

iWay

iWay Application Adapter for J.D. Edwards
OneWorld for BEA WebLogic User's Guide
Version 5 Release 5

October 15, 2004

DN3501473.1004

EDA, EDA/SQL, FIDEL, FOCCALC, FOCUS, FOCUS Fusion, FOCUS Vision, Hospital-Trac, Information Builders, the Information Builders logo, Parlay, PC/FOCUS, SmartMart, SmartMode, SNAPpack, TableTalk, WALDO, Web390, WebFOCUS and WorldMART are registered trademarks, and iWay and iWay Software are trademarks of Information Builders, Inc.

Due to the nature of this material, this document refers to numerous hardware and software products by their trademarks. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2004, by Information Builders, Inc and iWay Software. All rights reserved. Patent Pending. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Preface

This document is written for system integrators with programming backgrounds and an understanding of the J.D. Edwards OneWorld product in an application space. Extensive knowledge of J.D. Edwards OneWorld 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 J.D. Edwards OneWorld. For system integrators concerned with the development of a client/server interface between J.D. Edwards OneWorld and other applications, this guide addresses the OneWorld integration aspects. It does not cover other applications or application wrappers.

How This Manual Is Organized

The following table lists the numbers and titles of the chapters and appendixes for this manual with a brief description of the contents of each chapter and appendix.

Chapter/Appendix		Contents
1	Introducing the iWay Application Adapter for J.D. Edwards OneWorld	Introduces iWay Application Adapter for J.D. Edwards OneWorld.
2	Creating XML Schemas and Business Services for J.D. Edwards OneWorld	Describes how to create schemas for J.D. Edwards OneWorld functions or Web services for iWay Business Services Engine (iBSE) deployment.
3	Listening for Database Events	Describes how to configure agents and listeners for OneWorld.
4	Using Web Services Policy-Based Security	Describes how to configure Web services policy-based security.
5	Management and Monitoring	Describes how to configure Web services policy-based security.
A	Using iWay Application Explorer in BEA WebLogic Workshop	Describes the use of the iWay Application Explorer as implemented in the BEA WebLogic Workshop.

B	Configuring J.D. Edwards OneWorld for Outbound Transaction Processing	Describes how to enable outbound transaction processing in OneWorld and how to modify the jde.ini file for XML support.
C	Sample Files	Provides examples of the jdeRequest and jdeResponse XML structures for executing business functions within OneWorld.

Documentation Conventions

The following table lists the conventions that apply in this manual and a description of each.

Convention	Description
THIS TYPEFACE or <i>this typeface</i>	Denotes syntax that you must enter exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
<i>this typeface</i>	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.
<i>this typeface</i>	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 (...).
.	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 the iWay Application Adapter for J.D. Edwards OneWorld?

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 Adapter for J.D. Edwards OneWorld 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 when you call, please provide the following information:

- Your six-digit site code number (xxxx.xx).
- Your software configuration.

The following table lists the information to provide about your software configuration.

	Version-Build Date	HF/Service Pack	Patches	OS	Java Version
iWay Product					
Third-party Application Server					
EIS (adapter target)					

Note: For the EIS, ensure you record the application or database name and release level, including minor versions, for example, 4.6.1.

- The exact nature of the error or problem, specified as follows:
 - Steps to reproduce the problem.
 - Problem description (be as specific as possible).
 - Error message(s).
- To best define the problem, provide the following:
 - Screen captures of the error
 - Error output files
 - Trace files and log files
 - Log transaction
 - XML schemas and/or document instances
 - Other input documents (for example, transformations)
 - Configuration files (all are applicable):
 - .xch files
 - config.xml file
 - base.xml file
 - repository.xml file
 - ibserrepo.xml file

.dic files

.rules files

- Environment variable settings:

IWAY55

IWAY55OEM

CLASSPATH

JAVA_HOME

ACBDIR

CBDIR (UNIX)

- Has the process, procedure, or query ever worked in its current form? Has it changed recently? If so, how (provide specific details)? How often does the problem occur?
- Can this problem be reproduced? If so, how? Can it be consistently reproduced?
- Have you tried to reproduce your problem in the simplest form possible?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production?
- Do you just have questions about functionality or documentation?

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

1. Introducing the iWay Application Adapter for J.D. Edwards OneWorld	1-1
Executing a J.D. Edwards OneWorld Master Business Function	1-2
Resource Adapters	1-2
Accessing Data Stored in J.D. Edwards OneWorld	1-2
Propagating External Listeners Into J.D. Edwards OneWorld	1-3
Propagating Internal Listeners Out of J.D. Edwards OneWorld	1-3
J.D. Edwards OneWorld Interoperability Framework	1-4
Using iWay Application Explorer	1-6
Deployment Information for the iWay Application Adapter for J.D. Edwards OneWorld	1-6
Deployment Information Roadmap	1-7
The iWay Business Services Engine	1-7
The iWay Enterprise Connector for J2EE Connector Architecture	1-8
2. Creating XML Schemas and Business Services for J.D. Edwards OneWorld	2-1
Overview	2-2
Using GenJava to Generate a Schema	2-2
Defining a Target to J.D. Edwards OneWorld	2-2
Connecting to J.D. Edwards OneWorld	2-2
Managing a Connection to J.D. Edwards OneWorld	2-6
Creating an XML Schema for a J.D. Edwards OneWorld Master Business Function	2-7
Creating a Request and a Response Schema	2-8
Creating a Business Service	2-9
Testing a Business Service	2-12
Generating WSDL From a Web Service	2-15
Credential Mapping	2-16
3. Listening for Database Events	3-1
Understanding iWay Event Functionality	3-2
Creating an Event Port	3-2
Creating an Event Port From the iWay Adapters Tab	3-2
Creating an Event Port From the iWay Event Adapters Tab	3-4
Editing and Deleting an Event Port	3-18
Creating a Channel	3-19
Editing and Deleting a Channel	3-23
The OneWorld Event Listener	3-24
Configuring the OneWorld Event Listener	3-24
Creating the iwoevent.cfg File	3-25
Adding Connection Information	3-25
Logging and Error Handling	3-27

4. Using Web Services Policy-Based Security	4-1
Web Services Policy-Based Security	4-2
Configuring Web Services Policy-Based Security	4-3
Configuring the IP and Domain Restrictions Policy Type	4-10
5. Management and Monitoring	5-1
Managing and Monitoring Services and Events Using iBSE	5-2
Managing and Monitoring Services and Events Using the IVP	5-16
Testing the iWay Event Adapters Using the IVP	5-18
Monitoring Services	5-20
Setting Engine Log Levels	5-22
Configuring Connection Pool Sizes	5-23
A. Using iWay Application Explorer in BEA WebLogic Workshop	A-1
Overview	A-2
Using GenJava to Generate a Schema	A-2
Starting iWay Application Explorer in WebLogic Workshop	A-2
Creating a New Configuration	A-3
Defining a Target	A-5
Connecting to J.D. Edwards OneWorld	A-5
Disconnecting From or Deleting a Connection	A-9
Creating an XML Schema	A-10
Creating a Request and a Response Schema	A-10
Creating a Business Service	A-14
Testing a Business Service	A-16
Generating WSDL From a Web Service	A-17
Credential Mapping	A-17
Understanding iWay Event Functionality	A-17
Creating, Editing, and Deleting a Port	A-18
Creating an Event Port From the iWay Event Adapters Tab	A-18
Editing and Deleting an Event Port	A-29
Creating, Editing, and Deleting a Channel	A-30
Editing and Deleting a Channel	A-34
Deploying iWay Components in a Clustered BEA WebLogic Environment	A-35
Adding a Control for an iWay Resource in BEA WebLogic Workshop	A-42
Adding a Web Service Control to a BEA WebLogic Workshop Application	A-42
Extensible CCI Control	A-43
Overview	A-43
Using the Extensible CCI Control	A-43
B. Configuring J.D. Edwards OneWorld for Outbound Transaction Processing ...	B-1
Specifying Outbound Functionality for a Business Function	B-2
Outbound Transaction Processing	B-2
The Data Export Control Table and the Processing Log Table	B-3

Modifying the OneWorld jde.ini FileB-3

C. Sample FilesC-1

 Issuing a Single-Function RequestC-2

 Issuing a Multiple-Function RequestC-4

 Sample Sales Order Request C-16

 Sample Sales Order Response C-19

CHAPTER 1

Introducing the iWay Application Adapter for J.D. Edwards OneWorld

Topics:

- Executing a J.D. Edwards OneWorld Master Business Function
- Accessing Data Stored in J.D. Edwards OneWorld
- J.D. Edwards OneWorld Interoperability Framework
- Deployment Information for the iWay Application Adapter for J.D. Edwards OneWorld

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

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

Executing a J.D. Edwards OneWorld Master Business Function

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

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

Resource Adapters

The iWay Application Adapter for J.D. Edwards OneWorld 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 OneWorld

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

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

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

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 J.D. Edwards OneWorld. The information the listener receives is used to build an XML record and is forwarded to any specified disposition for further processing.

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

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

The XML format conforms to an XML schema.

The schema is based on metadata in the EIS.

Propagating External Listeners Into J.D. Edwards OneWorld

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

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

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

Propagating Internal Listeners Out of J.D. Edwards OneWorld

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

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

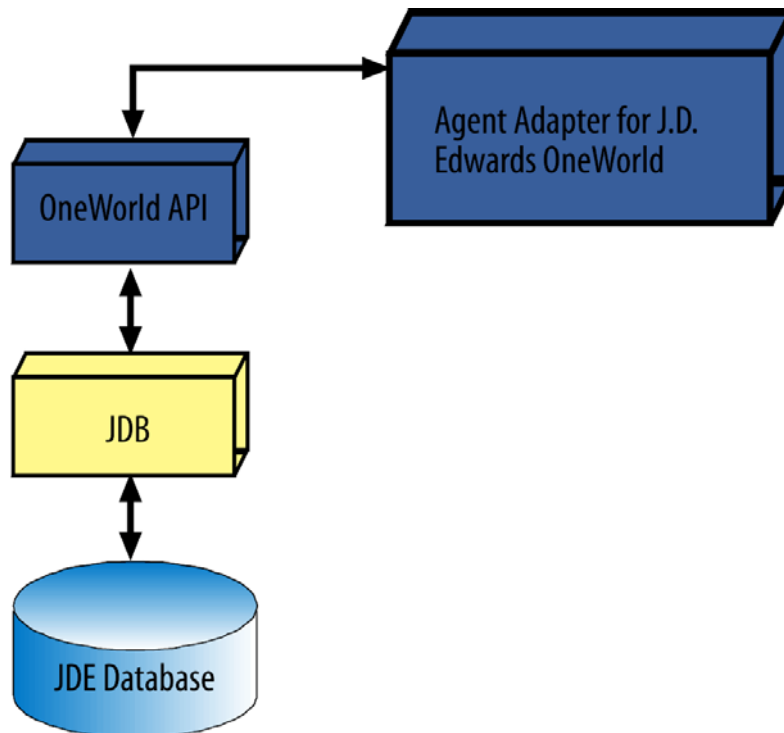
J.D. Edwards OneWorld Interoperability Framework

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

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

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

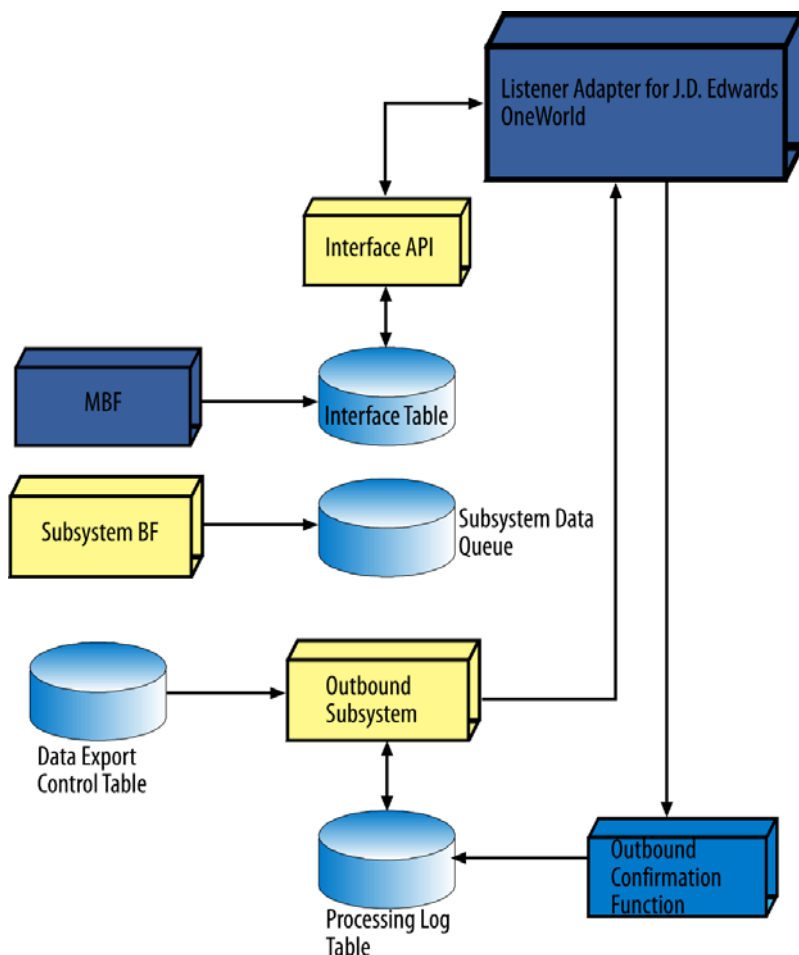
The following diagram illustrates the J.D. Edwards OneWorld inbound processing (from the EIS to WebLogic Server) framework. It shows the OneWorld components and the agent adapter in the inbound processing sequence.



J.D. Edwards Inbound Processing Framework

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

The following diagram illustrates the J.D. Edwards OneWorld outbound processing framework. It shows the OneWorld components and the listener adapter in the outbound processing sequence.



J.D. Edwards Outbound Processing Framework

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

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

Using iWay Application Explorer

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

Deployment Information for the iWay Application Adapter for J.D. Edwards OneWorld

The iWay Application Adapter for J.D. Edwards OneWorld works in conjunction with the following components:

- iWay Application Explorer

and

- iWay Business Services Engine (iBSE)

or

- 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	<ul style="list-style-type: none"> Chapters 2 and 3, and Appendix A of this guide <i>iWay Installation and Configuration for BEA WebLogic</i> <i>iWay Servlet Application Explorer for BEA WebLogic</i>
iWay Business Services Engine (iBSE)	<ul style="list-style-type: none"> <i>iWay Installation and Configuration for BEA WebLogic</i>
iWay Enterprise Connector for J2EE Connector Architecture (JCA)	<ul style="list-style-type: none"> <i>iWay Connector for JCA for BEA WebLogic User's Guide</i> <i>iWay Installation and Configuration for BEA WebLogic</i>

The iWay Business Services Engine

The 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.

CHAPTER 2

Creating XML Schemas and Business Services for J.D. Edwards OneWorld

Topics:

- Overview
- Defining a Target to J.D. Edwards OneWorld
- Managing a Connection to J.D. Edwards OneWorld
- Creating an XML Schema for a J.D. Edwards OneWorld Master Business Function
- Creating a Business Service

This section describes how to open a connection to J.D. Edwards OneWorld, how to create schemas for J.D. Edwards OneWorld functions, and how to create business services.

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

Overview

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

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

For more information on creating Web services and on Application Explorer in general, see the *iWay Application Explorer for BEA WebLogic 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 OneWorld utility called GenJava. You use Application Explorer to generate schemas against OneWorld GenJava wrappers.

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

Defining a Target to J.D. Edwards OneWorld

To browse the available Master Business Functions, 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.

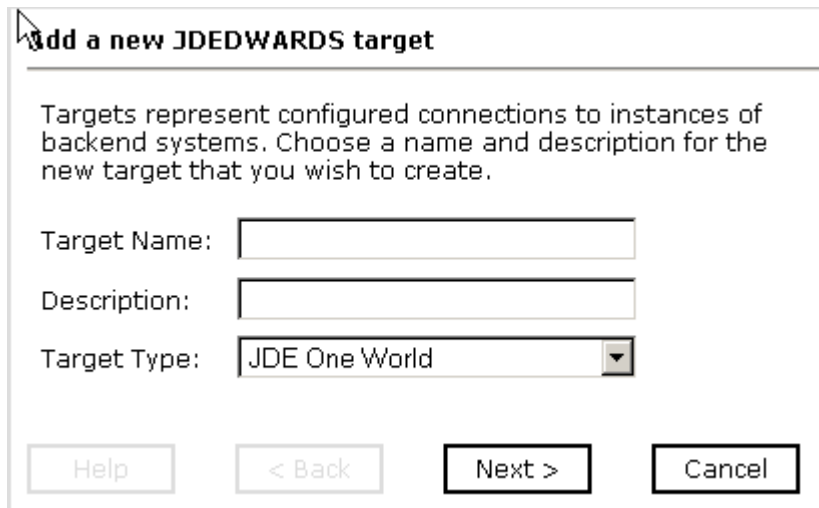
Connecting to J.D. Edwards OneWorld

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

Procedure How to Define a New Target to J.D. Edwards OneWorld

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 graphic 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. In this example, JDE OneWorld is selected from the list as the target type:



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:

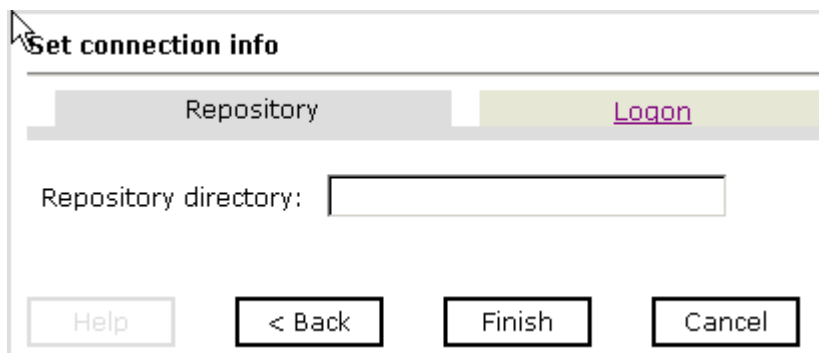
Description:

Target Type:

Help < Back Next > Cancel

- a. Type a name (for example, JDEConnection) and a brief description for the new target.
 - b. From the Target Type drop-down list, select a target type (for example, JDE OneWorld).
3. Click Next.

The following graphic shows the Repository tab active, with a field that prompts you to enter the repository directory.



Set connection info

Repository Login

Repository directory:

Help < Back Finish Cancel

4. Type the path to the GenJava repository.

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

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

5. Click *Logon*.

The following graphic shows the Logon dialog box that opens, with fields that prompt you to enter the information required for connecting to the server on which J.D. Edwards OneWorld is running:

Target connection info

[Repository](#) Logon

User id:

User password:

JDE Environment:

Application:

Server IP address:

Server port :

Help < Back Finish Cancel

- a. Type the appropriate information for your target type based on the following table. The table lists and describes the fields on the Logon dialog box.

