to 4-17
Create Partner Link dialog box, 4-11, 4-26 to 4-27
Create Variable dialog box, 4-14, 4-29
creating BPEL processes, 4-3, 4-11
creating BPEL projects, 4-10
creating channels, 4-21 to 4-23
creating copy rules, 4-17
creating repository projects, 2-8 to 2-9
creating schemas, 2-14 to 2-16

dash
Dashboard tab, 4-18
data
 - manipulating, 4-18
data dictionaries, 6-3
Data Export Control table, 1-4
data management, A-2
data queues, 1-4
Data Source Name (DSN), 2-25
database connections, 3-3
database repositories, 2-6
database tables, 1-3
databases, 3-3
date ranges, 6-3
Debug Level parameter, 2-4
DEBUG log level, 3-5
deploying adapters, 1-6, 3-1 to 3-2, 4-2
deploying outbound processes, 4-17
deployments
 - BSE, 1-6
Description field, 2-11, 2-19, 2-22, 7-3 to 7-5, 7-7
design time, 4-21, 7-8
 - configuring, 4-2, 5-1
design time service adapter connections, 3-4
designing BPEL processes, 4-3, 4-11
DNS Lookup option, 7-7
DNS name, 7-7
document types
 - request, 1-3, B-1 to B-3, B-11, B-13
 - response, 1-3, B-6, B-13 to B-17
 - XML, 1-3
Domain Name System (DNS), 7-7
domain names, 7-7
Domain option, 7-7
Domain Password field, 4-17
DSN (Data Source Name), 2-25

eclipse
Eclipse. See JDeveloper
Edit Invoke dialog box, 4-14
Edit Receive dialog box, 4-28
editing targets, 2-14
EIS (enterprise information systems), 1-3, 3-5
EJB (Enterprise Java Beans), 3-1
Encoding parameter, 2-4
Enterprise Connector for J2EE Connector Architecture (J2CA), 1-6, 2-8 to 2-10
Oracle Application Server and, 3-1
enterprise information systems (EIS), 1-3, 3-5
Enterprise Java Beans (EJB), 3-1
error messages, 4-16, 4-18, 6-2 to 6-11
target systems and, 6-5
event adapters, 2-21, 2-24

event data
 - receiving, 4-21 to 4-33
 - event integration, 4-21 to 4-33, 5-32
 - event listeners, 2-24 to 2-25
 - event messages, 4-21, 4-33
events, 1-1, 2-21
 - configuring, 2-21
 - verifying, 4-33
Execution Denied list, 7-6
Execution Granted list, 7-6
Existing Service Names list, 2-19
Export WSDL dialog box, 4-9
external listeners, 1-4

f
fault code elements, 6-5
fault string elements, 6-5
file system repositories, 2-5
flat files, 1-3 to 1-4
functional modeling, 5-1

g
generating schemas, 2-14 to 2-16
GenJava program, 2-12
GenJava repository, 2-12
GenJava utility, 2-16
GenJava wrappers, 2-16
Grant Access check box, 7-7
Group (of Computers) option, 7-7
Group node, 7-4
groups, 7-3

h
Home field, 2-9
Host parameter, 2-22, 4-22
Hostname parameter, 2-1 to 2-3, 2-9

i
inbound BPEL processes, 4-10, 4-25
inbound interactions, 4-25
inbound J2CA services, 4-21
inbound processing, 1-4, A-1
inbound services, 4-29
Initiate tab, 4-19
input records
 - creating, 3-5
Inspection date, 6-3
installation directories, 3-3
installation requirements
 - integration and, 5-1
 - instances of policy types, 7-1
Instances tab, 4-33
integration requirements, 5-1
integration scenarios, 5-1
interactions
 - creating, 3-5
 - executing, 3-5
inbound, 4-25
outbound, 4-9, 4-18
InterConnect
  adapter integration and, 5-1
  internal listeners, 1-4
interoperability framework, 1-4
  inbound processing, 1-4
  outbound processing, 1-5
invalid address numbers, 6-3
invalid records, 6-3
Invalid Settings warning, 4-16
Invoke activities, 4-14 to 4-17
IP (Mask)/Domain field, 7-7
IP addresses, 2-13, 2-25, 7-7
IP and Domain Restriction policy type, 7-6
Is Keep Alive parameter, 2-22, 4-22
Is Length Prefix parameter, 2-22, 4-22
Is XML parameter, 2-22, 4-22
Item Branch records, 6-3
IWAEC function, 3-5
IWAFCConnectionSpec, 3-4 to 3-5
IWAFIteractionSpec, 3-5
IWayConfig parameter, 3-3
IWayHome parameter, 3-3
IWayRepoPassword parameter, 3-3
IWayRepoURL parameter, 3-3
IWayRepoUser parameter, 3-3
IWOEvent listener exit, 2-24
iwoevent.cfg file, 2-25 to 2-26
iwoevent.log file, 2-24
iwse.ora file, 2-6

