

iWay

iWay Connector for JCA for BEA WebLogic
User's Guide
Version 5 Release 5



DN3501314.0404

April 14, 2004

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.

Contents

1. Introducing the iWay Connector for JCA	1-1
Overview of the iWay Connector for JCA	1-2
Support for Over 250 Adapters	1-3
Flexibility of Use	1-4
Distribution of the iWay Connector for JCA	1-6
Deployment of the iWay Connector for JCA	1-7
Using a JCA Resource Adapter	1-7
2. Deploying the iWay Connector for JCA	2-1
Deploying to BEA WebLogic Server	2-2
Deploying the iWay Connector for JCA to BEA WebLogic Server	2-2
3. iWay JCA Installation Verification Program	3-1
Overview of the IVP	3-2
Deploying the IVP for BEA WebLogic Server	3-2
Modifying the IVP for BEA WebLogic	3-5
4. Using the iWay Connector for JCA	4-1
Accessing the iWay Adapter for Telnet From BEA WebLogic Server	4-2
3270 Emulation	4-2
iWay Adapter Transaction Modules	4-2
Creating Adapter Transaction Modules	4-3
Sample CICS Transaction	4-4
Creating an Adapter Transaction Module	4-6
Adding a Screen Identifier	4-8
Adding Parameters and Metadata	4-10
Transaction Module XML Request Document	4-16
Connecting to Telnet From iWay Application Explorer	4-17
Deploying and Running the Sample Servlet	4-23
Accessing a CICS Transaction From BEA WebLogic Server	4-29
Viewing Metadata and Creating a Schema	4-35
Deploying and Running the Sample Servlet	4-36
Accessing an IMS Transaction From BEA WebLogic Server	4-42
Viewing Metadata and Creating a Schema	4-46
Deploying and Running the Sample Servlet	4-48
Accessing a J.D. Edwards OneWorld Business Function From BEA WebLogic Server	4-53
Preparing to Use J.D. Edwards OneWorld XE	4-53
Connecting to J. D. Edwards OneWorld From iWay Application Explorer	4-54
Viewing Metadata and Creating a Schema	4-60
Deploying and Running the Sample Servlet	4-66

Contents

Accessing a Lawson Business Function From BEA WebLogic Server	4-73
Preparing to Use the iWay Adapter for Lawson	4-73
Connecting to Lawson From iWay Application Explorer	4-74
Viewing Metadata and Creating a Schema	4-79
Deploying and Running the Sample Servlet	4-80
Accessing an Oracle Applications Business Process From a BEA WebLogic Application Server .	4-87
Connecting to Oracle Applications From iWay Application Explorer	4-87
Viewing Metadata and Creating a Schema	4-94
Deploying and Running the Sample Servlet	4-98
Accessing a PeopleSoft Business Function From BEA WebLogic Server	4-106
Preparing to Use PeopleSoft	4-106
Connecting to PeopleSoft From iWay Application Explorer	4-106
How to Establish a Connection to PeopleSoft From iWay Application Explorer	4-108
Viewing Metadata and Creating a Schema	4-111
Deploying and Running the Sample Servlet	4-113
Accessing an SAP Business Object From BEA WebLogic Server	4-118
Connecting to SAP From iWay Application Explorer	4-118
Viewing Metadata and Creating a Schema	4-125
Deploying and Running the Sample Servlet	4-126
Accessing a Siebel Business Object From BEA WebLogic Server	4-131
Preparing to Use Siebel	4-131
Connecting to Siebel From iWay Application Explorer	4-132
Viewing Metadata and Creating a Schema	4-138
Deploying and Running the Sample Servlet	4-139
Issuing a VSAM Parameterized Query From BEA WebLogic Server	4-147
Connecting to VSAM From iWay Application Explorer	4-147
Viewing Metadata and Creating a Schema	4-152
Creating a Parameterized Request	4-153
Deploying and Running the Sample Servlet	4-156
A. Servlet Sample Code	A-1
iWay Servlet Sample Code	A-2

Preface

This document is for iWay customers, consultants, and resellers who integrate J2EE™ application components with J2EE application servers using the iWay Connector for JCA. Before you begin, you should understand the J2EE Connector Architecture Specification, JSR016.

This documentation describes how to install and use the iWay Connector for JCA Connector for JCA. It is intended for those who are using the J2EE to Connector Architecture to connect to Enterprise Information Systems (EIS).

How This Manual Is Organized

This manual includes the following chapters:

Chapter/Appendix		Contents
1	Introducing the iWay Connector for JCA	Introduces the iWay Connector for JCA, including information on how the connector is distributed and deployed.
2	Configuring and Deploying the iWay Connector for JCA	Describes how to configure and deploy the iWay Connector for JCA.
3	iWay JCA Installation Verification Program	Describes the iWay JCA Installation Verification Program, used to test the functionality of the iWay Adapter Framework based J2EE-CA connector.
4	Using the iWay Connector for JCA	Illustrates how to access an Enterprise Information System (EIS) using the iWay Connector for JCA.
A	Servlet Sample Code	Contains servlet sample code for Enterprise JavaBeans.

Documentation Conventions

The following conventions apply throughout this manual:

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), a cross-reference, or an important term.
this typeface	Highlights a file name or command.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices; type one of them, not the braces.
[]	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
	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 Connector for JCA?

Call Customer Support Services (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 questions. Our consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code number (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.iwaysoftware.com>. It connects you to the tracking system and known-problem database at our 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 iWay Software representative about InfoResponse Online, or call (800) 969-INFO.

Information You Should Have

To help our consultants answer your questions most effectively, be ready to provide the following information when you call:

- Your six-digit site code (xxxx.xx).
- Your iWay Software configuration:
 - The iWay Software version and release.
 - The communications protocol (for example, TCP/IP or LU6.2), including vendor and release.
- The stored procedure (preferably with line numbers) or SQL statements being used in server access.
- The database server release level.
- The database name and release level.
- iWay XML Dictionary

- The exact nature of the problem:
 - Are the results or the format incorrect? Are the text or calculations missing or misplaced?
 - The error message and return code, if applicable.
 - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, your security system, communications protocol, or front-end software changed?
- Is this problem reproducible? If so, how?
- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing just the code to access the data source?
- 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 any opinion you can offer regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Documentation Feedback form on our Web site, <http://www.iwaysoftware.com>.

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>).

CHAPTER 1

Introducing the iWay Connector for JCA

Topics:

- Overview of the iWay Connector for JCA
- Distribution of the iWay Connector for JCA
- Deployment of the iWay Connector for JCA
- Using a JCA Resource Adapter

This section provides an overview of the iWay Connector for JCA and describes how it is distributed and deployed.

Overview of the iWay Connector for JCA

The J2EE Connector Architecture (JCA) defines a standard architecture for connecting the J2EE platform to a heterogeneous Enterprise Information System (EIS). Examples of an EIS include Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), Supply Chain Management (SCM), mainframe transaction processing, database systems, and legacy applications that are not written in the Java™ programming language. By defining a set of scalable, secure, and transactional mechanisms, JCA enables the integration of an EIS with an application server and enterprise applications.

The J2EE Connector Architecture permits an EIS vendor to provide a standard resource adapter for its EIS. The resource adapter plugs into an application server, providing connectivity to an EIS, and integrating it with the rest of the enterprise. If an application server vendor has extended its system to support JCA, it is assured of seamless connectivity to multiple Enterprise Information Systems.

iWay Software is the world's leading provider of integration tools and adapter technologies that offer fluid control over all your enterprise information assets. iWay's broad set of standards-based adapters provide rapid integration of applications, e-business documents, and databases. The iWay Intelligent Adapter Suite integrates a wide variety of IT assets into a single virtual information system with shared data structures, transactions, and business logic. No matter how diverse or dissimilar the components of your IT infrastructure are, iWay integration tools and adapters make them all accessible for e-business applications and integration projects.

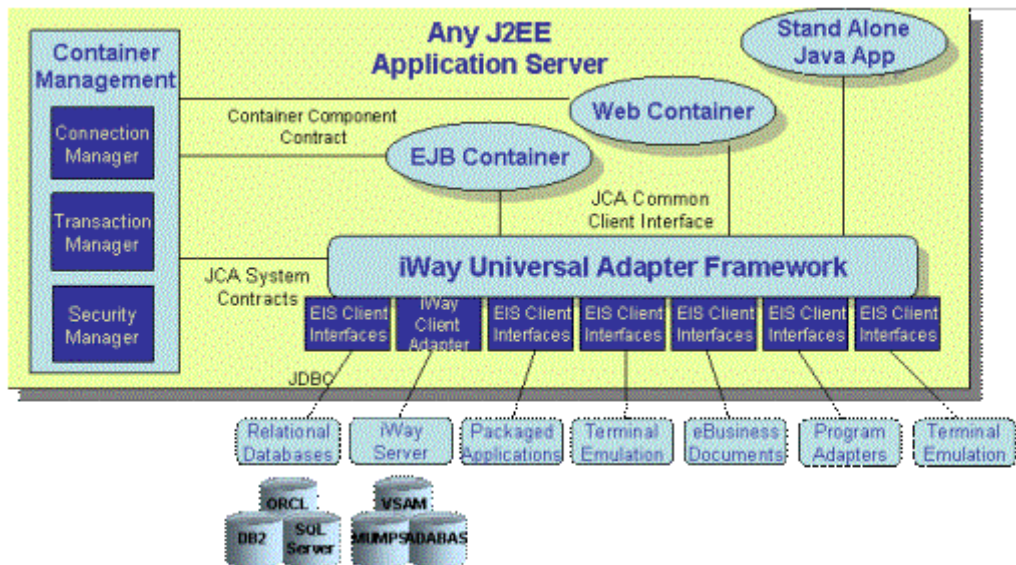
Companies have adopted J2EE application servers as a standard framework for supporting the development and maintenance of Internet applications. Separating non-functional tasks such as security, transactions, connection pooling, and data persistence from functional tasks such as order processing and customer value analysis enables you to concentrate on developing business logic and software vendors to focus on application infrastructure.

The JCA specification was created to establish a standardized mechanism for integrating Java-based applications with heterogeneous enterprise information assets such as packaged systems. JCA provides a standard way to connect several application servers with many Enterprise Information Systems. Using the iWay Connector for JCA, you can connect to any of over 250 adapters.

Support for Over 250 Adapters

The iWay Connector for JCA provides over 250 intelligent adapters, enabling you to integrate your J2EE applications with your Enterprise Information Systems, all within a unified adapter framework. iWay provides intelligent adapters for:

- Application packages, such as Siebel, SAP, PeopleSoft, Ariba, Lawson, Clarify, and others.
- Electronic business formats, such as SWIFT, FIX, HIPAA, ACORD, ebXML, and OAG BODs.
- Legacy database systems, such as ADABAS, Ingres, MUMPS, Teradata, Unisys, and many others.
- Relational databases, such as Oracle, MS SQL Server, Informix, Sybase, and many more.
- Transaction environments, such as CICS, IMS, and Tuxedo.
- 35 mainframe, midrange, UNIX, and PC operating platforms.
- Custom 3GL and 4GL applications.
- .NET Assemblies
- Messaging products like Sonic MQ, Oracle AQ, MQSeries, or JMS.
- Object technologies like COM, CORBA, and EJB™.
- Terminal emulation adapters that provide wrappers around the presentation tiers of applications of “green screen” 3270 and 5250 systems.



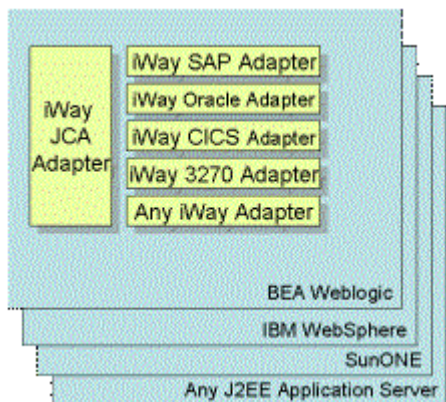
Flexibility of Use

Bidirectional Outbound/Inbound Use. You can use iWay adapters to invoke traditional response-request remote procedure calls (RPC) to your Enterprise Information Systems. This includes SQL requests to relational and non-relational data stores, as well as calls to applications such as SAP or Siebel. Additionally, iWay adapters work with many messaging technologies and have a native Java Message Service (JMS) interface. This enables you to augment your asynchronous messaging systems with iWay-enabled systems access.