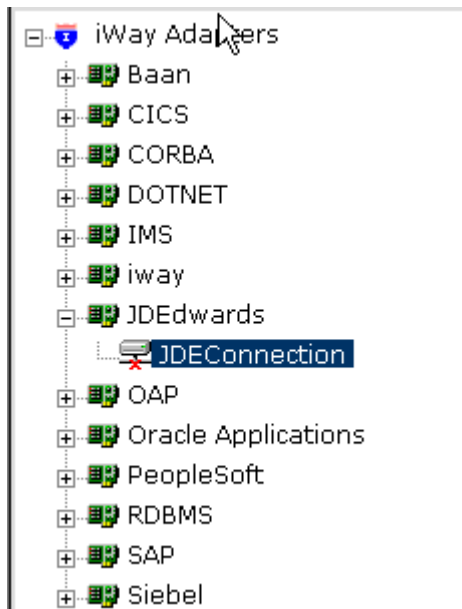
Target Parameter	Description
User id	Valid user ID for J.D. Edwards OneWorld.
User password	Password associated with the user ID.
JDE Environment	J.D. Edwards OneWorld environment, for example, DU7333. For more information about this parameter, see your J.D. Edwards OneWorld documentation or ask your OneWorld system administrator.

Target Parameter	Description
Application	XMLInterop or the application name in J.D. Edwards OneWorld. Optional.
Server IP address	Name of the server on which J.D. Edwards OneWorld is running. This can be the name of the server, for example, JDEOW, or its IP address, for example, 123.45.67.89.
Server port	Port number on which the server is listening, for example, 6009.

b. Click *Finish*.

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

The following graphic 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 J.D. Edwards OneWorld Master Business Function*.

Procedure How to Connect to a Defined J.D. Edwards OneWorld 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 your connection parameters. If required, provide the password and then click OK.
The following graphic 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 OneWorld

To manage J.D. Edwards OneWorld 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 J.D. Edwards OneWorld

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 graphic shows the x icon to the left of JDEConnection, indicating that the node is disconnected:



Procedure How to Delete a Connection to J.D. Edwards OneWorld

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 J.D. Edwards OneWorld Master Business Function

To execute a Master Business Function (MBF), the adapter must receive a request document through the J.D. Edwards OneWorld ThinNet API. The agent processes the request and sends an XML response document indicating the result. 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 for BEA WebLogic* manual and the *iWay Connector for JCA for BEA WebLogic User's Guide*.

Creating a Request and a Response Schema

The following procedure explains how to create request and response schemas for a J.D. Edwards OneWorld Master Business Function using Application Explorer.

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

- 1. If you are not connected to a J.D. Edwards OneWorld target, connect to one, as described in *How to Connect to a Defined J.D. Edwards OneWorld Target* on page 2-5.
- 2. Expand the *Services* node.
- 3. Expand the node of the Master Business Function (MBF) for which you want to create the schema.
- 4. Expand and then select the node beneath the MBF.
- 5. In the right pane, move the pointer over *Operations* and select *Generate Schema*.

Application Explorer creates the schemas.

The following graphic 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 graphic 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

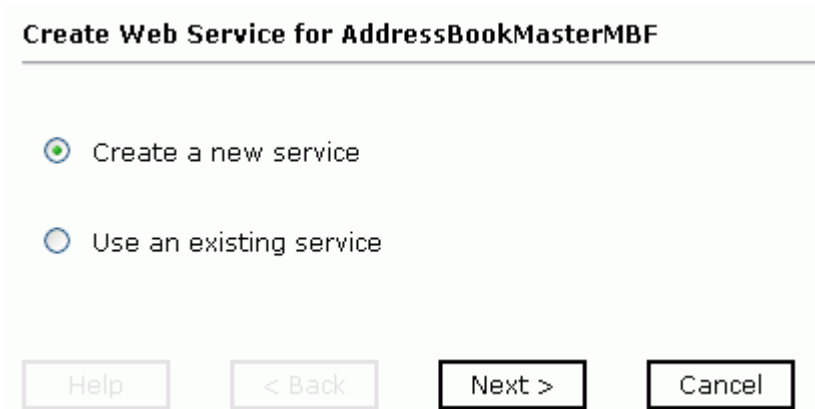
You can generate a business service (also known as a Web service). You can explore the business function repository and generate business services for the functions you want to use with the adapter.

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 for BEA WebLogic* manual and the *iWay Connector for JCA for BEA WebLogic User's Guide*.

Procedure How to Create a Business Service

1. Expand the *JDEdwards* node and then expand the *Services* 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 Service*.

The following graphic shows the Create Web Service dialog box that opens, with option buttons enabling you to choose between a new service and an existing service. In this example, the title on the dialog box indicates that the creation 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 dialog box in the sequence opens.

The following graphic shows the dialog box, with fields prompting you to enter the service name and description. A third field shows the available values for the license, from which you make a selection:

Create Web Service for AddressBookMasterMBF

Service Name:

Description:

License:
test

- a. Type a name and a brief description for the service.
 - b. Select one of the available licenses.
- If you select the Use an existing service option, another dialog box in the sequence opens.

The following graphic shows the dialog box, with a drop-down list from which you select the service:

Create Web Service for AddressBookMasterMBF

- a. Click the arrow next to the drop-down list.
- b. From the drop-down list, select a service.

4. Click *Next*.

Another dialog box 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 a business service is created, 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 link in the right pane.
5. In the right pane, click the named business services link.

The test option appears in the right pane.

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

The screenshot shows the iWay AddressUpdate web service interface. At the top is a purple header bar with the iWay Software logo and the text "Address An iWay Business Service". Below the header, a link "here" is provided for a complete list of operations. The main section is titled "AddressUpdate" with the subtitle "Update Address". Under the heading "Test", a note states: "To test the operation using the [SOAP protocol](#), click the 'Invoke' button." Below this is a form labeled "input xml:" containing a large text area for XML input. At the bottom of the form are four buttons: "Browse...", "Upload", "More", and "Invoke".

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 graphic 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:AddressUpdate"
    cid="BDF3FEF8CD73B26E42CF1722575DFA62">
  - <jdeResponse user="JDE" sessionid=""
    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

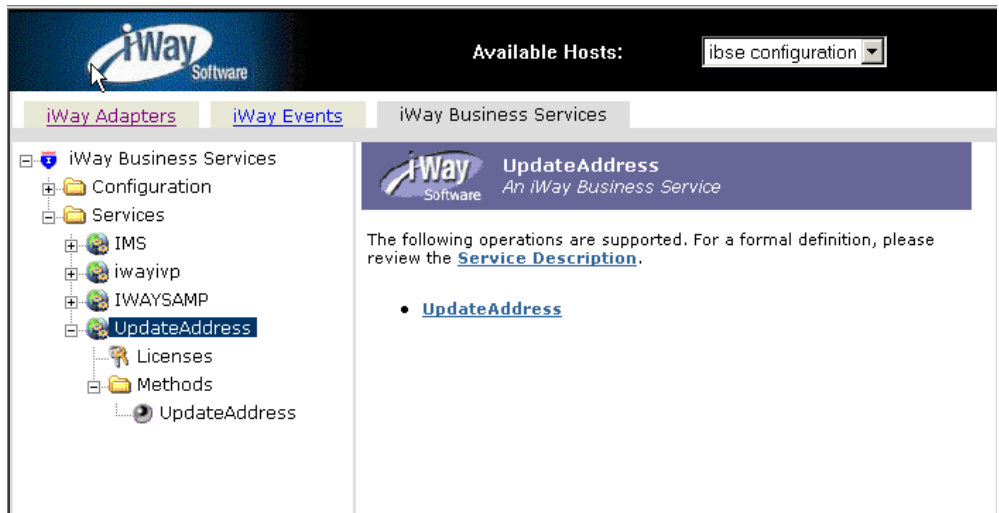
Generating Web Services Description Language (WSDL) from a Web service enables you to make the Web service available to other services within a host server such as BEA WebLogic Server.

Procedure How to Generate WSDL From a Web Service

1. Click the *iWay Business Services* tab.

The window that opens displays the current iWay Business Services in the left pane.

The following graphic shows a sample window in which the service UpdateAddress is expanded in the left pane. The right pane shows the supported operations:



2. In the left pane, expand the newly created Web service (for example, UpdateAddress).
3. In the right pane, right-click the *Service Description* link 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*.

Credential Mapping

For each SOAP request that is received, iBSE checks to see if a user name and password is included in the SOAP header. If a user name and password is available, iBSE acquires this information and replaces the values retrieved from the repository when pushing the request to the iWay Adapter.

CHAPTER 3

Listening for Database Events

Topics:

- Understanding iWay Event Functionality
- Creating an Event Port
- Creating a Channel
- The OneWorld Event Listener
- Configuring the OneWorld Event Listener
- Logging and Error Handling

This section describes how to use the iWay Application Adapter for J.D. Edwards OneWorld, deployed to a server such as BEA WebLogic Server, to listen for events.

Understanding iWay 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 iWay Servlet Application Explorer. To create an iWay event, you must create a port and a channel.

- Port

A port associates a particular business object exposed by the iWay 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 BEA WebLogic Server. For more information, see *Creating an Event Port*.

- 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 iWay Adapter. For more information, see *Creating a Channel* on page 3-19.

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.

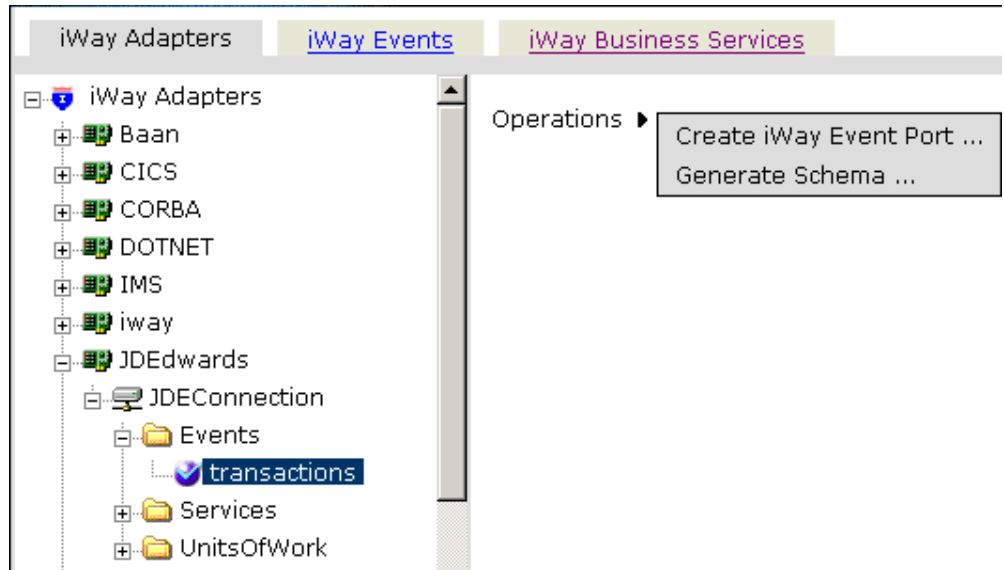
Creating an Event Port From the iWay Adapters Tab

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 J.D. Edwards OneWorld object for which you want to create an event port.

The following graphic 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 graphic shows the Create iWay 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:

Create iWay Event Port

The iWay Event Port is used to route events generated by external systems. An event port consists of an event schema bound to an event disposition and assigned to an event channel. Channels are configured by transport and you can assign multiple ports to each channel.

Event Port Name:

Event Port Description:

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 Event Adapters Tab*.

Creating an Event Port From the iWay Event Adapters Tab

The following procedures describe how to create an event port from the iWay Event Adapters 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.

Procedure How to Create an Event Port for File

1. Click the *iWay Events* tab.
The iWay Event Adapters window opens.
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 graphic 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-defir

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.

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*.

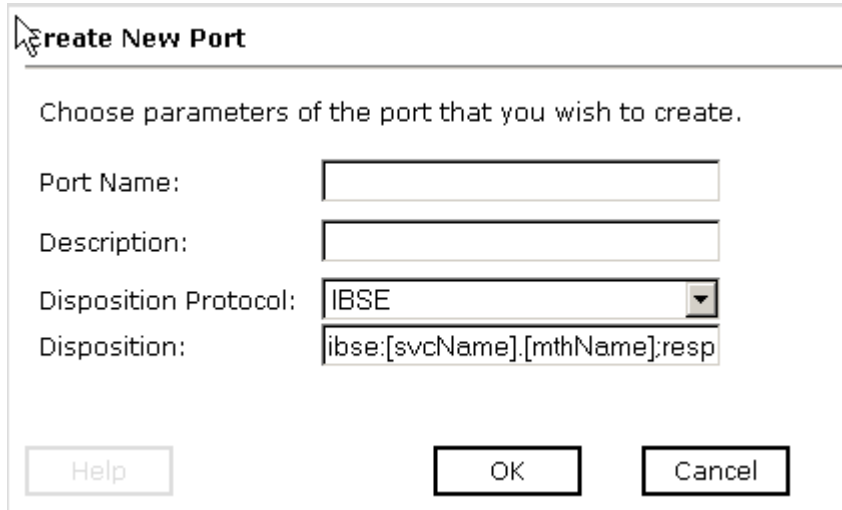
The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the event port you created.

You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-19.

Procedure How to Create an Event Port for iBSE

1. Click the *iWay Events* tab.
The iWay Event Adapters window opens.
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 graphic 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 iBSE, and a disposition has been entered:



The image shows a 'Create New Port' dialog box. It has a title bar with a mouse cursor icon and the text 'Create New Port'. Below the title bar is a horizontal line. The main area contains the text 'Choose parameters of the port that you wish to create.' followed by four input fields: 'Port Name:', 'Description:', 'Disposition Protocol:', and 'Disposition:'. The 'Disposition Protocol:' field is a drop-down menu with 'IBSE' selected. The 'Disposition:' field contains the text 'ibse:[svcName].[mthName]:resp'. At the bottom of the dialog box are three buttons: 'Help', 'OK', and 'Cancel'.

- 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 3-19.

Procedure How to Create an Event Port for MSMQ

1. Click the *iWay Events* tab.
The iWay Event Adapters window opens.
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 graphic 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 MSMQ, and a disposition has been entered:

Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- Type a name and a brief description for the event port.
- From the Disposition Protocol drop-down list, select *MSMQ*.
- 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**.

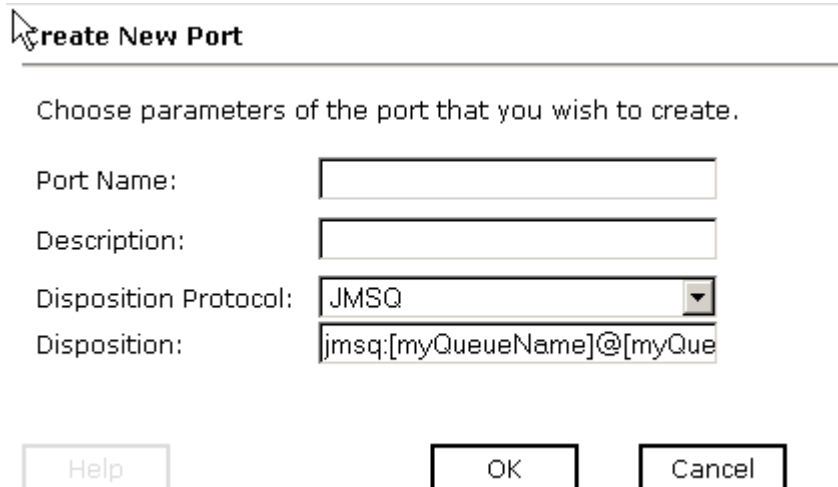
The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the event port you created.

You are now ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-19.

Procedure How to Create an Event Port for JMS Queue

1. Click the *iWay Events* tab.
The iWay Event Adapters window opens.
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 graphic 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 *JMSQ*, and a disposition has been entered:



The image shows a dialog box titled "Create New Port" with a mouse cursor icon next to the title. Below the title bar, there is a text prompt: "Choose parameters of the port that you wish to create." The dialog contains four input fields: "Port Name:" with an empty text box; "Description:" with an empty text box; "Disposition Protocol:" with a dropdown menu showing "JMSQ"; and "Disposition:" with a text box containing "jmsq:[myQueueName]@[myQue". At the bottom of the dialog, there are three buttons: "Help", "OK", and "Cancel".

- 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=;
```

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. The WebLogic connection factory is: <code>javax.jms.QueueConnectionFactory</code>
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. <code>java.naming.provider.url.</code> The URL of the WebLogic Server is <code>t3://host:port</code> where: <code>host</code> Is the machine name where WebLogic Server is installed. <code>port</code> Is the port on which WebLogic Server is listening. The default port, if not changed at installation, is 7001.
jndifactory	Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider. For WebLogic Server, the WebLogic factory is: <code>weblogic.jndi.WLInitialContextFactory.</code>
user	Valid user name required to access a JMS server.

Parameter	Description
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 3-19.

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.

The iWay Event Adapters window opens.

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 Event Port window opens in the right pane.

- a. Type a name for the event port and provide a brief description.
- b. From the Disposition Protocol drop-down list, select *SOAP*.
- c. In the Disposition field, enter a SOAP destination using the following format:

```
soap:wSDL-url;soapaction=action;responseTo=respDest;errorTo=errorDest
```

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

Parameter	Description
wSDL-url	<p>The URL to the WSDL file that is required to create the SOAP message. For example:</p> <p>http://localhost:7001/ibse/IBSEServlet/test/sw2xml2003MQ.ibs?wsdl</p> <p>This value can be found by navigating to the iWay Business Services tab and opening the <i>Service Description</i> link in a new window. The WSDL URL appears in the Address field.</p> <p>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.</p>
action	<p>The method that will be called by the disposition. For example:</p> <p>JDE.mt200Request@test@@</p> <p>where</p> <p>JDE</p> <p>Is the name of the Web service you created using Application Explorer.</p> <p>mt200</p> <p>Is the method being used.</p> <p>test</p> <p>Is the license that is being used by the Web service.</p> <p>This value can be found by navigating to the iWay Business Services tab and opening the <i>Service Description</i> link in a new window. Perform a search for <i>soapAction</i>.</p> <p>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.</p>
respDest	<p>The location to which responses are posted. A predefined port name or another full URL. Optional.</p> <p>A predefined port name or another disposition URL. The URL must be complete, including the protocol.</p>

Parameter	Description
errorDest	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**.

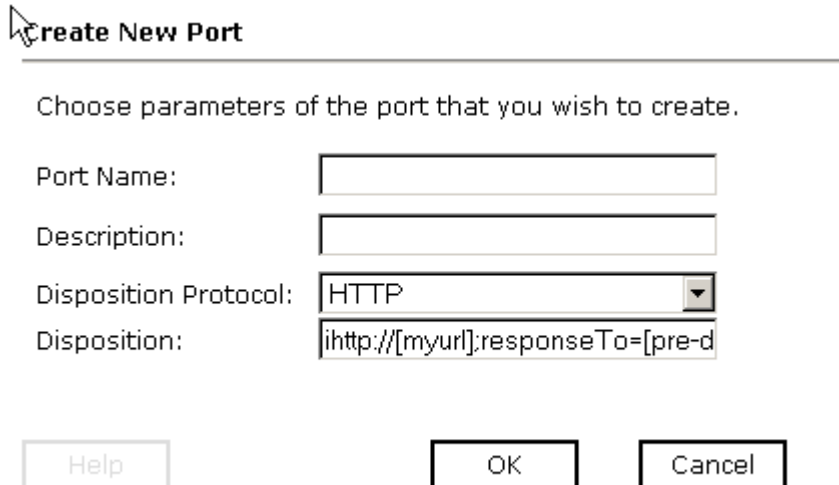
The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the port you created.

You are now ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-19.

Procedure How to Create an Event Port for HTTP

1. Click the *iWay Events* tab.
The iWay Event Adapters window opens.
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 graphic 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 HTTP, and a disposition has been entered:



Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- 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>

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 BEA WebLogic 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**.

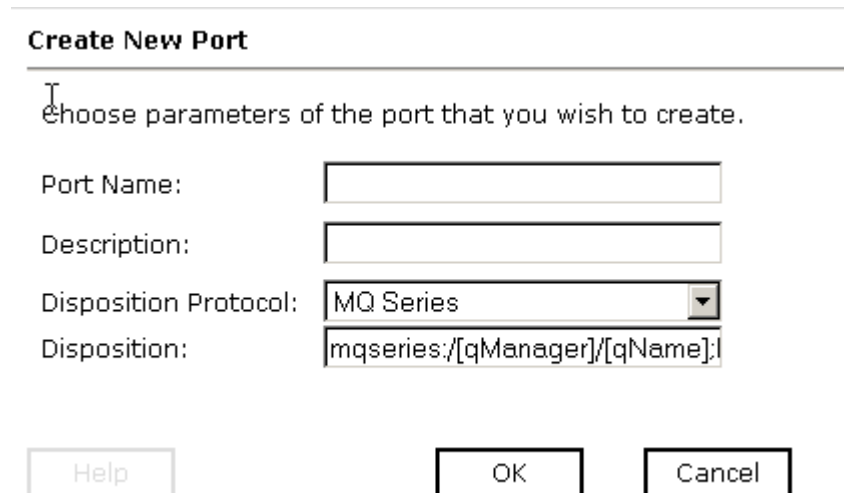
The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the event port you created.

You are now ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-19.

Procedure How to Create an Event Port for MQSeries

1. Click the *iWay Events* tab.
The iWay Event Adapters window opens.
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 graphic 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 MQSeries, and a disposition has been entered:



Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- 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=[channelname];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=
```

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 (for the MQ Client only).
port	Number to connect to an MQ server queue manager (for the MQ client only).
channel	Case-sensitive name of the channel that connects with the remote MQ server queue manager (for the 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 3-19.