J
J2CA (Enterprise Connector for J2EE Connector Architecture), 1-6, 2-8 to 2-10
J2CA configuration, 2-6
  Application Explorer and, 3-2
J2CA repositories, 7-11
J2CA resource adapters, 1-1, 3-1
J2CA services, 4-9 to 4-21
JAR files, 4-23, 6-2
Java application clients, 3-4
Java files, 2-12
Java program clients, 3-1
J.D. Edwards OneWorld, 3-5
  connecting to, 2-11 to 2-13, 6-2
J.D. Edwards OneWorld Event Listener, 2-24 to 2-25
J.D. Edwards OneWorld ThinNet API, 1-4, 1-8, 2-14,
  B-1
J.D. Edwards processes, 4-18
JDE Environment parameter, 2-13, 4-22, 4-23
JDE OneWorld dialog box, 2-11
JDE Service process link, 4-19
jdeTransactionName, 2-25
JDEdwards node, 4-21
jdeRequest documents, B-1 to B-3, B-11, B-13
jdeResponse documents, B-6, B-11, B-13 to B-17
JDeveloper, 4-2, 4-10
  BPEL processes and, 4-17, 4-18
JNDI lookup, 3-4

L
Language parameter, 2-4, 3-4
Last Service date, 6-3
License and Method dialog box, 2-19
License field, 2-19
licenses, 2-19
list of nodes, 6-2
listener configuration files, 2-25
listener exits, 2-24
listener types, 1-4
listeners, 1-4, 2-24 to 2-26
listeners. See also channels
log files, 6-1
log levels, 3-3 to 3-5
Loglevel parameter, 3-3 to 3-4
logon parameters, 6-2
Logon tab, 2-12

M
managed connector factories, 3-2
managed connector factory objects, 3-2
  multiple, 3-3
ManagedConnectionFactory parameter, 3-3 to 3-5
manipulating data, 4-18
mapping security, 3-5
mapping variables, 4-17
mappings
  verifying, 4-17
Master Business Functions (MBF), 1-3 to 1-6, 2-11,
  A-1
  executing, 2-14
MBF (Master Business Functions), 1-3 to 1-6, 2-11,
  A-1
  executing, 2-14
messages, 1-1 to 1-3
  error, 4-18
  event, 4-21, 4-33
  logging, 4-18
  warning, 4-16 to 4-18
Messages log area, 4-18
metadata, 2-5, 7-8
Method Name field, 2-19
methods, 7-1
migrating repositories, 7-8 to 7-10
migrating Web services, 7-10
multiple-function requests, B-2 to B-6
My Role field, 4-13
My Role list, 4-28

N
Name field, 2-11, 2-22, 7-3 to 7-5
Namespace field, 4-11, 4-26
New Configuration dialog box, 2-8 to 2-9
New Gallery window, 4-10, 4-25
New Group dialog box, 7-4
New Policy permissions dialog box, 7-6
New User dialog box, 7-2
Node list, 6-2
nodes, 2-8 to 2-9, 2-13, 2-21
Business Services, 7-2 to 7-5
Channels, 2-23, 4-21
Configuration under Business Services, 7-2 to 7-5, 7-7
Configurations, 2-8 to 2-9
Group, 7-4
JDEdwards, 4-21
Policies, 7-5
Process, 4-17
Security, 7-5, 7-7
Users, 7-3
Users and Groups, 7-2 to 7-3
Number of Async. Processors parameter, 2-4