Service Adapters. iWay resource adapters can invoke services on their targeted EIS.

Event Listener. iWay adapters can listen for events occurring on their targeted application server. For example, a new Siebel customer entry can trigger an event that causes the adapter to invoke Enterprise JavaBeans (EJB) within your application server, which in turn can invoke a service on CICS and SAP.

The iWay Connector for JCA can be used as a stand-alone application, or in conjunction with a supported application server such as a Sun™ J2EE Application Server, Oracle 9i Application Server Containers for Java, BEA WebLogic Server, Fujitsu Interstage Application Server, IBM WebSphere Application Server, or the Novell exteNd Application Server.



This documentation is not intended as a primer on JCA itself, except as required to illustrate the iWay Connector for JCA. For more information about JCA, see the *J2EE Connector Architecture Specification*, JSR016 (Java Community Process), Sun Microsystems, July 25, 2001.

The iWay Connector for JCA conforms to J2EE Connector Architecture (JCA) Specification version 1.0. This specification approaches the definition of a standard for application interaction with an Enterprise Information System (EIS). Before JCA, most EIS vendors offered vendor-specific architectures to provide connectivity between applications and their software; each program interacting with an EIS was required to be hand-tooled with a detailed knowledge of the peculiarities of the EIS.

Although common APIs such as ODBC and JDBC™ address application interaction with SQL data sources, the heterogeneous nature of EIS makes such APIs unwieldy. iWay Software provides a common interface to underlying EIS adapters. An access API that meets the requirements of both relational and packaged application sources required. JCA defines standard Java interfaces for simplifying the integration of applications with the EIS.

JCA addresses connection management, transaction management, and security. It includes these components:

- Application server
- Resource adapter
- Application

The resource adapter represents the interests of the underlying EIS. The application server is not strictly required. The application interacts with the resource adapter using what JCA calls *standard contracts*. Standard contracts define what interactions are to take place and how they appear. The contract between the application and the resource adapter is called the Common Client Interface (CCI). The resource adapter, in turn, interacts with the application manager under the Service Provider Interface (SPI), defining how the management of the resource adapter occurs. This includes:

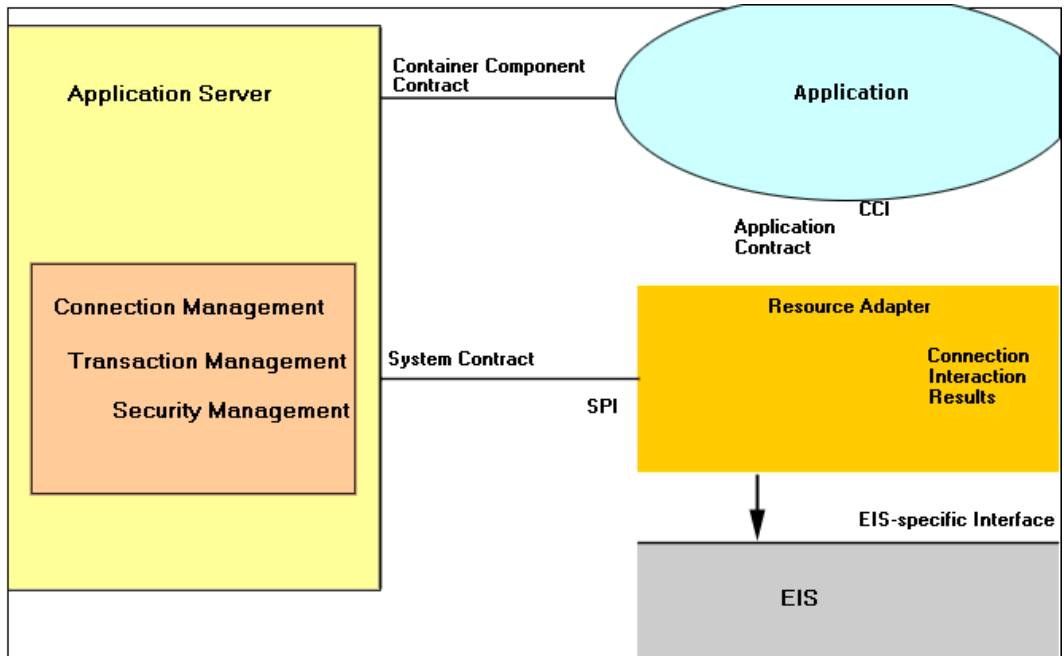
- Connectivity management
- Transaction demarcation
- Event listening (listeners receive notification of significant events; for example, a connection failure)
- Pooling of connections and other resources

In the normal course of events, the application server manages the transaction. First, the application uses a naming service to locate the appropriate resource adapter. The application server supplies the naming service, and so it recognizes that a request is being made to locate a resource adapter. In such a case, the application server interposes an intermediate object supplied by the resource adapter that interacts between the resource adapter and the application server. Through this intermediating object, the application server manages the items within the SPI contract below the awareness of the ultimate application.

During the application's use of the resource adapter, the application server manages the activity to the side, while it appears to the application that it is interacting directly with the resource adapter.

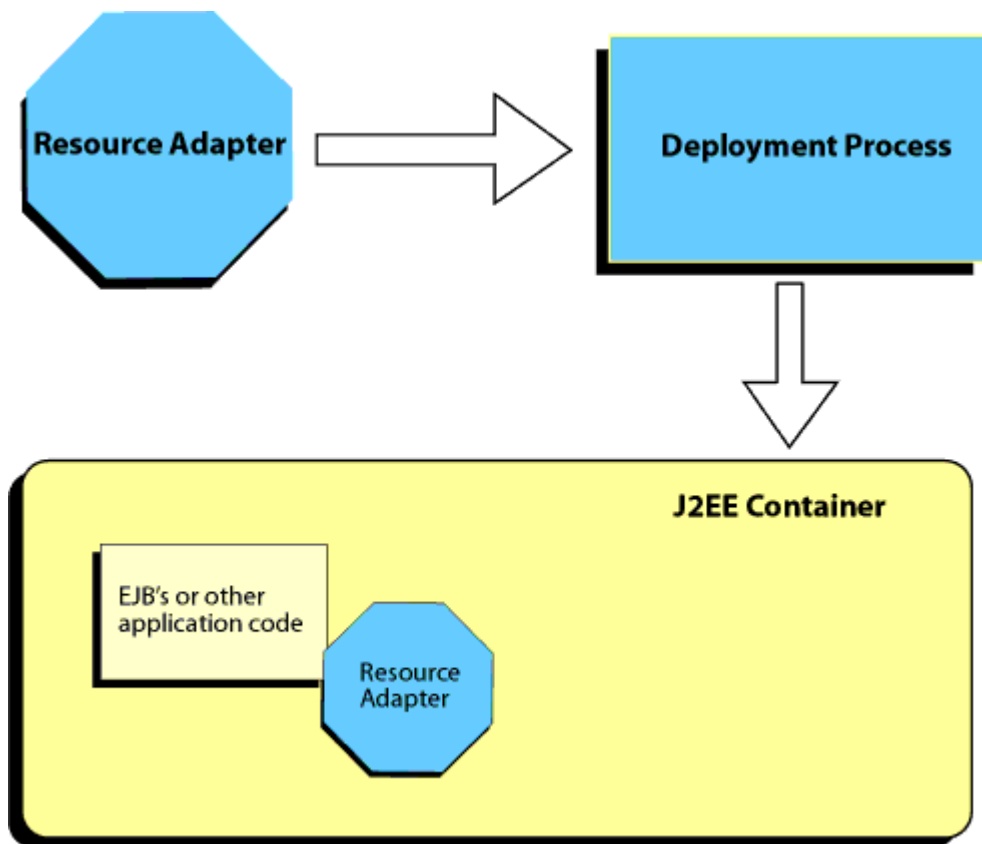
Distribution of the iWay Connector for JCA

The iWay Connector for JCA is a J2EE Connector Architecture resource adapter. It is distributed as both a standard Resource Adapter Archive (RAR) for deployment to the application server and as a JAR file for stand-alone application use. Thus, the iWay Connector for JCA can be employed in systems that are non-compliant, although services such as pooled connections are not available.



Deployment of the iWay Connector for JCA

The iWay Connector for JCA is a JCA resource adapter. For the connector to operate with an Enterprise JavaBean (EJB), it must be deployed to a Web application server. The RAR file is identified to the application server, which then uses the file as any other component. The connector's name is registered with the name service (JNDI) to enable the EJB (or servlet) to locate and instantiate it for use.



Using a JCA Resource Adapter

Using any JCA resource adapter (such as the iWay Connector for JCA) is, as of the 1.5 specification, a programming effort as well as an assembly effort. A JCA resource adapter appears to a programmer as two interacting parts. You can configure and serialize specification components with standard bean tools. These are specific to the adapter and require the programmer to understand the configuration properties that they offer.

Contract components meet stricter interface requirements and can be used by the JCA-compliant application exactly as described in the specification.

The adapter provides both the specification and contract components: the application programmer can write generic code to assemble the specification and contract components, and thus interact with the underlying Enterprise Information Systems (EIS). However, the tasks of preparing input and understanding output remain EIS-specific. The iWay Connector for JCA exposes the following iWay-specific components to the JCA application.

- **IWAFConnectionSpec.** A JavaBean encapsulating the properties required to perform a connection to the iWay service adapters.

You can use any standard bean tool to set the properties and serialize the bean.

Properties offered by the iWay Connector for JCA's connection specification are those required to control a local invocation of the iWay packaged adapters. The IWAFConnectionSpec has seven parameters:

- Adapter name
- Adapter configuration name
- Language
- Country
- User name
- Password
- Log level

The IWAFConnectionSpec supports connection pooling based on the above configuration parameters.

- **IWAFInboundConnectionSpec.** A JavaBean encapsulating the properties required to perform a connection to the iWay Event adapter channel.

You can use any standard bean tool to set the properties and serialize the bean.

Properties offered by the iWay Connector for JCA's connection specification are those required to control a local invocation of the iWay adapters. The IWAFInboundConnectionSpec has the following parameters:

- Adapter name
- Language
- Country
- Channel

- Disposition
- Log level
- **IWAFInteractionSpec.** A JavaBean encapsulating the properties required to manage one interaction with the adapter.

As is the case with the IWAFConnectionSpec, you can use this object with any bean tool.

The IWAFInteractionSpec uses the standard IndexedRecord to return the output.

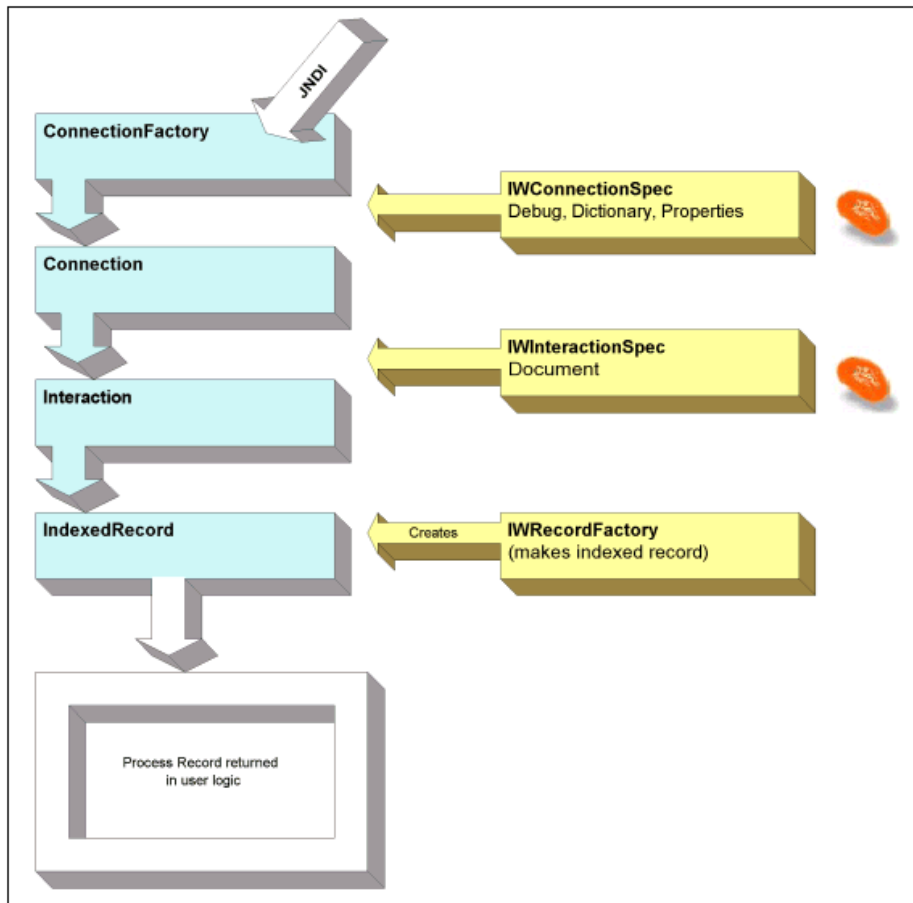
- **IWAFInboundInteractionSpec.** A JavaBean encapsulating the properties required to start and stop the iWay channel.

