iWay

iWay Applications System Adapter for J.D. Edwards
EnterpriseOne User's Guide
Version 5 Release 5

Updated for J2EE CA 1.5

Preface

This document is written for system integrators with programming backgrounds and an understanding of J.D. Edwards EnterpriseOne in an application space. Extensive knowledge of EnterpriseOne is not required but may be helpful in learning about the adapter.

This document describes how to work with the adapter tools to develop online interconnections to EnterpriseOne. For system integrators concerned with the development of a client/server interface between EnterpriseOne and other applications, this guide addresses the EnterpriseOne integration aspects. It does not cover other applications or application wrappers.

How This Manual Is Organized

The following table lists and describes the chapters and appendixes in this manual.

Chapter/Appendix		Contents
1	The iWay Application System Adapter for J.D. Edwards EnterpriseOne	Introduces the iWay Application System Adapter for J.D. Edwards EnterpriseOne.
2	Creating XML Schemas and Web Services for J.D. Edwards EnterpriseOne	Describes how to create schemas and Web services for J.D. Edwards EnterpriseOne functions.
3	Listening for Database Events	Describes how to use the iWay Application System Adapter for J.D. Edwards EnterpriseOne, deployed to a server to listen for events.
4	Using Web Services Policy-Based Security	Describes how to configure Web services policy-based security.
5	Management and Monitoring	Describes the management and monitoring tools provided by iBSE and JCA.
6	Troubleshooting	Provides troubleshooting information for the iWay Application System Adapter for J.D. Edwards EnterpriseOne.

Chapter/Appendix		Contents
A	Configuring EnterpriseOne for Outbound Transaction Processing	Describes how to enable outbound transaction processing in EnterpriseOne and how to modify the jde.ini file for XML support.
B	Sample Files	Provides examples of the jdeRequest and jdeResponse XML structures for executing business functions within EnterpriseOne.

Documentation Conventions

The following table lists and describes the conventions that apply in this manual.

Convention	Description
THIS TYPEFACE or this typeface	Denotes syntax that you must enter exactly as shown.
this typeface	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
this typeface	Represents a placeholder (or variable) in a text paragraph, a cross-reference, or an important term.
this typeface	Highlights a file name or command in a text paragraph that must be lowercase.
this typeface	Indicates a button, menu item, or dialog box option you can click or select.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices; type one of them, not the braces.
|	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (…).

Convention	Description
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

Visit our World Wide Web site, http://www.iwaysoftware.com, to view a current listing of our publications and to place an order. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have questions about iWay Application System Adapter for J.D. Edwards EnterpriseOne?

If you bought the product from a vendor other than iWay Software, contact your distributor.

If you bought the product directly from iWay Software, call Information Builders Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your iWay Application System Adapter for J.D. Edwards EnterpriseOne questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code (*xxxx.xx*) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, http://www.informationbuilders.com. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of www.informationbuilders.com also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

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 tables list the specifications our consultants require.

Platform	
Operating System	
OS Version	
Product List	
Adapters	
Adapter Deployment	For example, JCA, Business Services Engine, iWay Adapter Manager
Container Version	

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

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

The following table lists the types of Application Explorer. Specify the version (and platform, if different than listed previously) in the columns provided.

Application Explorer Type	Version	Platform
Swing		
Servlet		
ASP		

In the following table, specify the JVM version and vendor in the columns provided.

Version	Vendor

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

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

The following table lists error/problem files that might be applicable.

XML schema
XML instances
Other input documents (transformation)
Error screen shots
Error output files

XML schema
Trace and log files
Log transaction

User Feedback

In an effort to produce effective documentation, the Documentation Services staff welcomes your opinions regarding this manual. Please use the Reader Comments form at the end of this manual to communicate suggestions for improving this publication or to alert us to corrections. You also can go to our Web site, http://www.iwaysoftware.com and use the Documentation Feedback form.

Thank you, in advance, for your comments.

iWay Software Training and Professional Services

Interested in training? Our Education Department offers a wide variety of training courses for iWay Software and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our World Wide Web site, http://www.iwaysoftware.com or call (800) 969-INFO to speak to an Education Representative.

Interested in technical assistance for your implementation? Our Professional Services department provides expert design, systems architecture, implementation, and project management services for all your business integration projects. For information, visit our World Wide Web site, http://www.iwaysoftware.com.

Contents

Contents

The iWay Application System Adapter for J.D. Edwards EnterpriseOne

Topics:

- Executing a J.D. Edwards EnterpriseOne Master Business Function

- Accessing Data Stored in J.D. Edwards EnterpriseOne

- J.D. Edwards EnterpriseOne Interoperability Framework

- Deployment Information for iWay Application System Adapter for J.D. Edwards EnterpriseOne

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

This section provides information about the iWay Application System Adapter for J.D. Edwards EnterpriseOne to help you accomplish your integration projects.

Note: J.D. Edwards EnterpriseOne was formerly called J.D. Edwards OneWorld.

Executing a J.D. Edwards EnterpriseOne Master Business Function

You can use the iWay Application System Adapter for J.D. Edwards EnterpriseOne to invoke a J.D. Edwards EnterpriseOne Master Business Function (MFB), such as Address Book, Purchase Order, and Sales Order. You can also use the adapter as part of an iWay Business Services integration effort to connect EnterpriseOne with non-EnterpriseOne systems.

The adapter can receive an XML document, or it can run one or more MBFs by passing an XML document into EnterpriseOne through the J.D. Edwards EnterpriseOne ThinNet API.

Resource Adapters

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

Accessing Data Stored in J.D. Edwards EnterpriseOne

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

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

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

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

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

  The request document conforms to a request XML schema.

  The schema is based on metadata in the EIS.

- Calls the underlying function in the EIS and waits for its response.

- Transforms the response from the EIS-specific data format to an XML document.

  The response document conforms to a response XML schema for the adapter.

  The schema is generated by Application Explorer and is based on metadata in the EIS.

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

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

- Receives messages from an EIS client.

- Transforms the EIS-specific message format into an XML format.

  The XML format conforms to an XML schema.

  The schema is based on metadata in the EIS.

Propagating External Listeners Into EnterpriseOne

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

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

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

Propagating Internal Listeners Out of EnterpriseOne

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

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

J.D. Edwards EnterpriseOne Interoperability Framework

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

The iWay Application System Adapter for J.D. Edwards EnterpriseOne supports the following integration access methods:

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

The following diagram shows the J.D. Edwards EnterpriseOne inbound processing (from the EIS to application server) framework. It shows the EnterpriseOne components and the agent adapter in the inbound processing sequence.
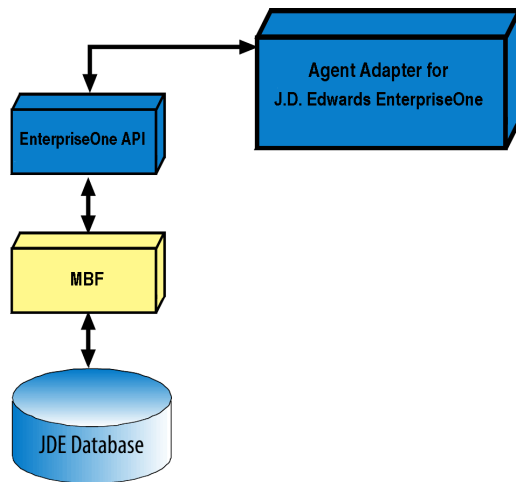
J.D. Edwards EnterpriseOne Inbound Processing Framework

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

J.D. Edwards EnterpriseOne Outbound Processing Framework

The following diagram shows the J.D. Edwards EnterpriseOne outbound processing framework. It shows the EnterpriseOne components and the listener adapter in the outbound processing sequence.
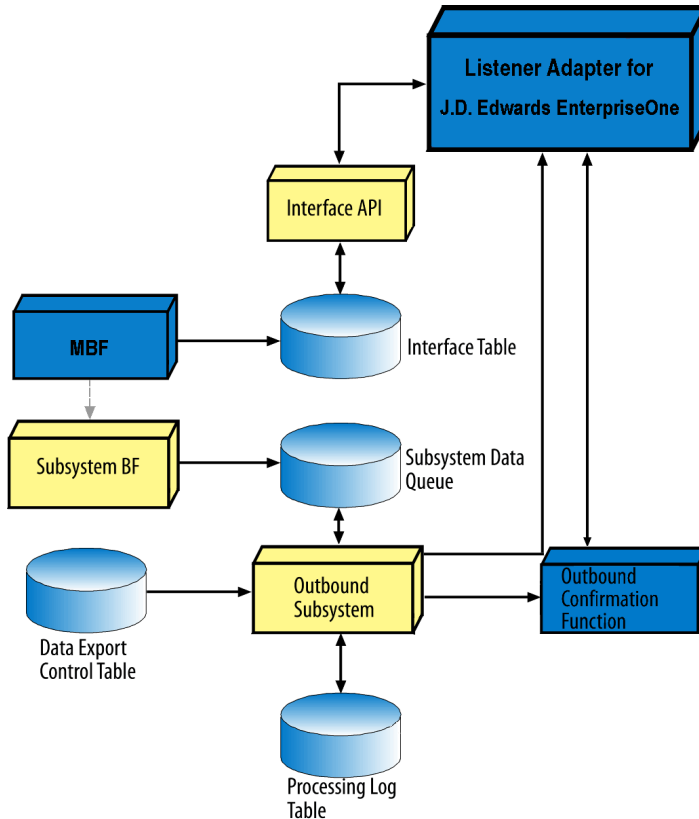
J.D. Edwards EnterpriseOne Outbound Processing Framework

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

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

Deployment Information for iWay Application System Adapter for J.D. Edwards EnterpriseOne

The iWay Application System Adapter for J.D. Edwards EnterpriseOne works with iWay Application Explorer in conjunction with one of the following components:

• iWay Business Services Engine (iBSE)

• iWay Enterprise Connector for J2EE™ Connector Architecture (JCA)

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

Both iBSE and the iWay Connector for JCA are deployed to an application server with iWay Application Explorer and the adapters.

Deployment Information Roadmap

The following table lists deployed components and describes where you can find information on each one. A description of the iWay Business Services Engine (iBSE) and the iWay Enterprise Connector for J2EE Connector Architecture (JCA) follows the table.

Deployed Component	For more information, see
iWay Application Explorer	*iWay Installation and Configuration*
iWay Business Services Engine (iBSE)	*iWay Installation and Configuration*
iWay Enterprise Connector for J2EE Connector Architecture (JCA)	*iWay Connector for JCA User's Guide* *iWay Installation and Configuration*

iWay Business Services Engine

iWay Business Services Engine (iBSE) exposes—as Web services—enterprise assets that are accessible from adapters regardless of the programming language or the particular operating system.

iBSE simplifies the creation and execution of Web services when running:

- Custom and legacy applications

- Database queries and stored procedures

- Packaged applications

- Terminal emulation and screen-based systems

- Transactional systems

Web services is a distributed programming architecture that overcomes hurdles with Enterprise Application Integration (EAI) that other programming models cannot. It enables programs to communicate with one another using a text-based platform- and language-independent message format called XML.

Coupled with a platform- and language-independent messaging protocol called SOAP (Simple Object Access Protocol), XML enables application development and integration by assembling previously built components from multiple Web services.

The iWay Enterprise Connector for J2EE Connector Architecture

The iWay Enterprise Connector for J2EE Connector Architecture (JCA) enables developers of JCA-compliant applications to deploy iWay adapters as JCA resources.

The iWay Connector for JCA is distributed as a standard Resource Adapter Archive (RAR) for deployment to the application server. Thus, the connector can be used in systems that are non-compliant, although services such as pooled connections are not available.

Two connectors are distributed in the iWay installation package. One conforms to the JCA 1.0 specification, with extensions that allow for the consumption of events. The other conforms to the JCA 1.5 specification. The JCA 1.0 connector provides for event functionality through the configuration of ports and channels. When using the adapter in conjunction with a JCA 1.5 connector, there is no need to create event ports to dispose of event data. However, you must create a channel to enable event listening capabilities. For more information on event capabilities of the iWay JCA connectors, see Chapter 3, *Listening for Oracle E-Business Suite Events*.

For more information on installing and deploying both connectors, see *iWay Installation and Configuration*.

Creating XML Schemas and Web Services for J.D. Edwards EnterpriseOne

Topics:

- Overview
- Starting Servlet Application Explorer
- Defining a Target to J.D. Edwards EnterpriseOne
- Managing a Connection to J.D. Edwards EnterpriseOne
- Creating an XML Schema for a EnterpriseOne Master Business Function
- Creating a Business Service

This section describes how to open a connection to J.D. Edwards EnterpriseOne, how to create schemas for EnterpriseOne functions, and how to create business services (Web services). It describes how to use iWay Application Explorer as deployed to Sun Java System Application Server.

Note: This guide is specifically for EnterpriseOne. iWay Software also has an iWay Application Systems Adapter for J.D. Edwards World.

Overview

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

External applications that access EnterpriseOne through the iWay Application System Adapter for J.D. Edwards EnterpriseOne use either XML schemas or Web services to pass data between the external application and the adapter. The following topics describe how to use iWay Application Explorer to create XML schemas and Web services for the J.D Edwards Master Business Functions (MBF) used with the adapter.

For more information on creating Web services and on Application Explorer in general, see the *iWay Application Explorer User's Guide.*

Using GenJava to Generate a Schema

To create schemas for the adapter, you must use GenJava wrappers. You create the GenJava wrappers using the EnterpriseOne utility called GenJava. You use Application Explorer to generate schemas against EnterpriseOne GenJava wrappers.

GenJava is supplied as a command line process with several run-time options. For more information on GenJava, see the *J.D. Edwards Interoperability Guide for OneWorld Xe.*

Starting Servlet Application Explorer

Before you can use Servlet Application Explorer, you must start your application server.

Procedure: **How to Open Servlet Application Explorer**

To open Application Explorer:

1. Ensure that your application server is running.

2. Enter the following URL in your browser:

   `http://hostname:port/iwae/index.html`

   where:

   hostname

   Is the name of the machine where your application server is running.

   port

   Is the port for the domain you are using for iWay.

After you start Application Explorer, the following window opens as shown in the following image. The iWay Adapters tab is active, and a list of the supported adapters appears. In the upper right, the Available Hosts drop-down list displays the Connector for JCA or Servlet iBSE instance you can access. A Welcome message appears in the right pane.
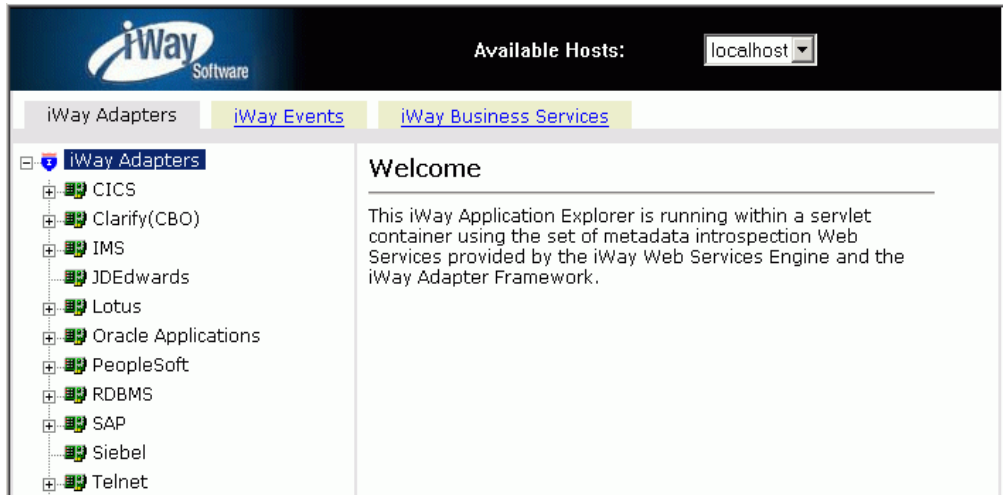
For more information on adding instances, see the *iWay Installation and Configuration* manual.

You are ready to create new targets for J.D. Edwards.

Defining a Target to J.D. Edwards EnterpriseOne

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 iWay adapters installed.

Connecting to EnterpriseOne

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

Procedure: How to Define a New Target to EnterpriseOne

To define a new target:

1. In the left pane, click the *JDEdwards* node.

2. In the right pane, move the pointer over *Operations* and select *Define a new target*.

   The following image shows the Add a new JDEDWARDS target dialog box that opens in the right pane, with fields that prompt you to enter the target name and description and a drop-down list from which to select the target type. The Next and Cancel buttons are active.

   Add a new JDEDWARDS target

   Targets represent configured connections to instances of backend systems. Choose a name and description for the new target that you wish to create.

   Target Name: [   ]

   Description: [   ]

   Target Type: [JDE One World   ▼]

   [Help]   [< Back]   [Next >]   [Cancel]

   a. In the Target Name field, type a name (for example, JDEConnection) for the target.

   b. In the Description field, type a brief description.

   c. From the Target Type drop-down list, select a target type (for example, JDE OneWorld).

3. Click *Next*.

The Set connection info pane opens as shown in following image with the Repository tab active and fields that prompts you to enter the repository directory and schema style.

Set connection info

Repository	Logon

Repository directory: [   ]

Schema style: [ELEMENT_STYLE   ▼]

[Help] [< Back] [Finish] [Cancel]

a. Type the path to the GenJava repository.

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

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

b. Select the Schema style of the schema for J.D. Edwards events and services. (ELEMENT_STYLE or ATTRIBUTE _STYLE).

4. Click *Finish*.

The following image shows the Logon tab active, with fields that prompt you to enter the information required for connecting to the server on which EnterpriseOne is running.

Note: The EnterpriseOne connection parameters are consistent with those found in your EnterpriseOne system. For more information on parameter values that are specific to your EnterpriseOne configuration, consult your EnterpriseOne system administrator.

5. Type the appropriate values for your target type based on the information in the following table.

   The following table lists and describes the parameters required in the Logon tab.

Target Parameter	Description
User ID	Valid user ID for J.D. Edwards EnterpriseOne.
User password	Password associated with the user ID.
JDE Environment	EnterpriseOne environment, for example, DV7333. For more information about this parameter, see your EnterpriseOne documentation or ask your EnterpriseOne system administrator.
Application	XMLInterop or the application name in EnterpriseOne. Optional.
Server IP address	Name of the server on which EnterpriseOne is running. This can be the name of the server, for example, JDEOW, or its IP address, for example, 123.45.67.89.
Server port	Port number where the server is listening, for example, 6009.
User role	User profile for the J.D. Edwards user (this identifies user privileges).

6. Click *Finish*.

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

   The following image shows a new target named JDEConnection under the JDEdwards node. The x icon to the left of JDEConnection indicates that the node is not connected.

   For information on how to create schemas for the adapter, see *Creating an XML Schema for a EnterpriseOne Master Business Function* on page 2-9.

Procedure: How to Connect to a Defined EnterpriseOne Target

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

2. Expand the *JDEdwards* node.

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

4. In the right pane, move the pointer over *Operations*, and select *Connect*.

   The Connect to JDEConnection dialog box opens, populated with values you entered for the connection parameters.

5. Verify the connection parameters. If required, provide the password and then, click *OK*.

   The following image shows that the x icon that appeared previously to the left of JDEConnection has disappeared, indicating that the node is now connected.

Managing a Connection to J.D. Edwards EnterpriseOne

To manage J.D. Edwards EnterpriseOne connections, you can:

- Disconnect from a connection that is not currently in use.

  Although you can maintain multiple open connections to different transaction processing systems, it is recommended to disconnect from connections not in use.

- Edit a connection.

- Delete a connection that is no longer required.

Procedure: How to Disconnect From a Connection to EnterpriseOne

To disconnect from a connection to J.D. Edwards:

1. Expand the *iWay Adapters* node.

2. Expand the *JDEdwards* node.

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

4. In the right pane, move the pointer over *Operations* and select *Disconnect*.

   The connection with JDEConnection is dropped, but the node remains.

   The following image shows the x icon to the left of JDEConnection, indicating that the node is disconnected.

Procedure: How to Delete a Connection to EnterpriseOne

1. Expand the *iWay Adapters* node.

2. Expand the *JDEdwards* node.

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

4. In the right pane, move the pointer over *Operations*, and select *Delete*.

   A message appears, prompting you to confirm the deletion of the node.