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 iWay event. All defined event ports must be associated with a channel. This topic also describes how to edit and delete channels.

Procedure How to Create a Channel

1. Click the *iWay Events* tab.

The iWay Event Adapters window opens. The iWay Adapters that appear in the left pane support events.

2. Expand the *iWay Adapter* 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 graphic 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 you 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:

Creating a Channel

- 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 graphic 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

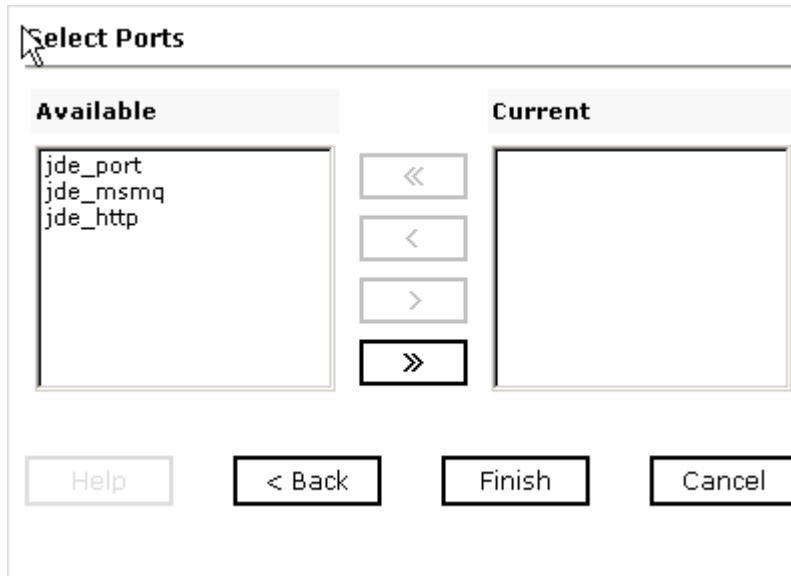
The following table lists and describes the fields required for the creation of a channel.

Parameter	Description
Host	Name or URL of the machine where the database is installed.
Server port	Port on which the host database is listening.
Synchronization Type	<ul style="list-style-type: none"> Select REQUEST_RESPONSE if the event application expects a reply sent back to it. Specify a preemitter. Select REQUEST_ACK when a TCP/IP acknowledgement (ACK) is sent back to the event application. Select REQUEST if the event application does not expect a return.
Is Length Prefix	For J.D. Edwards OneWorld events that send data back that is not in XML format. The TCP/IP event application must prefix the data with a 4-byte binary length field when writing the data to the TCP/IP port.
Is XML	For J.D. Edwards OneWorld events that send data back in XML format. No preparer is required.
Is Keep Alive	Maintains continuous communication between the event transaction and the channel.
User id	Valid user ID for J.D. Edwards OneWorld.
User Password	Password associated with the user ID.
JDE Environment	J.D. Edwards OneWorld environment, for example, DU7333. For more information about this parameter, see your J.D. Edwards OneWorld documentation or ask your OneWorld system administrator.
Application	XMLInterop or the application name in J.D. Edwards OneWorld. Optional.
Server IP address	Name of the server on which J.D. Edwards OneWorld is running. This can be the name of the server, for example, JDEOW, or its IP address, for example, 123.45.67.89.
Server port	Port number on which the server is listening, for example, 6009.

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

7. Click *Next*.

The following graphic shows the Select Ports dialog box that opens, with a list of available ports on the left, a list of current ports on the right, and buttons for moving a single port, selected ports, or all ports from one list to the other:



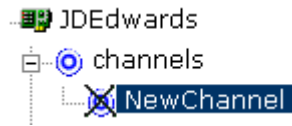
- a. Select an event port from the list of available ports. To select more than one, hold down the *Ctrl* key and click the ports.
 - b. Click the single right (>) arrow button to transfer the selected port(s) to the list of current ports. To transfer all event ports, click the double right (>>) arrow button.
8. Click *Finish*.

The following graphic shows the summary information for the new channel that appears in the right pane. It provides the channel, description, channel status, and ports:

Operations ►

Channel	New Channel for J.D. Edwards
Description	OneWorld
Channel Status	Disconnected
Ports	[jde_http, jde_port, jde_msmq]

The following graphic shows that the channel also appears under the channels node in the left pane. In this example, the name of the channel is NewChannel:



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

The following graphic shows that the channel named NewChannel is now active; the X over the icon has disappeared:



5. 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 OneWorld Event Listener

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

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

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

Configuring the OneWorld Event Listener

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

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

The OneWorld Event listener includes the following components:

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

Creating the iwoevent.cfg File

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

Procedure How to Create the iwoevent.cfg File

1. On the J.D. Edwards OneWorld Server, create an iwoevent.cfg file in the defined directory. For information about the contents of this file, see *Adding Connection Information* on page 3-25.
2. Create an environment variable, IWOEVENT_HOME, to point to the directory containing the iwoevent.cfg file.
 - On Windows, add IWOEVENT_HOME to the system environment variables.
 - On UNIX, add the following command to your start-up script:


```
export IWOEVENT_HOME =/directory_name
```

Adding Connection Information

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

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

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

- Common
- Alias
- Trans

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

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

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

where:

ipaddressordnsname

Is the IP address or DNS name for the server containing the Adapter for J.D. Edwards OneWorld (required).

port

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

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 J.D. Edwards OneWorld transactions to specified servers.

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

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

where:

jdeTransactionName

Is the JDE-defined name for the outbound transaction.

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 Adapter for J.D. Edwards OneWorld 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:\jdedwardsnewworld\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 BEA WebLogic Server.

Port

Is the TCP port waiting for the TCP request.

Example Sending a Request to BEA WebLogic Server (DSN Specified)

The following is a sample request sent to the BEA WebLogic 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>JDES00OUT</proc>
        <data>
          <ediUserId>islywm</ediUserId>
          <ediBatchNumber>100</ediBatchNumber>
          <ediTransactionNumber>100100
            </ediTransactionNumber>
        </data>
      </sp>
    </connection>
  </request>
</eda>
```

Example **Sending a Request to BEA WebLogic 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>JDES00OUT</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 OneWorld system, using the DSN information in the DataSource.cfg file.

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

Example **Sending Requests to J.D. Edwards OneWorld**

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

```
<jdeRequest environment="DV7333" user="JDE" type="trans"
sessionidle="300" session="" pwd="JDE">
    <transaction type="JDES00OUT" 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 OneWorld