As is the case with the IWAFInbound ConnectionSpec, you can use this object with any bean tool.

- **IWAFRecordFactory.** An object used to construct records to be passed between the application and the adapter at design time.

The following diagram shows the relationships between the components in a simple stand-alone application. In such an application, transactions issues and security are ignored. When an application uses the iWay Connector for JCA within a hosted application server (such as BEA WebLogic Server), it is referred to as a *managed* application. From the programmer's viewpoint, the most significant difference is the use of the Java JNDI name search facility to locate the connection factory.

After the connection factory is located and instantiated, the application control flow appears much the same. The iWay Connector for JCA supports non-transactional and locally controlled transactional workflows, with coordination by the Application Server.



CHAPTER 2

Deploying the iWay Connector for JCA

Topics:

- Deploying to BEA WebLogic Server

The following topics describe how to deploy the iWay Connector for JCA to BEA WebLogic Server.

Deploying to BEA WebLogic Server

The iWay Connector for JCA is supported under the following WebLogic releases:

- BEA WebLogic Server V6.1 SP2 and higher
- BEA WebLogic Server V7.0
- BEA WebLogic Server V8.1

Deploying the iWay Connector for JCA to BEA WebLogic Server

To deploy the iWay Connector for JCA, you must first connect to BEA WebLogic Server using the Administration Console.

Procedure How to Deploy the iWay Connector for JCA to BEA WebLogic Server

To deploy a new iWay Connector for JCA to BEA WebLogic Server:

1. Start WebLogic Server Administration Console in a browser by entering the following URL:

<http://host:port/console/>

where:

[host](#)

Is the machine where WebLogic Server is running.

[port](#)

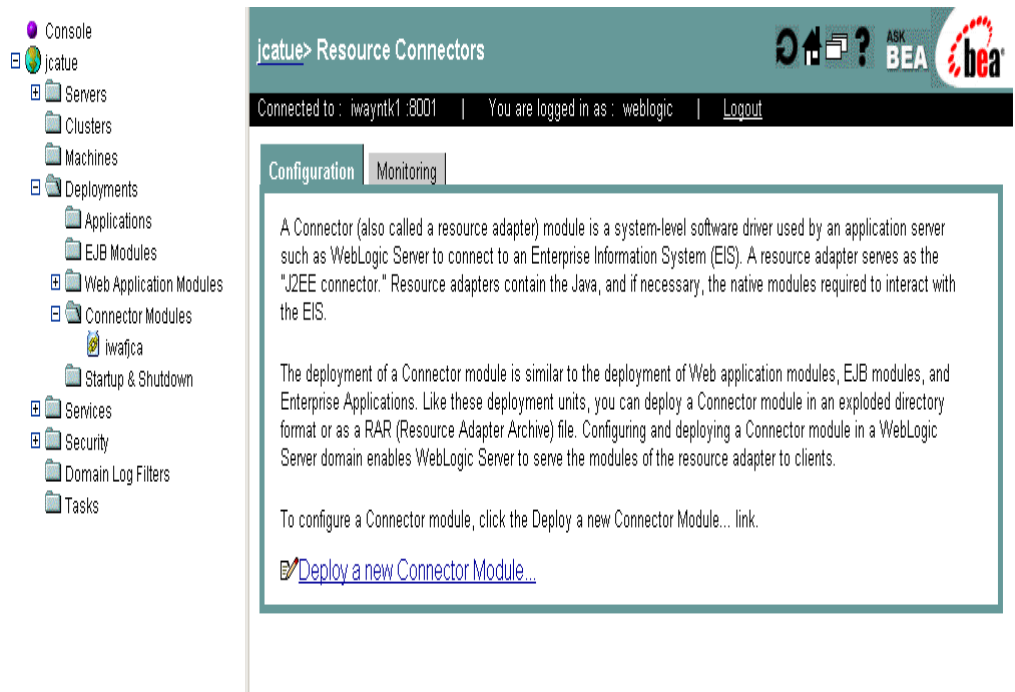
Is the port on which WebLogic Server is listening.

For example:

<http://localhost:7001/console/>

2. When prompted, enter the user name and password for the server.
The WebLogic Server Administration Console opens.
3. In the left pane, open *Deployments*.
4. Under Deployments, open *Connector Modules*.

The Resource Connectors pane opens.



5. Click *Deploy a new Connector Module*.
6. To upload the iwayfjca.rar file, click *Upload it through your browser*.
7. To select the iWay Connector for JCA, type the path of the iWay Connector for JCA RAR file in the text-entry field or click *Browse* to navigate in your file system.

The iwayfjca.rar file is located in

`iWayHome\etc\setup`

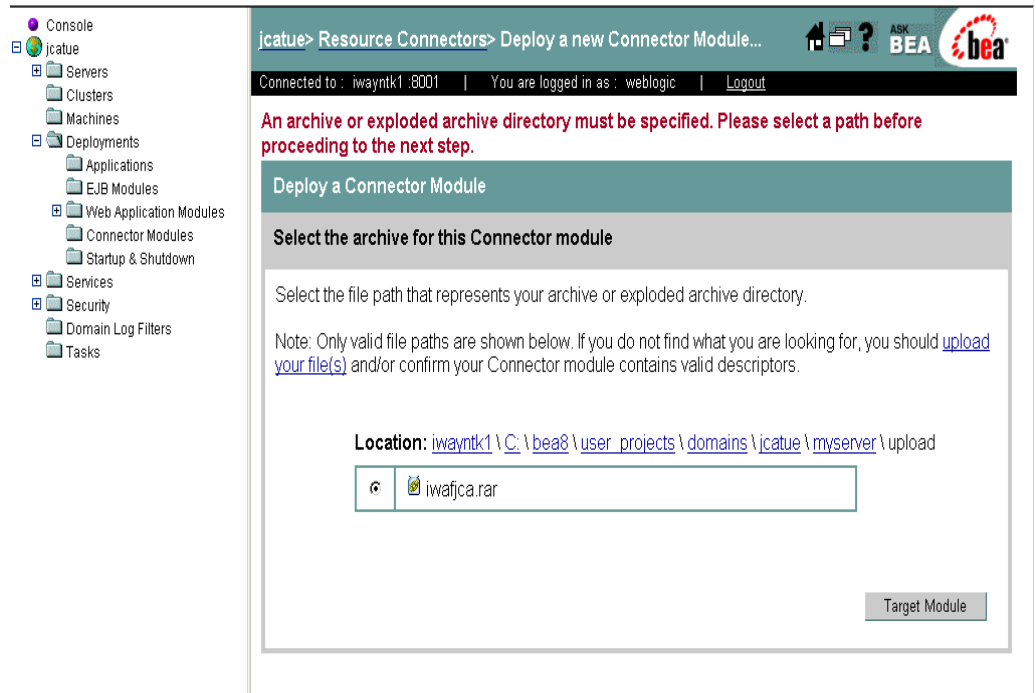
where:

`iWayHome`

Is the root directory where the iWay product is installed.

8. To begin the installation process for the iWay Connector for JCA, click *Upload*.
The RAR file is uploaded to the upload area.
9. Click *MyServer* and then, *Upload*.

The following pane opens.



- a. Select the *iwafjca.rar* file.
- b. Click *Target Module*.

A pane opens that provides details for the server where the RAR file will be deployed.

10. Click *Deploy* to complete the deployment process.

If deployment of the connector is successful, the following pane opens on the right.

The screenshot shows the BEA WebLogic console interface. On the left is a tree view with the following nodes: Console, jcatue, Servers, Clusters, Machines, Deployments, Applications, EJB Modules, Web Application Modules, Connector Modules, Startup & Shutdown, Services, Security, Domain Log Filters, and Tasks. The 'Deployments' node is expanded, and the 'Connector Modules' sub-node is selected.

The main pane displays the 'Resource Connectors' page for 'iwayfca'. The breadcrumb path is 'jcatue > Resource Connectors > iwayfca'. The status bar indicates 'Connected to : iwayntk1 :8001' and 'You are logged in as : weblogic'. The 'Deploy' tab is active, showing a message: 'This page allows you to view the deployment status of each Connector module, and to stop or redeploy individual Connector modules. You may also choose to stop and redeploy all Connector modules using the buttons at the bottom of the page. (To configure additional deployment targets for these Connector modules, click the Targets tab.)'

Module Status	Target	Target Type	Status of Last Action	Actions
Active	myserver	Server	Success	<input type="button" value="Stop"/> <input type="button" value="Redeploy"/>

Procedure How to View an iWay Connector for JCA Deployed to BEA WebLogic Server

To view an iWay Connector for JCA deployed to BEA WebLogic Server:

1. In the left pane of the Administration Console, select *Deployments*.
2. To view and confirm a list of deployed connectors in the table in the right pane, select *Connector Modules*.

The screenshot shows the BEA WebLogic Administration Console interface. On the left, the navigation tree is expanded to 'Deployments' > 'Connector Modules'. The main content area is titled 'jcatue> Resource Connectors' and shows the 'Configuration' tab selected. The page contains descriptive text about Connectors and a table listing the deployed connector modules.

Name	URI	Deployment Order	
iwafjca	iwafjca.rar	100	

For example, iwafjca appears in the Name column, and iwafjca.rar in the URI column.
The iWay Connector for JCA was successfully deployed to BEA WebLogic Server.

CHAPTER 3

iWay JCA Installation Verification Program

Topics:

- Overview of the IVP
- Deploying the IVP for BEA WebLogic Server

The iWay JCA Installation Verification Program (IVP) is a JSP™-based graphical tool for interacting with iWay adapters. This test tool is used to test the functionality of the J2EE-CA connector in the iWay adapter framework. There are several types of adapters available through this J2EE-CA connector.

Overview of the IVP

The iWay JCA Test Tool includes the following features:

- Support for execution of iWay Service requests.
- Monitoring of iWay JCA events.

The iWay JCA Installation Verification Program (IVP):

- Determines which iWay adapters are installed.
- Reads the iWay JCA configuration repository.
- Loads the configurations for each configured adapter.
- Executes iWay Service requests for a given adapter.

The IVP provides tools that enable you to test application performance early in the development cycle. This allows enough time to make architectural changes and implementation changes, reducing risk early in the cycle, and avoiding problems in final performance tests.

The iWay Connector for JCA includes deployable code and sample files. The sample files help you integrate the iWay JCA solution into the J2EE application and then test it. The following topic describes how to deploy and use the IVP.

The topic describes how to:

- Deploy the IVP.
- Modify the IVP to use other configurations.
- Use the IVP to execute a service request.
- Use the IVP to monitor iWay events.

Deploying the IVP for BEA WebLogic Server

The iWay Connector for JCA includes sample code that enables you to test iWay service and event adapters. The following procedure describes how to deploy the iWay JCA Installation Verification Program (IVP) to BEA WebLogic Server.

Procedure How to Deploy the IVP

To deploy the IVP:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

The following window opens.



The screenshot shows the BEA WebLogic Server Administration Console login interface. At the top, a teal header bar contains the text "Administration Console" and "BEA WebLogic Server 8.1". Below this, the main title "WebLogic Server Administration Console" is displayed in a large, bold, teal font. Underneath the title, a subtitle reads "Sign in to work with the WebLogic Server domain **jcawed**". The login form consists of two input fields: "Username:" with the text "weblogic" entered, and "Password:" which is empty. A "Sign In" button is located to the right of the password field.

- a. Type a user name and password.
 - b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from Deployed Resources.
3. Click *Deploy a New Web Application*.
4. Use the location link to navigate to the location of the exploded WAR file.