5. Click *OK*.

   The node disappears from the list of available connections.

Procedure: How to Edit a Target

To edit a target, you must first disconnect from the target.

1. In the left pane, click the target node.

2. In the right pane, move the pointer over *Operations* and select *Edit*.

   The Edit pane opens on the right.

3. Modify the target information.

4. To open another pane and modify additional information, click *Next*.

5. When you are finished editing, click *Finish*.

Creating an XML Schema for a EnterpriseOne Master Business Function

To execute a Master Business Function (MBF), the adapter must receive a request document through the J.D. Edwards EnterpriseOne 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.

Note: In a J2EE Connector Architecture (JCA) implementation of iWay adapters, Web services are not available. When the adapters are deployed to use the iWay Connector for JCA, the Common Client Interface provides integration services using the iWay adapters. For more information, see the *iWay Installation and Configuration* manual and the *iWay Connector for JCA User's Guide*.

Creating a Request and a Response Schema
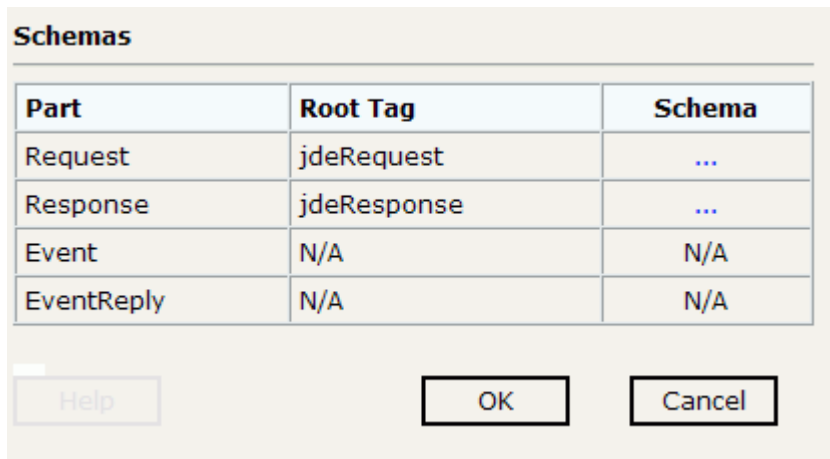
The following procedure explains how to create request and response schemas for an EnterpriseOne Master Business Function using Application Explorer.

Procedure: How to Create a Request Schema and a Response Schema

1. If you are not connected to EnterpriseOne target, connect to one, as described in *How to Connect to a Defined EnterpriseOne Target* on page 2-7.

2. Expand the *iWay Adapters* node.

3. Expand the node of the Master Business Function (MBF) for which you want to create the schema.

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

5. In the right pane, move the pointer over *Operations*, and select *Generate Schema*.

   Application Explorer creates the schemas.

   The following image shows the Schemas information window that opens in the right pane. The second column shows the root tag for the generated request and response schema, and the third column provides access to the schema XML.

Schemas

Part	Root Tag	Schema
Request	jdeRequest	...
Response	jdeResponse	...
Event	N/A	N/A
EventReply	N/A	N/A

Help   OK   Cancel

6. To view the XML for each schema, click the ellipsis (…).

The following image shows sample XML for a request schema generated by Application Explorer.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--  Generated by the iBSE 2004-02-06T19:23:15Z
   -->
- <xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <xsd:element name="jdeRequest">
    - <xsd:complexType>
      - <xsd:sequence>
        - <xsd:element name="callMethod">
          - <xsd:complexType>
            - <xsd:sequence>
              - <xsd:element
                  name="params">
                - <xsd:complexType>
                  - <xsd:sequence>
                    - <xsd:element
                        name="param"
                        minOccurs="0"
                        maxOccurs="9">
```

Creating a Business Service

You can generate a business service (also known as a Web service). You can explore the business function repository and generate business services for the functions you want to use with the adapter.

Note: In a J2EE Connector Architecture (JCA) implementation of iWay adapters, Web services are not available. When the adapters are deployed to use the iWay Connector for JCA, the Common Client Interface provides integration services using the iWay adapters. For more information, see the *iWay Installation and Configuration* manual and the *iWay Connector for JCA User's Guide*.

Procedure: How to Create a Business Service

1. Expand the *JDEdwards* node and then expand the *iWay Adapters* node.

2. Expand the node of the Master Business Function (MBF) for which you want to create a business service.

3. In the right pane, move the pointer over *Operations* and select *Create iWay Business Services*.

The following image shows the Create Web Service pane that opens, with option buttons enabling you to choose between creating a new service or using an existing service. The title indicates that the sample procedure applies to the AddressBookMaster MBF.
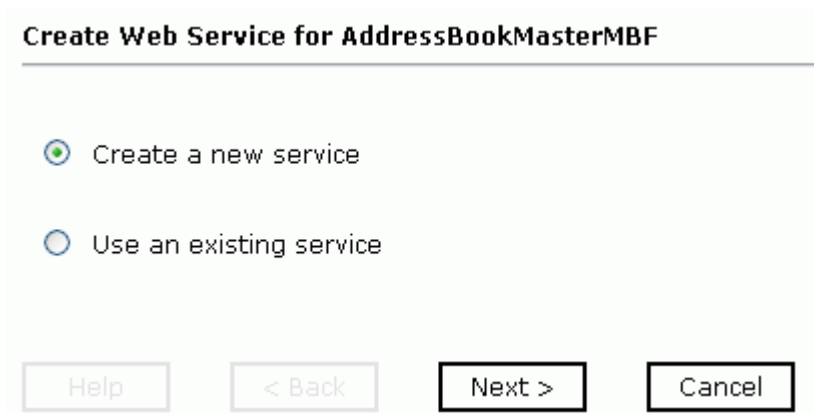
You can add the business function as a method for a new Web service or as a method for an existing one.

If you select the **Create a new service** option, another pane in the sequence opens as shown in the following image.

Create Web Service for AddressBookMasterMBF

Service Name:

Description:

License:
production
test

Help < Back Next > Cancel

a. In the Service Name field, type a name for the service.

b. In the Description field, type a brief description.

c. Select one of the available licenses.

If you select the **Use an existing service** option, a different pane opens as shown in the following image with a drop-down list from which you select the service.

Create Web Service for AddressBookMasterMBF

addbook

Help < Back Next > Cancel

4. After you choose to create a new service or select an existing service, click *Next*.

    Another pane with additional fields opens.

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

    b. In the Description field, type a brief description of the method.

5. Click *Finish*.

    Application Explorer switches the view to the iWay Business Services tab, and the new business service appears in the left pane.

Testing a Business Service

After you create a business service, test it to ensure that it functions properly. iWay provides a test tool for testing the business service.

Procedure: How to Test a Business Service

1. If you are not on the iWay Business Services tab of Application Explorer, click the tab to access business services.

2. If it is not expanded, expand the list of business services under iWay Business Services.

3. Expand the *Services* node.

4. Select the name of the business service you want to test.

    The business service name appears as a location in the right pane.

5. In the right pane, click the named business services location.

The test option appears in the right pane.

If you are testing a Web service that requires XML input, an input xml field appears.

The following image shows an input xml field, prompting you to enter XML code starting at the cursor location. An Invoke button enables you to test the operation.

6. In the input xml field, either type a sample XML document that queries the service, or browse to the location of an XML instance and click *Open*.

7. Click *Invoke*.

Application Explorer displays the results in the right pane.

The following image shows sample XML returned by Application Explorer.

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

Generating WSDL From a Web Service

Generating Web Services Description Language (WSDL) from a Web service enables you to make the Web service available to other services within a host server.

Procedure: How to Generate WSDL From a Web Service

1. Click the *iWay Business Services* tab.

2. In the left pane, expand the newly created Web service (for example, UpdateAddress).

3. In the right pane, right-click the *Service Description* hyperlink and select *Save Target as*.

The Save As dialog box opens.

4. Choose a location for the file and specify *.wsdl* for the extension.

   Note: The file extension must be .wsdl.

5. Click *Save*.

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. The user name and password values that you provided for J.D. Edwards during target creation using Application Explorer are overwritten for this Web service request. The following is a sample SOAP header that is included in the WSDL file for a Web service:

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

Note: You can remove the following tags from the SOAP header, as they are not required:

```
<m:disposition>String</m:disposition>
<m:language>String</m:language>
```

CHAPTER 3

Listening for Database Events

Topics:

- Understanding Event Functionality

- Creating an Event Port

- Creating a Channel

- The EnterpriseOne Event Listener

- Configuring the EnterpriseOne Event Listener

- Logging and Error Handling

This section describes how to use the iWay Application System Adapter for J.D. Edwards EnterpriseOne, deployed to a server, such as Sun Java System Application Server, to listen for events.

Understanding Event Functionality

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

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

The following is a description of how ports and channels work:

- Port

  A port associates a particular business object exposed by the adapter with a particular disposition. A disposition defines the protocol and location of the event data. The port defines the end point of the event consumption. For example, you can use the MSMQ protocol to route the result of a Purchase Order update in the J.D. Edwards system to a queue hosted by Sun Java System Application Server. For more information, see *Creating an Event Port on page 3-2*.

- Channel

  A channel represents configured connections to particular instances of back-end systems. A channel binds one or more event ports to a particular listener managed by the adapter. For more information, see *Creating a Channel* on page 3-15.

Important: When using the adapter in conjunction with the iWay Connector for JCA 1.5, there is no need to create event ports to dispose of event data. However, you must create a channel to enable event listening capabilities.

Creating an Event Port

Application Explorer enables you to create event ports from the iWay Adapters tab or from the iWay Events tab. You also can modify or delete an existing port.

Creating an Event Port From the iWay Adapters Tab

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

Procedure: How to Create an Event Port From the iWay Adapters Tab

1. Select the EnterpriseOne object for which you want to create an event port.

   The following image shows the selected transactions object under JDEConnection and Events in the left pane. It shows available operations in the right pane.
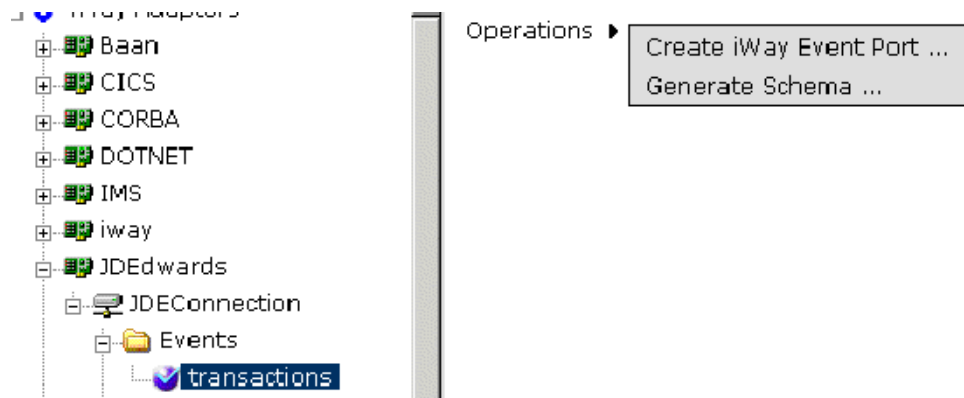
2. In the right pane, move the pointer over *Operations* and select *Create iWay Event Port*.

   The following image shows the Create Event Port dialog box that opens in the right pane, with fields prompting you to enter the event port name and description and a drop-down list from which you select the disposition protocol. In this example, FILE is the selected disposition protocol.

      a. Specify a name and a brief description for the event port.

      b. From the Disposition Protocol drop-down list, select the required disposition (for example, FILE).

3. Click *Next*.

The Specify Disposition dialog box opens in the right pane.

4. Specify the Disposition Url and click *Finish*.

For information on configuring port dispositions, see *Creating an Event Port From the iWay Events Tab on page 3-4*.

Creating an Event Port From the iWay Events Tab

The following procedures describe how to create an event port from the iWay Events tab for various dispositions using Application Explorer. You can switch between an iBSE and a JCA deployment by choosing one or the other from the drop-down menu in the upper right of Application Explorer.

The following dispositions are available when using Application Explorer in conjunction with an iBSE deployment.

- File
- iBSE
- MSMQ
- JMSQ
- SOAP
- HTTP
- MQSeries
- Mail

Note: The Mail disposition option will be supported in a future release.

The following dispositions are available when using Application Explorer in conjunction with a JCA connector deployment.

- File
- JMSQ
- HTTP
- MQSeries

You also can create an event port directly from the iWay Adapters tab. For more information, see *Creating an Event Port From the iWay Adapters Tab* on page 3-2.

Important: When using the adapter in conjunction with the iWay Connector for JCA 1.5, there is no need to create an event port to dispose of event data. However, you must create a channel to enable event listening capabilities.

Procedure: How to Create an Event Port for File

1. Click the *iWay Events* tab.

2. In the left pane, expand the *JDEdwards* node.

3. Select the *ports* node.

4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

   The following image shows the Create New Port dialog box that opens in the right pane, with fields prompting you to enter the port name and description, a drop-down list from which you select the disposition protocol, and a field prompting you to enter the disposition. In this example, the selected disposition protocol is FILE, and a disposition was entered.

   Create New Port

   Choose parameters of the port that you wish to create.

Port Name:	
Description:	
Disposition Protocol:	FILE
Disposition:	ifile://[location];errorTo=[pre-define

   Help   OK   Cancel

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

   b. From the Disposition Protocol drop-down list, select *FILE*.

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

When pointing Application Explorer to an **iBSE** deployment, specify the destination file using the following format:

```
ifile://[location];errorTo=[pre-defined port name or another
disposition url]
```

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

Important: When using the adapter in conjunction with the iWay Connector for JCA 1.5, there is no need to create event ports to dispose of event data. However, you must create a channel to enable event listening capabilities.

The following table lists and describes the disposition parameters for File.

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

5. Click *OK*.

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

You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-15.

Procedure: **How to Create an Event Port for iBSE**

1. Click the *iWay Events* tab.

2. In the left pane, expand the *JDEdwards* node.

3. Select the *ports* node.

4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

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

b. From the Disposition Protocol drop-down list, select *iBSE*.

c. In the Disposition field, type an iBSE destination using the following format:

```
ibse:svcName.mthName;responseTo=[pre-defined port name or another
disposition url];errorTo=[pre-defined port name or another
disposition url]
```

The following table lists and describes the disposition parameters for iBSE.

Parameter	Description
svcName	Name of the service created with iBSE.
mthName	Name of the method created for the Web service.
responseTo	Location where responses to the Web service are posted. A predefined port name or another full URL. Optional.
errorTo	Location where error documents are sent. A predefined port name or another full URL. Optional.

5. Click *OK*.

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

You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-15.

Procedure: How to Create an Event Port for MSMQ

1. Click the *iWay Events* tab.

2. In the left pane, expand the *JDEdwards* node.

3. Select the *ports* node.

4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

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

b. From the Disposition Protocol drop-down list, select *MSMQ*.

c. In the Disposition field, type an MSMQ destination using the following format:

```
msmq:/host/private$/qName;errorTo=[pre-defined port name or
another disposition url]
```

Note: This syntax is for a private queue. Private queues are queues that are not published in Active Directory. They appear only on the local computer that contains them. Private queues are accessible only by Message Queuing applications that recognize the full path name or format name of the queue.

The following table lists and describes the disposition parameters for MSMQ.

Parameter	Description
host	Machine name where the Microsoft Queuing system is running.
qName	Name of the private queue where messages are placed.
errorTo	Predefined port name or another disposition URL to which error logs are sent. Optional.

5. Click *OK*.

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

You are now ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-15.

Procedure: How to Create an Event Port for Sun Java System Message Queue

1. Click the *iWay Events* tab.

2. In the left pane, expand the *JDEdwards* node.

3. Select the *ports* node.

4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

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

   b. From the Disposition Protocol drop-down list, select *JMSQ*.

   c. In the Disposition field, type a JMS destination.

   When pointing Application Explorer to an **iBSE** deployment, specify the destination using the following format:

   ```
   jmsq:myQueueName@myQueueFac;jndiurl=[myurl];jndifactory=[myfactory
   ];user=[user];password=[xxx];errorTo=[pre-defined port name or
   another disposition url]
   ```

   When pointing Application Explorer to a **JCA** deployment, specify the destination using the following format:

   ```
   jms:jmsqueue@jmsfactory;jndiurl=;jndifactory=;
   ```

   Important: When using the adapter in conjunction with the iWay Connector for JCA 1.5, there is no need to create event ports to dispose of event data. However, you must create a channel to enable event listening capabilities.

The following table lists and describes the disposition parameters for JMSQ.

Parameter	Description
myQueueName or jmsqueue	JNDI name of a queue to which events are emitted.
myQueueFac or jmsfactory	A resource that contains information about the JMS Server.

jndiurl	The URL to use to contact the JNDI provider. The syntax of this URL depends on the JNDI provider being used. This value corresponds to the standard JNDI property.
	`java.naming.provider.url.`
	The URL of the Sun Java System Application Server is:
	`iiop://localhost:3700`
	3700 is a default port.

jndifactory	Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider. For Sun Java System Application Server, this is: `com.sun.jndi.cosnaming.CNCtxFactory`
user	Valid user name required to access a JMS server.
password	Valid password required to access a JMS server.
errorTo	Predefined port name or another disposition URL to which error logs are sent. Optional.

5. Click *OK*.

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

You are now ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-15.

Procedure: How to Create a Port for the SOAP Disposition

The SOAP disposition allows an event response to launch a Web service specified by a WSDL file. A soapaction is optional, the default is "".

To create a port for a SOAP disposition using Application Explorer:

1. Click the *iWay Events* tab.

2. In the left pane, expand the J.D. Edwards adapter node.

3. Select the *ports* node.

4. Move the pointer over *Operations* and select *Add a new port*.

   The Create New Port pane opens on the right.

   a. In the Port Name field, type a name for the event.

   b. In the Description field, type a brief description.

   c. From the Disposition Protocol drop-down list, select *SOAP*.

   d. In the Disposition field, enter an SOAP destination, using the following format:

   ```
   soap:[wsdl-url];soapaction=[myaction];method=[web service
   method];namespace=[namespace];responseTo=[pre-defined port name or
   another disposition URL];errorTo=[pre-defined port name or another
   disposition url]
   ```

The following table lists and defines the parameters for the disposition.

Parameter	Description
wsdl-url	The URL to the WSDL file that is required to create the SOAP message, for example, `http://localhost:7001/ibse/IBSEServlet/test/`*`webservice`*`.ibs?wsdl` where: *`webservice`* Is the name of the Web service you created using Application Explorer. This value can be found by navigating to the iWay Business Services tab and opening the Service Description option in a new window. The WSDL URL appears in the Address field. You can also open the WSDL file in a third party XML editor (for example, XMLSPY) and view the SOAP request settings to find this value.

soapaction	The method that will be called by the SOAP disposition, for example
	
	`webservice.method``@`*`test`*`@@`
	
	where:
	
	`webservice`
	   Is the name of the Web service you created using Application Explorer.
	
	`method`
	   Is the method being used.
	
	`test`
	   Is the license that is being used by the Web service.
	
	This value can be found by navigating to the iWay Business Services tab and opening the Service Description option in a new window. Perform a search for soapAction.

method	The Web service method you are using. This value can be found in the WSDL file.

Parameter	Description
namespace	The XML namespace you are using. This value can be found in the WSDL file.
responseTo	The location to which responses are posted, which can be a predefined port name or another URL. Optional.
	A predefined port name or another disposition URL. The URL must be complete, including the protocol.
errorTo	The location to which error logs are sent. Optional.
	A predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

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

You are now ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-15.

Procedure: How to Create an Event Port for HTTP

To create an event port for HTTP:

1. Click the *iWay Events* tab.

2. In the left pane, expand the *JDEdwards* node.

3. Select the *ports* node.

4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

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

   b. From the Disposition Protocol drop-down list, select *HTTP*.

   c. In the Disposition field, type an HTTP destination.

      When pointing Application Explorer to an **iBSE** deployment, specify the destination using the following format:

      ```
      ihttp://[myurl];responseTo=[pre-defined port name or another
      disposition url];
      ```

      When pointing Application Explorer to a **JCA** deployment, specify the destination using the following format:

      ```
      http://host:port/uri
      ```

Important: When using the adapter in conjunction with the iWay Connector for JCA 1.5, there is no need to create event ports to dispose of event data. However, you must create a channel to enable event listening capabilities.

The following table lists and describes the disposition parameters for HTTP when using an **iBSE** deployment.

Parameter	Description
myurl	URL target for the post operation, for example, http://myhost:1234/docroot
responseTo	Predefined port name or another disposition URL to which response documents are sent. Optional.

The following table lists and describes the disposition parameters for HTTP when using a **JCA** deployment.

Parameter	Description
host:port	Combination of the name of the host on which the server resides and the port on which the server is listening for the post operation.
uri	Universal resource identifier that completes the URL specification.

5. Click *OK*.

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

You are now ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-15.

Procedure: How to Create an Event Port for MQSeries

1. Click the *iWay Events* tab.

2. In the left pane, expand the *JDEdwards* node.

3. Select the *ports* node.

4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

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

   b. From the Disposition Protocol drop-down list, select *MQSeries*.

c. In the Disposition field, type an MQSeries destination.

When pointing Application Explorer to an **iBSE** deployment**,** specify the destination using the following format:

```
mqseries:/qManager/qName;host=[hostname];port=[port];channel=[chan
nelname];errorTo=[pre-defined port name or another disposition
url]
```

When pointing Application Explorer to a **JCA** deployment, specify the destination using the following format:

```
mq:qmanager@respqueue;host=;port=;channel=
```

Important: When using the adapter in conjunction with the iWay Connector for JCA 1.5, there is no need to create event ports to dispose of event data. However, you must create a channel to enable event listening capabilities.

The following table lists and describes the disposition parameters for MQSeries.

Parameter	Description
qManager	Name of the queue manager to which the server must connect.
qName or respqueue	Name of the queue where messages are placed.
host	Host on which the MQ server is located (MQ Client only).
port	Number to connect to an MQ server queue manager (MQ client only).
channel	Case-sensitive name of the channel that connects with the remote MQ server queue manager (MQ client only). The default channel name for MQSeries is SYSTEM.DEF.SVRCONN.
errorTo	Predefined port name or another disposition URL to which error logs are sent. Optional.

5. Click *OK*.

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

You are now ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-15.

Editing and Deleting an Event Port

The following procedures describe how to edit and delete an event port.

Procedure: How to Edit an Event Port

1. In the left pane, select the event port you want to edit.

2. In the right pane, move the pointer over *Operations* and select *Edit*.

   The Edit Port dialog box opens.

3. Make the required changes and click *OK*.

Procedure: How to Delete an Event Port

1. Select the event port you want to delete.

2. In the right pane, move the pointer over *Operations* and select *Delete*.

   A confirmation dialog box opens.

3. To delete the event port you selected, click *OK*.

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

Creating a Channel

The following procedure describes how to create a channel for your event. All defined event ports must be associated with a channel. This topic also describes how to start and stop a channel as well as how to edit and delete a channel.

Procedure: How to Create a Channel

1. Click the *iWay Events* tab.

   The adapters that appear in the left pane support events.

2. Expand the *JDEdwards* node.

   The ports and channels nodes appear in the left pane.

3. Click the *channels* node.

4. In the right pane, move the pointer over *Operations* and select *Add a new channel*.

The following image shows the Add a new JDEDWARDS channel dialog box that opens in the right pane, with fields prompting you to enter the channel name and description and a drop-down list from which to select the channel type. In this example, the selected channel type is TCP Listener.

Add a new JDEDWARDS channel

Choose a name and description for the new channel that you wish to create.

Channel Name: _____

Description: _____

Channel Type: [TCP Listener ▾]

[Help]   [< Back]   [Next >]   [Cancel]

a. Type a name (for example, NewChannel) and a brief description for the channel.

b. From the drop-down list, select *TCP Listener*.

5. Click *Next*.

The following image shows the Edit channels dialog box that opens, with fields, a drop-down list, and check boxes prompting you for information required for the creation of a channel. In this example, localhost has been entered in the Host field, and REQUEST_RESPONSE has been selected from the Synchronization Type drop-down list.

Edit channels

Host:	localhost
Port Number:	
Synchronization Type:	REQUEST_RESPONSE ▾
Is Length Prefix:	☐
Is XML:	☐
Is Keep Alive:	☐
User id:	
User password:	
JDE Environment:	
Application:	
Server IP address:	
Server port :	

| Help | < Back | Next > | Cancel |

6. Provide the system information that is specific to your J.D. Edwards environment.

The following table lists and describes the parameters required to create a channel.

Parameter	Description
Host	Name or URL of the machine where the database is installed.
Port number	Port number on which the server is listening, for example, 6009.
Synchronization Type	• If the event application expects a reply sent back to it, select *REQUEST_RESPONSE*. Specify a preemitter. • When a TCP/IP acknowledgement (ACK) is sent back to the event application, select *REQUEST_ACK*. • If the event application does not expect a return, select *REQUEST*.
Is Length Prefix	For EnterpriseOne events that send data back that is not in XML format. The TCP/IP event application must prefix the data with a 4-byte binary length field when writing the data to the TCP/IP port.
Is XML	For EnterpriseOne events that send data back in XML format. No preparser is required.
Is Keep Alive	Maintains continuous communication between the event transaction and the channel.
User ID	Valid user ID for EnterpriseOne.
User Password	Password associated with the user ID.
JDE Environment	EnterpriseOne environment, for example, DV7333. For more information about this parameter, see your EnterpriseOne documentation or ask your EnterpriseOne system administrator.
Application	XMLInterop or the application name in EnterpriseOne. Optional.
Server IP address	Name of the server on which EnterpriseOne is running. This can be the name of the server, for example, JDEOW, or its IP address, for example, 123.45.67.89.
Server port	Port number on which EnterpriseOne is running.

7. Click *Next*.

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

    b. Click the single right arrow button to transfer the selected port(s) to the list of current ports. To transfer all event ports, click the double right arrow button.

8. Click *Finish*.

An information summary for the new channel appears in the right pane. It provides the channel, description, channel status, and ports.

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

Procedure: How to Start and Stop a Channel

1. Expand the *iWay Events* node.

2. Expand the *JDEdwards* node.

3. Select the channel you want to start or stop.

To **start** the channel, move the pointer over *Operations* and select *Start the channel*.

To **stop** the channel, move the pointer over *Operations* and select *Stop the channel*.

Editing and Deleting a Channel

The following procedures describe how to edit and delete a channel.

Procedure: How to Edit a Channel

1. Expand the *iWay Events* node.

2. Expand the *JDEdwards* node.

3. In the left pane, select the channel you want to edit.

4. In the right pane, move the pointer over *Operations* and select *Edit*.

The Edit channels dialog box opens.

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

Procedure: How to Delete a Channel

1. Expand the *iWay Events* node.

2. Expand the *JDEdwards* node.

3. In the left pane, select the channel you want to delete.

4. In the right pane, move the pointer over *Operations* and select *Delete*.

A confirmation dialog box opens.

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

The channel disappears from the list in the left pane.

The EnterpriseOne Event Listener

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

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

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

Configuring the EnterpriseOne Event Listener

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

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

The EnterpriseOne Event listener includes the following components:

- Listener exit (IWOEvent)

  The file extension you use depends on your operating system, for example, for Windows, the exit is IWOEvent.dll.

- Listener configuration file (iwoevent.cfg)

- Outbound agent (XDJdeOutboundAgent)

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

Creating the iwoevent.cfg File

After EnterpriseOne invokes the EnterpriseOne 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 Sun Java System Application Server. If Sun Java System Application Server is unavailable or an exception occurs, the EnterpriseOne Event listener saves the event information in a file called batch.log. After the server becomes available, the listener sends the information. All of the log information is saved in a file called iwoevent.log.

Procedure: How to Create the iwoevent.cfg File

1. On the J.D. Edwards EnterpriseOne Server, create an iwoevent.cfg file in the defined directory. For information about the contents of this file, see *Adding Connection Information* on page 3-21.

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

   On **Windows**, add IWOEVENT_HOME to the system environment variables.

   On **UNIX**, add the following command to your start-up script:

   ```
   export IWOEVENT_HOME =/directory_name
   ```

Adding Connection Information

The EnterpriseOne Event listener requires connection information 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 EnterpriseOne 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 includes the following distinct sections:

• common

• alias

• trans

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

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

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

where:

`ipaddressordnsname`

> Is the IP address or DNS name for the server containing the iWay Application System Adapter for J.D. Edwards EnterpriseOne (required).

`port`

> Is the port defined for the adapter (required).

`on`

> Sets tracing on for the particular alias.

`off`

> Sets tracing off for the particular alias. Off is the default value.

The **trans** section of the configuration file contains transaction information required to route EnterpriseOne transactions to specified servers.

Note: If a particular EnterpriseOne 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.

`alias1,alias2,alias3`

> Is the list of aliases to which the transactions are sent.

Procedure: How to Add Connection Information to iwoevent.cfg

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 tracing on.

   `off`

   > Sets tracing off. Off is the default value.

Example: Adding Connection Information to iwoevent.cfg

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

```
common.trace=on

alias.edamcs1=172.1.1.1:3694
alias.edamcs1t=172.1.1.1:3694, trace=on
alias.edamcs2=222.2.2.2:1234

trans.JDESOW=edamcs1t,edamcs2
trans.JDEPOOUT=edamcs1
```

Logging and Error Handling

The client listener provides a log of each transaction it processes. The log is placed in iwoevent.log in the directory specified by the IWOEVENT_HOME environment variable.

When an event failure occurs, the event payload is saved to the local file system in a subdirectory of the IWOEVENT_HOME directory.

For example, if the IWOEVENT_HOME environment variable is set to d:\IWOEVENT, the iWay Application System Adapter for J.D. Edwards EnterpriseOne is not available, and you have the following alias:

`edamcs1`

The event information is saved to the following directory:

`d:\IWOEVENT\edamcs1`

The following is a sample portion of the log file.

```
...
Event call begin...
userId           : JDE
batchNumber      : 0
transactionNumber: 102628
lineNumber       : 2.000000
transactionType  : JDEWO
sequenceNumber   : 1.000000
Request xml:
========================
<?xml version="1.0" encoding="UTF-8"?><eda><request><connection><dsn
/><user /><password /><sp><proc>JDEWO    </proc><data><ediUserId>JDE
</ediUserId><ediBatchNumber>0
</ediBatchNumber><ediTransactionNumber>102628
</ediTransactionNumber></data></sp></connection></request></eda>
========================
Connection failed with Error

connect socket failed: IO_DRIVERERROR
WSAECONNREFUSED(274D)

Payload dumped into file
[g:\jdedwardsoneworld\ddp\b7333\outbound\ibiwfk\1055355515.xml]

Event call begin...
userId           : JDE
batchNumber      : 0
transactionNumber: 102629
lineNumber       : 2.000000
transactionType  : JDEWO
sequenceNumber   : 1.000000
Request xml:
========================
<?xml version="1.0" encoding="UTF-8"?><eda><request><connection><dsn />
 <user /><password /><sp><proc>JDEWO    </proc><data><ediUserId>JDE
</ediUserId><ediBatchNumber>0
</ediBatchNumber><ediTransactionNumber>102629
</ediTransactionNumber></data></sp></connection></request></eda>
========================
Connection failed with Error

connect socket failed: IO_DRIVERERROR

WSAECONNREFUSED(274D)
...
```

Example: **Supplying Connection Information**

The following example is an iwoevent.cfg file that supplies connection information.

```
DSN=jde
 Server=localhost
 Port=4575
```

where:

DSN

Is the name of the data source in dataSource.cfg (optional).

Server

Is the IP address of Sun Java System Application Server.

Port

Is the TCP port waiting for the TCP request.

Example: **Sending a Request to the Application Server (DSN Specified)**

The following is a sample request sent to Sun Java System Application Server when the DSN is specified.

```
<?xml version="1.0" encoding="UTF?8"?>

<eda>
        <request agent="XDJdeOutboundAgent">
              <connection>
                    <dsn>jde</dsn>
                    <user/>
                    <password/>
                    <sp>
                          <proc>JDESOOUT</proc>
                          <data>
                                <ediUserId>islywm</ediUserId>
                                      <ediBatchNumber>100</ediBatchNumber>
                                <ediTransactionNumber>100100
                                      </ediTransactionNumber>
                          </data>
                    </sp>
              </connection>
        </request>
</eda>
```

Example: **Sending a Request to the Application Server (DSN Not Specified)**

The following is the same request as in the previous example but without a specified DSN.

```
<?xml version="1.0" encoding="UTF?8"?>

<eda>
        <request>
              <connection>
                    <dsn />
                    <user />
                    <password/>
                    <sp>
                          <proc>JDESOOUT</proc>
                          <data>
                                <ediUserId>islywm</ediUserId>
                                <ediBatchNumber>100</ediBatchNumber>
                                <ediTransactionNumber>100100
                                      </ediTransactionNumber>
                          </data>
                    </sp>
              </connection>
        </request>
</eda>
```

When the integration server receives the XML request from the listener exit, it invokes the XDJdeOutboundAgent to process the request. The XDJdeOutboundAgent creates a J.D. Edwards XML request and executes the request against the EnterpriseOne system, using the DSN information in the DataSource.cfg file.

Note: No user ID or password information passes to the integration server from the EnterpriseOne Event listener. Secured communication from the EnterpriseOne Event listener to the adapter is not implemented.

Example: **Sending Requests to J.D. Edwards EnterpriseOne**

The following is a sample request sent to J.D. Edwards EnterpriseOne.

```
<jdeRequest environment="DV7333" user="JDE" type="trans"
sessionidle="300" session="" pwd="JDE">

        <transaction type="JDESOOUT" action="transactionInfo">
            <key>
              <column name="EdiUserId">islywm</column>
              <column name="EdiBatchNumber">100</column>
              <column name="EdiTransactionNumber">100100</column>
            </key>
        </transaction>
</jdeRequest>
```

Example: **Sending Responses From J.D. Edwards EnterpriseOne**

The following is a sample response from J.D. Edwards EnterpriseOne.

```
<jdeResponse type='trans' user='user' session='session1'
                                    environment='env'>