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

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

  <transaction type='JDES00UT' 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>
```

CHAPTER 4

Using Web Services Policy-Based Security

Topics:

- Web Services Policy-Based Security
- Configuring Web Services Policy-Based Security

iWay Servlet Application Explorer provides a security feature called Web services policy-based security. This section describes how this feature works and how to configure it.

Web Services Policy-Based Security

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

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

A policy is a set of privileges dealing with the execution of an iWay Business Service (iBS) that can be applied to an existing or new iBS. When you set specific rights or privileges inside a policy, you do not have to recreate privileges for every iBS that has security concerns in common with other iWay Business Services. Instead, you can use one policy for many iWay Business Services.

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

The iBS administrator creates an “instance” of a policy type, names it, associates individual users and/or groups (a collection of users), and then applies that policy to one or more iWay 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 that specific iBS. The policy type that is supported is Resource Execution, which 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, anybody can execute the iBS, until the Resource Execution policy is associated to the iBS. At that time, only those granted execution permission, or users who are not part of a group that was denied execution permissions, have access to the iBS.

Configuring Web Services Policy-Based Security

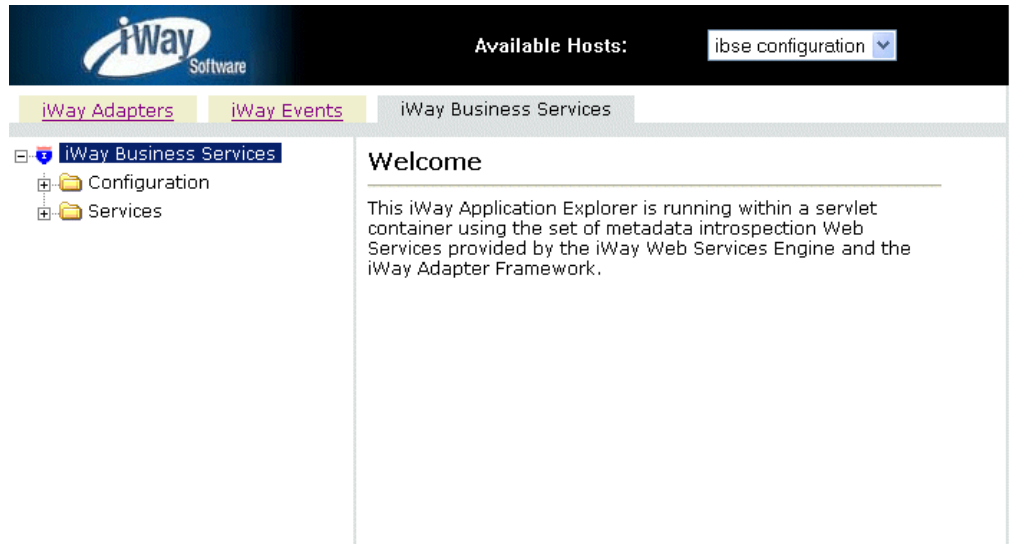
The following procedure describes how to configure iBSE policy-based security.

Procedure How to Create and Associate a User With a Policy

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

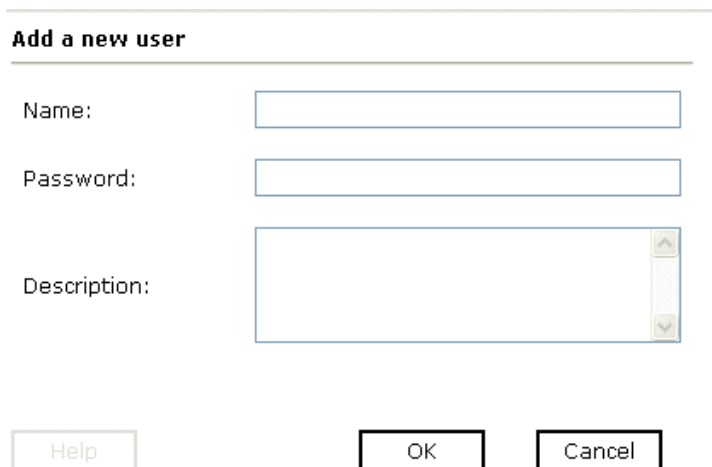
1. Open *iWay Application Explorer*.

The following window opens, showing three tabs: iWay Adapters, iWay Events, and iWay Business Services. The iWay Business Services tab is active and shows a Welcome screen on the right. The graphic shows the node selected and expanded in the left pane.



- a. Select 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 with fields where you enter a user name, password, and description of the user. To escape without making changes, you can click the Cancel button.



The image shows a dialog box titled "Add a new user". It contains three input fields: "Name:", "Password:", and "Description:". The "Description:" field is a text area with a vertical scrollbar. At the bottom of the dialog box, there are three buttons: "Help", "OK", and "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 graphic shows a new user added to the configuration. It includes a definition of users and a user ID and description.



The image shows a configuration screen for "Users". It has a title bar "Operations" with a right-pointing arrow. Below the title bar is a section header "Users" with a user icon. A paragraph of text explains that a user is an object that can be granted or denied permissions to run iWay Business Services. Below the text is a table with two columns: "User Id" and "Description". The table contains one row with the user ID "ibse1" and the description "new user".

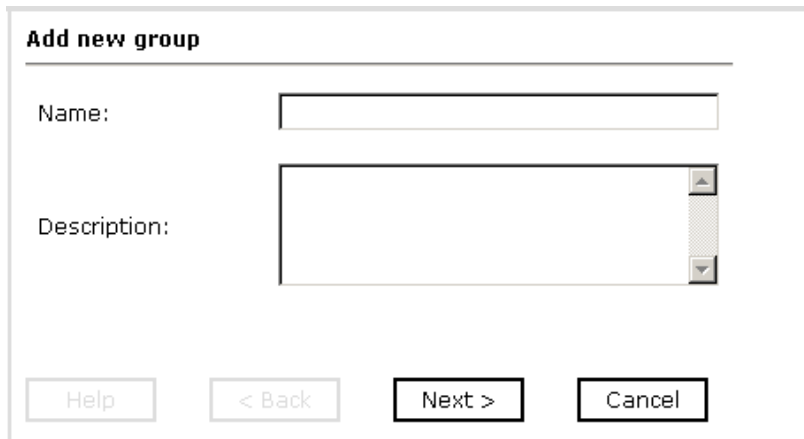
User Id	Description
ibse1	new user

Procedure How to Create a Group to Use With a Policy

To create a group to use with a policy:

1. Open *iWay Application Explorer*.
 - a. Select 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 graphic shows the Add new group pane that opens with fields where you enter a name and a description for the group. To continue, you click the Next button, or to escape without making any changes, you click the Cancel button.



Add new group

Name:

Description:

- 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 graphic shows the Modify Group Membership pane that opens, where you can move users to or from a group by moving them between the Current and Available lists and then clicking the Finish button. To return to the previous screen, you can click the Back button, or to escape you can click the Cancel button.

Modify Group Membership

Current

Available

ibse1

<<

<

>

>>

Help

< Back

Finish


Cancel

You can either highlight a single user in the list of available users and add it 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 graphic shows 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

☐ ibse_group

new group

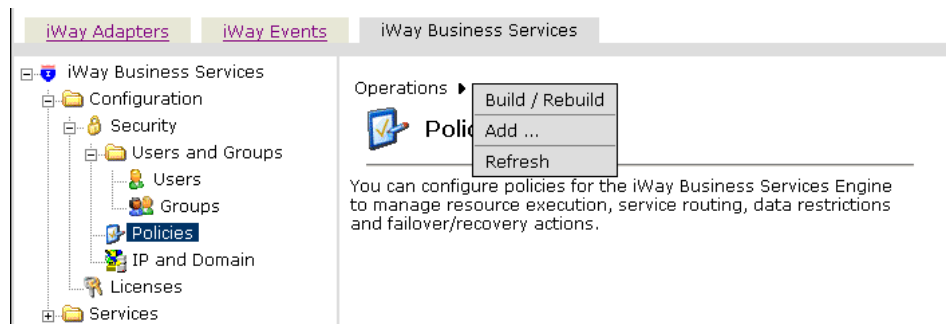
Procedure **How to Create an Execution Policy**

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

To create a group to use with a policy:

1. Open *iWay Servlet Application Explorer*.
 - a. Select the *iWay Business Services* tab.
 - b. Expand the *Configuration* node.
 - c. Select *Policies*.

The following graphic shows the Configuration node expanded in the left pane, and the Operations menu with its options in the right pane.



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

The following graphic shows the Add a new policy pane that opens, with fields for the name, type, and description of the policy. To continue, you click the Next button, or to escape without making changes, you click the Cancel button.

Add a new policy

Name:

Type:

Description:

- 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 graphic shows the Modify policy targets pane that opens with a list of current and available targets and arrow buttons to move targets from one list to the other. The pane includes buttons to return to the previous screen, go to the next screen, or to cancel out of the pane.

Modify policy targets

Current		Available
	<input type="button" value="«"/> <input type="button" value="<"/> <input type="button" value=">"/> <input type="button" value="»"/>	user.ibse1 group.ibse_group

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 graphic shows the Modify policy permissions pane that opens with drop-down lists where you can select to grant or deny permission to members and then click buttons to return to the previous screen, to finish, or to escape from the screen without making changes.

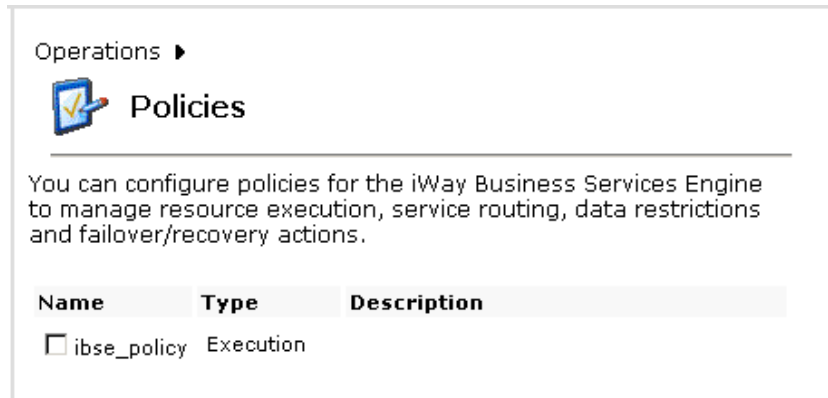
Modify policy permissions

Member Id	Permission
user.ibse1	Deny
group.ibse_group	Deny

You select whether users or groups may execute the iBS.

6. From the Permission drop-down lists, select *Grant* to permit execution or *Deny* to restrict execution.
7. Click *Finish*.

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



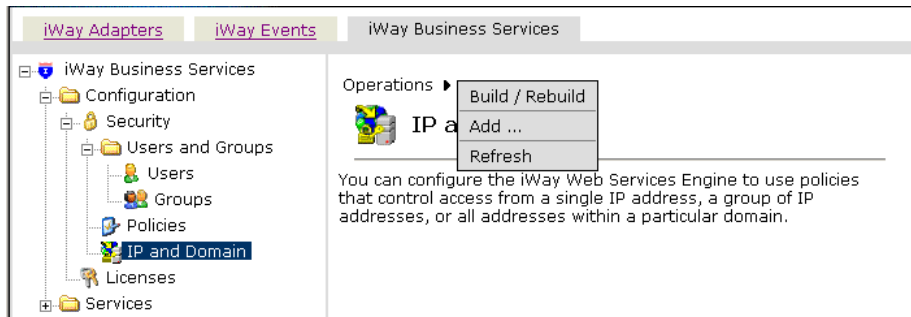
Configuring the IP and Domain Restrictions Policy Type

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

Procedure How to Configure IP and Domain Restrictions

1. Open *iWay 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*.

The following graphic shows the Security node expanded in the left pane, and the Operations menu with its options in the right pane.



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

The following graphic shows the Add a new IP/Domain pane that opens, where you enter information for the IP/Domain. Drop-down lists enable you to select the type and whether to grant access control. You can also add a description. To escape, you can click the Cancel button.

Add a new IP/Domain

IP(Mask)/Domain:

Type:

Access Control:

Description:

- 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 graphic shows the pane that summarizes your configuration, including the domain, whether access is granted or denied, and a description (optional).

Operations ▶

 **IP and Domain**

You can configure the iWay Web 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
<input type="checkbox"/> www.ibi.com	Deny	

CHAPTER 5

Management and Monitoring

Topics:

- Managing and Monitoring Services and Events Using iBSE
- Managing and Monitoring Services and Events Using the IVP
- Setting Engine Log Levels
- Configuring Connection Pool Sizes

When you have created services and events using iWay Application Explorer, you can use managing and monitoring tools provided by iBSE and JCA to gauge the performance of your run-time environment. This section describe how to configure and use these features.

Managing and Monitoring Services and Events Using iBSE

iBSE provides a console to manage and monitor services and events currently in use and 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 Access the Monitoring Console

To access the monitoring console:

1. Ensure that BEA WebLogic Server is started.
2. Enter the following URL in your Web browser:

<http://localhost:7001/ibse/IBSEConfig>

where:

[localhost](#)

Is where your application server is running.

The iBSE Settings page opens:

iWay Business Services Engine System Settings

Configure iWay Business Services Engine (iBSE) Settings.

iBSE Settings: Save

Property Name	Property Value
System	
Language	English
Adapter Lib Directory	C:\Program Files\iWay\SS\lib
Encoding	UTF-8
Debug Level	NONE
Number of Async. Processors	0
Security	
Admin User	iway
Admin Password	****
Policy	<input type="checkbox"/>
Repository	
Repository Type	File System
Repository Url	file://C:\Program Files\iWay\SS\bea\lib
Repository Driver	
Repository User	
Repository Password	
Repository Pooling	<input type="checkbox"/>
More configuration...	

Save

3. Scroll to the bottom of the page and click *More configuration*.

The iBSE Monitoring Settings page opens:

The screenshot shows the 'iWay Business Services Engine System Settings' page. The main heading is 'iBSE Monitoring Settings:'. Below this is a table with two columns: 'Property Name' and 'Property Value'. The table is divided into two sections: 'Monitoring' and 'Auditing'. In the 'Monitoring' section, there are fields for 'Repository Type' (set to 'File System'), 'Repository Url' (set to 'file:///C:/Program Files/iWay/55/ibse/'), 'Repository Driver', 'Repository User', 'Repository Password', and 'Repository Pooling' (unchecked). In the 'Auditing' section, there are fields for 'Store Message' (radio buttons for 'yes' and 'no', with 'no' selected) and 'Max Message Stored' (set to '10,000'). At the bottom of the table are four buttons: 'Save Configuration', 'Save History', 'View Events', and 'View Services'. Below the table is a large 'Start Monitoring' button.

Property Name	Property Value
Monitoring	
Repository Type	File System
Repository Url	file:///C:/Program Files/iWay/55/ibse/
Repository Driver	
Repository User	
Repository Password	
Repository Pooling	<input type="checkbox"/>
Auditing	
Store Message	<input type="radio"/> yes <input checked="" type="radio"/> no
Max Message Stored	10,000

Save Configuration Save History View Events View Services

Start Monitoring

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

<http://localhost:7001/ibse/IBSEStatus>

where:

[localhost](#)

Is where your application server is running.

Procedure How to Configure Monitoring Settings

To configure monitoring settings:

1. Ensure that BEA WebLogic Server is started.
2. Access the monitoring console.

The iBSE Monitoring Settings page opens:

Configure iWay Business Services Engine (iBSE) Settings.

iBSE Monitoring Settings:

Property Name	Property Value
Monitoring	
Repository Type	File System
Repository Url	file:///C:/Program Files/iWay/55/be...
Repository Driver	
Repository User	
Repository Password	
Repository Pooling	<input type="checkbox"/>
Auditing	
Store Message	<input type="radio"/> yes <input checked="" type="radio"/> no
Max Message Stored	10,000

Save Configuration Save History View Events View Services

Start Monitoring

3. Perform the following steps in the Monitoring section:
 - a. Select the type of repository you are using from the Repository Type drop-down list.
 - b. Enter a JDBC URL to connect to the database in the Repository URL field.
 - c. Enter a JDBC Class to connect to the database in the Repository Driver field.
 - d. Enter a user ID and password to access the monitoring repository database.
 - e. Click the *Repository pooling* check box if you want to enable pooling.
4. Perform the following steps in the Auditing section:
 - a. 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.
 - b. 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's resources, consult your system administrator.

5. Click *Save Configuration*.
6. Click *Start Monitoring*.

iBSE begins to monitor all services and events currently in use and store messages, if you selected this option. To stop monitoring, click *Stop Monitoring*.

***Procedure* How to Monitor Services**

To monitor services:

1. Ensure that BEA WebLogic Server is started.
2. Click *Start Monitoring* from the iBSE Monitoring Settings page.
3. Click *View Services*.

The System Level Summary page opens.

iWay Business Services Engine
System Level Summary

Drill down to view iWay Business Services Engine Statistics.

Service Statistics

Web Service Methods

Service:

Method:

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	<input type="text" value="select a correlation id"/>
Failed Invocations	<input type="text" value="select a correlation id"/>

[< home](#)

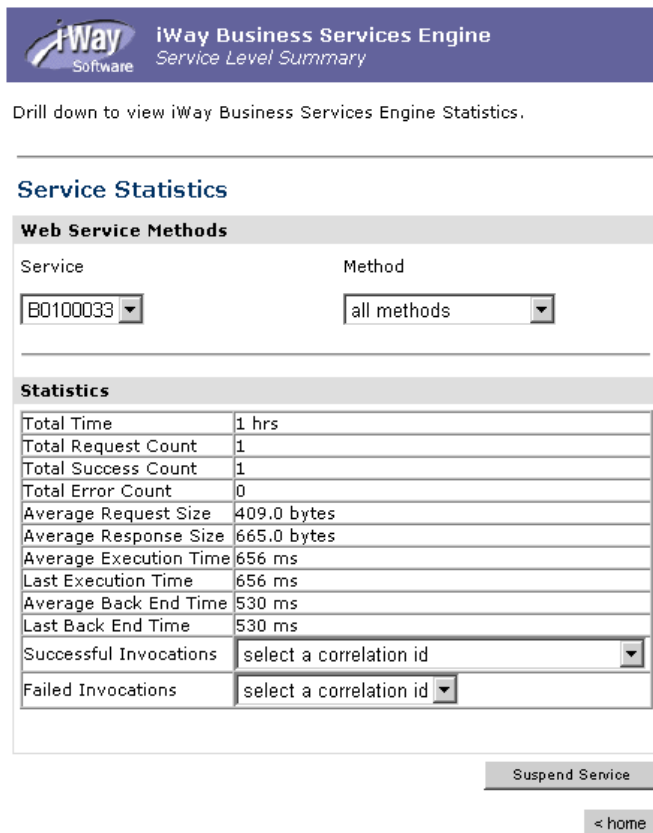
The system level summary provides services statistics at a system level. The following table provides a description of each statistic.

Statistic	Description
Total Time	The total amount of time iBSE is monitoring services. This time starts when you click <i>Start Monitoring</i> from the iBSE Monitoring Settings page.
Total Request Count	The total number of services requests that were made during this monitoring session.
Total Success Count	The total number of successful service executions.

Statistic	Description
Total Error Count	The total number of errors that were encountered.
Average Request Size	The average size of a service request that is available.
Average Response Size	The average size of a service response size that is available.
Average Execution Time	The average execution time for a service.
Last Execution Time	The last execution time for a service.
Average Back End Time	The average back end time.
Last Back End Time	The last back end time.
Successful Invocations	A list of successful services listed by correlation ID. Select a service from the drop-down list to retrieve more information for that service.
Failed Invocations	A list of failed services listed by correlation ID. Select a service from the drop-down list to retrieve more information for that service.

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

The Service Level Summary page opens.



The screenshot shows the 'iWay Business Services Engine Service Level Summary' page. At the top, there is a header with the iWay Software logo and the title 'iWay Business Services Engine Service Level Summary'. Below the header, a message says 'Drill down to view iWay Business Services Engine Statistics.' The main content area is titled 'Service Statistics' and contains a section for 'Web Service Methods'. This section has two dropdown menus: 'Service' (set to 'BD100033') and 'Method' (set to 'all methods'). Below this is a 'Statistics' table with various performance metrics. At the bottom right, there are two buttons: 'Suspend Service' and '< home'.

Web Service Methods	
Service	Method
BD100033	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 list of available methods for that service appears in the Method drop-down list.

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

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

The Method Level Summary page opens.

 **iWay Business Services Engine**
Method Level Summary

Drill down to view iWay Business Services Engine Statistics.

Service Statistics

Web Service Methods


Service	Method
<input type="text" value="B0100033"/>	<input type="text" value="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	<input type="text" value="select a correlation id"/>
Failed Invocations	<input type="text" value="select a correlation id"/>

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

The Invocation Level Statistics page opens.



Statistics for service *B0100033* and method *GetEffectiveAddress*.

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](#)

Information pertaining to the message and client is provided.


7. Click the *Request Message* link to view the XML request document in your Web browser.
You can also view the XML response document for the service.
8. Click *home* to return to the iBSE Monitoring Settings page.

Procedure How to Monitor Events

To monitor events:

1. Ensure that BEA WebLogic Server is started.
2. Click *Start Monitoring* from the iBSE Monitoring Settings page.
3. Click *View Events*.

The System Level Summary page opens.



iWay Business Services Engine
 System Level Event Summary

Drill down to view iWay Business Services Engine Channel Statistics.

Channel Statistics

Channels

Channels

all

Ports

Statistics

Total Event Count	4
Total Success Count	3
Total Error Count	1
Average Event Size	337.0 bytes
Average Event Reply Size	na
Average Delivery Time	1274.0 ms
Last Delivery Time	250 ms
Successful Events	<div style="border: 1px solid #ccc; padding: 2px;">select a correlation id</div>
Failed Events	<div style="border: 1px solid #ccc; padding: 2px;">select a correlation id</div>

< home

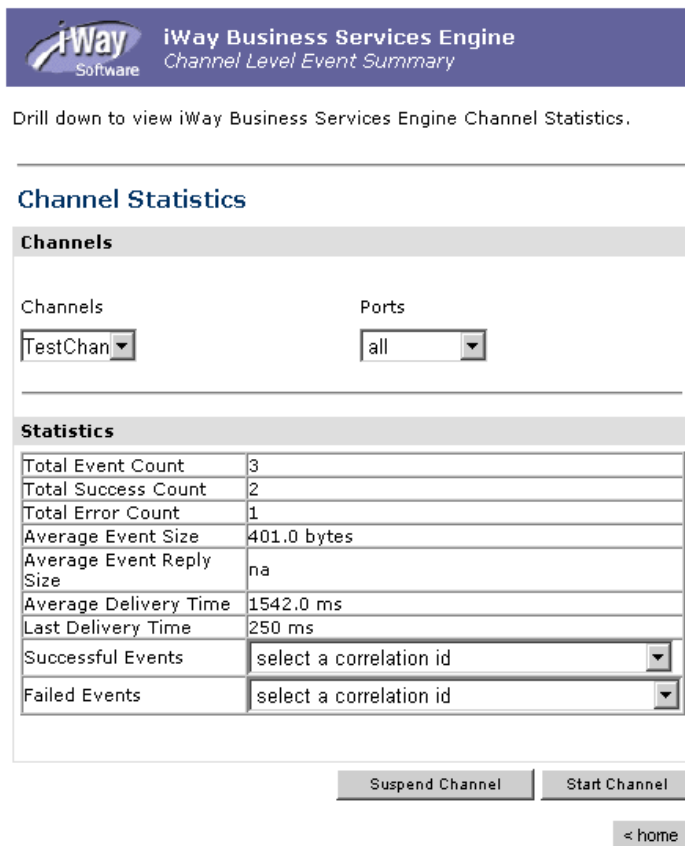
The system level summary provides event statistics at a system level. The following table provides a description of each statistic.

Statistic	Description
Total Event Count	The total number of events.
Total Success Count	The total number of successful event executions.
Total Error Count	The total number of errors that were encountered.
Average Event Size	The average size of an event request that is available.
Average Event Reply Size	The average size of an event response that is available.

Statistic	Description
Average Delivery Time	The average delivery time for an event.
Last Execution Time	The last execution time for an event.
Last Delivery Time	The last delivery time.
Successful Events	A list of successful events listed by correlation ID. Select an event from the drop-down list to retrieve more information for that event.
Failed Events	A list of failed events listed by correlation ID. Select an event from the drop-down list to retrieve more information for that event.

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

The Channel Level Event Summary page opens.

**iWay Business Services Engine**
Channel Level Event Summary

Drill down to view iWay Business Services Engine Channel Statistics.

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	<div>select a correlation id</div>
Failed Events	<div>select a correlation id</div>

Suspend Channel

Start Channel


< home

A list of available ports for that channel appears in the Ports drop-down list.

To stop a channel at any time, click *Suspend Channel*. To start the service, click *Start Channel*.

5. Select a port for the channel from the Ports drop-down list.

The Port Level Event Summary page opens.



iWay Business Services Engine
Port Level Event Summary

Drill down to view iWay Business Services Engine Channel Statistics.

Channel Statistics

Channels

Channels
TestChan ▼

Ports
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 ▼

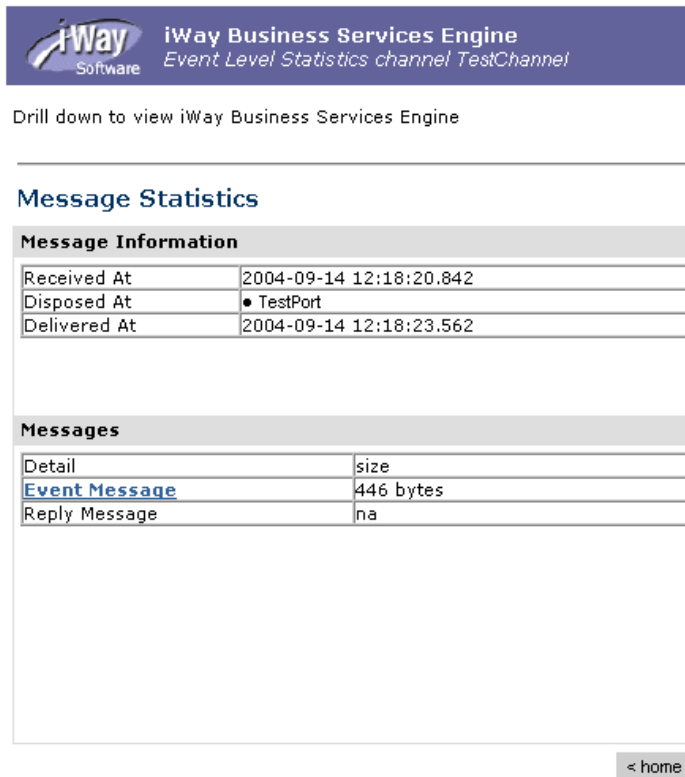
Suspend Channel

Start Channel

[< home](#)

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

The Event Level Statistics page for the channel and port you selected opens.



iWay Business Services Engine
Event Level Statistics channel TestChannel

Drill down to view iWay Business Services Engine

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

Information pertaining to the event message is provided.

7. Click the *Event Message* link to view the XML event document in your Web browser.
8. Click *home* to return to the iBSE Monitoring Settings page.

Managing and Monitoring Services and Events Using the IVP

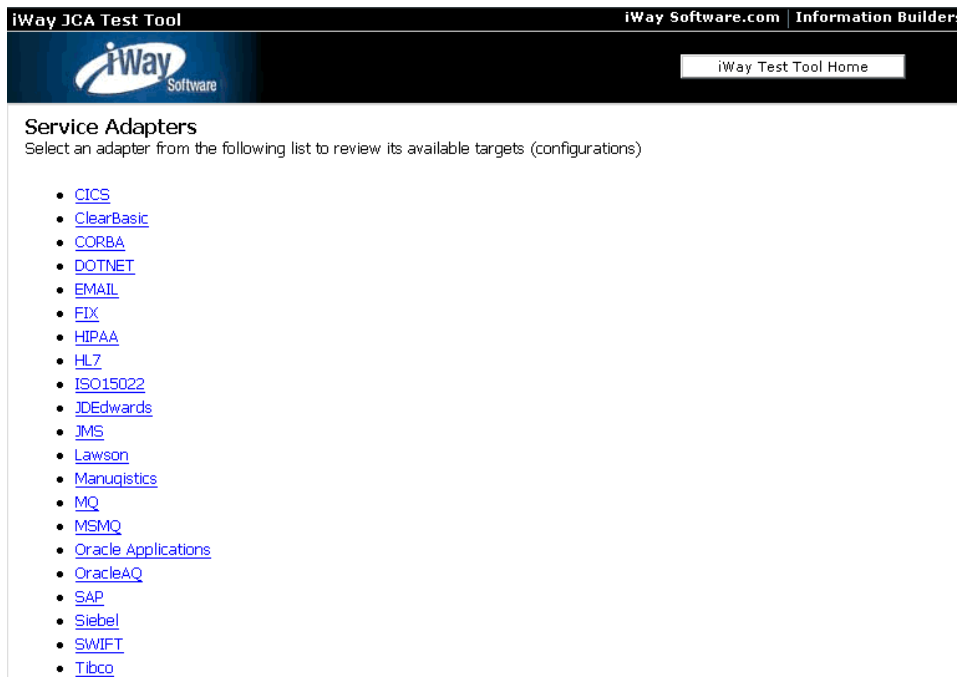
The following topics describe how to test service and event adapters using the iWay JCA Installation Verification Program (IVP).

Procedure How to Test the iWay Service Adapters

To test the iWay service adapters using the IVP:

1. To ensure that the targets you configured in iWay Application Explorer appear in the IVP, click *Refresh Manage Connection Factory*.
2. To display the available adapters, click the *Service adapters* link.

The following window, showing the list of deployed service adapters, opens.



3. Select the adapter that you want to test.

The adapter displays all of the targets currently configured in the iWay repository for that adapter.

The following window shows that there is one target, JDEConnection, configured for the iWay Adapter for J.D. Edwards OneWorld.

Targets for JDEDWARDS

- [JDEConnection](#)

4. Click the desired target, for example, *JDEConnection*.

The following pane, showing an input area in which you can provide XML code with which to test the adapter, opens.

Request for JDEDWARDS target JDEConnection

Enter the data for this interaction. The configured user/password will be used if the User name is not provided.

User:

Password:

Input Doc:

5. Enter a username and password to connect to J.D. Edwards OneWorld.
6. In the input area, enter a request document built from the iWay request schema.
7. Click *Send*.

A response is returned from J.D. Edwards OneWorld.

Testing the iWay Event Adapters Using the IVP

The iWay JCA Installation Verification Program (IVP) enables you to start and stop iWay event channels.

The tool also enables you to monitor events and provides statistics on channels.

Procedure How to Test the iWay Event Adapters

To test the iWay event adapters using the IVP:

1. Click *Refresh Manage Connection Factory*.
2. To display the available adapters, click the *Events adapter* link.
3. Select the adapter that you want to monitor, for example, JDEdwards.

The tool displays the channels that you already configured.

Channels for JDEWARDS

- [File1 start stop](#)
- [HTTPChann start stop](#)
- [TCP1 start stop](#)

4. Click the *start* hyperlink to start the channel.

Status for JDEWARDS channel File1 Current Statistics

Active:	true
Init. time:	Tue Sep 14 16:09:00 EDT 2004
Activate time:	Tue Sep 14 16:09:00 EDT 2004
Elapsed time:	1 min(s) and 20 sec(s)
Service count:	0
Error count:	0
Event count:	1
Avg. service time (msec):	0
Last service time (msec):	0

Statistics for the event channel are returned, including:

- The status of the channel.
 - The time the channel was initialized.
 - The number of events.
 - The event response time.
5. To stop the channel, click the *stop* hyperlink.

Monitoring Services

The following section describes how to use the iWay JCA Installation Verification Program (IVP) in Managed mode and monitor services in BEA WebLogic.

Procedure How to Use iWay JCA IVP in Managed Mode.

To use iWay JCA IVP in managed mode:

1. Open the *web.xml* file in a text editor.

It is located in the following directory:

```
<installDir>\bea\iwejcaivp\WEB-INF\web.xml
```

where:

```
<installDir>
```

Is the location of your iWay 5.5 installation.

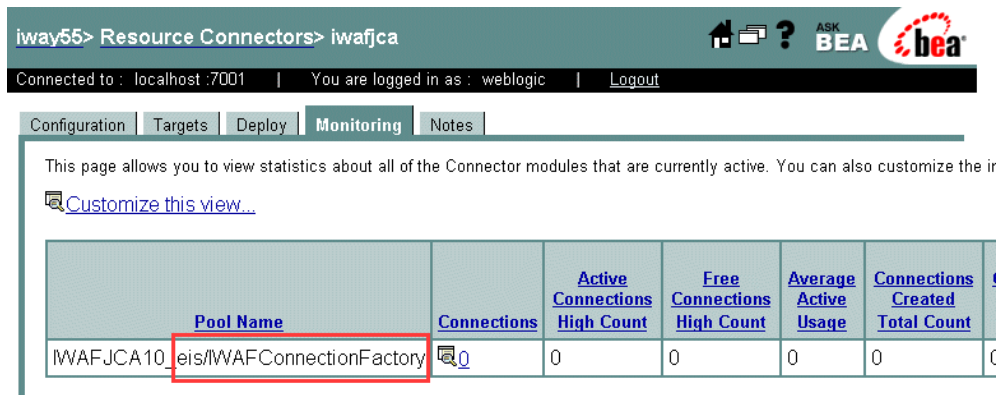
2. Locate the following lines:

```
<context-param><param-name>iway.jndi</param-name><param-value></param-value><description>JNDI name for the IWAF JCA Resource Adapter. If not provided, the application will create a new one based on iway.home, iway.config and iway.loglevel.</description></context-param>
```

3. Enter the path to the JCA module for the *iway.jndi* parameter, for example:

```
<param-value>eis/IWAFConnectionFactory</param-value>
```

You can find this value by browsing to the Resource Connectors section in BEA WebLogic and checking the Pool Name for the JCA connector module. For example:



The screenshot shows the BEA WebLogic iWay55 Resource Connectors monitoring page. The page has a navigation bar with tabs for Configuration, Targets, Deploy, Monitoring, and Notes. The Monitoring tab is selected. Below the navigation bar, there is a message: "This page allows you to view statistics about all of the Connector modules that are currently active. You can also customize the ir". Below this message is a link "Customize this view...". Below the link is a table with the following columns: Pool Name, Connections, Active Connections High Count, Free Connections High Count, Average Active Usage, and Connections Created Total Count. The table has one row with the following data: Pool Name: IWAFFJCA10, Connections: 0, Active Connections High Count: 0, Free Connections High Count: 0, Average Active Usage: 0, and Connections Created Total Count: 0. The 'Pool Name' column is highlighted with a red box, and the value 'eis/IWAFConnectionFactory' is visible in the box.

Pool Name	Connections	Active Connections High Count	Free Connections High Count	Average Active Usage	Connections Created Total Count
IWAFFJCA10 eis/IWAFConnectionFactory	0	0	0	0	0

4. Restart WebLogic Server or redeploy the JCA connector module.
5. Open a browser to:

<http://hostname:port/iwjcaivp>

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. The port for the default domain is 7001.

The iWay JCA Test Tool window opens and provides links for viewing iWay Service or Event adapters. Notice that it is now running in managed mode.

iWay JCA Test Tool

iWay Software.com | Information Builders

iWay Test Tool Home

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

Configuration

- Running in MANAGED mode.
- iway.jndi :eis/IWAFConnectionFactory:

Adapters

- [Service adapters](#)
- [Event adapters](#)

iWay JCA Test Tool
 Send comments to feedback@iwaysoftware.com
 Copyright © 2003-2004, iWay Software/Information Builders, Inc.
 All Rights Reserved.

6. Test a service you have created for an iWay Adapter using Application Explorer.
7. Return to the Resource Connectors section in BEA WebLogic.

ay55> Resource Connectors> iwafjca

Connected to: localhost:7001 | You are logged in as: weblogic | [Logout](#)

[Configuration](#) | [Targets](#) | [Deploy](#) | **[Monitoring](#)** | [Notes](#)

This page allows you to view statistics about all of the Connector modules that are currently active. You can also customize the information displayed.

[Customize this view...](#)

Pool Name	Connections	Active Connections High Count	Free Connections High Count	Average Active Usage	Connections Created Total Count	Connections Max Total Count
IWAFJCA10_eis/IWAFConnectionFactory	2	1	2	0	2	2

Monitoring statistics pertaining to the services you have executed are now available.

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 for BEA WebLogic* documentation.

Procedure How to Enable Tracing for Servlet iBSE

To enable tracing for Servlet iBSE:

1. Open the Servlet iBSE configuration page:

<http://hostname:port/ibse/IBSEConfig>

where:

[hostname](#)

Is the hostname of the application server machine.

[port](#)

Is the port for the domain you are using for iWay. The port for the default domain is 7001.

For example:

<http://localhost:7001/ibse/IBSEConfig>

2. In the top *System* area, specify the level of tracing from the *Debug* drop-down list.
3. Click *Save*.

The default location for the trace information on Windows is:

<C:\Program Files\bea\ibse\ibselogs>

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 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. Also, be sure to review logs generated by your application server.

Leave the remainder of this file unchanged.

3. Save the file and exit the editor.
4. Redeploy the connector.

Configuring Connection Pool Sizes

The following section describes how to configure connection pool sizes using JCA.

Procedure How to Configure Connection Pool Sizes

To configure connection pool sizes:

1. Open the extracted weblogic-ra.xml file in a text editor.
2. Locate and change the following setting:

pool-params. The JCA Resource Connector has an initial capacity value of 0 by default, and cannot be changed. The maximum capacity value is 10 by default and can be changed to a higher value.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE weblogic-connection-factory-dd (View Source for full
doctype...)>
- <weblogic-connection-factory-dd>
  <connection-factory-name>IWAJCA</connection-factory-name>
  <jndi-name>eis/IWAJConnectionFactory</jndi-name>
  - <pool-params>
    <initial-capacity>0</initial-capacity>
    <max-capacity>10</max-capacity>
    <capacity-increment>1</capacity-increment>
    <shrinking-enabled>false</shrinking-enabled>
    <shrink-period-minutes>200</shrink-period-minutes>
  </pool-params>
  <security-principal-map />
</weblogic-connection-factory-dd>
```

3. Save the file and exit the editor.
4. Redeploy the connector.

APPENDIX A

Using iWay Application Explorer in BEA WebLogic Workshop

Topics:

- Overview
- Starting iWay Application Explorer in WebLogic Workshop
- Creating a New Configuration
- Defining a Target
- Disconnecting From or Deleting a Connection
- Creating an XML Schema
- Creating a Business Service
- Understanding iWay Event Functionality
- Creating, Editing, and Deleting a Port
- Creating, Editing, and Deleting a Channel
- Deploying iWay Components in a Clustered BEA WebLogic Environment
- Adding a Control for an iWay Resource in BEA WebLogic Workshop
- Extensible CCI Control

This section describes the use of the iWay Application Explorer as implemented in the BEA WebLogic Workshop. The Application Explorer deployed in the WebLogic Workshop is functionally similar to the servlet Application Explorer.

Although this section describes the Java Swing implementation of Application Explorer, other implementations provide the same functionality using similar graphical user interfaces. For more information, see Chapter 2, *Creating XML Schemas and Business Services for J.D. Edwards OneWorld* and Chapter 3, *Listening for Database Events*.

Overview

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

External applications that access OneWorld through the iWay Application Adapter for J.D. Edwards OneWorld use either XML schemas or Web services to pass data between the external application and the adapter. The following topics describe how to use Application Explorer in BEA WebLogic Workshop to create XML schemas and Web services for the J.D Edwards Master Business Functions (MBFs) 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 OneWorld utility called GenJava. You use Application Explorer to generate schemas against OneWorld GenJava wrappers.

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

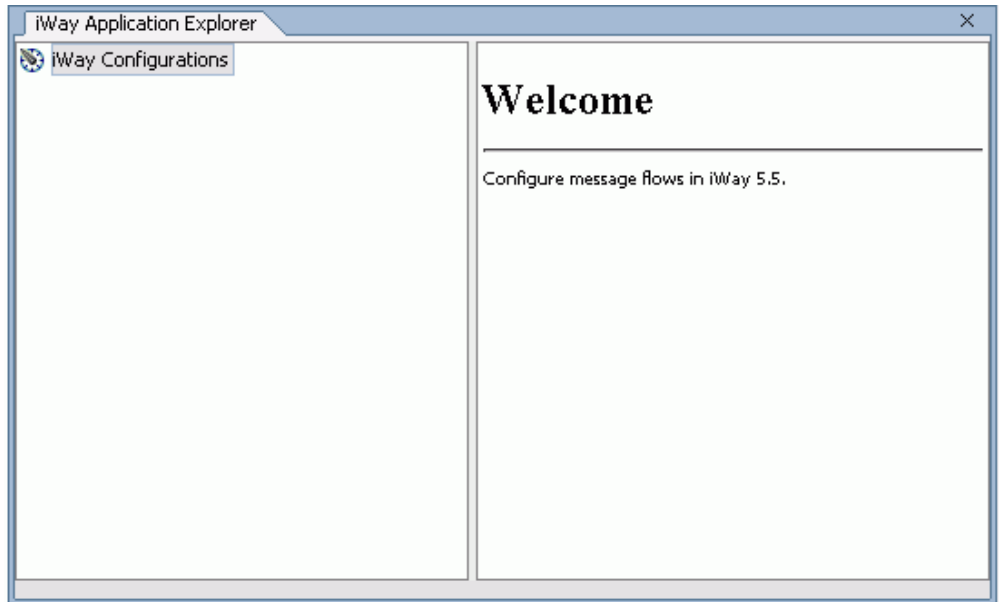
Starting iWay Application Explorer in WebLogic Workshop

You can use Application Explorer with a JCA or an iBSE configuration. If you want to use Application Explorer with a JCA configuration, you must use the servlet version of Application Explorer, which runs outside of WebLogic Workshop. For more information about the servlet version, see Chapter 2, *Creating XML Schemas and Business Services for J.D. Edwards OneWorld* and Chapter 3, *Listening for Database Events*.

Procedure How to Start Application Explorer in WebLogic Workshop

1. Start WebLogic Workshop.
2. Ensure that the server on which Application Explorer is deployed is started. If it is not started, select *WebLogic Server* from the Tools menu and then click *Start WebLogic Server*.
3. From the View menu, select *Windows* and then click *iWay Application Explorer*.

Application Explorer opens as a frame within the Workshop:

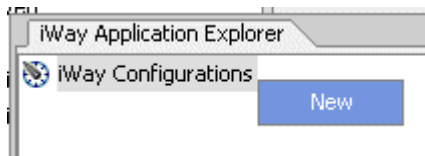


Creating a New Configuration

Before you can start using Application Explorer, you must define a new configuration for iBSE or JCA.

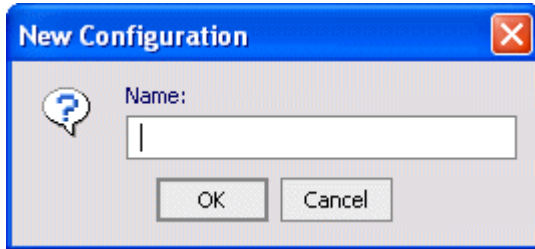
Procedure How to Create a New Configuration for iBSE or JCA

To create a new configuration:



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

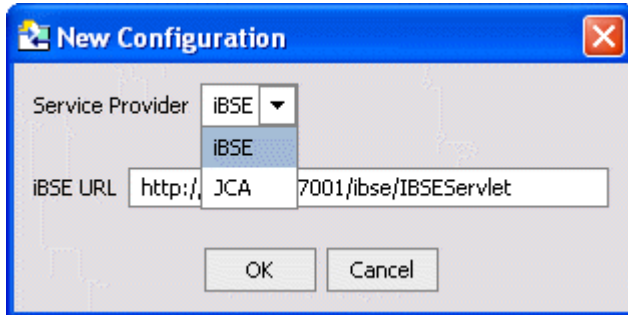
The New Configuration dialog box opens.



2. Type the name of the new configuration and click *OK*.

Note: If you are creating a new JCA configuration, type *base* in the name field. You must use this value if you are pointing to the default iWay configuration.

The following dialog box opens.



3. From the Service Provider drop-down list, select *iBSE* or *JCA*.

- If you select *iBSE*, type the URL for *iBSE*, for example,

<http://localhost:7001/ibse/IBSEServlet>

where:

[localhost](#)

Is where your application server is running.

- If you select *JCA*, enter the full path to the directory where iWay 5.5 is installed, for example,

[C:\Program Files\iWay55](#)

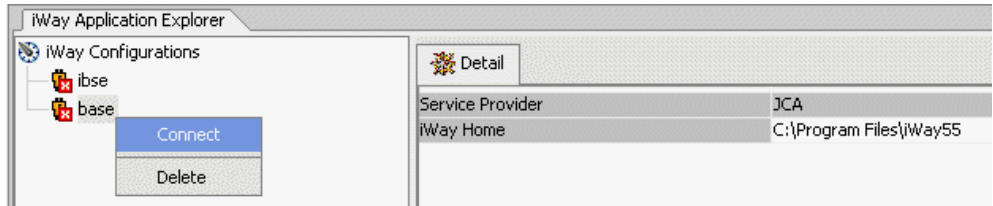
where:

[iWay55](#)

Is the full path to your iWay installation.

A node representing the new configuration appears under the iWay Configurations node. The right pane provides details of the configuration you created.

After you add your configuration, you must connect to it.

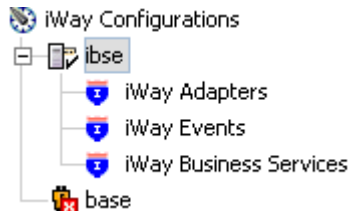


4. Right-click the configuration to which you want to connect, for example, base, and select *Connect*.

The iWay Adapters and iWay Events nodes appear.



When you connect to iBSE, the iWay Adapters, iWay Events, and iWay Business Services nodes appear.



5. To display the service and event adapters that are installed, expand each node.

Defining a Target

To browse the available Master Business Functions, 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.

Connecting to J.D. Edwards OneWorld

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

Procedure **How to Define a New Target to J.D. Edwards OneWorld**

1. Expand the *iWay Service Adapters* node.

The application systems supported by Application Explorer display as nodes based on the iWay adapters installed.

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

The Add Target dialog box opens:

The image shows a Windows-style dialog box titled "Add Target". It has a standard title bar with a minimize button, a maximize button, and a close button (X). The dialog contains three input fields: "Name:" with an empty text box, "Description:" with an empty text box, and "Type:" with a drop-down menu. The drop-down menu is currently open, showing "JDE One World" as the selected option. At the bottom of the dialog are two buttons: "OK" and "Cancel".

- a. Type a name (for example, JDEConnection) and a brief description for the new target.
- b. From the Type drop-down list, select the type of target (for example, JDE OneWorld).

4. Click OK. The Repository dialog box opens:



5. Type the path to the GenJava repository.

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

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

6. Click the *Logon* tab. The Logon dialog box opens:



Fields marked with * are required.

7. Type values for the following parameters. Fields with an asterisk are required.

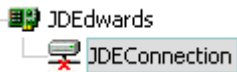
Target Parameter	Description
User id*	Valid user ID for J.D. Edwards OneWorld.
User password*	Password associated with the user ID.
JDE Environment*	J.D. Edwards OneWorld environment, for example, DU7333. For more information about this parameter, see your J.D. Edwards OneWorld documentation or ask your OneWorld system administrator.
Application	XMLInterop or the application name in J.D. Edwards OneWorld. Optional.
Server IP address*	Name of the server on which J.D. Edwards OneWorld is running. This can be the name of the server, for example, JDEOW, or its IP address, for example, 123.45.67.89.
Server port*	Port number on which the server is listening, for example, 6009.

8. Click OK.

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

Procedure How to Connect to a Defined J.D. Edwards OneWorld Target

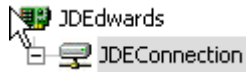
1. Expand the *iWay Service Adapters* node.
2. Expand the *JDEdwards* node.
3. Click the target name (for example, JDEConnection) under the JDEdwards node:



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

4. Verify your connection parameters. If required, provide the password.
5. Right-click the target name and select *Connect*.

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



Disconnecting From or Deleting a Connection

To manage J.D. Edwards OneWorld 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.

- Delete a connection that is no longer required.

Procedure How to Disconnect From a Connection to J.D. Edwards OneWorld

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

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

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



Procedure How to Delete a Connection to J.D. Edwards OneWorld

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

The node disappears from the list of available connections.