The WAR file is located in:

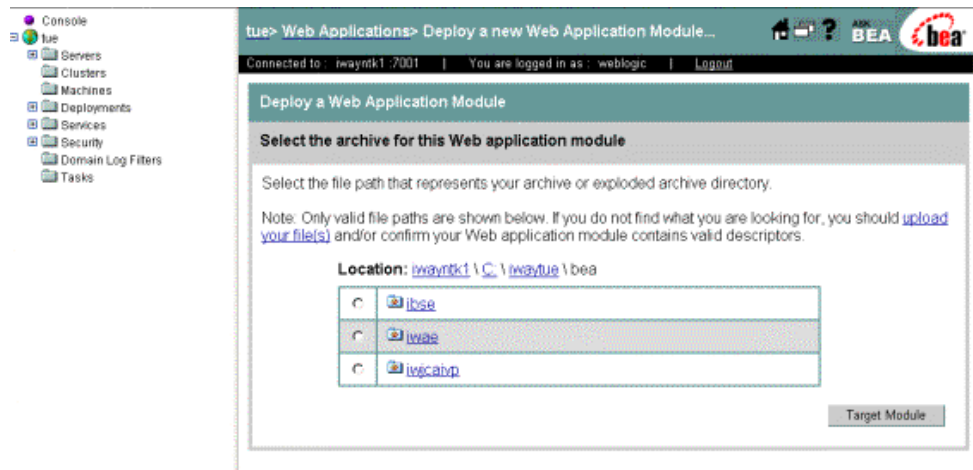
[iWayHome\BEA](#)

where:

[iWayHome](#)

Is the root directory where the iWay product is installed.

The following window opens.



5. Select *iwjcaivp*.
6. Click *Target Module*.

If deployment is successful, a window similar to the following appears.



Modifying the IVP for BEA WebLogic

The iWay JCA Installation Verification Program (IVP) is provided as an exploded Web application. It contains a web.xml file in the WEB-INF subdirectory that controls the configuration information for the IVP. This file determines whether the IVP obtains the connection information in a managed or non-managed fashion.

With BEA WebLogic Server, you can deploy an application in either exploded or archive form. An application in exploded form has a full directory structure.

The server manages exploded and archived applications the same way, so you can choose which form to use. Deploy your application in exploded form if:

- You are still testing your application.
- You want to modify static files (such as JSP or HTML files).

If the Web application is exploded, you can modify the JSP file and redeploy the application.

You can modify the web.xml file to use the iWay JCA verification program with configurations other than the default supplied configuration. The iWay installation program modifies the web.xml during the installation to include the location where the iWay adapters are installed. This value is included in the iWayhome tag. The web.xml file assumes that you are using a base configuration. This value is included in the iWayConfig xml tag.base.

Procedure How to Deploy the IVP When Running a Non-Default Configuration

To deploy the IVP when running a non-default configuration:

1. Edit the following web.xml file in the WEB-INF subdirectory:

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

<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <display-name>IWAF JCA IVP</display-name>
  <description>
    Installation verification tool for IWAF JCA.
  </description>
  <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>
  <context-param>
    <param-name>iway.home</param-name>
    <param-value>c:\Program Files\iway55</param-value>
    <description>
      ONLY USED IF IWAY.JNDI NOT SET.
      Absolute path of iway installation directory.
    </description>
  </context-param>
  <context-param>
    <param-name>iway.config</param-name>
    <param-value>base</param-value>
    <description>
      ONLY USED IF IWAY.JNDI NOT SET.
      configuration name
    </description>
  </context-param>
  <context-param>
    <param-name>iway.loglevel</param-name>
    <param-value>DEBUG</param-value>
    <description>
      ONLY USED IF IWAY.JNDI NOT SET.
      Log level: DEBUG FATAL ERROR INFO WARN
    </description>
  </context-param>
```

```
<session-config>
<session-timeout>
30
</session-timeout>
</session-config>
<welcome-file-list>
<welcome-file>
index.jsp
</welcome-file>
<welcome-file>
index.html
</welcome-file>
<welcome-file>
index.htm
</welcome-file>
</welcome-file-list>
</web-app>
```

where:

ibm.jndi

Is the connection factory name for the iWay Connector for JCA. The connection factory name under BEA WebLogic is eis/IWAFConnectionFactory.

or

ibm.home

Is the root directory where the iWay product is installed.

ibm.config

Is the iWay configuration to be used at run time.

ibm.loglevel

Is the desired level of tracing. Valid levels are debug, info, and error.

The iWay JCA Test Tool attempts to connect to the adapter through JNDI.

If a value for iWayJNDI is set, the iWay JCA IVP obtains connection information from the JNDI connection factory.

2. If you do not want to connect using JNDI, modify the *ibm.jndi* name to point to the iWay Connection factory or modify the *ibm.home* *ibm.config* to reflect the appropriate parameters.
3. To redeploy the IVP, log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

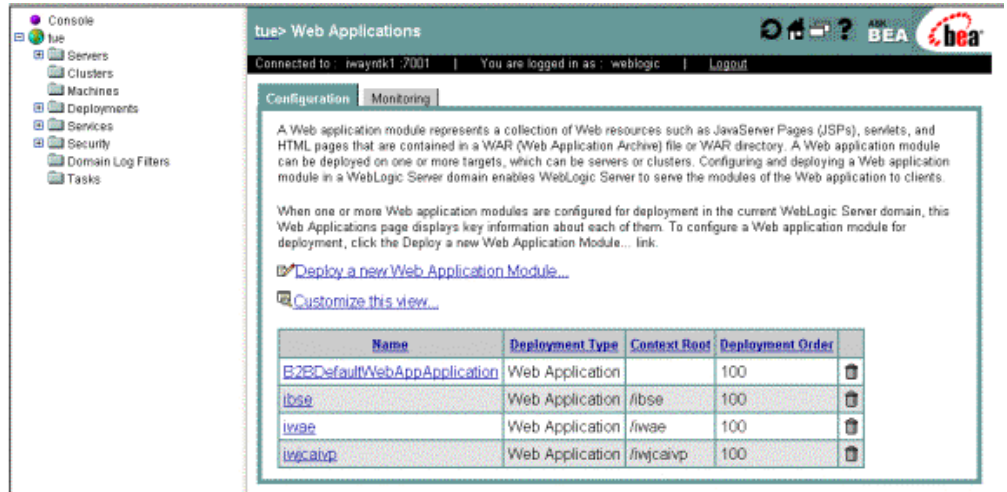
The following window opens.



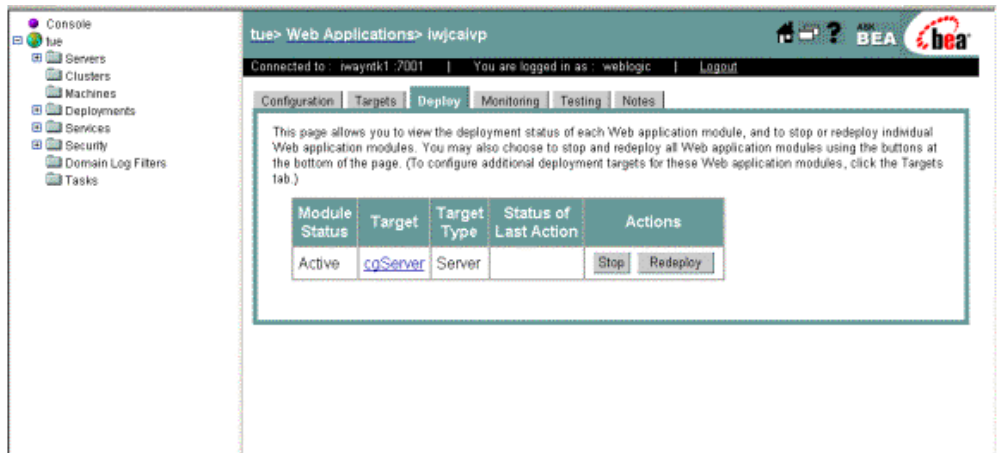
The screenshot shows the BEA WebLogic Server Administration Console login interface. At the top, there is a teal header bar with the text "Administration Console" and "BEA WebLogic Server 8.1". Below this, the main content area has a title "WebLogic Server Administration Console" and a subtitle "Sign in to work with the WebLogic Server domain jcwed". There are two input fields: "Username:" with the text "weblogic" entered, and "Password:" which is empty. A "Sign In" button is located to the right of the password field.

- a. Type a user name.
 - b. Type the password associated with the user name.
 - c. Click *Sign In*.
4. When the WebLogic Server console opens, select *Web Application Modules* from Deployed Resources.
 - a. Click *Deploy a New Web Application Module*.
 - b. To navigate to the location of the exploded WAR file, click the location link.

The following window opens.



- Click the *iwjcaivp* link.



- Select the *Deploy* tab.
- To ensure that the changes made to the web.xml file take effect, click the *Redeploy* button.

Procedure How to Run the IVP

To run the iWay JCA Installation Verification Program (IVP):

- Open a browser.

2. Enter the following URL:

<http://host:port/iwjcaivp>

where:

[host](#)

Is the IP address or DNS name where the application server is installed.

[port](#)

Is the port on which the application server is listening. For a BEA WebLogic Server, the default is 7001.

[iwjcaivp](#)

Is the context root where the Web application is deployed.

For WebLogic, the default URL is

<http://localhost:7001/iwjcaivp>

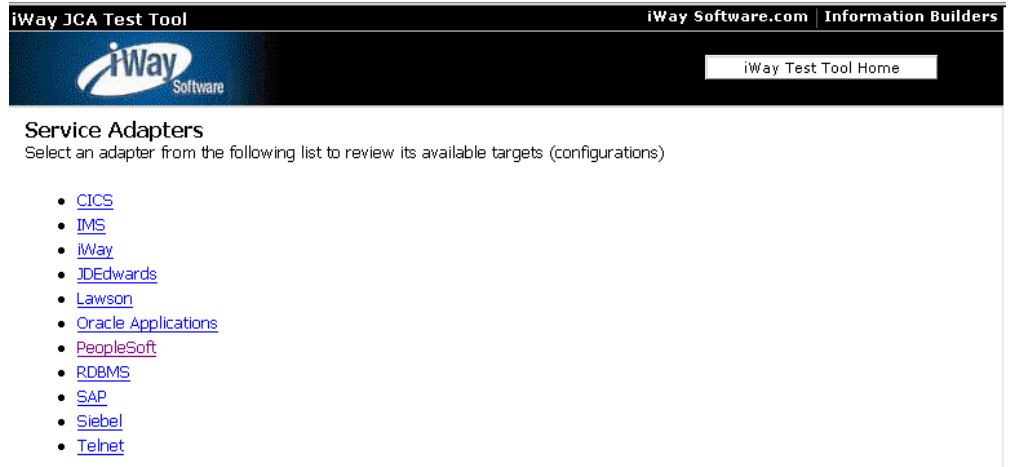
The iWay JCA Test Tool window opens.



Procedure How to Test the iWay Service Adapters Using the IVP

1. To display the available adapters, click the *Service adapters* link.

The following window opens.

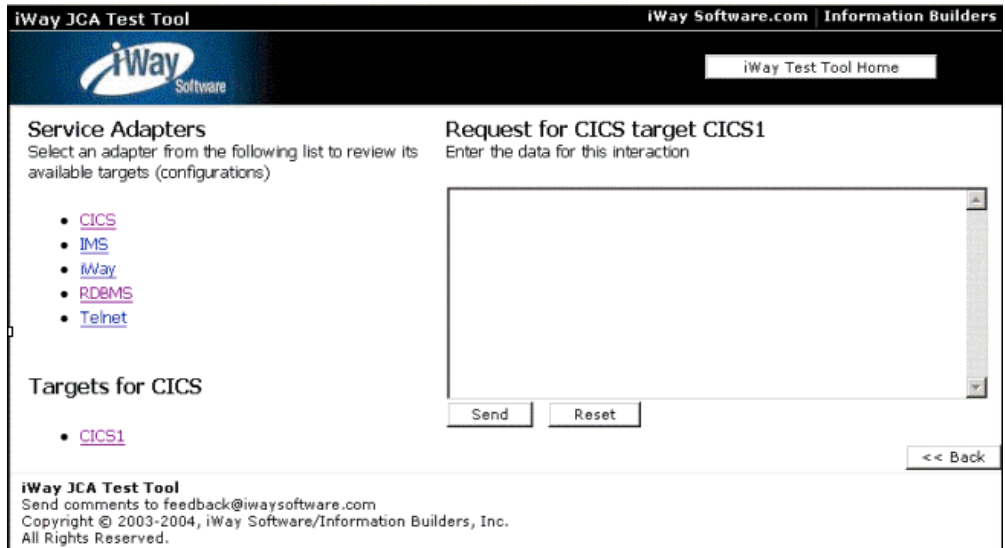


2. Click the adapter that you want to test.

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, CICS1, configured for the iWay Adapter for CICS.

3. Click the desired target, for example, *CICS1*.



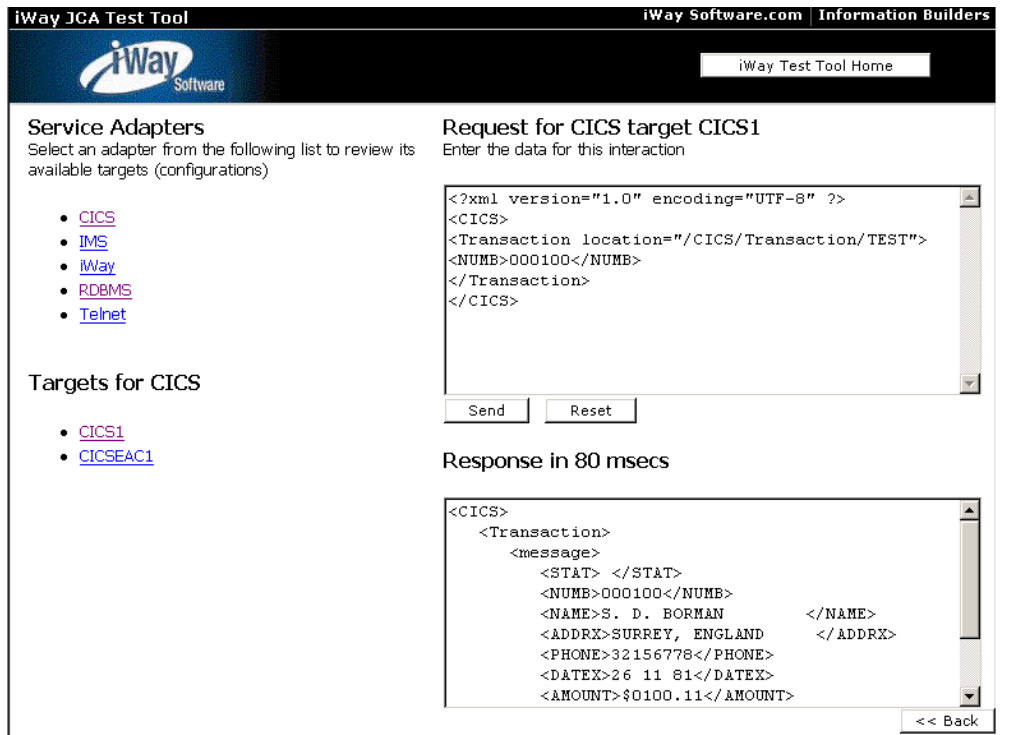
4. Enter a request document built from the iWay Request schema, for example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
<Transaction location="/CICS/Transaction/TEST">
<NUMB>000100</NUMB>
</Transaction>
</CICS>
```