    <transaction type='JDESOOUT' action='transactionInfo'>
        <returnCode code='0'>XML Request OK</returnCode>
            <key>
                <column name='EdiUserId'></column>
                <column name='EdiBatchNumber'></column>
                <column name='EdiTransactNumber'></column>
            </key>
            <table name='F4201Z1' type='header'>
                <column name='EdiUserId'></column>
                <column name='EdiBatchNumber'></column>
            </table>
            <table name='F4211Z1' type='detail'>
                <column name='EdiUserId'></column>
                <column name='EdiBatchNumber'></column>
            </table>
            <table name='F49211Z1' type='additionalHeader'>
                <WARNING>No record found</WARNING>
            </table>
    </transaction>
</jdeResponse>
```

Using Web Services Policy-Based Security

Topics:

- iWay Business Services Policy-Based Security
- Configuring iWay Business Services Policy-Based Security

Servlet Application Explorer provides a security feature called iWay Business Services policy-based security. The following topics describe how this feature works and how to configure it.

iWay Business Services Policy-Based Security

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

Servlet Application Explorer controls the use of business services that use adapters with a feature called policy-based security. This feature enables an administrator to apply *policies* to iWay Business Services (iBS) to deny or permit their execution.

A *policy* is a set of privileges associated with the execution of a business service that can be applied to an existing or new iBS. When you assign specific rights or privileges inside a policy, you need not recreate privileges for every iBS that has security issues in common with other iWay Business Services. Instead, you can use one policy for many iWay Business Services.

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

The iBSE administrator creates an instance of a policy type, names it, associates individual users and/or groups (a collection of users), and then applies the policy to one or more business services.

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

When a policy is not applied, the default value for an iBS is to "grant all." For example, anyone can execute the iBS until the Resource Execution policy is associated to the iBS. At that time, only users granted execution permission, or those who do not belong to a group that was denied execution permissions, have access to the iBS.

Configuring iWay Business Services Policy-Based Security

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 Servlet Application Explorer. For more information, see *How to Create a User to Associate With a Policy* on page 4-3 or *How to Create a Group to Associate With a Policy* on page 4-5.

An execution policy governs who can execute the business service to which the policy is applied. For more information, see *How to Create an Execution Policy* on page 4-9.

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 iBSE and therefore, need not be applied to an individual business service. You need not create a policy, however, you must enable the Security Policy option in Servlet Application Explorer. For more information, see *How to Configure IP and Domain Restrictions* on page 4-13.

Procedure: How to Create a User to Associate With a Policy

To create a user to associate with a policy:

1. Open Servlet Application Explorer.

   The following image shows the window that opens and includes three tabs corresponding to iWay Adapters, iWay Events, and iWay Business Services. The iWay Business Services tab is active and displays a Welcome screen on the right. The image shows the iWay Business Services node expanded in the left pane.
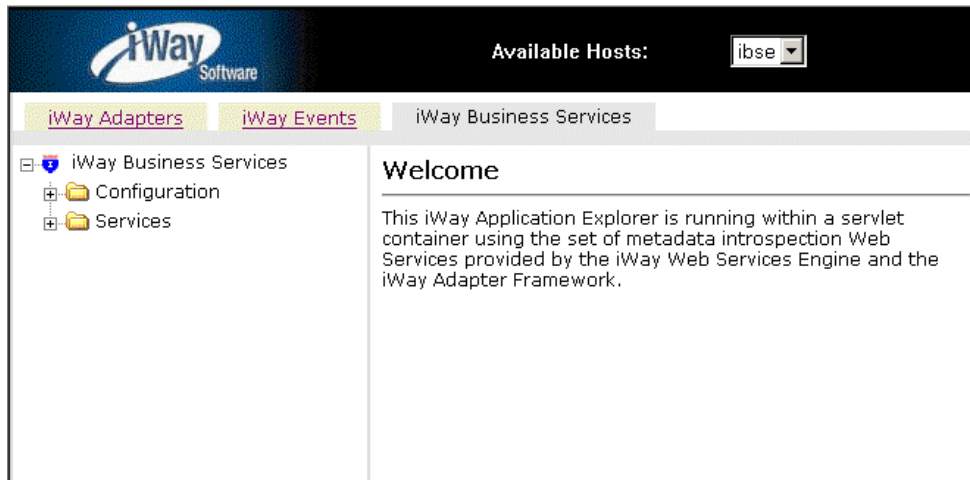
   a. Click the *iWay Business Services* tab.

   b. Expand the *Configuration* node.

      c. Expand the *Security* node.

      d. Expand the *Users and Groups* node.

      e. Select *Users*.

2. In the right pane, move the pointer over *Operations* and select *Add*.

The following image shows the Add a new user pane that opens and includes fields where you enter a user name, a password, and a description of the user. The pane includes a Help button, an OK button to instruct the system to accept inputs, and a Cancel button to escape from the pane.

Add a new user

Name:

Password:

Description:

Help   OK   Cancel

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

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

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

3. Click *OK*.

The following image opens and shows a new user added to the configuration. It includes a definition of a user and a user ID and description.

Operations ▶

Users

A user is an object that can be granted or denied permissions to run iWay Business Services. A user can be belong to one or more groups. Policies that specify particular rights can be associated with user.

User Id	Description
☐ bse1	
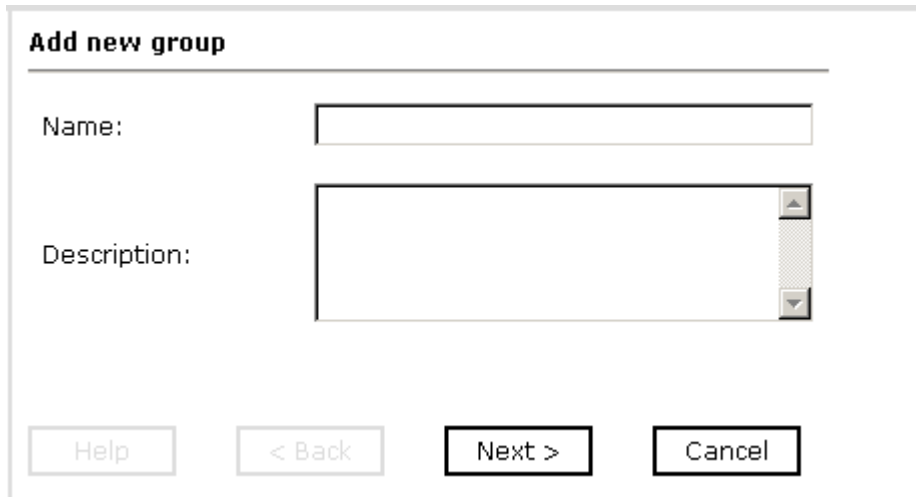
Procedure: How to Create a Group to Associate With a Policy

To create a group to associate with a policy:

1. Open Servlet Application Explorer.

  a. Click the *iWay Business Services* tab.

  b. Expand the *Configuration* node.

  c. Expand the *Security* node.

  d. Expand the *Users and Groups* node.

  e. Select *Groups*.

2. In the right pane, move the pointer over *Operations* and click *Add*.

The following image shows the Add new group pane that opens with fields where you enter a name and a description for the group. To continue after typing inputs, click the Next button. The pane also includes a Help button, a Back button to return to the previous screen, and a Cancel button to escape from the pane.

Add new group

Name:

Description:

| Help | < Back | Next > | Cancel |

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

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

3. Click *Next*.

The following image shows the Modify Group Membership pane where you can move users to or from a group using the arrow keys to move them between the Current and Available lists and then clicking the Finish button. The pane includes a Help button, a Back button to return to the previous screen, and a Cancel button to escape from the pane.

Modify Group Membership

Current		Available
	«	bse1
	‹	
	›	
	»	

Help	< Back	Finish	Cancel
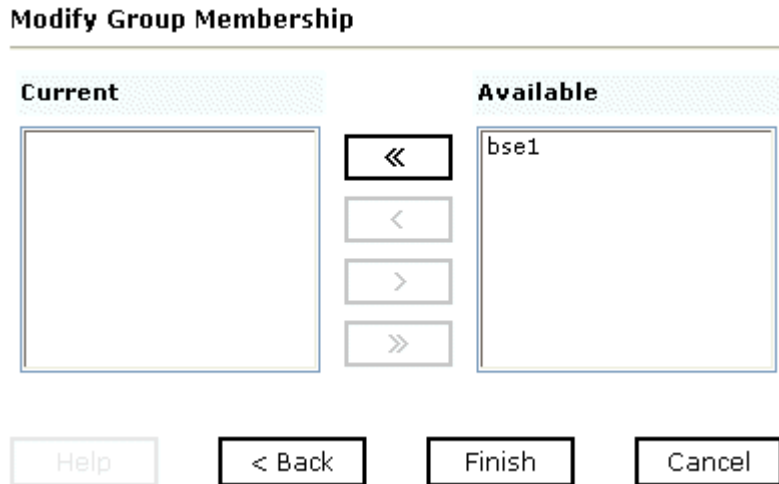
You can either highlight a single user in the list of available users and add it to the current list by clicking the left arrow, or you can click the double left arrow to add all users in the list of available users to the group.

4. After you select a minimum of one user, click *Finish*.

The new group is added.

The following image shows a pane with a new group added to the configuration. It includes a definition of a group and the group name and description.

Operations ▶

Groups

A group is an object that can be granted or denied permissions to run iWay Business Services. A group is used as a container for one or more users. Policies that specify particular rights can be associated with a group.

Group name	Description
☐ newgroup	

Procedure: How to Create an Execution Policy

To create an execution policy:

1. Open Servlet Application Explorer.

   a. Click the *iWay Business Services* tab.

   b. Expand the *Configuration* node.

   c. Select *Policies*.

   The following image shows the Policies pane on the right where you apply a policy. The Operations menu becomes available with three options, Build/Rebuild, Add, and Refresh.

2. Move the pointer over *Operations* and click *Add*.

The following image shows the Add a new policy pane that opens with fields for entering the name, type, and description of the policy. To continue, click the Next button. The pane includes a Help button, a Back button to return to the previous screen, and a Cancel button to escape from the pane.

Add a new policy

Name:

Type: Execution ▼

Description:

| Help | < Back | Next > | Cancel |

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

b. From the Type drop-down list, select *Execution*.

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

3. Click *Next*.

The following image shows the Modify policy targets pane that opens and includes a list of current and available targets and arrow buttons to move targets from one list to the other. The pane also includes a Help button, a Back button to return to the previous screen, a Next button to continue to the next screen, and a Cancel button to escape from the pane.

Modify policy targets

Current   **Available**

                                     user.ibse1
                                     group.ibse_group

Help   < Back   Next >   Cancel
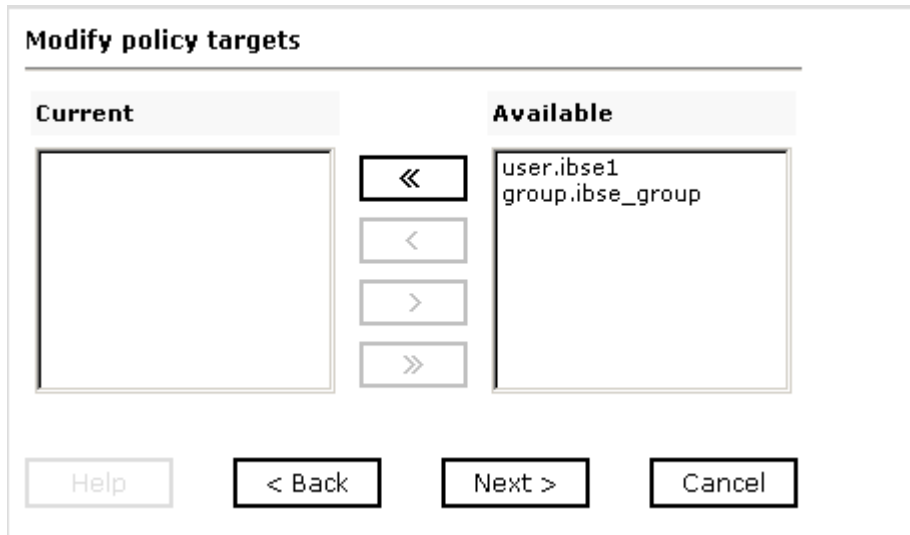
4. Select a minimum of one user or group from the Available pane.

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

5. Click *Next*.

The following image shows the Modify policy permissions pane that opens and includes drop-down lists where you can select to grant or deny permission to members and then click a button to finish. The pane also includes a Help button, a Back button to return to the previous screen, and a Cancel button to escape from the pane.

6. To assign whether users or groups may execute the iBSE, select *Grant* to permit execution or *Deny* to restrict execution from a Permission drop-down list.

7. Click *Finish*.

The following image shows the pane that summarizes your configuration. It includes a definition of policies and the name, type, and description of the policies.

Procedure: How to Configure IP and Domain Restrictions

To configure IP and domain restrictions:

1. Open Servlet Application Explorer.

    a. Select the *iWay Business Services* tab.

    b. Expand the *Configuration* node.

    c. Expand the *Security* node.

    d. Select *IP and Domain*.

2. In the right pane, move the pointer over *Operations* and click *Add*.

    The following image shows the Add a new IP/Domain pane that opens where you enter information for the IP/Domain in four fields. You must select a type of restriction from a drop-down list before you can enter information in the IP(Mask)/Domain field. The pane also includes a Help button, an OK button to instruct the system to accept inputs, and a Cancel button to escape from the pane.

a. From the Type drop-down list, select the type of restriction.

b. In the IP(Mask)/Domain field, type the IP or domain name using the following guidelines.

If you select Single (Computer) from the Type drop-down list, you must provide the IP address for that computer. If you only know the DNS name for the computer, click *DNS Lookup* to obtain the IP Address based on the DNS name.

If you select Group (of Computers), you must provide the IP address and subnet mask for the computer group.

If you select Domain, you must provide the domain name, for example, yahoo.com.

3. From the Access Control drop-down list, select *Grant* to permit access or *Deny* to restrict access for the IP addresses and domain names you are adding.

4. Click *OK*.

The following image shows the pane that opens and summarizes your configuration including the domain name, whether access is granted or denied, and a description (optional).

Operations ▶

IP and Domain

You can configure the iWay Business Services Engine to use policies that control access from a single IP address, a group of IP addresses, or all addresses within a particular domain.

IP(Mask) / Domain	Access	Description
☐ test	Deny	

CHAPTER 5

Management and Monitoring

Topics:

- Managing and Monitoring Services and Events Using iBSE

- Managing and Monitoring Services and Events Using the JCA Test Tool

- Setting Engine Log Levels

- Migrating Repositories

- Exporting or Importing Targets

- Retrieving or Updating Web Service Method Connection Information

- Starting or Stopping a Channel Programmatically

After you create services and events using Servlet Application Explorer, you can use managing and monitoring tools provided by the iWay Business Services Engine (iBSE) and the iWay Connector for JCA to measure the performance of your run-time environment. This section describes how to configure and use these features.

Managing and Monitoring Services and Events Using iBSE

iWay Business Services Engine (iBSE) provides a console to manage and monitor services and events currently in use and to display resource usage and invocation statistics. These indicators can help you adjust your environment for optimum efficiency.

The following monitoring levels are available for services:

- System
- Service
- Method

The following monitoring levels are available for events:

- System
- Channel
- Port

Procedure: **How to Configure Monitoring Settings**

To configure monitoring settings:

1. Ensure that your application server is started.

2. To access the monitoring console, enter the following URL in your Web browser:

   `http://localhost:port/ibse/IBSEConfig`

   where:

   `localhost`

   Is the machine where the application server is running.

   `port`

   Is the HTTP port for the application server.

   The following image shows the iBSE Settings window that opens. It lists property names and includes fields where you can enter values for each property. To configure system settings, the System pane contains drop-down lists for selecting language, encoding, the debug level, and the number of asynchronous processors. It also contains a field where you can enter a path to the adapters lib directory.

   To configure security settings, the Security pane contains fields for typing the Admin User name and the associated password and a check box for specifying policy.

To configure repository settings, the Repository pane contains a drop-down list for selecting the repository type, fields to type information for the repository URL, driver, user, and password, and a check box where you can enable repository pooling. In the upper and lower right of the window is a Save button. In the lower left of the window is an option to click to access more configuration settings.

3. Click *More configuration*.

Tip: To access the monitoring console directly, enter the following URL in your Web browser:

`http://localhost:port/ibse/IBSEStatus`

where:

localhost

   Is the machine where the application server is running.

port

   Is the HTTP port for the application server.

The following image shows the iBSE Monitoring Settings window that opens. It lists property names and includes a corresponding field where you can enter values for each property. The Monitoring pane contains a drop-down list for selecting the repository type, fields to type information for the repository URL, driver, user, and password, and a check box where you can enable repository pooling. The Auditing pane contains an option button to click to specify whether to store a message and a drop-down list where you can select the maximum messages to store. At the bottom of the window is a row of buttons that you can click to save your configuration, view events, or view services. The Save History button is inactive. After you enter properties and choose whether to save or view, you can click the Start Monitoring button.

iBSE Monitoring Settings:

Property Name	Property Value
Monitoring	
Repository Type	File System ▼
Repository Url	file://C:\Program Files\iWay55\bea
Repository Driver	
Repository User	
Repository Password	
Repository Pooling	☐
Auditing	
Store Message	○ yes ◉ no
Max Message Stored	10,000 ▼

Save Configuration	Save History	View Events	View Services

Start Monitoring

    a. In the Monitoring pane, from the Repository Type drop-down list, select the type of repository you are using.

    b. To connect to the database in the Repository Url field, type a JDBC URL.

    c. To connect to the database in the Repository Driver field, type a JDBC Class.

    d. To access the monitoring repository database, type a user ID and password.

    e. To enable pooling, click the *Repository Pooling* check box.

    f. In the Auditing pane, select *yes* if you want to store messages.

    This option is disabled by default.

    Note: You must start and then, stop monitoring to enable this option.

    g. Select the maximum number of messages you want to store.

    By default, 10,000 is selected.

    Note: Depending on your environment and the number of messages that are exchanged, storing a large number of messages may affect system performance. If you need more information about your system resources, consult your system administrator.

    h. Click *Save Configuration*.

4. Click *Start Monitoring*.

iBSE begins to monitor all services and events currently in use. If you selected the option to store messages, iBSE stores messages.

5. To stop monitoring, click *Stop Monitoring*.

Procedure: How to Monitor Services

To monitor services:

1. Ensure that your application server is started.

2. From the iBSE Monitoring Settings window, click *Start Monitoring*.

3. Click *View Services*.

The following image shows the System Level Summary (Service Statistics) window that opens. The Web Service Methods pane contains a drop-down list where you select a service. On the right, space is reserved for a drop-down list of methods that will appear. The Statistics pane contains a table with a summary of service statistics and two drop-down lists where you can select a successful or failed invocation to view more information about that service. At the bottom of the window is a home button to click to return to the iBSE Monitoring Settings window.

Service Statistics

Web Service Methods

Service   Method

all ▼

Statistics

Total Time	55 min
Total Request Count	1
Total Success Count	1
Total Error Count	0
Average Request Size	409.0 bytes
Average Response Size	665.0 bytes
Average Execution Time	656 ms
Last Execution Time	828 ms
Average Back End Time	530 ms
Last Back End Time	765 ms
Successful Invocations	select a correlation id ▼
Failed Invocations	select a correlation id ▼

< home

The system level summary provides services statistics at a system level.

The following table consists of two columns, one that lists the name of each statistic and the other that describes the corresponding service statistic.

Statistic	Description
Total Time	Total amount of time iBSE monitors services. The time starts after you click Start Monitoring in the iBSE Monitoring Settings window.
Total Request Count	Total number of services requests that were made during the monitoring session.
Total Success Count	Total number of successful service executions.
Total Error Count	Total number of errors that were encountered.
Average Request Size	Average size of an available service request.
Average Response Size	Average size of an available service response size.
Average Execution Time	Average execution time for a service.
Last Execution Time	Last execution time for a service.
Average Back End Time	Average back end time for a service.
Last Back End Time	Last back end time for a service.
Successful Invocations	A list of successful services arranged by correlation ID. To retrieve more information for a service, you can select the service from the drop-down list.
Failed Invocations	A list of failed services arranged by correlation ID. To retrieve more information for a service, you can select the service from the drop-down list.

4. Select a service from the drop-down list.

The following image shows the System Level Summary (Service Statistics) window that opens. The Web Service Methods pane contains a drop-down list on the left where you select a service and a drop-down list on the right where you select a service method. The Statistics pane contains a table with a summary of service statistics and two drop-down lists. To view more information about that service, you can select it from the Successful Invocations or Failed Invocations drop-down list. To suspend or resume a service, you can click a button in the lower right. To return to the iBSE Monitoring Settings window, you click the home button (also located in the lower right).

Service Statistics

Web Service Methods

Service	Method
B0100033 ▼	all methods ▼

Statistics

Total Time	1 hrs
Total Request Count	1
Total Success Count	1
Total Error Count	0
Average Request Size	409.0 bytes
Average Response Size	665.0 bytes
Average Execution Time	656 ms
Last Execution Time	656 ms
Average Back End Time	530 ms
Last Back End Time	530 ms
Successful Invocations	select a correlation id ▼
Failed Invocations	select a correlation id ▼

Suspend Service

< home

a. To stop a service at any time, click *Suspend Service*.

b. To restart the service, click *Resume Service*.

5. Select a method for the service from the Method drop-down list.

The following image shows the Method Level Summary (Service Statistics) window that opens. The Web Service Methods pane contains a drop-down list on the left where you select a service and a drop-down list on the right where you select a service method. The Statistics pane contains a table with a summary of service statistics and two drop-down lists. To view more information about that service, you can select it from the Successful Invocations or Failed Invocations drop-down list. To suspend or resume a service, you can click a button in the lower right. To return to the iBSE Monitoring Settings window, you click the home button (also located in the lower right).

Service Statistics

Web Service Methods

Service

B0100033

Method

GetEffectiveAddress

Statistics

Total Time	1 hrs
Total Request Count	1
Total Success Count	1
Total Error Count	0
Average Request Size	409.0 bytes
Average Response Size	665.0 bytes
Average Execution Time	656 ms
Last Execution Time	656 ms
Average Back End Time	530 ms
Last Back End Time	530 ms
Successful Invocations	select a correlation id
Failed Invocations	select a correlation id

Suspend Service

< home

6. For additional information about a successful service and its method, select a service based on its correlation ID from the Successful Invocation drop-down list.

The following image shows the Invocation Level Statistics window that opens. The Message Information pane contains a table of information about the message. The Client Information pane contains a table of information about the client. The Detail pane contains a table that shows the size of the request and response messages, with options to click to view the respective XML documents. In the lower right of the window is a home button to click to return to the iBSE Monitoring Settings window.

Invocation Statistics

Message Information

Received	2004-09-14 12:04:16.312
Sent to adapter	2004-09-14 12:04:16.406
Received from adapter	2004-09-14 12:04:16.936
Responded	2004-09-14 12:04:16.968
Status	SUCCESS

Client Information

Client IP	127.0.0.1
Client Host Name	127.0.0.1
User Name	

Detail

Message	Size
Request Message	409 bytes
Response Message	665 bytes

< home

7. To view the XML request document in your Web browser, click *Request Message*.

You can also view the XML response document for the service.

8. To return to the iBSE Monitoring Settings window, click *home*.

Procedure: How to Monitor Events

To monitor events:

1. Ensure that your application server is started.

2. In the iBSE Monitoring Settings window, click *Start Monitoring*.

3. Click *View Events.*

The following image shows the System Level Summary (Channel Statistics) window that opens. The Channels pane contains a drop-down list on the left where you select a channel. On the right, space is reserved for a drop-down list of ports that will appear. The Statistics pane contains a table with a summary of event statistics and two drop-down lists where you can select a successful or failed event to view more information about that event. In the lower right of the window is a home button to click to return to the iBSE Monitoring Settings window.
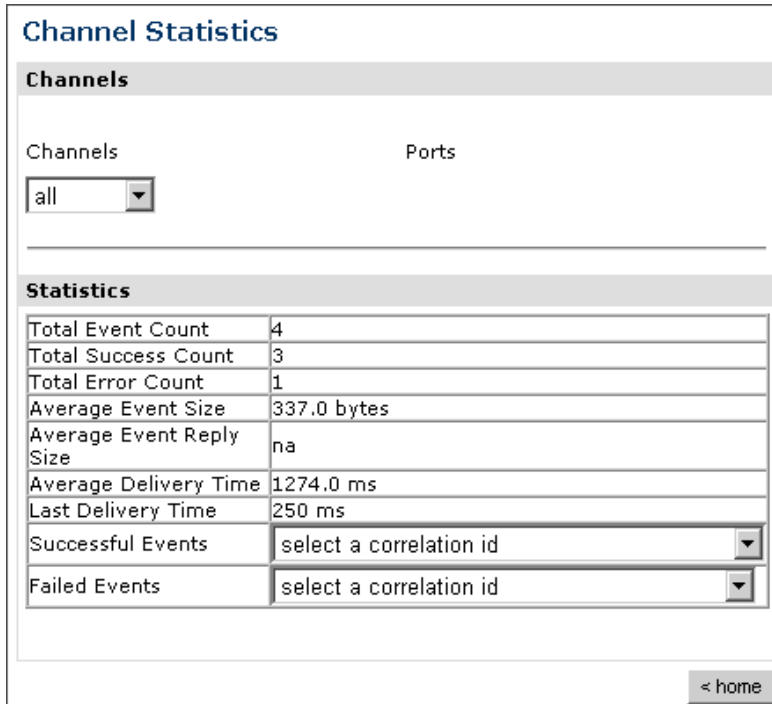
The system level summary provides event statistics at a system level.

The following table consists of two columns, one that lists the name of each statistic and the other that describes the corresponding event statistic.

Statistic	Description
Total Event Count	Total number of events.
Total Success Count	Total number of successful event executions.
Total Error Count	Total number of errors that were encountered.

Statistic	Description
Average Event Size	Average size of an available event request.
Average Event Reply Size	Average size of an available event response.
Average Delivery Time	Average delivery time for an event.
Last Delivery Time	Last delivery time for an event.
Successful Events	List of successful events arranged by correlation ID. To retrieve more information for an event, select the event from the drop-down list.
Failed Events	List of failed events arranged by correlation ID. To retrieve more information for an event, select the event from the drop-down list.

4. Select a channel from the drop-down list.

The following image shows the Channel Level Event Summary (Channel Statistics) window that opens. The Channels pane contains a drop-down list on the left where you select a channel and a drop-down list on the right where you select a port. The Statistics pane contains a table with a summary of event statistics and two drop-down lists where you can select a successful or failed event to view more information about that event. In the lower right of the window is a button to click to suspend or resume a channel and a home button to click to return to the iBSE Monitoring Settings window.

Channel Statistics

Channels

Channels	Ports
TestChan ▾	all ▾

Statistics

Total Event Count	3
Total Success Count	2
Total Error Count	1
Average Event Size	401.0 bytes
Average Event Reply Size	na
Average Delivery Time	1542.0 ms
Last Delivery Time	250 ms
Successful Events	select a correlation id ▾
Failed Events	select a correlation id ▾

[Suspend Channel] [Start Channel]

[< home]

a. To stop a channel at any time, click *Suspend Channel*.

b. To start the channel, click *Start Channel*.

5. From the Ports drop-down list, select a port for the channel.

The following image shows the Port Level Event Summary (Channel Statistics) window that opens. The Channels pane contains a drop-down list on the left where you select a channel and a drop-down list on the right where you select a port. The Statistics pane contains a table with a summary of event statistics and two drop-down lists where you can select a successful or failed event to view more information about that event. In the lower right of the window is a button to click to suspend or resume a channel and a home button to click to return to the iBSE Monitoring Settings window.

Channel Statistics

Channels

Channels   Ports
TestChan ▼   TestPort ▼

Statistics

Total Event Count	2
Total Success Count	2
Total Error Count	0
Average Event Size	446.0 bytes
Average Event Reply Size	na
Average Delivery Time	2189.0 ms
Last Delivery Time	na
Successful Events	select a correlation id ▼
Failed Events	select a correlation id ▼

[Suspend Channel] [Start Channel]

[< home]

6. For more information about a successful event and its port, select an event based on its correlation ID from the Successful Events drop-down list.

The following image shows the Event Level Statistics (Message Statistics) window that opens. The Message Information pane contains a table of information pertaining to the event message. The Messages pane contains a table that shows the size of the event and reply messages, with an option to view an XML document of the event message. In the lower right of the window is a home button to click to return to the iBSE Monitoring Settings window.

Message Statistics

Message Information

Received At	2004-09-14 12:18:20.842
Disposed At	• TestPort
Delivered At	2004-09-14 12:18:23.562

Messages

Detail	size
Event Message	446 bytes
Reply Message	na

`< home`

a. To view the XML event document in your Web browser, click *Event Message*.

b. To return to the iBSE Monitoring Settings window, click *home*.

Managing and Monitoring Services and Events Using the JCA Test Tool

The JCA Test Tool, which is also known as the JCA Installation Verification Program (IVP), provides a console to manage and monitor services and events currently in use and to display resource usage and invocation statistics. These indicators can help you adjust your environment for optimum efficiency.

Procedure: How to Manage and Monitor Services Using the JCA Test Tool

To manage and monitor services using the JCA Test Tool:

1. Open a Web browser to:

`http://localhost:port/iwjcaivp`

where:

localhost

Is the name of the machine where your application server is running.

port

Is the HTTP port for the application server, for example:

`http://localhost:7001/iwjcaivp`

The following image shows the JCA Test Tool page that opens. The page contains a description of the function of the tool and configuration information, including options to change your connection settings. It also provides options for viewing service or event adapters.