Creating an XML Schema

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

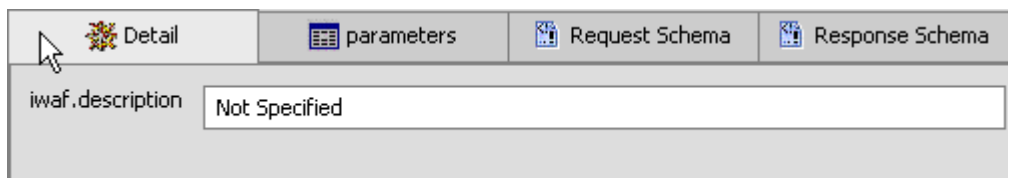
Creating a Request and a Response Schema

The following procedure explains how to create request and response schemas for a J.D. Edwards OneWorld Master Business Function using Application Explorer.

Procedure How to Create a Request Schema and a Response Schema

1. If you are not connected to a J.D. Edwards OneWorld target, connect to one, as described in *How to Connect to a Defined J.D. Edwards OneWorld Target* on page A-9.
2. Expand the *Services* node.
3. Expand the node of the Master Business Function (MBF) for which you want to create the schema.
4. Expand and then select the node beneath the MBF.

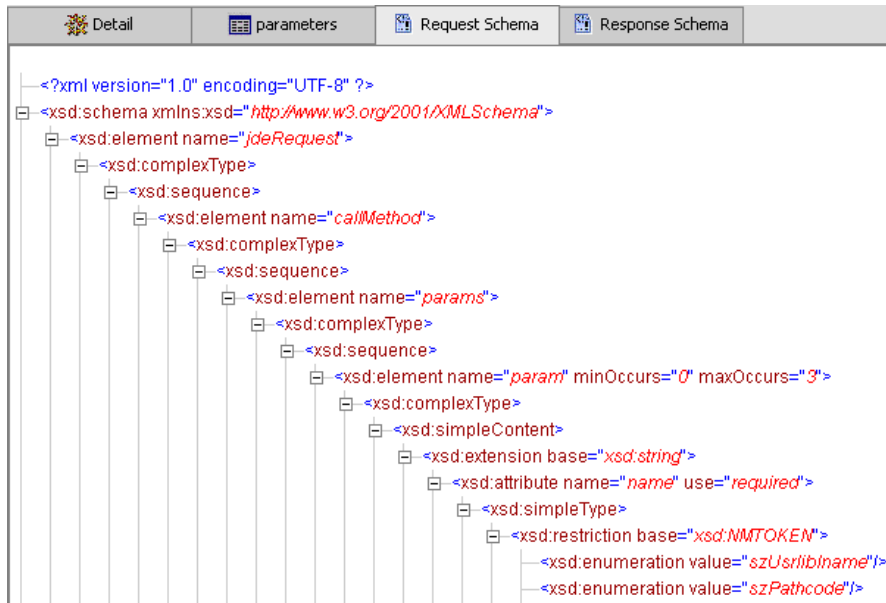
The following screen appears in the right pane:



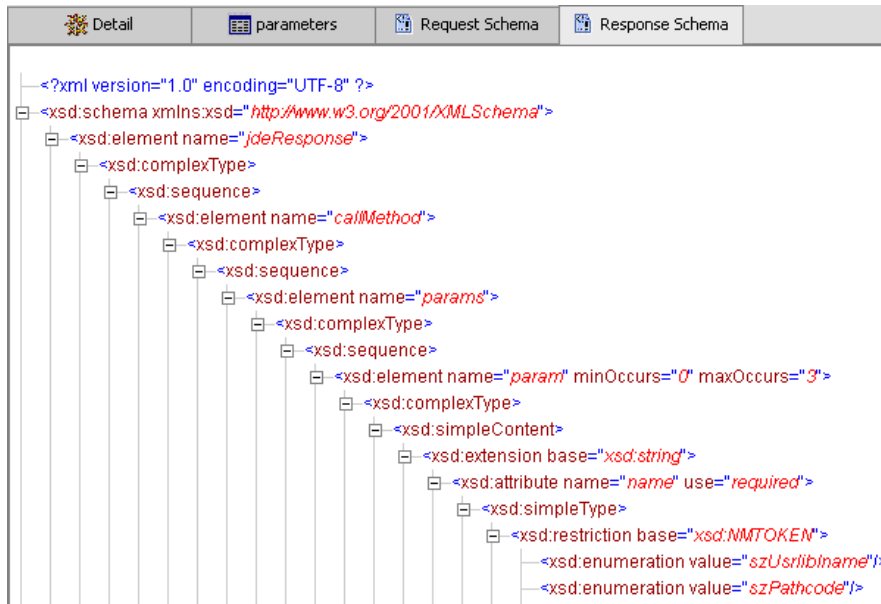
5. Click the *parameters* tab to view the parameter information:

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

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



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

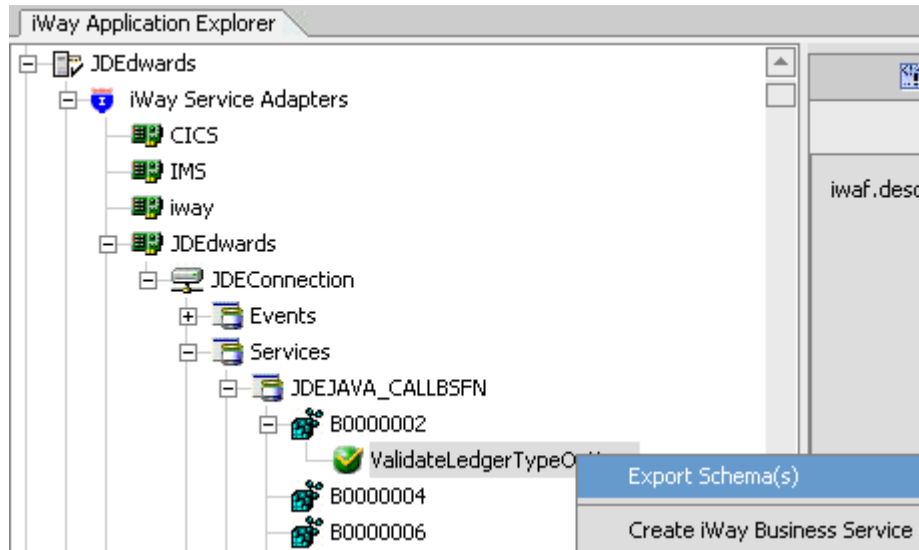


For information about exporting schemas, see *How to Export a Schema* on page A-12.

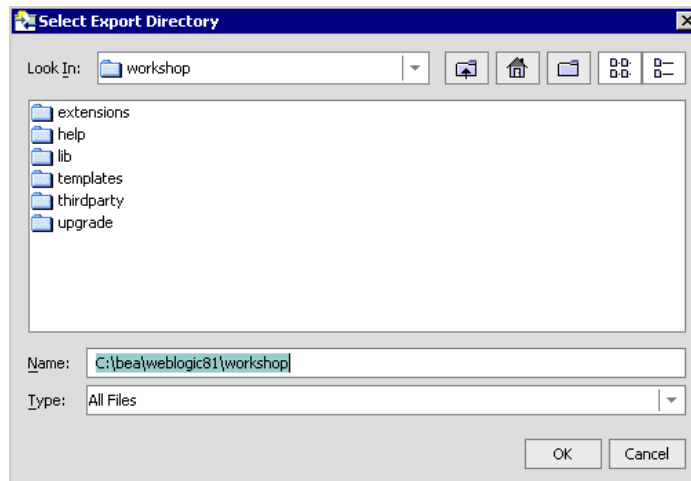
Procedure How to Export a Schema

1. If you are not connected to a J.D. Edwards OneWorld target, connect to one, as described in *How to Connect to a Defined J.D. Edwards OneWorld Target* on page A-9.
2. Expand the *Services* node.
3. Expand the node of the Master Business Function (MBF) for which you want to create the schema.
4. Expand and then select the node beneath the MBF.

5. Right-click the node from which you want to export a schema, and select *Export Schema(s)*:



6. The Select Export Directory dialog box opens:



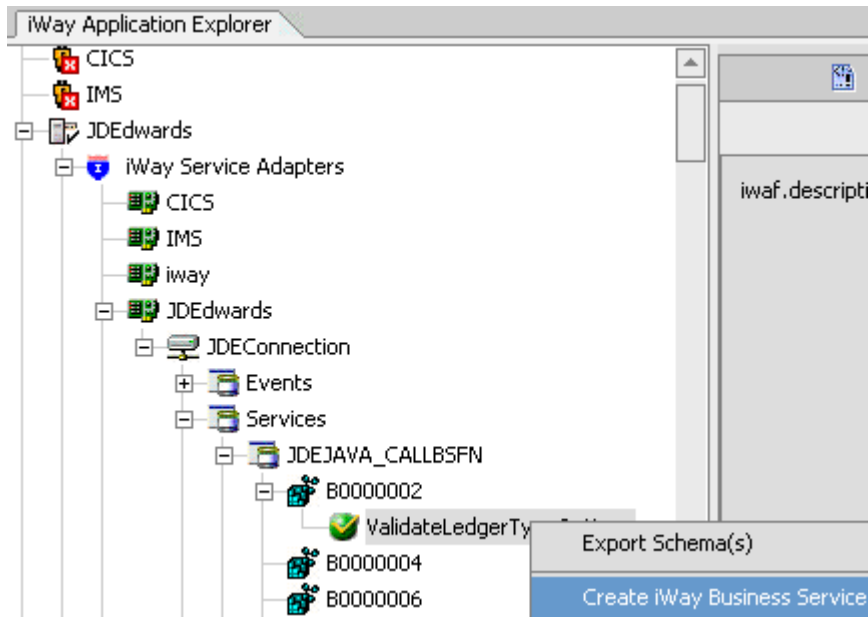
7. Select the directory to which you want to save the schema and click *OK*.

Creating a Business Service

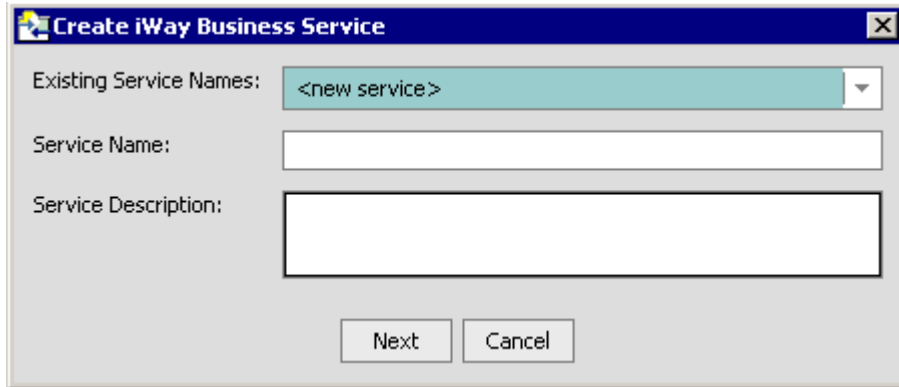
You can generate a business service (also known as a Web service). You can explore the business function repository and generate business services for the functions you want to use with the adapter.

Procedure How to Create a Business Service

1. If you are not connected to a J.D. Edwards OneWorld target, connect to one, as described in *How to Connect to a Defined J.D. Edwards OneWorld Target* on page A-9.
2. Expand the *Services* node.
3. Expand the node of the Master Business Function (MBF) for which you want to create a business service.
4. Expand and then select the node beneath the MBF.
5. Right-click the node from which you want to create a business service, and select *Create iWay Business Service*:



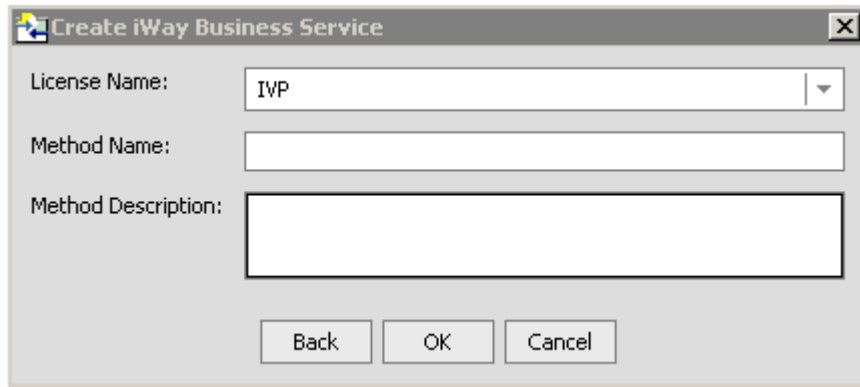
6. The service information dialog box opens:



The dialog box titled "Create iWay Business Service" has a blue title bar with a close button. It contains three input fields: "Existing Service Names:" with a dropdown menu showing "<new service>", "Service Name:" with a text box, and "Service Description:" with a larger text area. At the bottom are "Next" and "Cancel" buttons.

- a. Select either a new service or an existing service from the Existing Service Names drop-down list.
 - b. Type a service name if you are creating a new service. This name identifies the Web service in the list of services under the iWay Business Services node.
 - c. Type a description for the service.
7. Click *Next*.

The license and method dialog box opens:



The dialog box titled "Create iWay Business Service" has a gray title bar with a close button. It contains three input fields: "License Name:" with a dropdown menu showing "IVP", "Method Name:" with a text box, and "Method Description:" with a larger text area. At the bottom are "Back", "OK", and "Cancel" buttons.

- a. In the License field, select one or more license codes to assign to the Web Service. To select more than one, hold down the *Ctrl* key and click the licenses.
- b. In the Method Name field, type a descriptive name for the method.
- c. In the Description field, type a brief description for the method.

8. Click OK.

Application Explorer expands the iWay Business Services node in the left pane to show the newly created business service and presents a test input area in the right pane.

Testing a Business Service

After a business service is created, use the test tool to ensure that it functions properly.

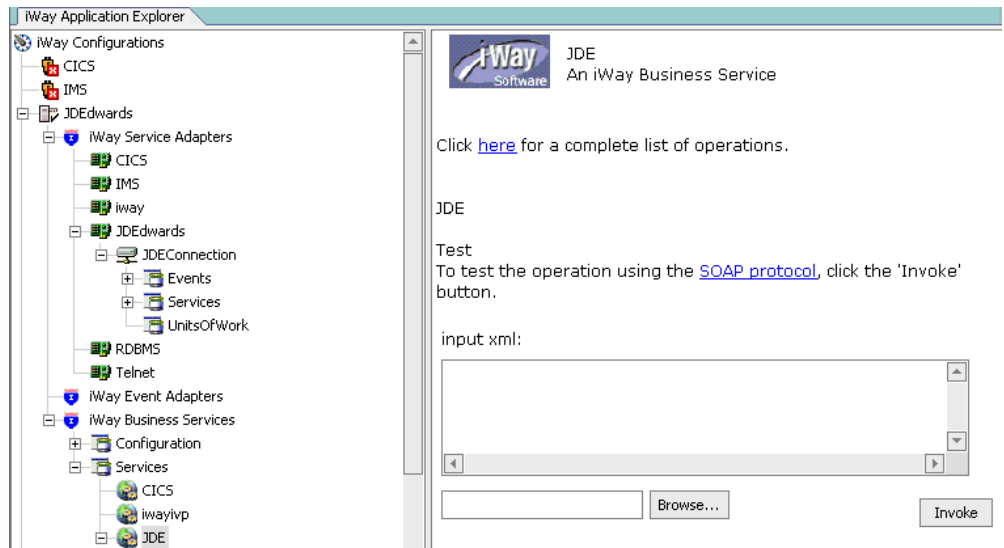
Procedure How to Test the Business Service

1. If you are not in the iWay Business Services node of Application Explorer, click the node 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 (for example, JDE).

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

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

The test option appears in the right pane:



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.

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 such as BEA WebLogic Server.

Procedure How to Generate WSDL From a Web Service

1. Expand the *iWay Business Services* node.
2. Expand the *Services* node to display the Web service for which you want to generate WSDL.
3. Right-click the Web service and select *Export WSDL*.

The Save dialog box opens.

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

Note: The file extension must be *.wsdl*.

5. Click *Save*.

Credential Mapping

For each SOAP request that is received, iBSE checks to see if a user name and password is included in the SOAP header. If a user name and password is available, iBSE acquires this information and replaces the values retrieved from the repository when pushing the request to the iWay Adapter.

Understanding iWay 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 Application Explorer. To create an iWay event, you must create a port and a channel.

- Port

A port associates a particular business object exposed by the iWay 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 BEA WebLogic Server. For more information, see *Creating, Editing, and Deleting a Port* on page A-18.

- 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 iWay Adapter. For more information, see *Creating, Editing, and Deleting a Channel* on page A-30.

Creating, Editing, and Deleting a 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 Event Adapters Tab

The following procedures describe how to create an event port from the iWay Event Adapters window for various dispositions using Application Explorer. The following dispositions are available when using Application Explorer in conjunction with an iBSE deployment:

- File
- iBSE
- MSMQ
- JMSQ
- SOAP
- HTTP
- MQ Series

Note: You can switch between an iBSE and a JCA deployment using the servlet Application Explorer. For more information, see *Creating an Event Port* in Chapter 3, *Listening for Database Events*.

Procedure How to Create an Event Port for File

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

The Add Port dialog box opens:

The screenshot shows a standard Windows-style dialog box titled "Add Port". It has a close button (X) in the top right corner. The dialog contains four labeled input areas: "Name:" with a single-line text box; "Description:" with a multi-line text box; "Protocol:" with a drop-down menu currently showing "FILE"; and "URL:" with a multi-line text box containing the placeholder text `file:///location];errorTo=[pre-defined port name or another disposition url]`. At the bottom of the dialog are two buttons: "OK" and "Cancel".

- a. Type a name and a brief description for the event port.
- b. From the Protocol drop-down list, select *File*.
- c. In the URL field, type a File destination using the following format:


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

The following table defines the parameters for the File disposition.

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.

4. Click OK.

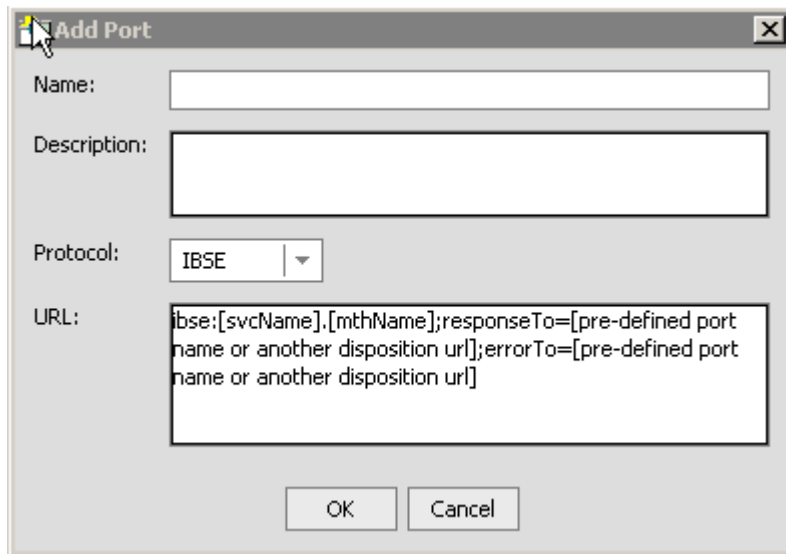
The port appears under the ports node in the left pane. To review the port settings, select the port name. A table summarizing the port settings appears in the right pane.