The following results appear:

```
<CICS>
  <Transaction>
    <message>
      <STAT> </STAT>
      <NUMB>000100</NUMB>
      <NAME>S. D. BORMAN </NAME>
      <ADDRX>SURREY, ENGLAND </ADDRX>
      <PHONE>32156778</PHONE>
      <DATEX>26 11 81</DATEX>
      <AMOUNT>$0100.11</AMOUNT>
      <COMMENTS>*****</COMMENTS>
    </message>
  </Transaction>
</CICS>
```

The window looks similar to the following:



CHAPTER 4

Using the iWay Connector for JCA

Topics:

- Accessing the iWay Adapter for Telnet From BEA WebLogic Server
- Accessing a CICS Transaction From BEA WebLogic Server
- Accessing an IMS Transaction From BEA WebLogic Server
- Accessing a J.D. Edwards OneWorld Business Function From BEA WebLogic Server
- Accessing a Lawson Business Function From BEA WebLogic Server
- Accessing an Oracle Applications Business Process From a BEA WebLogic Application Server
- Accessing a PeopleSoft Business Function From BEA WebLogic Server
- Accessing an SAP Business Object From BEA WebLogic Server
- Accessing a Siebel Business Object From BEA WebLogic Server
- Issuing a VSAM Parameterized Query From BEA WebLogic Server

The following topics illustrate how to access an Enterprise Information System (EIS) using the iWay Connector for JCA. Each topic includes procedures that describe how to use the connector with a particular adapter.

For information on deploying and configuring the iWay Connector for JCA, see the *iWay 5.5. Installation and Configuration* manual.

Accessing the iWay Adapter for Telnet From BEA WebLogic Server

The iWay Adapter for Telnet captures mainframe 3270 type screens and their output to create an adapter transaction module, which is an XML file that can be run within the BEA WebLogic Server environment. The iWay Adapter for Telnet enables you to view your online application as an object that can be run as a service that executes the existing mainframe transaction. If created appropriately, online screen information is returned as an answer set. In other words, the adapter wraps your 3270 mainframe screens, data entered, and keystrokes inside a transaction module to transform the output of the screens into an answer set.

The following topics describes the steps required to use the iWay Adapter for Telnet in the BEA WebLogic Server environment.

The steps describe how to:

- Create a transaction module for your 3270 application.
- Use the iWay Application Explorer to create schemas against which your application validates.
- Deploy the sample servlet.
- Test the transaction module with the sample servlet.

3270 Emulation

The iWay Adapter for Telnet (built on top of a standard 3270 emulation product) offers programming that translates your legacy application programs into new user

interfaces. This means that you can continue to use the logic and data associated with the programs, even though these legacy programs were written to communicate with input or output devices and user interfaces that now may be obsolete. On occasion, some emulation products are referred to as screen scraping or advanced terminal emulation. A screen scraping program takes the data from the legacy program that is formatted for an older type of terminal, such as an IBM 3270 display, and reformats it.

iWay Adapter Transaction Modules

The iWay Adapter for Telnet contains advanced terminal emulation functionality along with the ability to quickly and easily create adapter transaction modules.

These adapter transaction modules contain the captured screens, keystrokes, entered data, parameters, and output parameters based on the options chosen during an Application Explorer development session. These modules are the procedures that are executed as a service within the BEA WebLogic Application Server framework.

The adapter transaction module is an internal XML file created by the iWay Application Explorer. The transaction module can be treated as a remote procedure call (RPC) that is executed through an XML request document. Its associated service request and response schemas are created using Application Explorer.

Creating Adapter Transaction Modules

The following topics describe how to create adapter transaction modules and include:

- Sample CICS transaction
- Creating an adapter transaction module
- Capturing the contents of a table
- Transaction module XML request document

A transaction module is the encapsulated procedure that contains all of the information required to execute an existing online mainframe application. You must create the transaction module by navigating through the mainframe screens that comprise the online transaction. While navigating, you *recognize* each screen, which enables the recording of all data and keystrokes entered.

Sample CICS Transaction

The following topics illustrate an actual mainframe CICS application that is running at iWay Software. As mentioned earlier, it is crucial that you know the exact online transaction that you want to run within the BEA WebLogic Server environment. A particular transaction must be selected, and its particular input and output parameters must be known, so that you can navigate through all the necessary screens. In this example, the main VTAM screen is displayed for the host name IBICICS. On the mainframe, the CICS system is accessed using an APPLID (pronounced apple ID). The APPLID is EDBGM010.



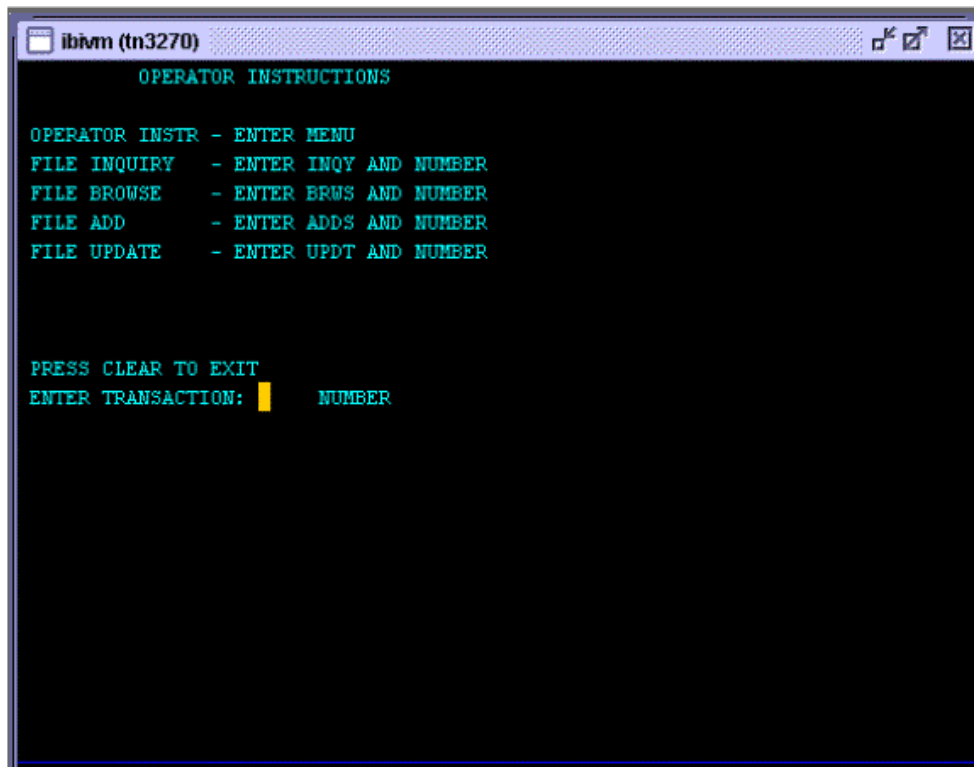
Procedure How to Create a CICS Transaction

To create a CICS transaction:

1. Type *LOGON APPLID(EDBGM010)* to enter the CICS region.
A blank CICS window opens.
2. Type *MENU* anywhere in the window to invoke the IBM CICS MENU Application.

The MENU application is a common CICS application that is simple to use and allows you to maintain account information. There are inquiry, browse, add, and update transactions within this application.

For the purposes of this document, create a transaction module against the INQY transaction.



3. Type *INQY* in the ENTER TRANSACTION field and *000100* in the NUMBER field.
4. Press *Enter*.

The following window displays the record detail for account number 000100:

```
FILE INQUIRY

NUMBER: 000100
NAME:   J. D. BORMAN
ADDRESS: SURREY, ENGLAND
PHONE:  32156778
DATE:   26 11 81
AMOUNT: $0100.11
COMMENT: *****

PRESS ENTER TO CONTINUE
```

The detail information appears. The transaction module you created returns some of this information in an answer set format. After the transaction module is created, the Application Explorer creates request and response service schemas. This enables a service request to execute.

The particular service (the transaction module) you create accepts a six byte account number and returns detail information, such as the number, name, and address for the particular account number.

Creating an Adapter Transaction Module

This following topic illustrates the steps required to create an adapter transaction module called CICSINQ. The transaction module CICSINQ accepts a six byte account number to use as input to the CICS inquiry transaction. The purpose of this module (service) is to return detail information, in particular, the account number, the name, and address.

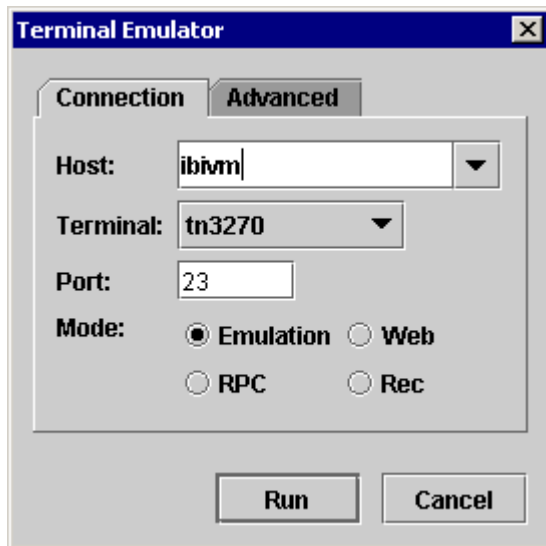
The transaction module connects to the mainframe system, logs on to CICS, and executes the IBM Menu application executing the INQY (inquiry) transaction. The module does this in exactly the same way as if a mainframe user were entering the information directly in CICS.