```
This JSP application is used to test the functionality of the J2EE-CA connector.
There are several types of adapters available thru this J2EE-CA connector.

Configuration

   • Running in MANAGED mode.
   • iway.jndi :eis/IWAFConnectionFactory:
   • Refresh Connection Factory after redeployment - Destroy Connection Factory for redeployment

Adapters

   • Service adapters
   • Event adapters
```

The JCA Test Tool runs in managed mode by default.

2. Perform the following steps to monitor the latest service adapter configuration.

Note: You must perform these steps for every new adapter target that is created using a JCA implementation of Application Explorer. In addition, you also must perform these steps for every new JCA configuration that is created using Application Explorer.

   a. Click *Destroy Connection Factory for redeployment*.

   b. Redeploy the JCA connector.

   c. In the JCA Test Tool, click *Refresh Connection Factory after redeployment*.

3. Click *Service adapters*.

The following image shows the Service Adapters page that opens. The page provides a live list of available service adapters and a list of targets configured for a specific adapter. In the lower right is a Back button to click to return to the previous page.
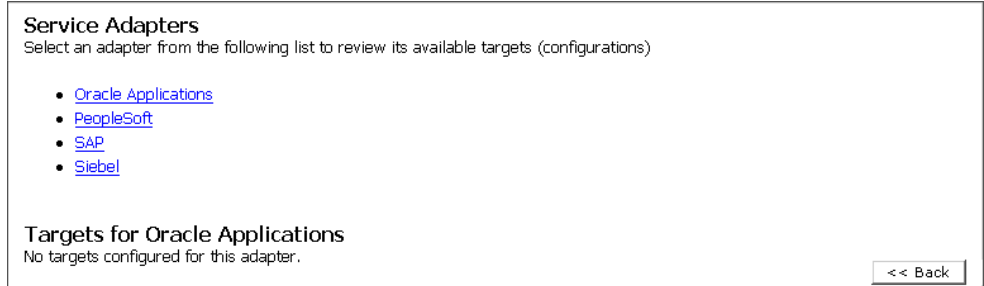
4. Select a service adapter to monitor.

The following image shows the page that opens. The left side provides a live list of available service adapters and a list of any targets configured for a specific adapter. The upper right side shows statistics for a selected target. The middle right has a User field and a Password field. The lower right contains a box where you type or paste an input document. Below the input box is a Send button to click to send a request for a test service and a Reset button to click to reset the fields. In the lower right is a Back button to click to return to the previous page.

a. Click the desired target for your service adapter.

b. In the Request area, enter a user name and password.

c. In the Input Doc area, enter a request document that was created from the request schema for your service.

5. Click *Send*.

The following image shows the updated statistics that appear for your service if the request is successful. The statistics include the total number of requests, successes, and errors and the average and last execution time in milliseconds.

```
TotalRequestCount      : 0
TotalSuccessCount      : 0
TotalErrorCount        : 0
AverageExcecutionTime  : 0 msec.
LastExcecutionTime     : 0 msec.
```

Procedure: How to Manage and Monitor Events Using the JCA Test Tool

To manage and monitor events using the JCA Test Tool:

1. Open a Web browser to:

`http://localhost:port/iwjcaivp`

where:

`localhost`

Is the name of the machine where your application server is running.

`port`

Is the HTTP port for the application server, for example:

`http://localhost:7001/iwjcaivp`

The following image shows the JCA Test Tool page that opens. The page contains a description of the function of the tool and configuration information, including options to change your connection settings. It also provides options for viewing service or event adapters.

> This JSP application is used to test the functionality of the J2EE-CA connector.
> There are several types of adapters available thru this J2EE-CA connector.
>
> **Configuration**
>
> - Running in MANAGED mode.
> - iway.jndi :eis/IWAFConnectionFactory:
> - Refresh Connection Factory after redeployment - Destroy Connection Factory for redeployment
>
> **Adapters**
>
> - Service adapters
> - Event adapters

The JCA Test Tool runs in managed mode by default.

2. Perform the following steps to monitor the latest event adapter configuration.

Note: You must perform these steps for every new adapter target that is created using a JCA implementation of Application Explorer. In addition, you must also perform these steps for every new JCA configuration that is created using Application Explorer.

a. Click *Destroy Connection Factory for redeployment*.

b. Redeploy the JCA connector.

c. In the JCA Test Tool, click *Refresh Connection Factory after redeployment*.

3. Click *Event adapters*.

The Event Adapters page opens.

4. Select the event adapter to monitor.

5. Click the desired channel for your event adapter.

6. Click *start*.

The following image shows the updated statistics for your channel and the port. The statistics include the total number of requests, successes, and errors and the average and last execution time in milliseconds. There are options to click in the upper right of the page to start or refresh the channel.

```
Current channel Statistics
Commands: start refresh
Active: false
TotalRequestCount        : 0
TotalSuccessCount        : 0
TotalErrorCount          : 0
AverageExcecutionTime : 0 msec.
LastExcecutionTime       : 0 msec.

Statistics for port 'fileIN'
TotalRequestCount        : 0
TotalSuccessCount        : 0
TotalErrorCount          : 0
AverageExcecutionTime : 0 msec.
LastExcecutionTime       : 0 msec.
```

Setting Engine Log Levels

The following section describes how to set engine log levels for Servlet iBSE and JCA. For more information, see the *iWay Installation and Configuration* documentation.

Procedure: How to Enable Tracing for Servlet iBSE

To enable tracing for Servlet iBSE:

1. Open the Servlet iBSE configuration page at:

    `http://`*`localhost:port`*`/ibse/IBSEConfig`

    where:

    `localhost`

    Is the name of the machine where your application server is running.

    `port`

    Is the HTTP port for the application server, for example:

    `http://localhost:7001/ibse/IBSEConfig`

2. In the System pane, from the Debug drop-down list, select the level of tracing.

3. Click *Save*.

Tracing information is written to the ibselogs directory where your application server accesses or has expanded Servlet iBSE.

Procedure: How to Enable Tracing for JCA

To enable tracing for JCA:

1. Open the extracted ra.xml file in a text editor.

2. Locate and change the following setting:

    LogLevel. This setting can be set to DEBUG, INFO, or ERROR.

    ```
    <context-param>
    <config-property>
        <config-property-name>LogLevel</config-property-name>
        <config-property-type>java.lang.String</config-property-type>
        <config-property-value></config-property-value>
    </config-property>
    ```

    For example:

    `<config-property-value>DEBUG</config-property-value>`

    A directory in the configuration directory contains the logs.

    a. Review the logs generated by your application server.

    b. Leave the remainder of the previous file unchanged.

3. Save the file and exit the editor.

4. Redeploy the connector.

Migrating Repositories

During design time, a 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. For more information on configuring repositories, see the *iWay Installation and Configuration* documentation.

The information in the repository also is referenced at run time. For management purposes, you can migrate iBSE and JCA repositories to new destinations without affecting your existing configuration. For example, you may want to migrate a repository from a development environment to a production environment.

File Repositories

If you want to migrate a File repository to another destination, copy the ibserepo.xml file from the following path:

drive:\Program Files\iWay55\ibse\ibserepo.xml

where:

drive

Is the location of your iWay 5.5 installation.

You can place the ibserepo.xml file in a new location that is a root directory of the iBSE Web application, for example:

drive:\ProductionConfig\ibse\ibserepo.xml

iBSE Repositories

The following topic describes how to migrate an iBSE repository that is configured for Oracle. You can follow the same procedure if you want to migrate an iBSE repository that is configured for Microsoft SQL Server 2000, Sybase, or DB2. However, when you are configuring a new environment, you must execute the script that creates the repository tables for your database. In addition, verify that all required files and drivers for your database are in the class path. For more information on configuring repositories, see the *iWay Installation and Configuration* documentation.

Note: The following procedure allows you to migrate only Web services. If migrating event handling information is one of your requirements, you must migrate at the database level. For more information, see *Migrating Event Handling Configurations* on page 5-27.

Procedure: How to Migrate an iBSE Repository Configured for Oracle

To migrate an iBSE repository that is configured for Oracle:

1. Copy the iBSE configuration service URL, for example:

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

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

   The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.
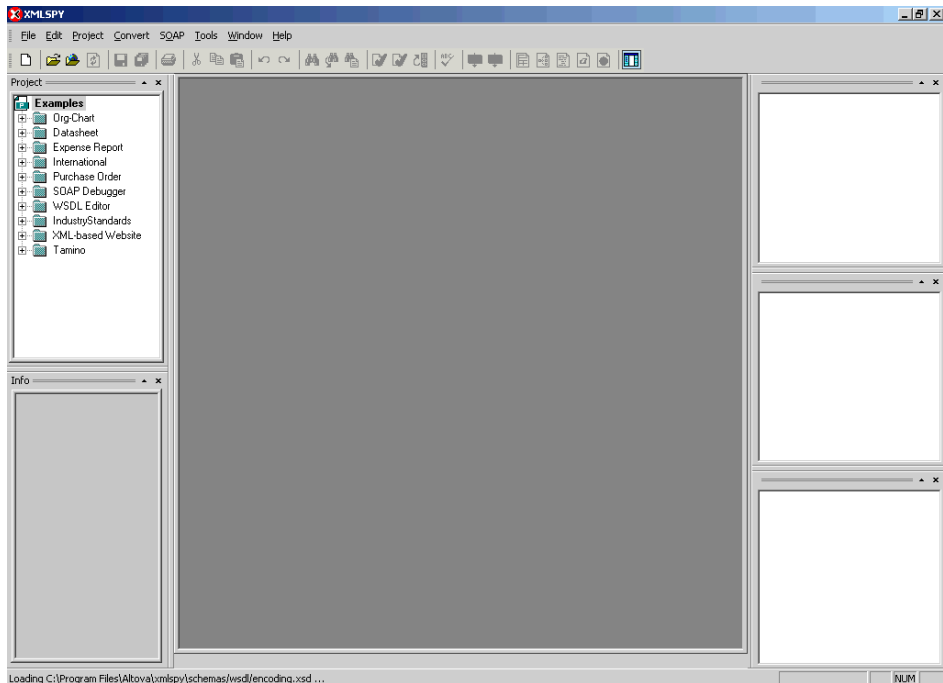
3. From the SOAP menu, select *Create new SOAP request*.

   The following image shows the WSDL file location dialog box that opens, where you enter a local path or URL. The dialog includes Browse, Window, OK, and Cancel buttons.
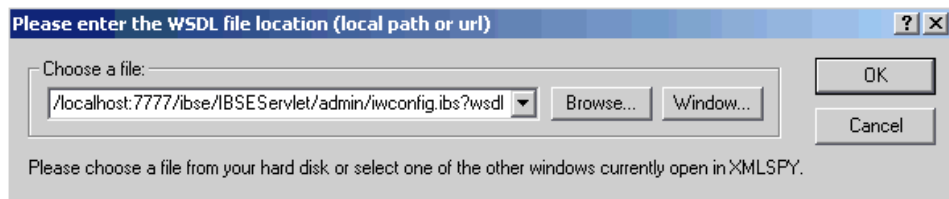
4. In the Choose a file field, paste the iBSE configuration service URL.

5. Click *OK*.

The following image shows the soap operation name dialog box that opens with a list of available control methods. You can select from the list and click OK or to escape from the dialog box, you can click Cancel.
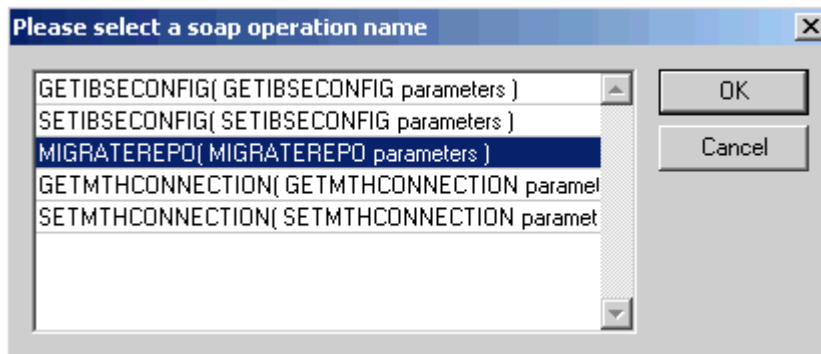
6. Select the *MIGRATEREPO(MIGRATEREPO parameters)* control method and click *OK*.

   The following image shows a portion of the window that opens with the structure of the SOAP envelope. It includes information about location and schemas.
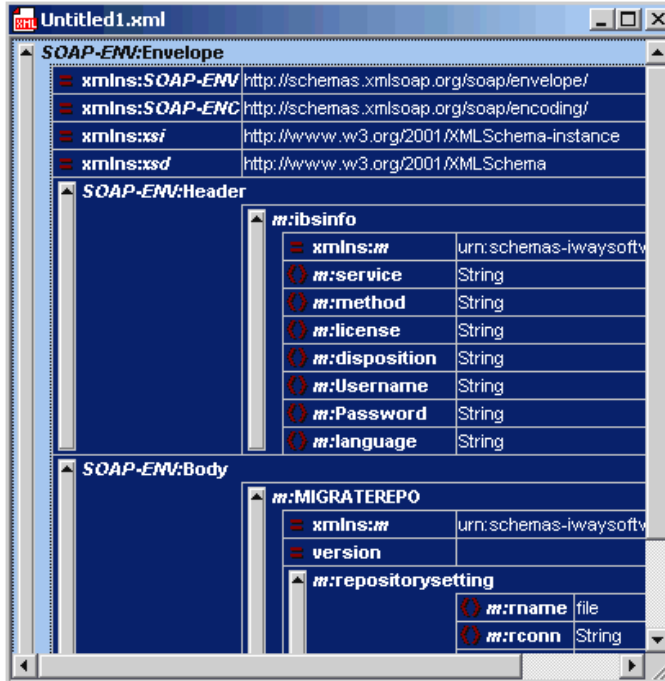
7. Locate the *Text view* icon in the tool bar.

   In the following image, the pointer points to the Text view icon.

8. To display the structure of the SOAP envelope as text, click the *Text view* icon.

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

9. Locate the following section:

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

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

For example, 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.

Note: This is an optional tag. If you do not specify a value, the default Oracle JDBC driver is used.

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 iBSE repository, enter the Web service name in the <m:servicename> tag, for example:

```
<m:servicename>Service1</m:servicename>
```

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