O

OC4J (Oracle Application Server Containers for J2EE), 3-1
deploying, 3-1 to 3-7
OC4J-ra.xml file, 3-3
OneWorld Event Listener, 2-24 to 2-25
Oracle Application Server
Enterprise Connector for J2EE Connector Architecture and, 3-1
installation directory and, 3-3
Oracle Application Server Containers for J2EE (OC4J), 3-1
deploying, 3-1 to 3-7
Oracle BPEL Console, 4-18
Oracle BPEL Server, 4-17, 4-27
connecting to, 4-10
Oracle databases, 3-3
Oracle JDeveloper, 4-2, 4-10
BPEL processes and, 4-17, 4-18
Oracle repositories
configuring, 2-5 to 2-6
migrating, 7-8 to 7-10
OracleAS Adapter Application Explorer. See Application Explorer
OracleAS Adapter Business Services Engine (BSE), 1-6
configuring, 2-2, 2-8
troubleshooting, 6-4 to 6-11
OracleAS Adapter for J.D. Edwards OneWorld
BPEL Process Manager and, 4-1
configuring, 2-1
deploying, 1-1, 4-2
integrating with J.D. Edwards OneWorld, 4-9
InterConnect and, 5-1
troubleshooting, 6-1 to 6-11
OracleAS Adapter J2CA, 3-5
Application Explorer and, 6-3
connecting to, 6-3
OracleAS InterConnect adapter integration and, 5-1
outbound agents, 2-24
outbound integration, 4-9
outbound interactions, 4-9, 4-18
outbound JCA services, 4-9
outbound processes, 4-17
outbound processing, 1-5, 2-24
outbound transactions, 2-24

P

parameter types
channel configuration, 2-22, 4-22
configuration, 3-3
connection, 2-13, 3-4, 4-3, 6-2
repository, 2-5
repository migration, 7-9
security, 2-4
system, 2-4
Parameters tab, 2-15
Partner Link Type field, 4-13
partner links, 4-11, 4-26 to 4-28
Partner Role field, 4-13, 4-28
Partner Role list, 4-13
Password field, 7-3
Password parameter, 2-2 to 2-3, 3-4 to 3-5, 6-2
Password Prompt dialog box, 4-17, 4-18
passwords, 3-3 to 3-5, 4-17, 4-18, 4-29
BPEL domain and, 4-33
permissions, 7-1, 7-6
policies, 7-1, 7-4
Policies node, 7-5
Policy parameter, 2-4
policy types
instances of, 7-1
IP and Domain Restriction, 7-6
policy-based security, 7-1 to 7-8
Port Number parameter, 2-22, 4-22
port numbers, 2-13
Port parameter, 2-2 to 2-3
ports, 2-21, 2-25
channels and, 2-21
Preparser tab, 2-22, 4-22
privileges, 7-1
process activities, 4-14
Process Activities pane, 4-11, 4-17, 4-26 to 4-28
PROCESS function, 3-5
Process Manager. See BPEL Process Manager
Process node, 4-17
processes
BPEL, 4-10
creating, 4-11
inbound, 4-25
outbound, 4-9
synchronous, 4-10
projects
BPEL, 4-3
properties, 3-3
Protocol list, 2-22, 4-21