Now that you are familiar with the transaction on the mainframe, you are ready to create an adapter transaction module. To do this, use the iWay Terminal Emulator and the iWay Application Explorer.

Procedure **How to Establish a Connection to the Mainframe**

To establish a connection to the mainframe:

1. Start the iWay Emulation Adapter by using the iWay Telnet command from the `iway55\tools telnet` directory.
2. Select the session folder.
3. To start the emulator session, select *Tools*, and then *Emulator*.



4. Enter the appropriate parameter values:

where:

Host

Is the host address of the OS/390 machine where CICS is running. It is the connection name.

Port

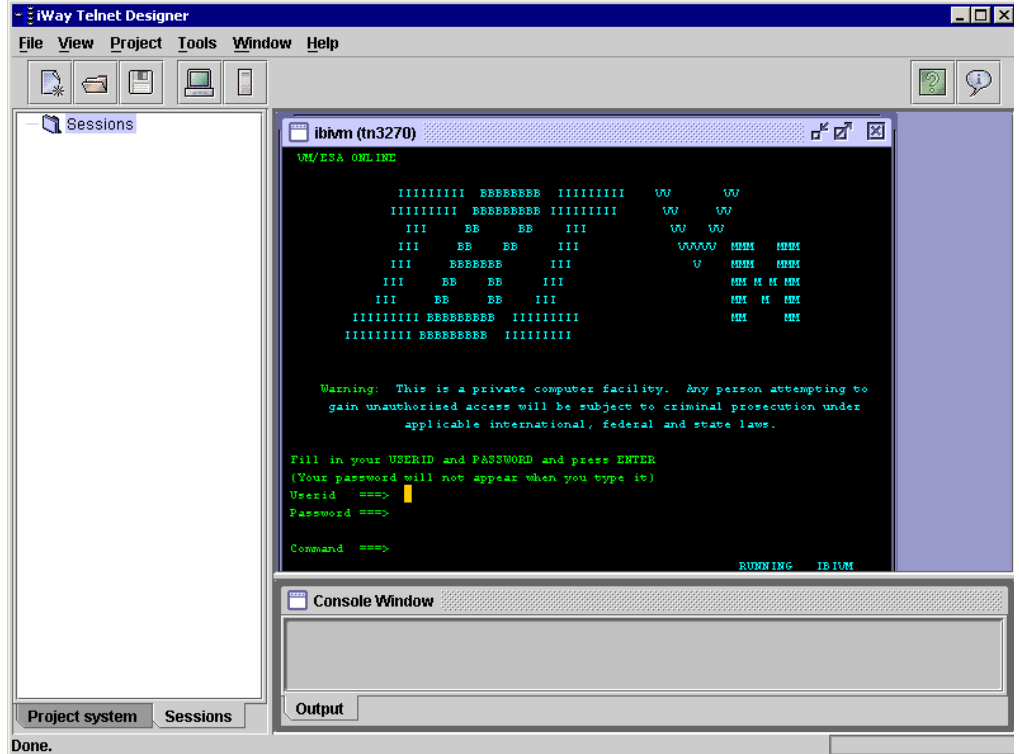
Is the Telnet port number. The default value is 23.

5. Select the RPC radio button.
6. Click the *Advanced* tab and select the *Extended Attributes* check box.

This enables several language options.

7. Click the *Connection* tab, and then click *Run* to connect to the mainframe and start the emulation session.

The following window opens, which represents the initial VTAM mainframe session:



You must identify each screen so that the adapter *recognizes* it when it encounters it during execution of the service; in this case, the service that runs the transaction module that you are creating.

The Add Parameter option is used when you set up an input parameter for the transaction. The input parameter(s) are received from the XML request when running the service. Similarly, the Add Metadata option is used to create output fields to be displayed as part of the response from the transaction module. The Add Parameter and Add Metadata options are illustrated in the following topics.

Adding a Screen Identifier

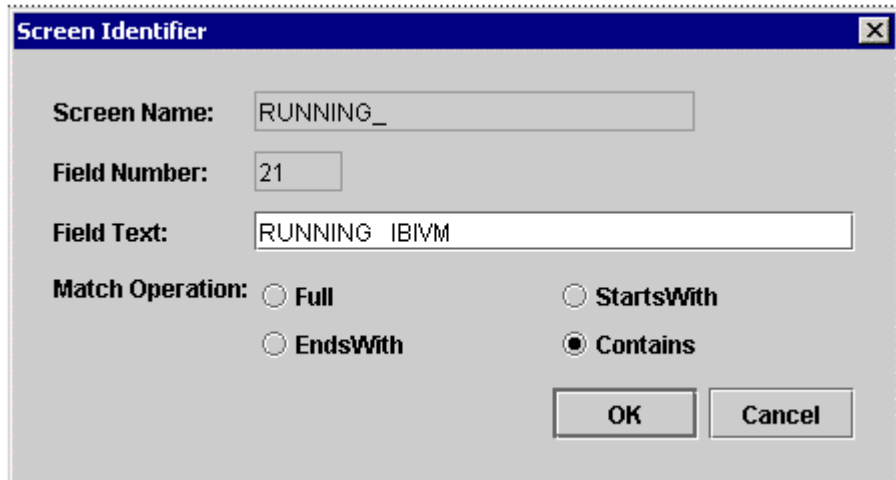
The Add Screen Identifier option enables you to add patterns (filters) based on screen fields. You can click the region of the screen that uniquely identifies the current screen.

The emulator highlights the field and a dialog box opens, enabling string matching. You enter the screen name and click *OK* to update the screen database.

By adding a screen identifier, the adapter records the screen and the key data entered in a local database. After the screen is identified, it is recognized by the adapter.

1. Select *Add Screen Identifier*.

The Screen Identifier dialog box opens.



The screenshot shows a Windows-style dialog box titled "Screen Identifier". It has a blue title bar with a close button (X). The dialog contains the following fields and controls:

- Screen Name:** A text box containing "RUNNING_".
- Field Number:** A text box containing "21".
- Field Text:** A text box containing "RUNNING IBIVM".
- Match Operation:** Four radio buttons are arranged in two rows:
 - Top row: ☐ Full, ☐ StartsWith
 - Bottom row: ☐ EndsWith, ☒ Contains
- Buttons:** "OK" and "Cancel" buttons are located at the bottom right.

You can name the screen and select match operations.

When a match is made, the adapter proceeds with the keystroke you pressed, such as the enter key or PFKEY. At run time, the screen is processed accordingly.

2. Click *OK*. The screen is recognized and added to the internal database.

Note: You must add screen identifiers for all screens. Since the database contains the screens, you must recognize the screen only once.

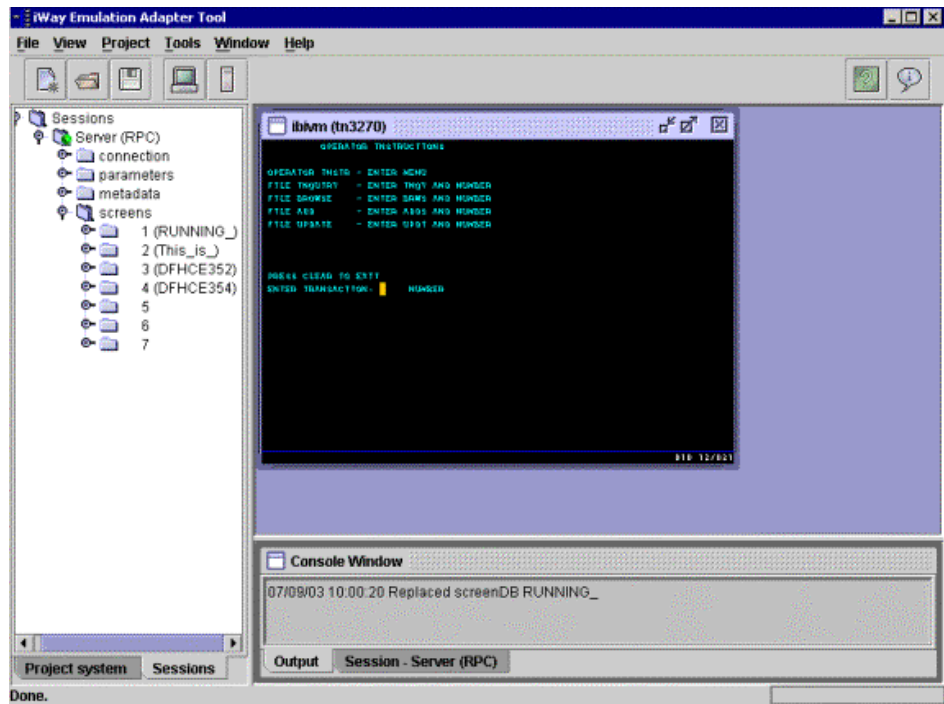
3. Continue navigating through the application.

Ensure you identify all screens in the process.

You need not identify a blank screen, such as the one you encounter after you navigate to CICS.

4. Type the CICS transaction MENU and press *Enter*.

When the following menu screen opens, remember to add a screen identifier to identify the screen.



Adding Parameters and Metadata

When you encounter a screen that contains an input field, you must use the Add Parameter option for each parameter to be used as input to the transaction module. When you encounter a screen that contains an output field, you must use the Add Metadata option for each output field that you want returned.

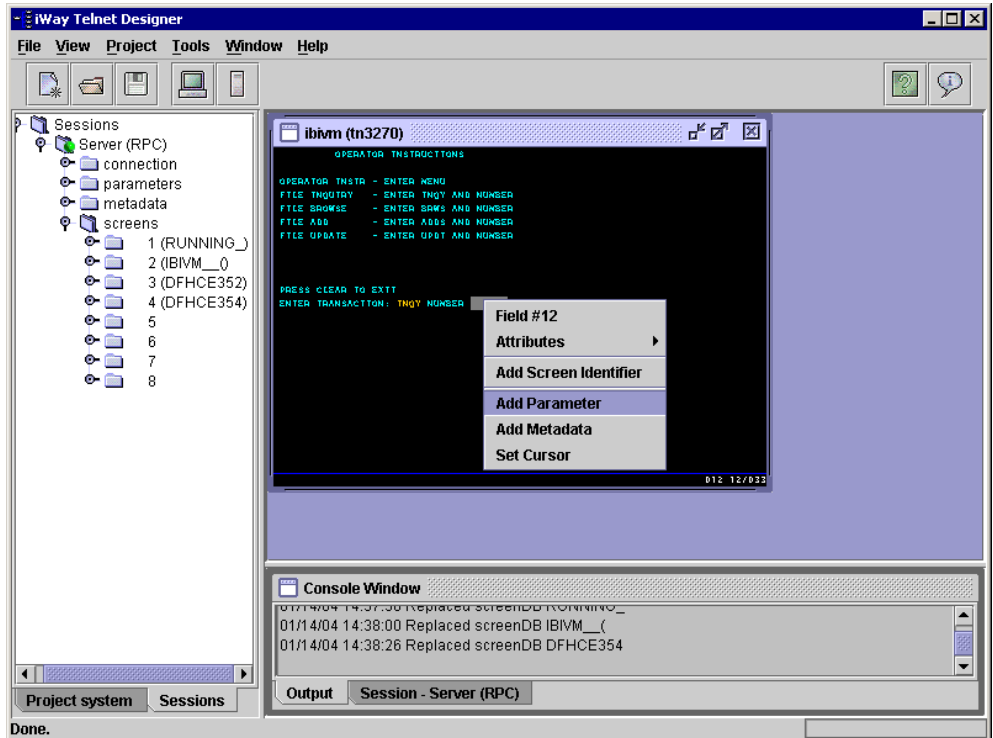
For the transaction module being created, INQY is the transaction, and NUMBER is an input parameter. As a result, the NUMBER field is an input parameter passed in the request.