```
<m:servicename>Service1</m:servicename>
<m:servicename>Service2</m:servicename>
```

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

11. From the SOAP menu, select *Send request to server*.

Your iBSE repository and the Web services you specified migrate to the new Oracle repository URL that you specified.

JCA Repositories

The following procedure describes how to migrate a JCA repository. For more information on configuring JCA repositories, see the *iWay Installation and Configuration* documentation.

Procedure: How to Migrate a JCA Repository

To migrate a JCA repository:

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

   `C:\Program Files\iway55\config\base`

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

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

   Your JCA repository migrates to the new JCA configuration directory.

Migrating Event Handling Configurations

This topic describes how to migrate your iBSE repositories at a database level for Microsoft SQL Server 2000, Oracle, Sybase, or DB2. You can use this information to migrate event handling information, for example, port or channel configurations.

Procedure How to Migrate a Microsoft SQL Server 2000 Repository

To migrate a Microsoft SQL Server 2000 repository:

1. Open a command prompt and navigate to the iWay setup directory. The default location on Windows is:

    `C:\Program Files\iWay55\etc\setup`

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

    `iwse.sql`

    You can use iwse.sql to create the database tables that are used by iBSE. For example, the following image shows the tree in the left pane and tables in the right pane. The tables are listed by name in one column with corresponding columns for information about owner, type, and the date the table was created.

    For more information on configuring the Microsoft SQL Server 2000 repository, see the *iWay Installation and Configuration* documentation.

2. To migrate the tables that were created by the iwse.sql script for iBSE, use your Microsoft SQL Server 2000 database tool set. For more information, consult your database administrator.

Procedure   How to Migrate an Oracle Repository

To migrate an Oracle repository:

1. Open a command prompt and navigate to the iWay setup directory. The default location on Windows is:

    `C:\Program Files\iWay55\etc\setup`

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

    For Oracle 8:

    `iwse.ora`

    For Oracle 9:

    `iwse.ora9`

2. To create the Oracle database tables that are used by iBSE, use the SQL script as shown in the example in the following image that shows a list of tables.

    For more information on configuring the Oracle repository, see the *iWay Installation and Configuration* documentation.

3. To migrate the tables that were created by the SQL script for iBSE, use your Oracle database tool set. For more information, consult your database administrator.

Procedure **How to Migrate a Sybase Repository**

To migrate a Sybase repository:
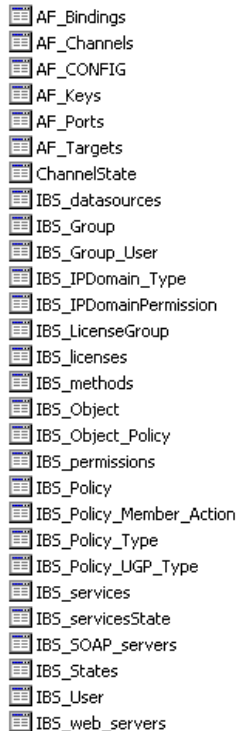
1. Open a command prompt and navigate to the iWay setup directory. The default location on Windows is:

   `C:\Program Files\iWay55\etc\setup`

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

   `sybase-iwse.sql`

2. To create the Sybase database tables that are used by iBSE, use the SQL script as shown in the example in the following image that shows a list of tables.

   AF_Bindings
   AF_Channels
   AF_CONFIG
   AF_Keys
   AF_Ports
   AF_Targets
   ChannelState
   IBS_datasources
   IBS_Group
   IBS_Group_User
   IBS_IPDomain_Type
   IBS_IPDomainPermission
   IBS_LicenseGroup
   IBS_licenses
   IBS_methods
   IBS_Object
   IBS_Object_Policy
   IBS_permissions
   IBS_Policy
   IBS_Policy_Member_Action
   IBS_Policy_Type
   IBS_Policy_UGP_Type
   IBS_services
   IBS_servicesState
   IBS_SOAP_servers
   IBS_States
   IBS_User
   IBS_web_servers

   For more information on configuring the Sybase repository, see the *iWay Installation and Configuration* documentation.

3. To migrate the tables that were created by the SQL script for iBSE, use your Sybase database tool set. For more information, consult your database administrator.

Procedure   How to Migrate a DB2 Repository

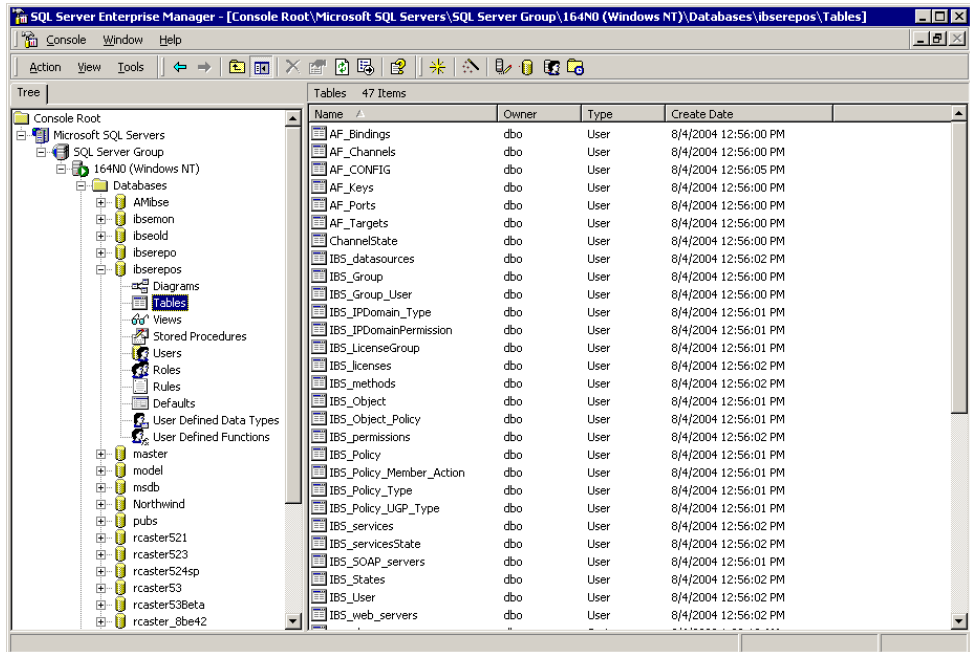To migrate a DB2 repository:
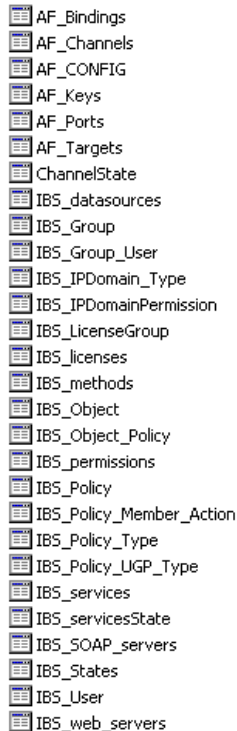
1. Open a command prompt and navigate to the iWay setup directory. The default location on Windows is:

   `C:\Program Files\iWay55\etc\setup`

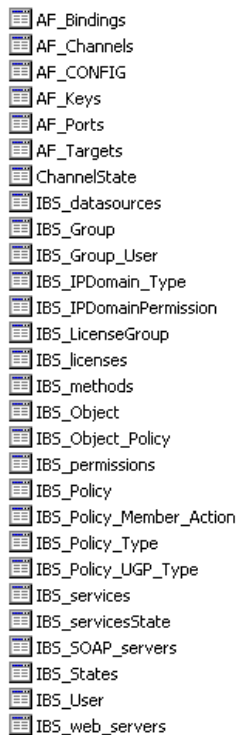   This directory contains SQL to create the repository tables in the following file:

   `db2-iwse.sql`

2. To create the DB2 database tables that are used by iBSE, use the SQL script as shown in the example in the following image that shows a list of tables.

   AF_Bindings
   AF_Channels
   AF_CONFIG
   AF_Keys
   AF_Ports
   AF_Targets
   ChannelState
   IBS_datasources
   IBS_Group
   IBS_Group_User
   IBS_IPDomain_Type
   IBS_IPDomainPermission
   IBS_LicenseGroup
   IBS_licenses
   IBS_methods
   IBS_Object
   IBS_Object_Policy
   IBS_permissions
   IBS_Policy
   IBS_Policy_Member_Action
   IBS_Policy_Type
   IBS_Policy_UGP_Type
   IBS_services
   IBS_servicesState
   IBS_SOAP_servers
   IBS_States
   IBS_User
   IBS_web_servers

   For more information on configuring the DB2 repository, see the *iWay Installation and Configuration* documentation.

   You can migrate the tables that were created by the SQL script for iBSE using your DB2 database toolset. For more information, consult your database administrator.

Exporting or Importing Targets

After you migrate your repository, you can export or import targets with their connection information and persistent data between repositories.

Procedure: How to Export a Target

To export a target:

1. Copy the iBSE administrative services for Application Explorer URL, for example:

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

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

    The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.
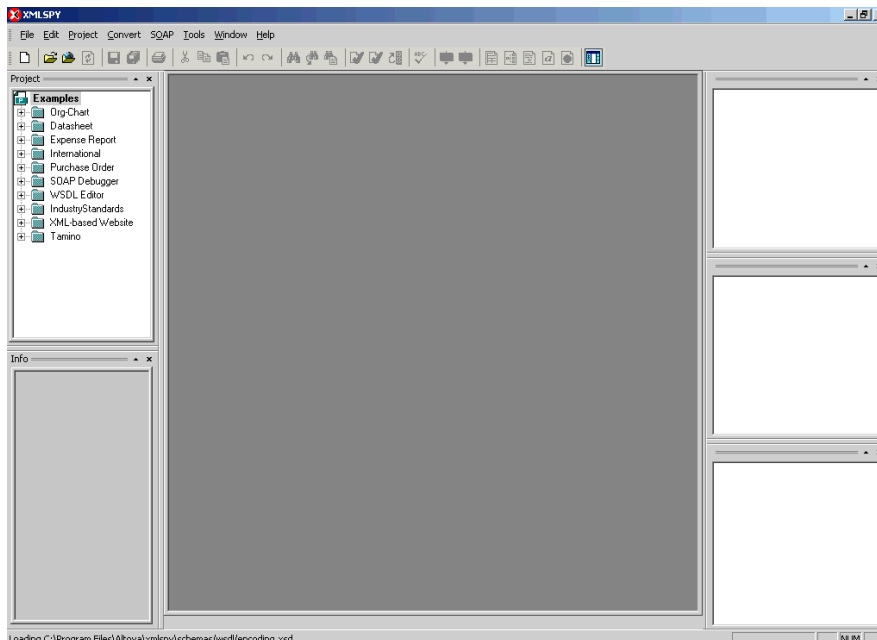
3. From the SOAP menu, select *Create new SOAP request*.

    The WSDL file location dialog box opens.

4. In the Choose a file field, paste the iBSE administrative services for Application Explorer URL.

5. Click *OK*.

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

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

A window opens that shows the structure of the SOAP envelope.

7. Locate the *Text view* icon in the tool bar.

In the following image, the pointer points to the Text view icon.

8. To display the structure of the SOAP envelope as text, click the *Text view* icon.

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

9. Locate the following section:

```
<m:EXPORTTARGET
xmlns:m="urn:schemas-iwaysoftware-com:dec2002:iwse:af">
<m:target>String</m:target>
<m:name>String</m:name>
</m:EXPORTTARGET>
```

a. For the <m:target> tag, replace the String placeholder with the EIS target system name as it appears in Application Explorer and verify whether this value is case sensitive.

b. For the <m:name> tag, replace the String placeholder with the name of the target you want to export.

10. From the SOAP menu, select *Send request to server*.

A response is returned that contains the <m: exporttime> and <m: contents> elements. You must use these elements when importing your target.

Procedure: How to Import a Target

To import a target:

1. Copy the iBSE administrative services for Application Explorer URL, for example:

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

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

The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.
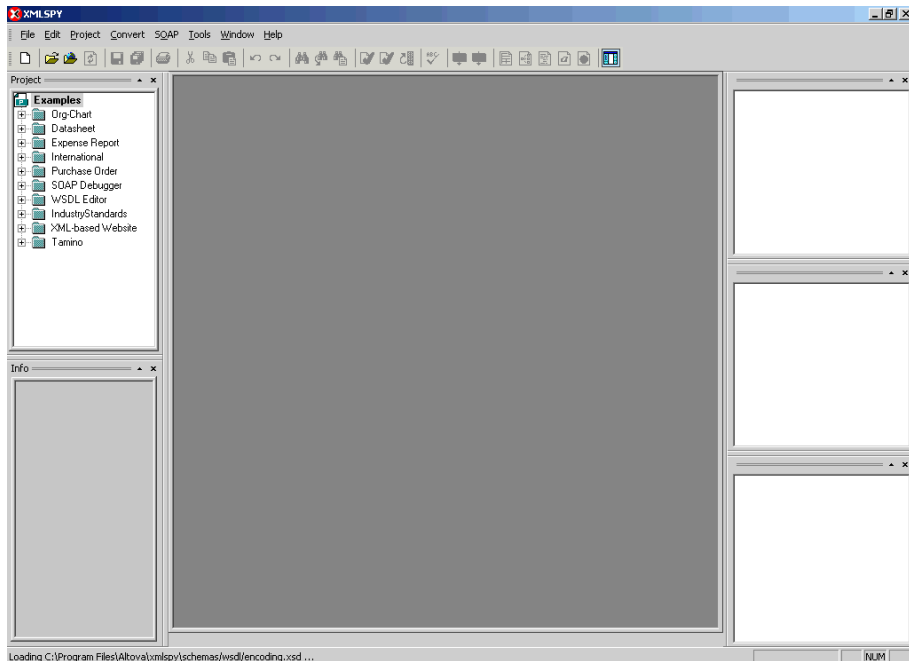
3. From the SOAP menu, select *Create new SOAP request*.

The WSDL file location dialog box opens.

4. In the Choose a file field, paste the iBSE administrative services for Application Explorer URL and click *OK*.

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

5. Select the *IMPORTTARGET(IMPORTTARGET parameters)* control method and click *OK*.

A window opens, which shows the structure of the SOAP envelope.

6. Locate the *Text view* icon in the toolbar.

In the following image, the pointer points to the Text view icon.

7. 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.

8. Locate the following section:

```
<m:IMPORTTARGET
xmlns:m="urn:schemas-iwaysoftware-com:dec2002:iwse:af">
<m:targetinstance>
<m:target>String</m:target>
<m:name>String</m:name>
<m:description>String</m:description>
<m:repositoryid>String</m:repositoryid>
<m:exporttime>2001-12-17T09:30:47-05:00</m:exporttime>
<m:contents>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</m:contents>
</m:targetinstance>
</m:IMPORTTARGET>
```

a. For the <m:target> tag, replace the String placeholder with the EIS target system name.

b. For the <m:name> tag, replace the String placeholder with the new name of the target you want to import.

c. For the <m:description> tag, replace the String placeholder with a description of the target.

d. For the <m:repositoryid> tag, copy and paste the contents of the <m:repositoryid> tag that was returned when you exported your target.

e. For the <m: exporttime> tag, copy and paste the contents of the <m: exporttime> tag that was returned when you exported your target.

f. For the <m: contents> tag, copy and paste the contents of the <m: contents> tag that was returned when you exported your target.

9. From the SOAP menu, select *Send request to server.*

Retrieving or Updating Web Service Method Connection Information

After you migrate your repository, you can retrieve or update connection information for your Web service methods.

Procedure: How to Retrieve Web Service Method Connection Information

To retrieve Web service method connection information:

1. Copy the iBSE configuration service URL, for example:

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

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

    The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.

3. From the SOAP menu, select *Create new SOAP request*.

    The WSDL file location dialog box opens.

4. In the Choose a file field, paste the iBSE configuration service URL, and click *OK*.

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

5. Select the *GETMTHCONNECTION(GETMTHCONNECTION parameters)* control method and click *OK*.

A window opens, which shows the structure of the SOAP envelope.

6. Locate the *Text view* icon in the toolbar.

In the following image, the pointer points to the Text view icon.

7. 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.

8. Locate the following section:

```
<m:GETMTHCONNECTION
xmlns:m="urn:schemas-iwaysoftware-com:jul2003:ibse:config">
<m:servicename>String</m:servicename>
<m:methodname>String</m:methodname>
</m:GETMTHCONNECTION>
```

a. For the <m:servicename> tag, replace the String placeholder with the name of the Web service.

b. For the <m:methodname> tag, replace the String placeholder with name of the Web service method.

9. From the SOAP menu, select *Send request to server*.

A response is returned that contains the <m: descriptor> element. You must use this element when updating your Web service method.

Procedure: How to Update Web Service Method Connection Information

To update Web service method connection information:

1. Copy the iBSE configuration service URL, for example:

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

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

   The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.

3. From the SOAP menu, select *Create new SOAP request*.

   The WSDL file location dialog box opens.

4. In the Choose a file field, paste the iBSE configuration service URL, and click *OK*.

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

5. Select the *SETMTHCONNECTION(SETMTHCONNECTION parameters)* control method and click *OK*.

   A window opens that shows the structure of the SOAP envelope.

6. Locate the *Text view* icon in the toolbar.

In the following image, the pointer points to the Text view icon.

7. 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.

8. Locate the following section:

```
<m:SETMTHCONNECTION
xmlns:m="urn:schemas-iwaysoftware-com:jul2003:ibse:config">
<m:servicename>String</m:servicename>
<m:methodname>String</m:methodname>
<m:descriptor format="" channel="">
     <m:option title="">
        <m:group title="">
            <m:param/>
        </m:group>
     </m:option>
</m:descriptor>
</m:SETMTHCONNECTION>
```

a. For the <m:servicename> tag, replace the String placeholder with the name of the Web service.

b. For the <m:methodname> tag, replace the String placeholder with the name of the Web service method.

c. For the <m: descriptor> tag, copy and paste the contents of the <m: descriptor> tag that was returned when you retrieved Web Service method connection information.

9. Modify the contents of the <m: descriptor> tag to change the existing Web Service method connection information.

10. From the SOAP menu, select *Send request to server*.

Starting or Stopping a Channel Programmatically

The following topic describes how to start or stop a channel programmatically.

Procedure: How to Start a Channel Programmatically

To start a channel programmatically:

1. Copy the iBSE control event URL, for example:

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

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

   The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.
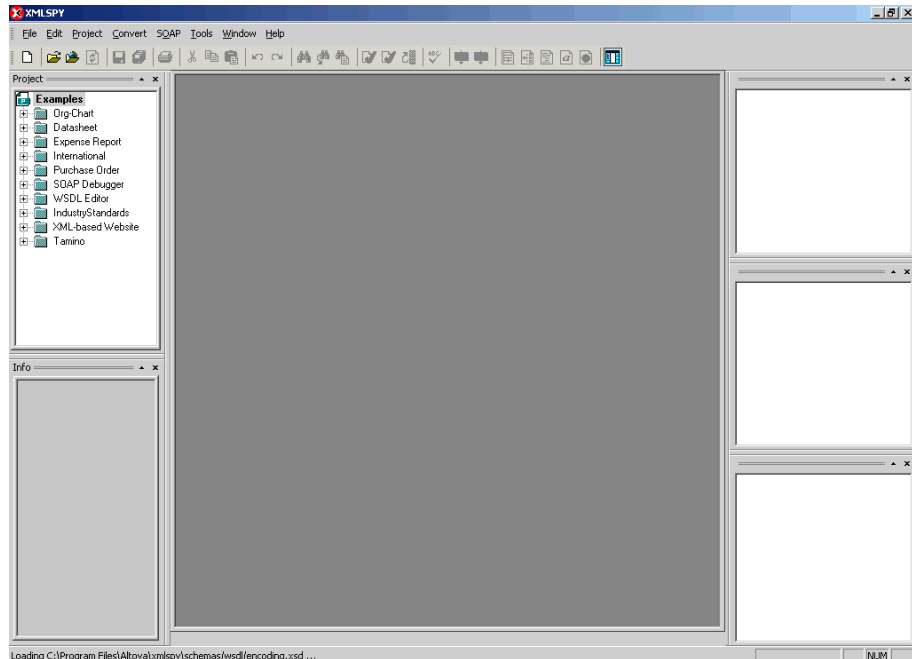
3. From the SOAP menu, select *Create new SOAP request*.

   The WSDL file location dialog box opens.

4. In the Choose a file field, paste the iBSE control event URL, and click *OK*.

The following image shows the soap operation name dialog box that opens with a list of available control methods. You can select one and click OK or to escape from the dialog box, you can click Cancel.

5. Select the *STARTCHANNEL(STARTCHANNEL parameters)* control method and click *OK*.

A window opens, which shows the structure of the SOAP envelope.

6. Locate the *Text view* icon in the toolbar.

In the following image, the pointer points to the Text view icon.

7. 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.

8. Locate the following section:

```
<SOAP-ENV:Body>
  <m:STARTCHANNEL
    xmlns:m="urn:schemas-iwaysoftware-com:dec2002:iwse:event">
        <m:channel>String</m:channel>
  </m:STARTCHANNEL>
</SOAP-ENV:Body>
```

9. For the <m:channel> tag, replace the String placeholder with the name of the Channel you want to start.

10. From the SOAP menu, select *Send request to server*.

Procedure: How to Stop a Channel Programmatically

To stop a channel programmatically:

1. Copy the iBSE control event URL, for example:

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

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

   The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.
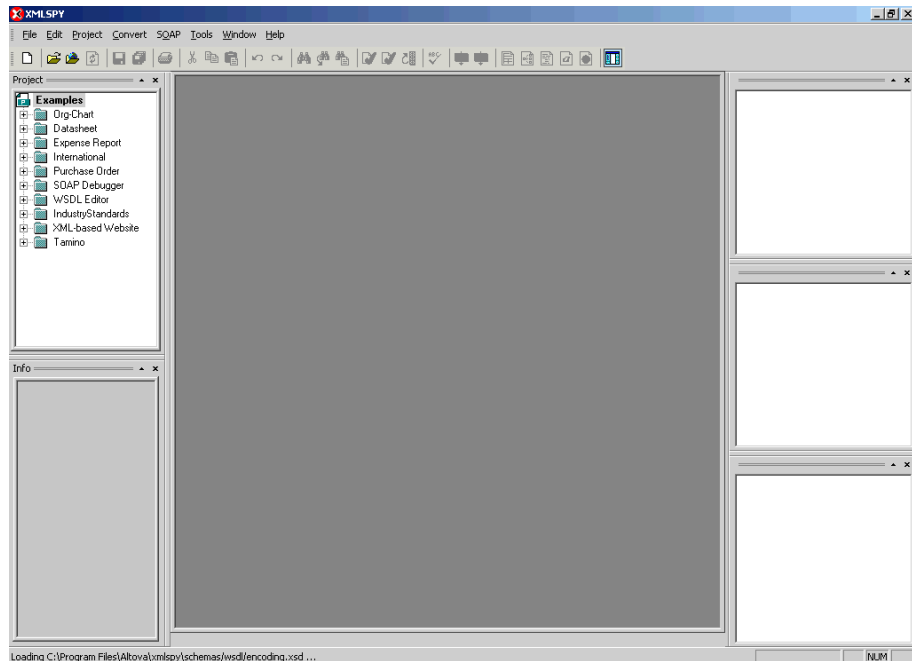
3. From the SOAP menu, select *Create new SOAP request*.

   The WSDL file location dialog box opens.

4. In the Choose a file field, paste the iBSE control event URL, and click *OK*.

The following image shows the soap operation name dialog box that opens with a list of available control methods. You can select one and click OK or to escape from the dialog box, you can click Cancel.

5. Select the *STOPCHANNEL(STOPCHANNEL parameters)* control method and click *OK*.

A window opens, which shows the structure of the SOAP envelope.

6. Locate the *Text view* icon in the toolbar.

In the following image, the pointer points to the Text view icon.

7. 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.

8. Locate the following section:

```
<SOAP-ENV:Body>
  <m:STOPCHANNEL
     xmlns:m="urn:schemas-iwaysoftware-com:dec2002:iwse:event">
         <m:channel>String</m:channel>
  </m:STOPCHANNEL>
</SOAP-ENV:Body>
```

9. For the <m:channel> tag, replace the String placeholder with the name of the Channel you want to stop.

10. From the SOAP menu, select *Send request to server*.

CHAPTER 6

Troubleshooting

Topics:

- Troubleshooting
- Error Messages in Application Explorer
- Error Messages in J.D. Edwards Enterprise One
- Error Messages in JCA
- Error Messages in iWay Business Services Engine

The following topics explain the limitations and workarounds when connecting to EnterpriseOne.

The adapter-specific errors listed in this section can occur when you are using the adapter with a JCA or with an iWay Business Services Engine (iBSE) configuration.

Troubleshooting

This topic provides troubleshooting information for J.D. Edwards, separated into the following categories:

- Application Explorer

- J.D. Edwards

- JCA

- iBSE

Error Messages in Application Explorer

The following table describes errors and corresponding solutions for Application Explorer.

Error	Solution
Cannot connect to the iWay Application System Adapter for J.D. Edwards EnterpriseOne from Application Explorer.	Ensure that: • J.D. Edwards is running. • The J.D. Edwards user ID and password are correct. • The port number is correct. • The custom Component Interface is properly installed.

The following error message appears:
 	
`java.lang.IllegalStateException:`
`java.lang.Exception: Error Logon`
`to J.D. Edwards OneWorld System`
	You provided invalid connection information for J.D. Edwards EnterpriseOne or the wrong JAR file is in the lib directory.

J.D. Edwards does not appear in the Application Explorer Adapter node list.	Ensure that the J.D. Edwards JAR files are added to the lib directory.

Error Messages in J.D. Edwards Enterprise One

The following table describes errors and corresponding causes and solutions for J.D. Edwards EnterpriseOne.

Error	Cause	Solution
Action code invalid.	In the Sales Order request, the Action code appears as "H," an invalid action code.	Use: • "I" for inquiry. • "C" for change. • "D" for delete. • "A" to add a new record.
Invalid address number.	The address number does not exist in the Address Book Master file (F0101).	Enter an address number using the Address Book Revisions program (PO1051). Ensure that the number entered is correct.
Record invalid	The record being processed either already exists for an ADD function or does not exist for an INQUIRY, CHANGE, or DELETE function.	If you are attempting to inquire, change, or delete a record you added previously, data base problems might exist in your production library. Contact your data processing department.
Item Branch record does not exist.	An Item Branch record (F4102) does not exist for this item in the Branch/Plant specified.	Correct the Branch or enter an Item Branch record for this item in Branch Plant Item Information (P41026).
&1 does not match any of the valid values.	The &1 does not match any of the valid values specified in the Data Dictionary for this field.	Enter a valid value.
Date out of range.	The Last Service Date and the Inspection Date must be within the range of the effective dates of the Service Contract.	Change the date to be greater than or equal to the beginning effective date and less than or equal to the ending effective date of the Service Contract.

Error Messages in JCA

The following table describes an error and a corresponding solution for JCA.

Error	Solution
In Application Explorer, the following error message appears when you attempt to connect to a JCA configuration: `Could not initialize JCA`	In the Details tab in the right pane, ensure that the directory specified in the Home field points to the correct directory, for example, C:\Program Files\iWay55.

Error Messages in iWay Business Services Engine

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

General Error Handling in iBSE

iWay Business Services Engine (iBSE) serves as both a SOAP gateway into the adapter framework and as the engine for some of the adapters. In both design time and execution time, various conditions can cause errors in iBSE when Web services that use adapters are running. Some of these conditions and resulting errors are exposed the same way, regardless of the specific adapter; others are exposed differently, based on the adapter being used. This topic explains what you can expect when you encounter some of the more common error conditions on an adapter-specific basis.

Usually, the SOAP gateway (agent) inside iBSE passes a SOAP request message to the adapter required for the Web service. If an error occurs, the way 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, when the SOAP agent inside iBSE 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 iBSE receives an invalid SOAP request:

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

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

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

Adapter-Specific Error Handling

When an adapter raises an exception during execution, the SOAP agent in iBSE 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. Because adapters use the target system interfaces and APIs, whether or not an exception is raised depends on how the target systems interface or API treats the error condition. If a SOAP request message is passed to an adapter by the SOAP agent in iBSE, and that request is invalid based on the WSDL for that service, the adapter may raise an exception yielding a SOAP fault.

Although 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.

Example: **iWay Application System Adapter for J.D. Edwards EnterpriseOne Invalid SOAP Request**

When the adapter receives a SOAP request message that does not conform to the WSDL for the Web service being executed, 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:CARRIERResponse xmlns:m="urn:schemas-iwaysoftware-com:iwse"
   xmlns="urn:schemas-iwaysoftware-com:iwse"
   cid="2A3CB42703EB20203F91951B89F3C5AF">
   <PS8>
      <error>Cannot find Component Interface {VARRIER}
(91,2)Initialization
    failed (90,7)Not Authorized (90,6)Failed to execute PSSession request
Cannot find Component Interface {VARRIER} (91,2)</error>
        </PS8>
      </m:CARRIERResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example: **Invalid SOAP Request**

When the adapter receives a SOAP request message that does not conform to the WSDL for the Web service being executed, the following SOAP response is generated.

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

APPENDIX A

Configuring EnterpriseOne for Outbound Transaction Processing

Topics:

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

EnterpriseOne enables you to specify outbound functionality for Master Business Functions (MBF).

This section describes how to enable outbound transaction processing in EnterpriseOne and how to modify the jde.ini file for XML support.

Specifying Outbound Functionality for a Business Function

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

Outbound Transaction Processing

To process outbound data, you use the:

- Data Export Control table

- Processing Log table

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

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

Procedure: How to Enable Outbound Transaction Processing

To enable outbound transaction processing:

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

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

2. From the shortcut menu, select *Prompt for Values*.

3. Click either the *Outbound* tab or the *Interop* tab.

4. Enter the transaction type.

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

   You are not required to set the *before* image function.

The Data Export Control Table and the Processing Log Table

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

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

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

Procedure: How to Use the Data Export Controls

To use the data export controls:

1. On the Work With Data Export Controls pane, click *Add*.

2. Type values in the Transaction Type and Order Type fields.

3. For each detail row, enter either a batch process name or version or a function name and the library.

4. To launch the vendor-specific object for an add or insert, type *1*.

5. For the update, delete, and inquiry actions, type *1*.

6. In the Launch Immediately column, type *1*.

7. Click *OK*.

Modifying the EnterpriseOne jde.ini File

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

Example: **Modifying a jde.ini File for XML Support**

The following is a sample of the modifications required to implement XML support on Windows NT. Add the following blocks of code:

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

[JDENET_KERNEL_DEF15]
krnlName=XML TRANSACTION KERNEL
dispatchDLLName=XMLTransactions.dll
dispatchDLLFunction=_XMLTransactionDispatch@28
maxNumberOfProcesses=1
numberOfAutoStartProcesses=1
```

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

APPENDIX B

Sample Files

Topics:

- Issuing a Single-Function Request

- Issuing a Multiple-Function Request

- Sample Sales Order Request

- Sample Sales Order Response

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

- Aggregate business function calls into a single object.

- Use the EnterpriseOne ThinNet API.

- Access both Z files and business functions.

This section provides examples of the jdeRequest and jdeResponse XML structures for executing business functions within EnterpriseOne.

Issuing a Single-Function Request

The following example, GetEffectiveAddress, is a single-function call to EnterpriseOne, 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.

Example: **Executing a Business Function With a Single-Function Call**

The following code is a sample GetEffectiveAddress jdeRequest.

```
<jdeRequest type="callmethod" user="JDE" pwd="JDE" environment="DV7333"
session="">

<callMethod name="GetEffectiveAddress" app="BSE" runOnError="no">
<params>
   <param name="mnAddressNumber">1001</param>
   <param name="jdDateBeginningEffective"></param>
   <param name="cEffectiveDateExistence10"></param>
   <param name="szAddressLine1"></param>
   <param name="szAddressLine2"></param>
   <param name="szAddressLine3"></param>
   <param name="szAddressLine4"></param>
   <param name="szZipCodePostal"></param>
   <param name="szCity"></param>
   <param name="szCountyAddress"></param>
   <param name="szState"></param>
   <param name="szCountry"></param>
   <param name="szUserid"></param>
   <param name="szProgramid"></param>
   <param name="jdDateupdated"></param>
   <param name="szWorkstationid"></param>
   <param name="mnTimelastupdated"></param>
   <param name="szNamealpha"></param>
</params>
<onError abort="yes"></onError>
</callMethod>
</jdeRequest>
```

The following code 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>
      <param name="szAddressLine3">                         </param>
      <param name="szAddressLine4">                         </param>
      <param name="szZipCodePostal">80237 </param>
      <param name="szCity">Denver          </param>
      <param name="szCountyAddress">      </param>
      <param name="szState">CO             </param>
      <param name="szCountry"/>
      <param name="szUserid"/>
      <param name="szProgramid"/>
      <param name="jdDateupdated"/>
      <param name="szWorkstationid"/>
      <param name="mnTimelastupdated">0   </param>
      <param name="szNamealpha">J.D. Edwards & Company         </param>
    </params>
  </callMethod>
</jdeResponse>
```

Issuing a Multiple-Function Request

The following example, GetEffectiveAddress, is a multiple-function call to EnterpriseOne, 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.

Example: Executing a Business Function With a Multiple-Function Call

The following code is a sample Purchase Order in the jdeRequest format. The XML contains return parameter specifications as well as 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>
   <onError abort='yes'>
   </onError>
   </callMethod>
   <callMethod app='XMLTest' name='F4311InitializeCaching'
      runOnError='no'>
   <params>
      <param name='cUseWorkFiles'>2</param>
   </params>
   </callMethod>
   <callMethod app='XMLTest' name='F4311FSBeginDoc' runOnError='no'
      returnNullData='yes'>
   <params>
      <param name='mnJobNumber' id='jobNumber'></param>
      <param name='szComputerID' idref='machineKey'></param>
      <param name='cHeaderActionCode'>A</param>
      <param name='cProcessEdits'>1</param>
      <param name='cUpdateOrWriteToWorkFile'>2</param>
      <param name='cRecordWrittenToWorkFile'>0</param>
      <param name='szOrderCOmpany' id='orderCompany'>00200</param>
      <param name='szOrderType'>OP</param>
      <param name='szOrderSuffix'>000</param>
      <param name='szBranchPlant'>M30</param>
      <param name='mnSupplierNumber'
         id='supplierNumber'>4343</param>
      <param name='mnShipToNumber'>0.0</param>
      <param name='jdOrderDate'>2000/03/02</param>
      <param name='cEvaluatedReceiptsFlag'>N</param>
      <param name='cCurrencyMode'>D</param>
```

```
    <param name='szTransactionCurrencyCode'>USD</param>
    <param name='mnCurrencyExchangeRate'>0.0</param>
    <param name='szOrderedPlacedBy'>SUBSTITUTE</param>
    <param name='szProgramID'>EP4310</param>
    <param name='szPurchaseOrderPrOptVersion'
        id='Version'>ZJDE0001</param>
    <param name='szUserID'>SUBSTITUTE</param>
    <param name='mnProcessID' id='processID'></param>
    <param name='mnTransactionID' id='transactionID'></param>
</params>
<onError abort='yes'>
<callMethod app='XMLTest' name='F4311ClearWorkFiles'
    runOnError='yes' returnNullData='yes'>
<params>
    <param name='szComputerID' idref='jobNumber'></param>
    <param name='mnJobNumber' idref='machineKey'></param>
    <param name='cClearHeaderFile'>1</param>
    <param name='cClearDetailFile'>1</param>
    <param name='mnLineNumber'>0</param>
    <param name='cUseWorkFiles'>2</param>
    <param name='mnProcessID' idref='processID'></param>
    <param name='mnTransactionID' idref='transactionID'></param>
</params>
</callMethod>
</onError>
</callMethod>
<!-- This is the first EditLine entry -->
<callMethod app='XMLTest' name='F4311EditLine' runOnError='yes'
    returnNullData='no'>
<params>
    <param name='mnJobNumber' idref='jobNumber'></param>
    <param name='szComputerID' idref='machineKey'></param>
    <param name='cDetailActionCode'>A</param>
    <param name='cProcessEdits'>1</param>
    <param name='cUpdateOrWriteWorkFile'>2</param>
    <param name='cCurrencyProcessingFlag'>Y</param>
    <param name='szPurchaseOrderPrOptVersion'
    idref='version'></param>
    <param name='szOrderCompany' idref='orderCompany'></param>
    <param name='szOrderType'>OP</param>
    <param name='szOrderSuffix'>000</param>
    <param name='szBranchPlant'>M30</param>
    <param name='mnSupplierNumber'
        idref='supplierNumber'></param>
    <param name='mnShipToNumber'>0.0</param>
    <param name='jdRequestedDate'>2000/03/02</param>
    <param name='jdTransactionDate'>2000/03/02</param>
    <param name='jdPromisedDate'>2000/03/02</param>
```

```
            <param name='jdGLDate'>2000/03/02</param>
            <param name='szUnformattedItemNumber'>1001</param>
            <param name='mnQuantityOrdered'>1</param>
            <param name='szDetailLineBranchPlant'>M30</param>
            <param name='szLastStatus'>220</param>
            <param name='szNextStatus'>230</param>
            <param name='cEvaluatedReceipts'>N</param>
            <param name='szTransactionCurrencyCode'>USD</param>
            <param name='cSourceRequestingPOGeneration'>0</param>
            <param name='szProgramID'>XMLTest</param>
            <param name='szUserID'>SUBSTITUTE</param>
            <param name='szAgreementNumber'></param>
            <param name='mnAgreementSupplement'>0</param>
            <param name='jdEffectiveDate'></param>
            <param name='szPurchasingCostCenter'></param>
            <param name='szObjectAccount'></param>
            <param name='szSubsidiary'></param>
            <param name='mnProcessID' idref='processID'></param>
            <param name='mnTransactionID' idref='transactionID'></param>
        </params>
        </callMethod>
        <!-- This is the second EditLine entry -->
        <callMethod app='XMLTest' name='F4311EditLine' runOnError='yes'
            returnNullData='no'>
        <params>
            <param name='mnJobNumber' idref='jobNumber'></param>
            <param name='szComputerID' idref='machineKey'></param>
            <param name='cDetailActionCode'>A</param>
            <param name='cProcessEdits'>1</param>
            <param name='cUpdateOrWriteWorkFile'>2</param>
            <param name='cCurrencyProcessingFlag'>Y</param>
            <param name='szPurchaseOrderPrOptVersion'
                idref='version'></param>
            <param name='szOrderCompany' idref='orderCompany'></param>
            <param name='szOrderType'>OP</param>
            <param name='szOrderSuffix'>000</param>
            <param name='szBranchPlant'>M30</param>
            <param name='mnSupplierNumber'
                idref='supplierNumber'></param>
            <param name='mnShipToNumber'>0.0</param>
            <param name='jdRequestedDate'>2000/03/02</param>
            <param name='jdTransactionDate'>2000/03/02</param>
            <param name='jdPromisedDate'>2000/03/02</param>
            <param name='jdGLDate'>2000/03/02</param>
            <param name='szUnformattedItemNumber'>2001</param>
            <param name='mnQuantityOrdered'>3</param>
            <param name='szDetailLineBranchPlant'>M30</param>
            <param name='szLastStatus'>220</param>
```

```
    <param name='szNextStatus'>230</param>
    <param name='cEvaluatedReceipts'>N</param>
    <param name='szTransactionCurrencyCode'>USD</param>
    <param name='cSourceRequestingPOGeneration'>0</param>
    <param name='szProgramID'>XMLTest</param>
    <param name='szUserID'>SUBSTITUTE</param>
    <param name='szAgreementNumber'></param>
    <param name='mnAgreementSupplement'>0</param>
    <param name='jdEffectiveDate'></param>
    <param name='szPurchasingCostCenter'></param>
    <param name='szObjectAccount'></param>
    <param name='szSubsidiary'></param>
    <param name='mnProcessID' idref='processID'></param>
    <param name='mnTransactionID' idref='transactionID'></param>
</params>
</callMethod>
<callMethod app='XMLTest' name='F4311EditDoc' runOnError='no'
    returnNullData='no'>
<params>
    <param name='szOrderSuffix'>000</param>
    <param name='szComputerID' idref='machineKey'></param>
    <param name='mnJobnumber' idref='jobNumber'></param>
    <param name='mnAddressNumber' idref='supplierNumber'></param>
    <param name='szOrderType'>OP</param>
    <param name='szOrderCompany' idref='orderCompany'></param>
    <param name='szVersionProcOption' idref='version'></param>
    <param name='cActionCode'>A</param>
    <param name='mnProcessID' idref='processID'></param>
    <param name='mnTransactionID' idref='transactionID'></param>
</params>
</callMethod>
<callMethod app='XMLTest' name='F4311EndDoc' runOnError='no'
    returnNullData='no'>
<params>
    <param name='szComputerID' idref='machineKey'></param>
    <param name='mnJobNumber' idref='jobNumber'></param>
    <param name='szCallingApplicationName'>XMLTest</param>
    <param name='szVersion' idref='version'></param>
    <param name='szUserID'>SUBSTITUTE</param>
    <param name='mnOrderNumberAssigned'
        id='orderNumberAssigned'></param>
    <param name='cUseWorkFiles'>2</param>
    <param name='cConsolidateLines'>0</param>
    <param name='mnProcessID' idref='processID'></param>
    <param name='mnTransactionID' idref='transactionID'></param>
</params>
</callMethod>
<returnParams runOnError='yes' returnNullData='no'>
```