Q

queues, 1-4
R
ra.xml file, 2-2
Receive activities, 4-14, 4-28
Receive dialog box, 4-29
record identifiers, 2-24
record types, 3-5
records
   creating, 3-5
   processing, 6-3
Reply activities, 4-17
repositories
   configuring, 2-5 to 2-6
   migrating, 7-8 to 7-10
Repository Driver parameter, 2-5
repository information, 2-5
repository migration parameters, 7-9
repository parameters
   Driver, 2-5
   Password, 2-5
   Pooling, 2-5
   Type, 2-5
   URL, 2-5
   User, 2-5
Repository Password parameter, 2-5
Repository Pooling parameter, 2-5
repository projects
   creating, 2-8 to 2-9
   Web services and, 2-8
Repository tab, 2-12
repository tables, 2-6
Repository Type parameter, 2-5
Repository URL parameter, 2-3, 2-5
repository URLs, 7-10
Repository User parameter, 2-5
repository.xml file, 7-11
request documents, 1-3, B-1 to B-3, B-11, B-13
Request Schema tab, 2-15
request schemas, 2-14 to 2-16
request-response services, 4-18
   creating, 4-9
resource adapters, 1-1, 1-6, 3-1
Resource Execution policy type, 7-1
response documents, 1-3, B-1, B-6, B-13 to B-17
Response Schema tab, 2-15
response schemas, 2-14 to 2-16
runtime, 7-8
   configuring, 5-1
runtime service adapter connections, 3-4
S
sample files, B-1 to B-17
   multiple-function requests, B-1 to B-2
   purchase order, B-3 to B-6
   sales order, B-11 to B-17
   single-function requests, B-1 to B-2
scenarios
   integration, 5-1
Schema style parameter, 4-23
schemas, 1-3
creating, 2-14 to 2-16
security, 7-1 to 7-8
   Application Explorer and, 7-1, 7-6
   configuring, 7-2
   mapping, 3-5
Security node, 7-2 to 7-5, 7-7
security parameters
   Admin Password, 2-4
   Admin User, 2-4
   Policy, 2-4
security policies, 7-1, 7-4
Security Policy option, 7-6
Selected list, 7-4 to 7-5
Server IP address parameter, 2-13, 4-23
server name, 2-13
Server Port parameter, 2-13, 4-23
service adapter connections, 3-4
Service Contract date, 6-3
service integration, 4-9, 5-2
service names, 2-19
Service Provider list, 2-8 to 2-9
Service-Oriented Architecture (SOA), 4-1
services, 1-1
   creating, 2-18 to 2-19
   testing, 2-20
servlets, 3-1
   WSDL, 4-27
Single (Computer) option, 7-7
single-function requests, B-1 to B-2
SOA (Service-Oriented Architecture), 4-1
SOAP agents, 6-5
SOAP envelopes, 7-9
SOAP faults, 6-5
soap operation name dialog box, 7-9
SOAP requests, 6-5 to 6-8, 7-1, 7-6
   creating, 7-8
   errors and, 6-5 to 6-6, 6-8
SOAP responses, 6-5 to 6-8
software reuse, 5-1
storing metadata, 7-8
Synchronization Type list, 4-22
Synchronization Type parameter, 2-22
synchronous access, 1-4
synchronous processes, 4-10
system parameters
   Adapter Lib Directory, 2-4
   Debug Level, 2-4
   Encoding, 2-4
   Language, 2-4
   Number of Async. Processors, 2-4
system settings, 2-3 to 2-5
T
target systems, 6-5
Target Type list, 2-11
targets
   connecting to, 2-11 to 2-13
   defining, 2-11 to 2-13
   deleting, 2-14
disconnecting from, 2-14
editing, 2-14
TCP Listener dialog box, 2-22
Template list, 4-11, 4-26
ThinNet API, 1-4, 1-8, 2-14, B-1
trace information, 6-1
trace settings, 2-25
trans section of iwoevent.cfg file, 2-25
transaction processing, 1-1, A-1
transaction tables, 1-4
transactions
calling, 4-18
storing, 2-5
troubleshooting, 6-1 to 6-11
Application Explorer, 6-1 to 6-2
BSE, 6-4 to 6-11
Web services, 6-4 to 6-11
Type list, 7-5, 7-7

U
updating adapter configuration, 3-2
User id parameter, 2-2, 2-12, 2-13, 2-22, 4-22
User password parameter, 2-13, 2-22 to 2-23, 4-22
UserName parameter, 3-4 to 3-5, 6-2
users
associating, 7-2
Users and Groups node, 7-2 to 7-3
Users node, 7-3

V
values, 6-3
variables, 4-17
verifying events, 4-33
verifying mappings, 4-17
visual editors, 4-11, 4-26 to 4-27

W
warning messages, 4-16 to 4-18
Web service names, 2-19
Web services, 1-1
creating, 2-18 to 2-19
delivering, 2-5
deploying, 6-5, 7-1
integrating, 4-1
migrating, 7-10
repository projects and, 2-8
testing, 2-20
troubleshooting, 6-4 to 6-11
Web Services Description Language (WSDL), 4-21, 4-27
generating, 4-9, 4-23
Web services policy-based security, 7-1 to 7-8
workspaces, 4-10, 4-25
WSDL (Web Services Description Language), 4-21, 4-27
generating, 4-9, 4-23
WSDL Chooser dialog box, 4-12 to 4-13, 4-27
WSDL documents, 4-1