Procedure How to Add Parameters and Metadata

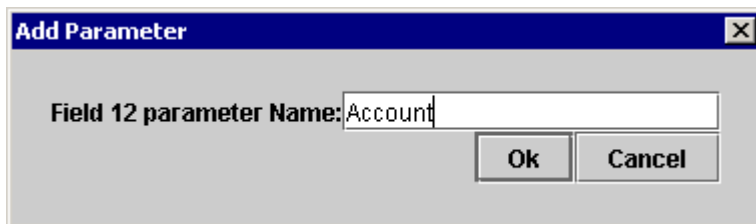
To add parameters and metadata

1. Type the INQY transaction in the ENTER TRANSACTION field.

2. Move the cursor to the NUMBER field, right-click, and select *Add Parameter*.



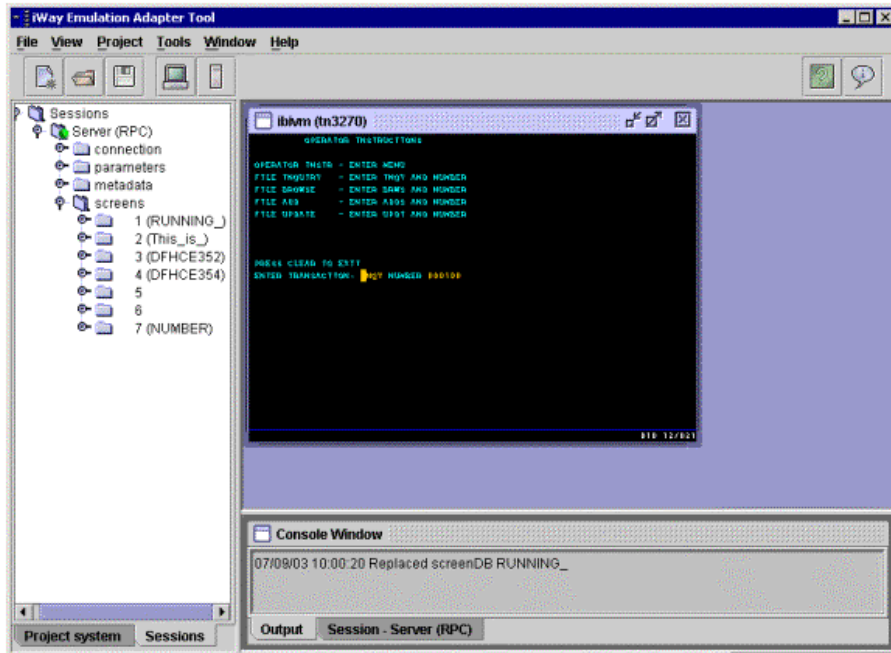
The Add Parameter dialog box opens:



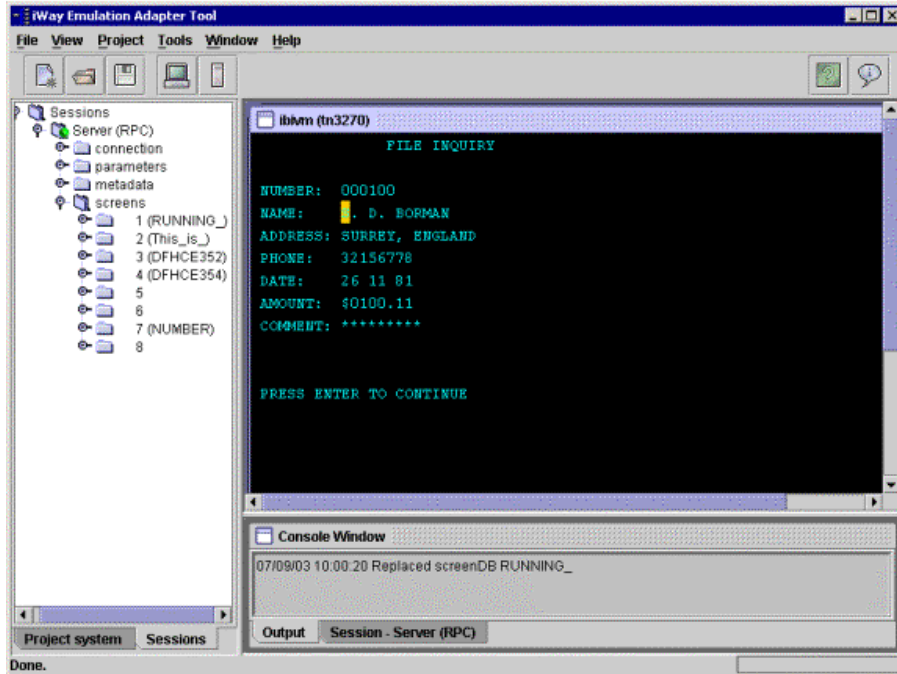
3. Type a name (for example, ACCOUNT).
4. Click *OK*.

Continue navigating through the following screen. In this case, the screen is the detail screen.

5. Type an existing account number (for example, 000100).

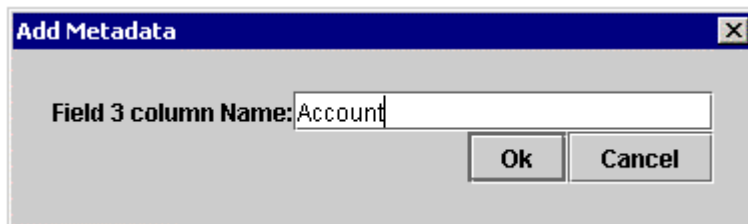


6. Press *Enter* to view the detail screen.



The detail screen contains information that you must describe as output. The output fields are returned as part of the answer set by the transaction module you created.

7. Right-click one of the output fields (for example, the NUMBER field) and select *Add Metadata*.

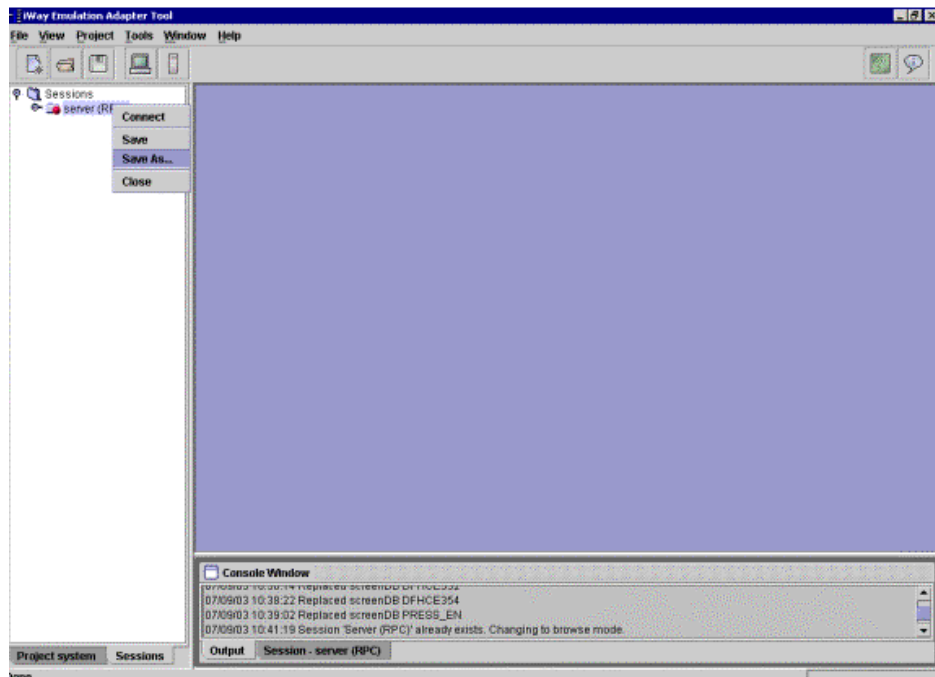


8. Type a name (for example, Account).
9. Click *OK*.
10. Repeat these steps for the remaining two fields you want the module to return as output.

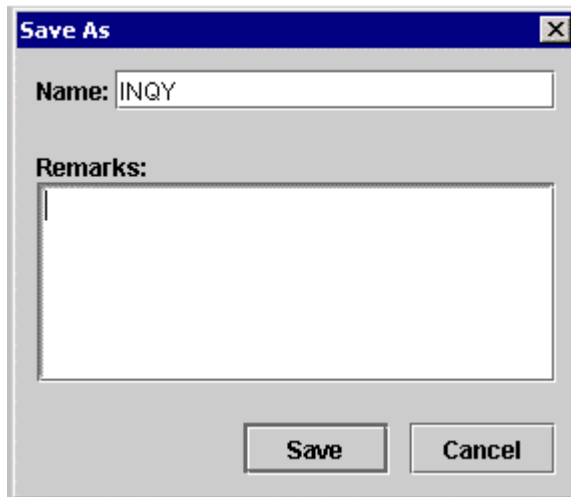
When you are finished assigning output fields, you must navigate through the screens to log off the CICS region. The transaction module signs off CICS in the same manner.

The emulator must be closed before saving the created XML file.

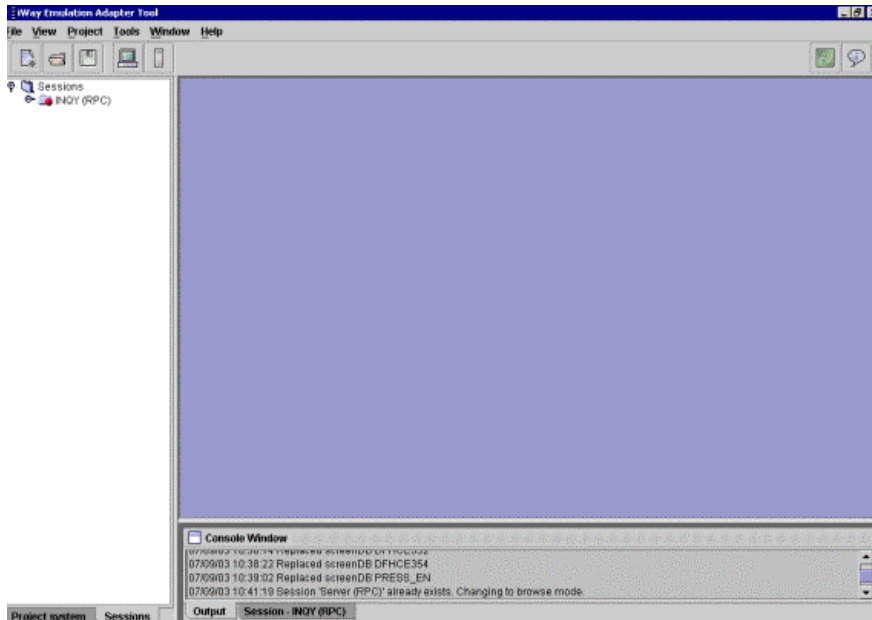
- 11.** Right-click the Server (RPC) icon and select *Save As*.



The Save As dialog box opens:



12. Type a name (for example, INQY) and click *Save*.



The transaction module called INQY is saved. Notice the module name called INQY in left pane of the window.

Transaction Module XML Request Document

An XML request document, used as input to the request, is created during the development process when creating a transaction module. You can use the request XML to test the adapter. For the module INQY, an XML request file called INQY.XML has been added to the folder C:\iway55\tools\telnet\projects\system\rpcs