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting a Channel* on page A-30.

Procedure How to Create an Event Port for iBSE

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

The Add Port dialog box opens:

The image shows a Windows-style dialog box titled "Add Port". It has a standard title bar with a minimize button, a maximize button, and a close button. The dialog contains four labeled fields: "Name:" with a single-line text box; "Description:" with a multi-line text box; "Protocol:" with a drop-down menu currently showing "IBSE"; and "URL:" with a multi-line text box containing the text "ibse:[svcName].[mthName];responseTo=[pre-defined port name or another disposition url];errorTo=[pre-defined port name or another disposition url]". At the bottom of the dialog are two buttons: "OK" and "Cancel".

- a. Type a name and a brief description for the event port.
- b. From the Protocol drop-down list, select *iBSE*.
- c. In the URL 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 defines the parameters for the iBSE disposition.

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	Predefined port name or another disposition URL to which error logs are sent. Optional.

4. Click OK.

The port appears under the ports node in the left pane. To review the port settings, select the port name. A table summarizing the port settings appears in the right pane.

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting a Channel* on page A-30.

Procedure How to Create an Event Port for MSMQ

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

The Add Port dialog box opens:

The screenshot shows the 'Add Port' dialog box with the following fields and values:

- Name:** (empty text box)
- Description:** (empty text box)
- Protocol:** MSMQ (dropdown menu)
- URL:** msmq://[machineName]/private\$/[qName];errorTo=[pre-defined port name or another disposition url] (text box)
- Buttons:** OK, Cancel

- a. Type a name and a brief description for the event port.
- b. From the Protocol drop-down list, select *MSMQ*.
- c. In the URL field, type an MSMQ destination using the following format:

`msmq://[machineName]/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 defines the parameters for the MSMQ disposition.

Parameter	Description
machineName	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.

4. Click *OK*.

The port appears under the ports node in the left pane. To review the port settings, select the port name. A table summarizing the port settings appears in the right pane.

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting a Channel* on page A-30.

Procedure How to Create an Event Port for JMSQ

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

The Add Port dialog box opens:

Add Port

Name:

Description:

Protocol: JMSQ ▼

URL:

- Type a name and a brief description for the event port.
- From the Protocol drop-down list, select *JMSQ*.
- In the URL field, type a JMS 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]
```

The following table defines the parameters for the JMSQ disposition.

Parameter	Description
myQueueName	JNDI name of a queue to which events are emitted.
myQueueFac	Resource that contains information about the JMS Server. The WebLogic connection factory is: <code>javax.jms.QueueConnectionFactory</code>

Parameter	Description
jndiurl	<p>URL used to contact the JNDI provider. The syntax of this URL depends on which JNDI provider is being used. This value corresponds to the standard JNDI property.</p> <p><code>java.naming.provider.url.</code></p> <p>The URL of the WebLogic Server is</p> <p><code>t3://host:port</code></p> <p>where:</p> <p><code>host</code></p> <p>Is the machine name where WebLogic Server is installed.</p> <p><code>port</code></p> <p>Is the port on which WebLogic Server is listening. The default port, if not changed at installation, is 7001</p>
jndifactory	<p>Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider.</p> <p>For WebLogic Server, the WebLogic factory is:</p> <p><code>weblogic.jndi.WLInitialContextFactory.</code></p>
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.

4. Click OK.

The port appears under the ports node in the left pane. To review the port settings, select the port name. A table summarizing the port settings appears in the right pane.

You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting a Channel* on page A-30.

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.

The iWay Event Adapters window opens.

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 Event Port window opens in the right pane.

- a. Type a name for the event port and provide a brief description.
- b. From the Disposition Protocol drop-down list, select *SOAP*.
- c. In the Disposition field, enter a SOAP destination using the following format:

```
soap:wSDL-url;soapaction=action;responseTo=respDest;errorTo=errorDest
```

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

Parameter	Description
wSDL-url	<p>The URL to the WSDL file that is required to create the SOAP message. For example:</p> <pre>http://localhost:7001/ibse/IBSEServlet/test/sw2xml2003MQ.ibs?wSDL</pre> <p>This value can be found by navigating to the iWay Business Services tab and opening the <i>Service Description</i> link in a new window. The WSDL URL appears in the Address field.</p> <p>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.</p>

Parameter	Description
action	<p>The method that will be called by the disposition. For example:</p> <p><code>JDE.mt200Request@test@@</code></p> <p>where</p> <p><code>JDE</code></p> <p>Is the name of the Web service you created using Application Explorer.</p> <p><code>mt200</code></p> <p>Is the method being used.</p> <p><code>test</code></p> <p>Is the license that is being used by the Web service.</p> <p>This value can be found by navigating to the iWay Business Services tab and opening the <i>Service Description</i> link in a new window. Perform a search for <i>soapAction</i>.</p> <p>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.</p>
respDest	<p>The location to which responses are posted. A predefined port name or another full URL. Optional.</p> <p>A predefined port name or another disposition URL. The URL must be complete, including the protocol.</p>
errorDest	<p>The location to which error logs are sent. Optional.</p> <p>A predefined port name or another disposition URL. The URL must be complete, including the protocol.</p>

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, Editing, and Deleting a Channel* on page A-30.

Procedure How to Create an Event Port for HTTP

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

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

The Add Port dialog box opens:

The screenshot shows a standard Windows-style dialog box titled "Add Port". It contains four labeled input fields: "Name:" (a single-line text box), "Description:" (a multi-line text box), "Protocol:" (a drop-down menu currently showing "HTTP"), and "URL:" (a multi-line text box containing the text `http://[myurl];responseTo=[pre-defined port name or another disposition url]`). At the bottom right are "OK" and "Cancel" buttons.

- a. Type a name and a brief description for the event port.
- b. From the Protocol drop-down list, select *HTTP*.
- c. In the URL field, type an HTTP destination using the following format:
`http://[myurl];responseTo=[pre-defined port name or another disposition url]`

The following table defines the parameters for the HTTP disposition.

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

4. Click *OK*.

The port appears under the ports node in the left pane. To review the port settings, select the port name. A table summarizing the port settings appears in the right pane.

You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting a Channel* on page A-30.

Procedure How to Create an Event Port for MQ Series

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

The Add Port dialog box opens:

- a. Type a name and a brief description for the event port.
- b. From the Protocol drop-down list, select *MQ Series*.
- c. In the URL field, type an MQ Series destination using the following format:
`mqseries://[qManager]/[qName];host=[hostname];port=[port];
channel=[channelname];errorTo=[pre-defined port name or another
disposition url]`

The following table defines the parameters for the MQ Series disposition.

Parameter	Description
qManager	Name of the queue manager to which the server must connect.
qName	Name of the queue where messages are placed.
host	Host on which the MQ Server is located (for the MQ Client only).

Parameter	Description
port	Number to connect to an MQ Server queue manager (for the MQ client only).
channel	Case-sensitive name of the channel that connects with the remote MQ Server queue manager (for the 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.

4. Click *OK*.

The port appears under the ports node in the left pane. To review the port settings, select the port name. A table summarizing the port settings appears in the right pane.

You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting a Channel* on page A-30.

Editing and Deleting an Event Port

The following procedures provide information on how to edit and delete an event port using Application Explorer.

Procedure How to Edit an Event Port

1. Expand the *iWay Event Adapters* node.
2. Expand the *JDEdwards* node.
3. Right-click the event port you want to edit and select *Edit*.
The Edit Port window opens.
4. Make the required changes and click *OK*.

Procedure How to Delete an Event Port

1. Expand the *iWay Event Adapters* node.
2. Expand the *JDEdwards* node.
3. Right-click the event port you want to delete and select *Delete*.
The event port disappears from the list in the left pane.

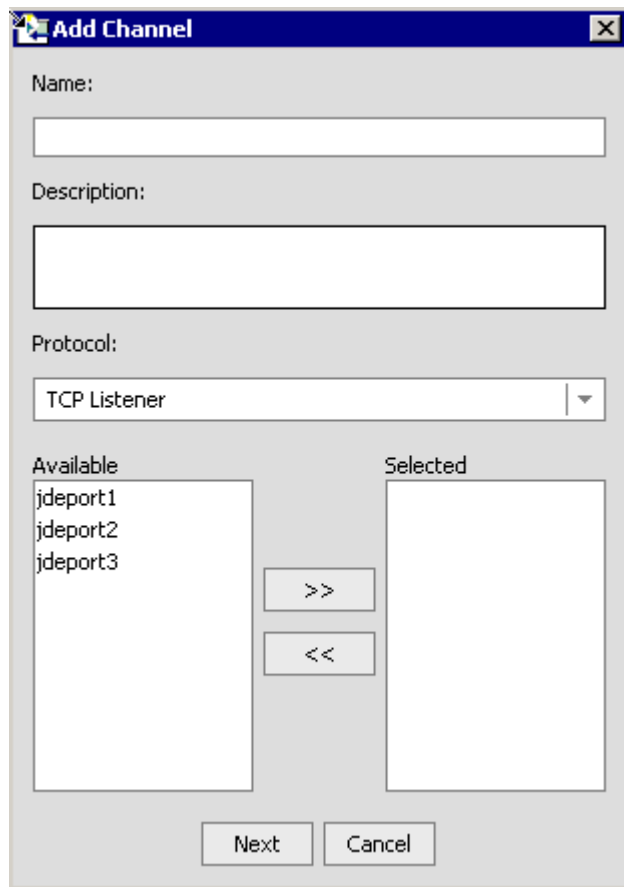
Creating, Editing, and Deleting a Channel

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

Procedure How to Create a Channel

1. Expand the *iWay Event Adapters* node.
2. Expand the *JDEdwards* node.
3. Right-click the *Channels* node and select *Add Channel*.

The Add Channel dialog box opens:



The image shows a Windows-style dialog box titled "Add Channel". It has a standard title bar with a minimize button, a maximize button, and a close button (X). The dialog box contains the following fields and controls:

- Name:** A text input field.
- Description:** A larger text input field.
- Protocol:** A dropdown menu currently showing "TCP Listener".
- Available:** A list box containing three items: "jdeport1", "jdeport2", and "jdeport3".
- Selected:** An empty list box.
- Navigation buttons:** Two buttons between the list boxes, labeled ">>" and "<<".
- Action buttons:** Two buttons at the bottom, labeled "Next" and "Cancel".

4. Specify information for the channel you are creating.

- a. Type a name (for example, JDEChannel) and a brief description for the channel.
 - b. From the *Protocol* drop-down list, select a protocol (for example, TCP Listener).
 - c. Select an event port from the list of available ports. To select more than one, hold down the *Ctrl* key and click the ports.
 - d. Click the double right arrow (>>) to transfer the port(s) to the list of selected ports.
5. Click Next.

The Basic settings in the TCP Listener dialog box appear:

Tcp Listener

Basic preparser

Host* localhost

Port Number*

Synchronization Type RECEIVE_REPLY ▼

☐ Is Length Prefix

☐ Is XML

☐ Is Keep Alive

OK Cancel

Fields marked with * are required.

6. Specify values for the following parameters. Fields with an asterisk are required.

Property	Description
Host*	Name or URL of the machine where the database is installed.
Port Number*	Port on which the Host database is listening.
Synchronization Type	<ul style="list-style-type: none">• Select RECEIVE_REPLY if the event application expects a reply sent back to it. Specify a preemitter.• Select RECEIVE_ACK when a TCP/IP acknowledgement (ACK) is sent back to the event application.• Select RECEIVE if the event application does not expect a return.
Is Length Prefix	For J.D. Edwards OneWorld events that send data back that is not in XML format. The TCP/IP event application must prefix the data with a 4-byte binary length field when writing the data to the TCP/IP port.
Is XML	For J.D. Edwards OneWorld events that send data back in XML format. No preparser is required.
Is Keep Alive	Maintains continuous communication between the event transaction and the channel.

7. Click the preparser tab.

The preparser settings in the TCP Listener dialog box appear:

tcp Listener

Basic preparser

User id*

User password*

JDE Environment*

Application

Server IP address*

Server port *

OK Cancel

Fields marked with * are required.

8. Specify values for the following parameters. Fields with an asterisk are required.

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

9. Click **OK**.

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



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

To review the settings for the channel, select the channel. In the right pane, Detail, TCP Listener, and preparser tabs summarize the channel settings.

Procedure How to Start and Stop a Channel

1. Expand the *iWay Event Adapters* node.
2. Expand the *JDEdwards* node.
3. To start a channel, right-click the channel node and select *Start*.
The channel becomes active and the X over the icon disappears.
4. To stop a channel, right-click the connected channel node and select *Stop*.
The channel becomes inactive and the X appears over the icon.

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 Event Adapters* node.
2. Expand the *JDEdwards* node.
3. Right-click the channel you want to edit and select *Edit*.
The Edit Channel dialog box opens.
4. Make the required changes to the channel configuration and click **OK**.

Procedure How to Delete a Channel

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

The channel disappears from the list in the left pane.

For More Information

See the following topics in Chapter 3, *Listening for Database Events*:

- *The OneWorld Event Listener*
- *Configuring the OneWorld Event Listener*
- *Logging and Error Handling*

Deploying iWay Components in a Clustered BEA WebLogic Environment

iWay events can be configured in a clustered BEA WebLogic environment.

A cluster consists of multiple server instances running simultaneously, yet appears to clients to be a single server instance. The server instances that contain a cluster can be run on one machine, but are usually run on multiple machines.

Clustering provides the following benefits:

- Load balancing
- High availability

Service requests are processed through the HTTP router and routed to an available managed server.

Events are server-specific and are not processed through the HTTP router. You must configure each server separately.

Procedure How to Deploy iWay Components in a Clustered Environment

To deploy iWay components in a clustered environment:

1. Using the BEA Configuration Wizard:
 - a. Configure an administrative server to manage the managed servers.
 - b. Add and configure as many managed servers as required.
 - c. Add and configure an HTTP router. This does not have to be a part of WebLogic and can be an outside component.

- d. If you configure the HTTP router within WebLogic, start it by entering the following command:

```
StartManagedWebLogic HTTPROUTER http://localhost:7001
```

where:

HTTPROUTER

Is the name of the server on which the HTTP router is running.

http://localhost:7001

Is the location of the admin console.

- e. Add the managed servers to your cluster/clusters.

For more information on configuring WebLogic Integration for deployment in a clustered environment, see *Deploying WebLogic Integration Solutions*.

2. Start the WebLogic Server and open WebLogic Server Console.
3. Deploy iBSE to the cluster by selecting *Web Application Modules* from the Domain Configurations section, and clicking *Deploy a new Web Application Module*.

A page appears for you to specify where the Web application is located.

Note: You can deploy JCA to a cluster, but you can only point it to one directory, and to the machine on which it is installed.

4. To deploy iBSE, select the option button next to the ibse directory and then click *Target Module*.

Deploy a Web Application Module

Select the archive for this Web application module

Select the file path that represents your archive or exploded archive directory.

Note: Only valid file paths are shown below. If you do not find what you are looking for, [your file\(s\)](#) and/or confirm your Web application module contains valid descriptors.

Location: [localhost](#) \ [C:](#) \ [iWay55](#) \ bea

	ibse
	iwa
	iwaycaivp

5. To deploy servlet Application Explorer, select the option button next to the iwae directory and then click *Target Module*.

If you are using servlet Application Explorer, deploy it only on the admin server or one of the managed servers.

Deploy a Web Application Module

Select the archive for this Web application module

Select the file path that represents your archive or exploded archive directory.

Note: Only valid file paths are shown below. If you do not find what you are looking for, you should [upload your file\(s\)](#) and/or confirm your Web application module contains valid descriptors.

Location: [localhost](#) \ [C:](#) \ [Program Files](#) \ [iWay55](#) \ bea

<input type="radio"/>	ibse
<input checked="" type="radio"/>	iwae
<input type="radio"/>	iwjcaivp

Target Module

The following window opens.

Select targets for this Web application module

Select the servers and/or clusters on which you want to deploy your new Web Application module

Independent Servers

<input type="checkbox"/> AdminServer
<input type="checkbox"/> HTTPROUTER

Clusters

<input checked="" type="checkbox"/> MYCluster <ul style="list-style-type: none"> <input checked="" type="radio"/> All servers in the cluster <input type="radio"/> Part of the cluster <ul style="list-style-type: none"> <input type="checkbox"/> MS1 <input type="checkbox"/> MS2
--

6. Select the servers and/or clusters on which you want to deploy the application and click *Continue*.

The following window opens.

Source Accessibility

During runtime, a targeted server must be able to access this Web Application module's files. This access can be accomplished by either copying the Web Application module onto every server, or by defining a single location where the files exist.

How should the source files be made accessible?

- ☐ Copy this Web Application module onto every target for me.

During deployment, the files in this Web Application module will be copied automatically to each of the targeted locations.

- ☒ I will make the Web Application module accessible from the following location:

C:\Way55\bea\ibse

Provide the location from where all targets will access this Web Application module's files. You must ensure the Web Application module's files exist in this location and that each target can reach the location.

7. Select the *I will make the Web Application module accessible from the following location* option button and provide the location from which all targets will access iBSE.

iWay Software recommends that you use a single instance of iBSE, rather than copying iBSE onto every target.

Note: iBSE must use a database repository (SQL or Oracle). Do not use a file repository. You can select this in the Repository Type drop-down list in the iBSE monitoring page. After configuring a database repository, you must restart all of the managed servers.

<http://hostname:port/ibse/IBSEConfig/>

where:

[hostname](#)

Is where your application server is running. Use the IP address or machine name in the URL; do not use localhost.

[port](#)

Is the port specific to each server, since you deploy iBSE to an entire cluster. For example, 8001, 8002, or any other port that is specified for each managed node.

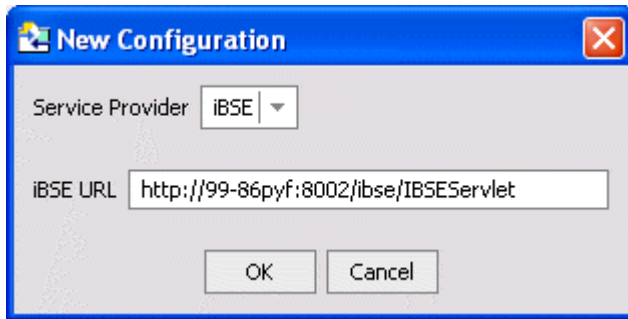
8. Click *Deploy*.

Procedure Configuring Ports and Channels in a Clustered Environment

To configure ports and channels in a clustered environment:

1. Open Swing Application Explorer in BEA WebLogic Workshop.

2. Create a new connection to the iBSE instance. For information on creating a new configuration, see *Creating a New Configuration on page A-3*.

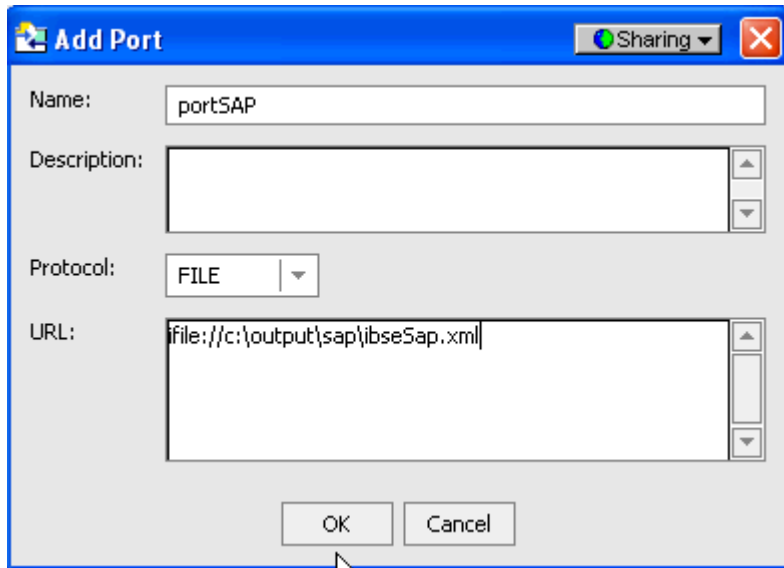


Note: Use the IP address or machine name in the URL; do not use localhost.

3. Connect to the new configuration and select the iWay Events node in the left pane of Application Explorer.

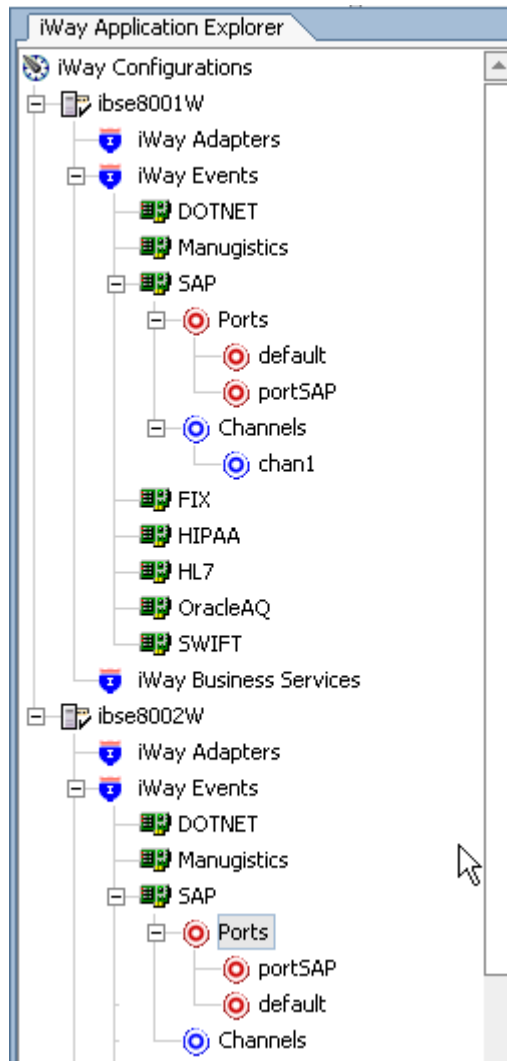


4. Select an adapter from the adapter list (in this example, SAP) and add a new port. For more information, see *Creating, Editing, and Deleting a Port* on page A-18.



5. Create a channel and add the port you created. For more information, see *Creating, Editing, and Deleting a Channel* on page A-30.
6. Click *Next* and enter the application server parameters.
7. Start the channel.
8. Create a new configuration and connect to the second iBSE instance.
The connection to iBSE must be configured to each instance of the managed server.

The following graphic shows two configurations.



The following operations performed on one managed server will be replicated on all other managed servers:

- Create port and channel: Creates the channel and port under all available servers.
- Delete port and channel. Deletes the port and channel under all available servers.

The following operations must be performed on each server:

- Start channel. Starts the channel for the specific server.

- Stop channel. Stops the channel for the specific server.

Adding a Control for an iWay Resource in BEA WebLogic Workshop

Java controls provide a convenient way to incorporate access to iWay resources. You can add controls in BEA WebLogic Workshop to use Web services created by Application Explorer, or you can add controls that enable you to take advantage of the JCA resources of Application Explorer.

Adding a Web Service Control to a BEA WebLogic Workshop Application

After you create an iWay Web service using Application Explorer and export the WSDL file, you can create a control for the Web service.

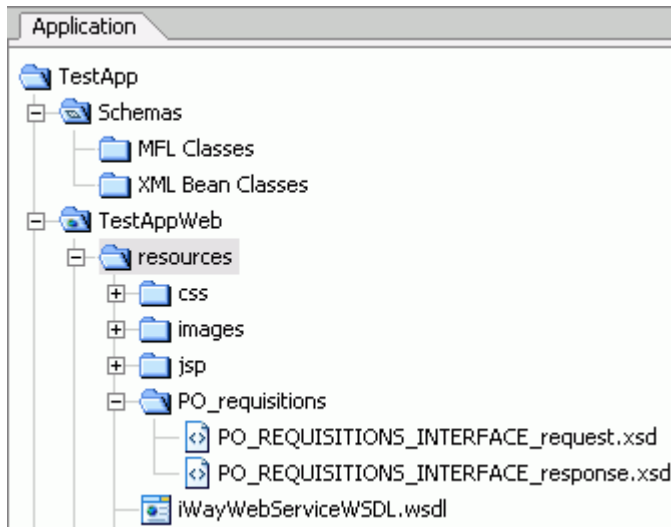
For more information on exporting a WSDL file, see *How to Generate WSDL From a Web Service* on page A-17.

Procedure How to Add a Web Service Control

To add a Web service control:

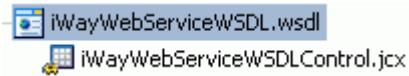
1. After exporting the WSDL file from Application Explorer, locate the file in the Application tab of your BEA WebLogic Workshop application.

For example, a WSDL file saved to the \resources directory in your BEA WebLogic Workshop Web application directory structure appears as follows:



2. Right-click the WSDL file and select *Generate Service Control*.

The control for the WSDL appears below the WSDL file in the resources tree.



Extensible CCI Control

The following section describes the enhanced CCI control, which is extensible and provides JCX with typed inputs and outputs for JCA in BEA WebLogic Workshop.

Overview

The extensible iWay CCI control now offers:

- **Method and tag validation.** BEA WebLogic Workshop provides warnings regarding invalid methods and tags.
- **Improved error handling.**

You can now define new methods that rely on the generic *service* and *authService* methods. For example, you can define a JCX with a new method such as the following, without having to write casting code or explicit transformations:

```
sapComDocumentSapRfcFunctions.BAPIMATERIALGETDETAILResponseDocument
getDetail(sapComDocumentSapRfcFunctions.BAPIMATERIALGETDETAILDocument
aRequest) throws java.lang.Exception
```

In addition, the extensible CCI control now generates a JCX file to which you can add your own methods.

Using the Extensible CCI Control

The extensible CCI control functions much like a database control since it generates JCX files to which you can add your own methods.

Your own methods can use the correct input and output types rather than the generic *XmlObject* types that the JCA control uses. Since the control is just a proxy that uses a reflection to call the relevant method, it will take care of the casting for you. There is no longer a need to write custom code that does the cast or transformations that are cast between an *XmlObject*.

For example, instead of the generic *XmlObject*:

```
XmlObject service(XmlObject input) throws java.lang.Exception;
```

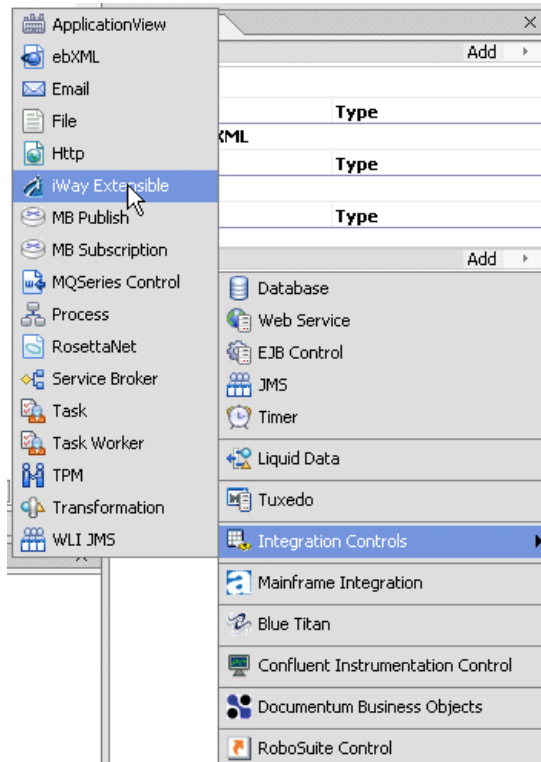
you will be calling:

```
BAPIMATERIALGETDETAILResponseDocument
getDetail(BAPIMATERIALGETDETAILDocument aRequest) throws
java.lang.Exception;
```

Example Defining a Control Using the Extensible CCI Control

The following sample JCX demonstrates how to define a control that uses the SAP BAPI_MATERIAL_GET_DATA using the extensible CCI control in BEA WebLogic Workshop.

1. Start BEA WebLogic Workshop and create a new project.
2. Click *Integration Controls* and select *iWay Extensible*.



The Insert Control - iWay Extensible dialog box opens.

Insert Control - iWay Extensible

STEP 1 Variable name for this control: SAPjcx

STEP 2 I would like to :

☐ Use an iWay Extensible control already defined by a JCX file

JCX file: Browse...

☒ Create a new iWay Extensible control to use.

New JCX name: SAPjcx

☐ Make this a control factory that can create multiple instances at runtime

STEP 3

Adapter Name: SAP

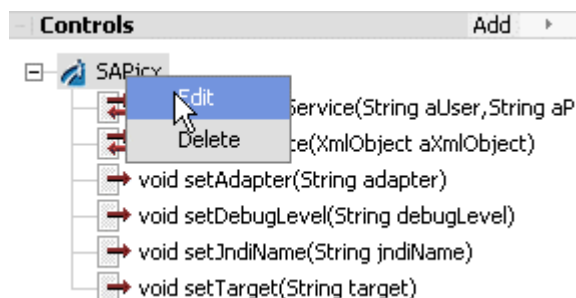
Target Name: sapconnection

Debug Level: ERROR

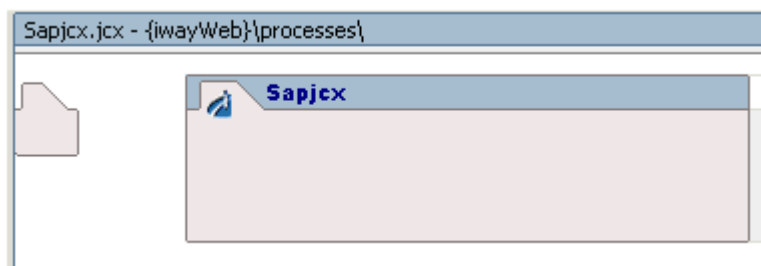
Create Cancel

3. Perform the following steps:
 - a. Provide a variable name for this control.
 - b. Click *Create a new iWay Extensible control to use* and provide a new JCX name.
 - c. Enter the adapter name, target name, and select a debug level from the drop-down list.
 4. Click *Create*.
- A new JCX file is created.

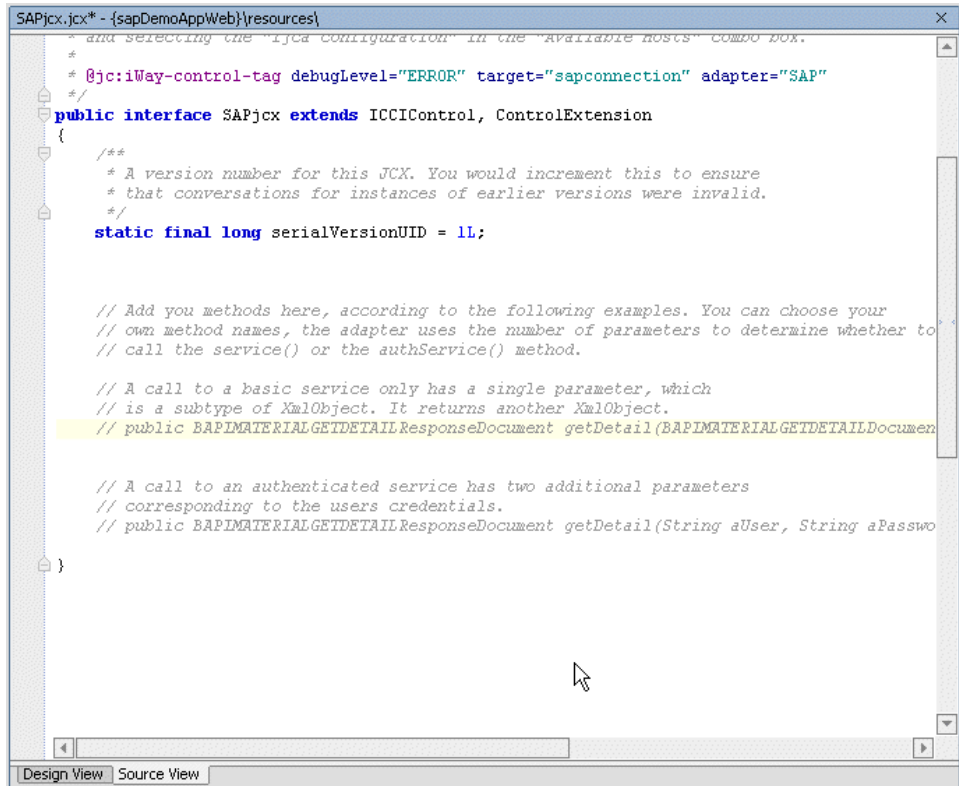
To edit an existing control, right click the control and select *t Edit*.



The Design view is displayed.



5. Click Source View.



You can add your own methods that call the adapter's services.

APPENDIX B

Configuring J.D. Edwards OneWorld for Outbound Transaction Processing

Topics:

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

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

This section describes how to enable outbound transaction processing in OneWorld 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 OneWorld event.

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

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

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

The Data Export Control Table and the Processing Log Table

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

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

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

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 OneWorld jde.ini File

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

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

The following is a sample of the modifications required to implement XML support.

1. 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
```

2. Change the following block of code:

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

Note: Change maxKernelRanges to 15.

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

APPENDIX C

Sample Files

Topics:

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

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

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

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

Issuing a Single-Function Request

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

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 version="1.0"?>
<!DOCTYPE jdeResponse>
<jdeResponse environment="DV7333"
    pwd="JDE"
    session="516.1029417972.68"
    type="callmethod"
    user="JDE">
    <callMethod app="BSE"
        name="GetEffectiveAddress"
        runOnError="no">
        <returnCode code="0"/>
        <params>
            <param name="mnAddressNumber">1001</param>
            <param name="jdDateBeginningEffective"/>
            <param name="cEffectiveDateExistence10"/>
            <param name="szAddressLine1">8055 Tufts Avenue, Suite 1331
        </param>
            <param name="szAddressLine2">
        </param>
            <param name="szAddressLine3">
        </param>
            <param name="szAddressLine4">
        </param>
            <param name="szZipCodePostal">80237 </param>
            <param name="szCity">Denver </param>
            <param name="szCountyAddress"> </param>
            <param name="szState">CO</param>
            <param name="szCountry"/>
            <param name="szUserid"/>
            <param name="szProgramid"/>
            <param name="jdDateupdated"/>
            <param name="szWorkstationid"/>
            <param name="mnTimeLastupdated">0</param>
            <param name="szNamealpha">J.D. Edwards & Company
        </param>
        </params>
    </callMethod>
</jdeResponse>
```

Issuing a Multiple-Function Request

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

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 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>
    </params>
  </callMethod>
</jdeRequest>
```

```

    <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>
  </params>

```

Issuing a Multiple-Function Request

```
<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'>

```

Issuing a Multiple-Function Request

```
<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>
```

Issuing a Multiple-Function Request

```
<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>

```

Issuing a Multiple-Function Request

```
<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>

```

Issuing a Multiple-Function Request

```
<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 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='cCMPProcessEdits'>1</param>
      <param name='szCMComputerID' idref='2'></param>
      <param name='cCMUpdateWriteToWF'>2</param>
      <param name='szCMPProgramID'>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'>
    </onError>
  <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>
    </params>
  </callMethod>
</jdeRequest>
```

```

<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='cCMPProcessEdits'>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='szCMPProgramID'>XMLInterop</param>
    <param name='szCMVersion'>ZJDE0001</param>
    <param name='cWKSSourceOfData'></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='cCMPProcessEdits'>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='szCMPProgramID'>XMLInterop</param>
    <param name='szCMVersion'>ZJDE0001</param>
    <param name='cWKSSourceOfData'></param>
  </params>
  <onError abort='no'>
</onError>
</callMethod>
<callMethod name='F4211FSEndDoc' app='XMLInterop'
  runOnError='no'>

```

Sample Sales Order Request

```
<params>
  <param name='mnCMJobNo' idref='1'></param>
  <param name='szCMComputerID' idref='2'></param>
  <param name='szCMPProgramID'>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="cCMPProcessEdits">1</param>
    <param name="szCMComputerID" idref="2">XEENT</param>
    <param name="cCMErrorConditions">1</param>
    <param name="cCMUpdateWriteToWF">2</param>
    <param name="szCMPProgramID">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>
```

Sample Sales Order Response

```
<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="szWKCcreditMesg"> </param>
<param name="szWKTempCreditMesg"> </param>
<param name="cWKSsourceOfData"/>
<param name="cWKProcMode"/>
<param name="mnWKSsuppressProcess">0</param>
<param name="szPricingGroup">PREFER </param>
<param name="mnProcessID">2252</param>
<param name="mnTransactionID">4</param>
</params><errors><error code="2597">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="cCMPProcessEdits">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="szCMPProgramID">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

```

Sample Sales Order Response

```
Exceeds what'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="szCMPProgramID">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="mnWKTotCost">321</param>
  <param name="mnWKForeignTotalCost">321</param>
  <param name="cWKSourceOfData"/>
```

```

<param name="cWKCheckAvailability">1</param>
<param name="mnLastLineNoAssigned">2</param>
<param name="cStockingType">P</param>
<param name="cParentItmMethdOfPriceCalcn">1</param>
<param name="mnShortItemNo">60003</param>
<param name="szSalesOrderFlags">0</param>
<param name="jdPriceEffectiveDate">2000/03/29</param>
<param name="jdPromisedShip">2000/03/29</param>
<param name="mnQuantityAvailable">-44</param>
<param name="mnItemVolume_ITVL">22,5</param>
<param name="szVolumeUOM_VLUM">FC</param>
<param name="szRevenueBusinessUnit">M30</param>
<param name="mnProcessID">2252</param>
<param name="mnTransactionID">4</param>
</params><errors><error code="030B">Warning: Order Quantity
Exceeds what'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="szCMPProgramID">XMLInterop</param>
<param name="szCMVersion">ZJDE0001</param>
<param name="mnTimeOfDay">174220</param>
<param name="cCMUseWorkFiles">2</param>
<param name="cCMPProcessEdits">1</param>
<param name="mnProcessID">2252</param>
<param name="mnTransactionID">4</param>
</params>
</callMethod><returnParams failureDestination="ERROR.Q"
successDestination="SUCCESS.Q">
</returnParams>
</jdeResponse>

```

Sample Sales Order Response

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