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

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

```
<?xml version="1.0" encoding="utf-8" ?>
<jdeResponse environment="DV7333" user="JDE" type="callmethod"
sessionidle="" session="2612.1026498135.5" pwd="JDE">
   <callMethod name="GetLocalComputerId" runOnError="no"
       app="XMLTest">
   <returnCode code="0"/>
   <params>
       <param name="szMachineKey" id="machineKey">XEENT</param>
   </params>
   </callMethod>
   <callMethod name="F4311InitializeCaching" runOnError="no"
       app="XMLTest">
   <returnCode code="0"/>
   <params>
       <param name="cUseWorkFiles">2</param>
   </params>
   </callMethod>
   <callMethod name="F4311FSBeginDoc" returnNullData="yes"
       runOnError="no" app="XMLTest">
   <returnCode code="0"/>
   <params>
       <param name="mnJobNumber" id="jobNumber">3</param>
```

```
<param name="szComputerID" idref="machineKey">XEENT</param>
<param name="cHeaderActionCode">1</param>
<param name="cProcessEdits">1</param>
<param name="cUpdateOrWriteToWorkFile">2</param>
<param name="cRecordWrittenToWorkFile">1</param>
<param name="cCurrencyProcessingFlag">Z</param>
<param name="szOrderCOmpany" id="orderCompany">00200</param>
<param name="mnOrderNumber">0</param>
<param name="szOrderType">OP</param>
<param name="szOrderSuffix">000</param>
<param name="szBranchPlant">M30</param>
<param name="szOriginalOrderCompany"/>
<param name="szOriginalOrderNumber"/>
<param name="szOriginalOrderType"/>
<param name="szRelatedOrderCompany"/>
<param name="szRelatedOrderNumber"/>
<param name="szRelatedOrderType"/>
<param name="mnSupplierNumber"
    id="supplierNumber">17000</param>
<param name="mnShipToNumber">6074</param>
<param name="jdRequestedDate">2002/07/12</param>
<param name="jdOrderDate">2000/03/02</param>
<param name="jdPromisedDate">2002/07/12</param>
<param name="jdCancelDate"/>
<param name="szReference01"/>
<param name="szReference02"/>
 <param name="szDeliveryInstructions01">
</param>
     <param name="szDeliveryInstructions02">
</param>
<param name="szPrintMessage"/>
<param name="szSupplierPriceGroup"/>
<param name="szPaymentTerms"/>
<param name="szTaxExplanationCode"/>
<param name="szTaxRateArea"/>
<param name="szTaxCertificate">                          </param>
<param name="cAssociatedText"/>
<param name="szHoldCode"/>
<param name="szFreightHandlingCode"/>
<param name="mnBuyerNumber">0</param>
<param name="mnCarrierNumber">0</param>
<param name="cEvaluatedReceiptsFlag">N</param>
<param name="cSendMethod"/>
<param name="szLandedCostRule">   </param>
<param name="szApprovalRouteCode"/>
<param name="mnChangeOrderNumber">0</param>
<param name="cCurrencyMode">D</param>
<param name="szTransactionCurrencyCode">USD</param>
```

```
      <param name="mnCurrencyExchangeRate">0</param>
      <param name="szOrderedPlacedBy">SUBSTITUTE</param>
      <param name="szOrderTakenBy"/>
      <param name="szProgramID">EP4310</param>
      <param name="szApprovalRoutePO"/>
      <param name="szPurchaseOrderPrOptVersion"
         id="Version">ZJDE0001</param>
      <param name="szBaseCurrencyCode">USD</param>
      <param name="szUserID">SUBSTITUTE</param>
      <param name="cAddNewLineToExistingOrder"/>
      <param name="idInternalVariables">0</param>
      <param name="cSourceOfData"/>
      <param name="mnSODOrderNumber">0</param>
      <param name="szSODOrderType"/>
      <param name="szSODOrderCompany"/>
      <param name="szSODOrderSuffix"/>
      <param name="mnRetainage">0</param>
      <param name="szDescription"/>
   <param name="szRemark"/>
   <param name="jdEffectiveDate"/>
   <param name="jdPhysicalCompletionDate"/>
   <param name="mnTriangulationRateFromCurrenc">0</param>
   <param name="mnTriangulationRateToCurrency">0</param>
   <param name="cCurrencyConversionMethod"/>
   <param name="szPriceAdjustmentScheduleN"/>
   <param name="cAIADocument"/>
   <param name="mnProcessID" id="processID">2612</param>
   <param name="mnTransactionID" id="transactionID">4</param>
</params>
</callMethod>
<callMethod name="F4311EditLine" returnNullData="no"
   runOnError="yes" app="XMLTest">
<returnCode code="0"/>
<params>
   <param name="mnJobNumber" idref="jobNumber">3</param>
   <param name="szComputerID" idref="machineKey">XEENT</param>
   <param name="mnOrderLineNumber">1</param>
   <param name="cDetailActionCode">1</param>
   <param name="cProcessEdits">1</param>
   <param name="cUpdateOrWriteWorkFile">2</param>
   <param name="cRecordWrittenToWorkFile">1</param>
   <param name="cCurrencyProcessingFlag">Y</param>
   <param name="szPurchaseOrderPrOptVersion"
      idref="version">ZJDE0001</param>
   <param name="szOrderCompany"
      idref="orderCompany">00200</param>
   <param name="szOrderType">OP</param>
   <param name="szOrderSuffix">000</param>
```

```
 <param name="szBranchPlant">M30</param>
 <param name="mnSupplierNumber"
    idref="supplierNumber">17000</param>
 <param name="mnShipToNumber">6074</param>
 <param name="jdRequestedDate">2000/03/02</param>
 <param name="jdTransactionDate">2000/03/02</param>
 <param name="jdPromisedDate">2000/03/02</param>
 <param name="jdGLDate">2000/03/02</param>
 <param name="szUnformattedItemNumber">1001</param>
 <param name="mnQuantityOrdered">1</param>
 <param name="mnUnitPrice">32,1000</param>
 <param name="mnExtendedPrice">32,1</param>
 <param name="szLineType">S</param>
 <param name="szDescription1">Bike Rack - Trunk Mount</param>
<param name="szDescription2">                           </param>
<param name="szDetailLineBranchPlant">M30</param>
<param name="szLocation">   .    .              </param>
<param name="szLotNumber">                    </param>
<param name="szTransactionUoM">EA</param>
<param name="szPurchasingUoM">EA</param>
<param name="szLastStatus">220</param>
<param name="szNextStatus">230</param>
<param name="mnDiscountFactor">1</param>
<param name="szInventoryPriceRule">        </param>
<param name="szPrintMessage"> </param>
<param name="cTaxable">Y</param>
<param name="szGLClassCode">IN30</param>
<param name="mnBuyerNumber">8444</param>
<param name="szPurchasingCategoryCode1"> </param>
<param name="szPurchasingCategoryCode2"> </param>
<param name="szPurchasingCategoryCode3"> </param>
<param name="szPurchasingCategoryCode4">240</param>
<param name="szLandedCostRule"> </param>
<param name="mnWeight">80</param>
<param name="szWeightUoM">OZ</param>
<param name="mnVolume">2,25</param>
<param name="szVolumeUoM">FC</param>
<param name="cEvaluatedReceipts">N</param>
<param name="cInventoryInterface">Y</param>
<param name="szTransactionCurrencyCode">USD</param>
<param name="szBaseCurrencyCode">USD</param>
<param name="cSourceRequestingPOGeneration">0</param>
<param name="szProgramID">XMLTest</param>
<param name="szUserID">SUBSTITUTE</param>
<param name="szAgreementNumber"/>
<param name="mnAgreementSupplement">0</param>
<param name="jdEffectiveDate"/>
<param name="szPurchasingCostCenter"/>
```

```
          <param name="szObjectAccount"/>
          <param name="szSubsidiary"/>
          <param name="cStockingType">P</param>
          <param name="mnProcessID" idref="processID">2612</param>
          <param name="mnTransactionID" idref="transactionID">4</param>
          <param name="mnIdentifierShortItem">60003</param>
      </params>
      </callMethod>
      <callMethod name="F4311EditLine" returnNullData="no"
            runOnError="yes" app="XMLTest">
      <returnCode code="0"/>
      <params>
          <param name="mnJobNumber" idref="jobNumber">3</param>
          <param name="szComputerID" idref="machineKey">XEENT</param>
          <param name="mnOrderLineNumber">2</param>
          <param name="cDetailActionCode">1</param>
          <param name="cProcessEdits">1</param>
          <param name="cUpdateOrWriteWorkFile">2</param>
          <param name="cRecordWrittenToWorkFile">1</param>
          <param name="cCurrencyProcessingFlag">Y</param>
          <param name="szPurchaseOrderPrOptVersion"
              idref="version">ZJDE0001</param>
          <param name="szOrderCompany"
              idref="orderCompany">00200</param>
          <param name="szOrderType">OP</param>
          <param name="szOrderSuffix">000</param>
          <param name="szBranchPlant">M30</param>
          <param name="mnSupplierNumber"
           idref="supplierNumber">17000</param>
          <param name="mnShipToNumber">6074</param>
          <param name="jdRequestedDate">2000/03/02</param>
          <param name="jdTransactionDate">2000/03/02</param>
          <param name="jdPromisedDate">2000/03/02</param>
          <param name="jdGLDate">2000/03/02</param>
          <param name="szUnformattedItemNumber">2001
          </param>
          <param name="mnQuantityOrdered">3</param>
          <param name="mnUnitPrice">164,0817</param>
          <param name="mnExtendedPrice">492,2451</param>
          <param name="szLineType">S</param>
          <param name="szDescription1">Cro-Moly Frame, Red          </param>
          <param name="szDescription2">                              </param>
          <param name="szDetailLineBranchPlant">M30</param>
          <param name="szLocation">  .   .                </param>
          <param name="szLotNumber">                              </param>
          <param name="szTransactionUoM">EA</param>
          <param name="szPurchasingUoM">EA</param>
          <param name="szLastStatus">220</param>
```

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

```
            <param name="mnProcessID" idref="processID">2612</param>
            <param name="mnTransactionID" idref="transactionID">4</param>
    </params>
      </callMethod>
      <callMethod name="F4311EndDoc" returnNullData="no"
            runOnError="no" app="XMLTest">
      <returnCode code="0"/>
      <params>
        <param name="szComputerID" idref="machineKey">XEENT</param>
        <param name="mnJobNumber" idref="jobNumber">3</param>
        <param name="szCallingApplicationName">XMLTest</param>
        <param name="szVersion" idref="version">ZJDE0001</param>
        <param name="szUserID">SUBSTITUTE</param>
        <param name="mnOrderNumberAssigned"
            id="orderNumberAssigned">4884</param>
        <param name="cUseWorkFiles">2</param>
        <param name="cConsolidateLines">0</param>
        <param name="mnProcessID" idref="processID">2612</param>
        <param name="mnTransactionID" idref="transactionID">4</param>
      </params>
      </callMethod>
      <returnParams>
        <param name="JobNumber" idref="machineKey">XEENT</param>
        <param name="ComputerID" idref="jobNumber">3</param>
        <param name="OrderNumberAssigned" idref="orderNumberAssigned">4884</
param>
      </returnParams>
      </jdeResponse>
```

Sample Sales Order Request

The following is a sample Sales Order request.

Example: Executing a Sales Order Request

The following code 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>
   <onError abort='yes'>
   </onError>
   </callMethod>
   <callMethod name='F4211FSBeginDoc' app='XMLInterop'
         runOnError='no'>
   <params>
     <param name='mnCMJobNumber' id='1'></param>
     <param name='cCMDocAction'>A</param>
     <param name='cCMProcessEdits'>1</param>
     <param name='szCMComputerID' idref='2'></param>
     <param name='cCMUpdateWriteToWF'>2</param>
     <param name='szCMProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
     <param name='szOrderType'>SO</param>
     <param name='szBusinessUnit'>M30</param>
     <param name='mnAddressNumber'>4242</param>
     <param name='jdOrderDate'>2000/03/29</param>
     <param name='szReference'>10261</param>
     <param name='cApplyFreightYN'>Y</param>
     <param name='szCurrencyCode'></param>
     <param name='cWKSourceOfData'></param>
     <param name='cWKProcMode'></param>
     <param name='mnWKSuppressProcess'>0</param>
   </params>
   <onError abort='yes'>
   <callMethod name='F4211ClearWorkFile' app='XMLInterop'
      runOnError='yes'>
   <params>
<param name='mnJobNo' idref='1'></param>
<param name='szComputerID' idref='2'></param>
<param name='mnFromLineNo'>0</param>
<param name='mnThruLineNo'>0</param>
<param name='cClearHeaderWF'>2</param>
<param name='cClearDetailWF'>2</param>
```

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

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

Sample Sales Order Response

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

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

Example: **Using the Sales Order Response**

The following code is the jdeResponse document.

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

<jdeResponse environment="DV7333" user="JDE" type="callmethod" pwd="JDE">
   <callMethod name="GetLocalComputerId" runOnError="no"
      app="XMLInterop">
   <returnCode code="0"/>
   <params>
      <param name="szMachineKey" id="2">XEENT</param>
   </params>
   </callMethod><callMethod name="F4211FSBeginDoc" runOnError="no"
      app="XMLInterop">
   <returnCode code="1"/>
   <params>
     <param name="mnCMJobNumber" id="1">3</param>
     <param name="cCMDocAction">A</param>
     <param name="cCMProcessEdits">1</param>
     <param name="szCMComputerID" idref="2">XEENT</param>
     <param name="cCMErrorConditions">1</param>
     <param name="cCMUpdateWriteToWF">2</param>
     <param name="szCMProgramID">XMLInterop</param>
     <param name="szCMVersion">ZJDE0001</param>
     <param name="szOrderCo">00200</param>
     <param name="szOrderType">SO</param>
     <param name="szBusinessUnit">M30</param>
     <param name="mnAddressNumber">4242</param>
     <param name="mnShipToNo">4242</param>
     <param name="jdRequestedDate">2000/03/29</param>
     <param name="jdOrderDate">2000/03/29</param>
     <param name="jdPromisedDate">2000/03/29</param>
     <param name="szReference">10261</param>
     <param name="szDeliveryInstructions1">            </param>
     <param name="szDeliveryInstructions2">            </param>
     <param name="szPrintMesg">        </param>
     <param name="szPaymentTerm">       </param>
```

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

```
            <param name="jdPromisedDate">2000/03/29</param>
            <param name="jdPromisedDlvryDate">2000/03/29</param>
            <param name="szItemNo">1001                 </param>
            <param name="szLocation">  .   .    </param>
            <param name="szDescription1">Bike Rack Trunk Mount </param>
            <param name="szDescription2">                            </param>
            <param name="szLineType">S</param>
            <param name="szLastStatus">900</param>
            <param name="szNextStatus">540</param>
            <param name="mnQtyOrdered">1</param>
            <param name="mnQtyBackordered">1</param>
            <param name="mnUnitPrice">44,99</param>
            <param name="mnUnitCost">32,1000</param>
            <param name="szPrintMesg">        </param>
            <param name="cPaymentInstrument"> </param>
            <param name="cSalesTaxableYN">N</param>
            <param name="cAssociatedText"> </param>
            <param name="szTransactionUOM">EA</param>
            <param name="szPricingUOM">EA</param>
            <param name="mnItemWeight">80</param>
            <param name="szWeightUOM">OZ</param>
            <param name="mnForeignUnitPrice">44,99</param>
            <param name="mnForeignUnitCost">32,1000</param>
            <param name="mnDiscountFactor">1</param>
            <param name="mnCMLineNo">1</param>
            <param name="szCMProgramID">XMLInterop</param>
            <param name="szCMVersion">ZJDE0001</param>
            <param name="mnSupplierNo">4343</param>
            <param name="mnWKOrderTotal">44,99</param>
            <param name="mnWKForeignOrderTotal">44,99</param>
            <param name="mnWKTotalCost">32,1</param>
            <param name="mnWKForeignTotalCost">32,1</param>
            <param name="cWKSourceOfData"/>
            <param name="cWKCheckAvailability">1</param>
            <param name="mnLastLineNoAssigned">1</param>
            <param name="cStockingType">P</param>
            <param name="cParentItmMethdOfPriceCalcn">1</param>
            <param name="mnShortItemNo">60003</param>
            <param name="szSalesOrderFlags">0</param>
            <param name="jdPriceEffectiveDate">2000/03/29</param>
            <param name="jdPromisedShip">2000/03/29</param>
            <param name="mnQuantityAvailable">-34</param>
            <param name="mnItemVolume_ITVL">2,25</param>
            <param name="szVolumeUOM_VLUM">FC</param>
            <param name="szRevenueBusinessUnit">  M30</param>
            <param name="mnProcessID">2252</param>
            <param name="mnTransactionID">4</param>
        </params>
```

```
   <errors><error code="030B">Warning: Order Quantity
    Exceeds what&apos;s Available</error></errors>
</callMethod><callMethod name="F4211FSEditLine" runOnError="yes"
    app="XMLInterop"><returnCode code="1"/><params>
   <param name="mnCMJobNo" idref="1">3</param>
   <param name="cCMLineAction">A</param>
   <param name="cCMProcessEdits">1</param>
   <param name="cCMWriteToWFFlag">2</param>
   <param name="cCMRecdWrittenToWF">1</param>
   <param name="szCMComputerID" idref="2">XEENT</param>
   <param name="cCMErrorConditions">1</param>
   <param name="szOrderCo">00200</param>
   <param name="szOrderType">SO</param>
   <param name="szBusinessUnit">M30</param>
   <param name="mnShipToNo">4242</param>
   <param name="jdRequestedDate">2000/03/29</param>
   <param name="jdPromisedDate">2000/03/29</param>
   <param name="jdPromisedDlvryDate">2000/03/29</param>
   <param name="szItemNo">1001                    </param>
   <param name="szLocation">   .    .    </param>
   <param name="szDescription1">Bike Rack-Trunk Mount    </param>
   <param name="szDescription2">                           </param>
   <param name="szLineType">S</param>
   <param name="szLastStatus">900</param>
   <param name="szNextStatus">540</param>
   <param name="mnQtyOrdered">10</param>
   <param name="mnQtyBackordered">10</param>
   <param name="mnUnitPrice">44,99</param>
   <param name="mnUnitCost">32,1000</param>
   <param name="szPrintMesg">              </param>
   <param name="cPaymentInstrument"> </param>
   <param name="cSalesTaxableYN">N</param>
   <param name="cAssociatedText"> </param>
   <param name="szTransactionUOM">EA</param>
   <param name="szPricingUOM">EA</param>
   <param name="mnItemWeight">800</param>
   <param name="szWeightUOM">OZ</param>
   <param name="mnForeignUnitPrice">44,99</param>
   <param name="mnForeignUnitCost">32,1000</param>
   <param name="mnDiscountFactor">1</param>
   <param name="mnCMLineNo">2</param>
   <param name="szCMProgramID">XMLInterop</param>
   <param name="szCMVersion">ZJDE0001</param>
   <param name="mnSupplierNo">4343</param>
   <param name="mnWKOrderTotal">494,89</param>
   <param name="mnWKForeignOrderTotal">494,89</param>
   <param name="mnWKTotalCost">321</param>
   <param name="mnWKForeignTotalCost">321</param>
```

```xml
          <param name="cWKSourceOfData"/>
          <param name="cWKCheckAvailability">1</param>
          <param name="mnLastLineNoAssigned">2</param>
          <param name="cStockingType">P</param>
          <param name="cParentItmMethdOfPriceCalcn">1</param>
          <param name="mnShortItemNo">60003</param>
          <param name="szSalesOrderFlags">                0    </param>
          <param name="jdPriceEffectiveDate">2000/03/29</param>
          <param name="jdPromisedShip">2000/03/29</param>
          <param name="mnQuantityAvailable">-44</param>
          <param name="mnItemVolume_ITVL">22,5</param>
          <param name="szVolumeUOM_VLUM">FC</param>
          <param name="szRevenueBusinessUnit">M30</param>
          <param name="mnProcessID">2252</param>
          <param name="mnTransactionID">4</param>
        </params><errors><error code="030B">Warning: Order Quantity
          Exceeds what&apos;s Available</error></errors>
      </callMethod><callMethod name="F4211FSEndDoc" runOnError="no"
          app="XMLInterop"><returnCode code="0"/>
      <params>
          <param name="mnCMJobNo" idref="1">3</param>
          <param name="mnSalesOrderNo">2623</param>
          <param name="szCMComputerID" idref="2">XEENT</param>
          <param name="cCMErrorCondition">0</param>
          <param name="szOrderType">SO</param>
          <param name="szKeyCompany">00200</param>
          <param name="mnOrderTotal">494,89</param>
          <param name="szWorkstationID">XEENT</param>
          <param name="szCMProgramID">XMLInterop</param>
          <param name="szCMVersion">ZJDE0001</param>
          <param name="mnTimeOfDay">174220</param>
          <param name="cCMUseWorkFiles">2</param>
          <param name="cCMProcessEdits">1</param>
          <param name="mnProcessID">2252</param>
          <param name="mnTransactionID">4</param>
        </params>
      </callMethod><returnParams failureDestination="ERROR.Q"
          successDestination="SUCCESS.Q">
        </returnParams>
    </jdeResponse>
```

Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

Mail:	Documentation Services - Customer Support
	Information Builders, Inc.
	Two Penn Plaza
	New York, NY 10121-2898
Fax:	(212) 967-0460
E-mail:	books_info@ibi.com
Web form:	http://www.informationbuilders.com/bookstore/derf.html

Name:_____

Company:_____

Address:_____

Telephone:_____Date:_____

E-mail:_____

Comments:

Reader Comments