The following is the sample request file that is created:

```
<sequence NAM
E='INQY' MODE='RPC' VERSION='2.0' >

  <parameters>
    <parameter NAME="Account" LENGTH='6'>000100</parameter>
  </parameters>
  <metadata>
<segment NAME="Screen8" >
  <column NAME="NUMBER" LENGTH='6' POSITION='3' />
  <column NAME="NAME" LENGTH='20' POSITION='6' />
  <column NAME="ADDRESS" LENGTH='20' POSITION='9' />
</segment>
</metadata>

  <connection HOST='ibivm' PORT='23' EMULATION='tn3270' EXTENDED='OFF'
LANGUAGE='Cp037' TIMERTIMEOUT='15' />

  <screen ID='1' NAME='RUNNING_' METHOD='execute' >
    <filter FIELD='21' TYPE='MATCHOP_CONTAINS' VALUE='RUNNING IBIVM' />
    <action NAME='setFieldContent' FIELD='20' VALUE='d vtam' />
    <attention NAME='Session.AID_ENTER' VALUE='0' />
  </screen>

  <screen ID='2' NAME='IBIVM__(' >
    <filter FIELD='1' TYPE='MATCHOP_CONTAINS' VALUE='IBIVM (CNM04)
ACF/VTAM Ver 4 Rel 2.0' />
    <action NAME='setFieldContent' FIELD='5' VALUE='logon
applid(edbgm010)' />
    <attention NAME='Session.AID_ENTER' VALUE='0' />
  </screen>

  <screen ID='3' NAME='Signon_t' >
    <filter FIELD='1' TYPE='MATCHOP_CONTAINS' VALUE='Signon to CICS' />
    <attention NAME='Session.AID_F3' VALUE='4' />
  </screen>

  <screen ID='4' NAME='DFHCE354' >
    <filter FIELD='1' TYPE='MATCHOP_CONTAINS' VALUE='DFHCE3543 You have
cancelled your sign-on request. Sign-on is terminated.' />
    <attention NAME='Session.AID_ENTER' VALUE='0' />
```

```

</screen>

<screen ID='5' >
  <attention NAME='Session.AID_CLEAR' VALUE='1' />
</screen>

<screen ID='6' TYPE='unformatted' >
  <action NAME='setContent' FIELD='-1' VALUE='MENU' />
  <attention NAME='Session.AID_ENTER' VALUE='0' />
</screen>

<screen ID='7' >
  <action NAME='setFieldContent' FIELD='10' VALUE='INQY' />
  <action NAME='setFieldContent' FIELD='12' LABEL='Account'
VALUE='$LABEL$' />
  <attention NAME='Session.AID_ENTER' VALUE='0' />
</screen>

<screen ID='8' >
  <action NAME='getFieldContent' FIELD='3' LABEL='NUMBER'
VALUE='Screen8' />
  <action NAME='getFieldContent' FIELD='6' LABEL='NAME'
VALUE='Screen8' />
  <action NAME='getFieldContent' FIELD='9' LABEL='ADDRESS'
VALUE='Screen8' />
  <attention NAME='Session.AID_ENTER' VALUE='0' />
</screen>

<screen ID='9' >
  <attention NAME='Session.AID_CLEAR' VALUE='1' />
</screen>

<screen ID='10' TYPE='unformatted' >
  <action NAME='setContent' FIELD='-1' VALUE='CESF logoff' />
  <attention NAME='Session.AID_ENTER' VALUE='0' />
</screen>

<screen ID='11' NAME='IBIVM__(' >
  <filter FIELD='1' TYPE='MATCHOP_CONTAINS' VALUE='IBIVM (CNM04)
ACF/VTAM Ver 4 Rel 2.0' />
</screen>
</sequence>

```

Connecting to Telnet From iWay Application Explorer

The following procedure describes how to connect to CICS from the iWay Application Explorer (iAE). The procedure uses a sample application servlet supplied with the iWay Connector for JCA.

Procedure How to Connect to Telnet From iWay Application Explorer

To connect to Telnet from iWay Application Explorer:

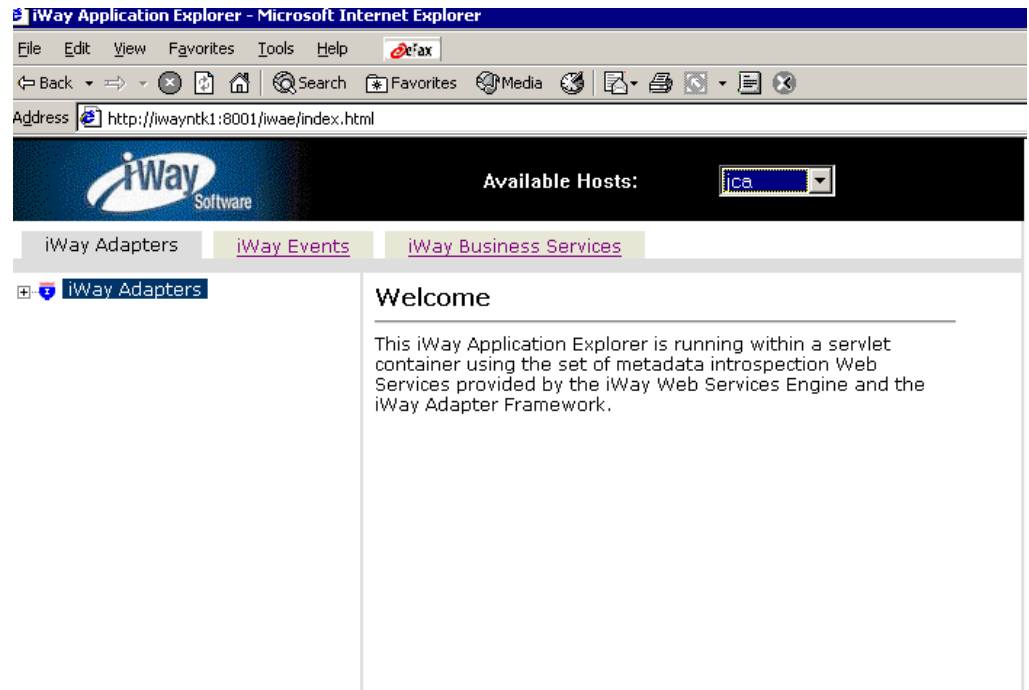
1. Follow the installation steps to install the iWay Application Explorer, as described in the iWay installation manual.
2. Determine the hostname and port for your application server, and then type the following URL:

<http://hostname:port/iwae/index.html>

For WebLogic with the default port and domain on your localhost, go to:

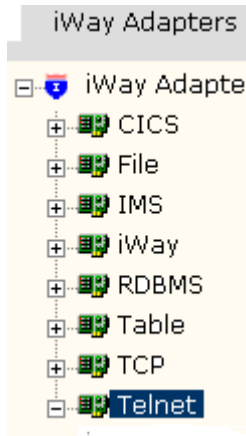
<http://localhost:7001/iwae/index.html>

Application Explorer opens.



3. Select *ijca configuration* from the drop-down list of Available Hosts. For information on how to configure available hosts, see the installation manual.

You can expand the iWay Adapter folder to display the list of iWay service adapters currently installed on your system.

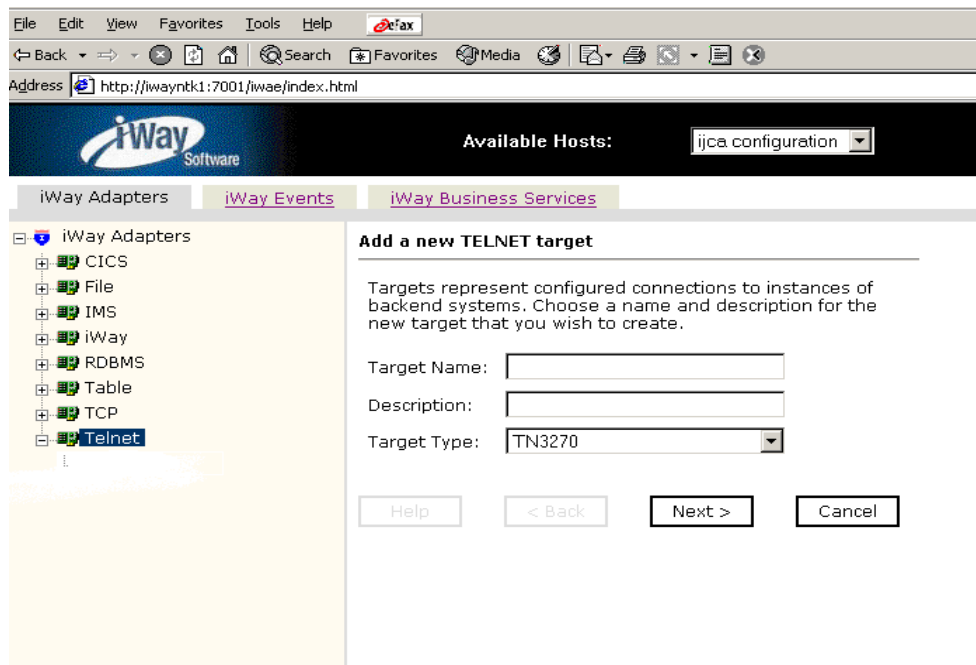


Procedure How to Establish a Connection to Telnet From iWay Application Explorer

To create a new connection to your Telnet system:

1. Expand the iWay Adapters node in Application Explorer.
2. Right-click the *Telnet* node.

3. In the left pane, move your pointer over *Operations*, and select *Define a new target*.



4. In the Add a new Telnet target dialog box:
 - a. In the Target Name field, type a name for the connection, for example, CICS1.
The name is used to build a repository entry as well as to identify the connection.
 - b. In the Description field, type a description for the target name you just created. For example, Connection to CICS1 on zos14.
 - c. In the Target Type drop-down list, select *TN3270*.

5. Click *Next*.

The connection name is verified for the system. If you entered an invalid instance name, a new dialog box opens and prompts you for an instance name again.

The Set connection info dialog box appears.

6. Type the connection parameters to create a new connection to Telnet.

You can obtain this information from the MVS or VM systems programmer. This information should be the same for all transactions and messages in a single CICS region.

The following table lists the parameters.

Note: The fields marked with an asterisk (*) are required.

Parameter	Description
Host	The DNSNAME or IP Address of your mainframe system.
Terminal	3270

7. Click *Finish*.

The newly created connection is added under the Telnet service adapter. The configuration information is stored in the repository for the configuration you defined at installation time.

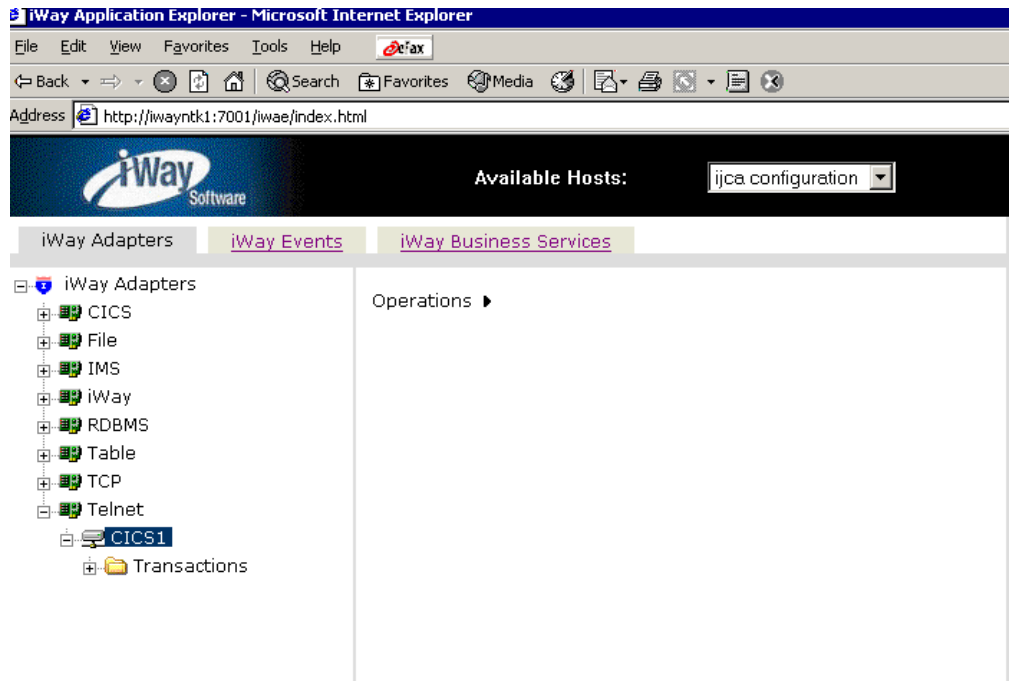
Procedure How to Connect to Telnet From iWay Application Explorer

To connect to Telnet from iWay Application Explorer:

1. To connect to Telnet, move your pointer over *Operations*, and select *Connect*.

The Connect to CICS1 dialog box appears, populated with the values you entered for the connection parameters.

2. Verify your connection parameters.
3. Click *OK*.



iWay Application Explorer connects and loads the Telnet metadata.

Now you can create schemas for transactions. The schemas are stored in

`c:\iWay55\config\base\Telnet\CICS1`

where:

`cics1`

Is the symbolic session name.

Procedure How to Import a 3270 transaction

To import a 3270 transaction:

1. In the left pane, expand the connection name to display CICS1.
2. Click *Transactions*, move your pointer over *Operations*, and select *Import Transaction*.
3. Type *inqy.xml* to import the newly created transaction.
Note: Currently the inqy.xml file must reside in the BEAHOME directory.
4. Click *Import*.

Deploying and Running the Sample Servlet

The iWay Connector for JCA includes sample code that enables you to test iWay service and event adapters. This topic describes how to configure and deploy the iWay JCA Test Tool to the BEA WebLogic Server.

Procedure How to Deploy the Sample Servlet

To deploy the sample servlet:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

The following window opens.



Administration Console
BEA WebLogic Server 8.1

WebLogic Server Administration Console
Sign in to work with the WebLogic Server domain **jcwed**

Username:

Password:

- a. Type a user name and password.
 - b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from *Deployed Resources*.
3. Click *Deploy a New Web Application*.

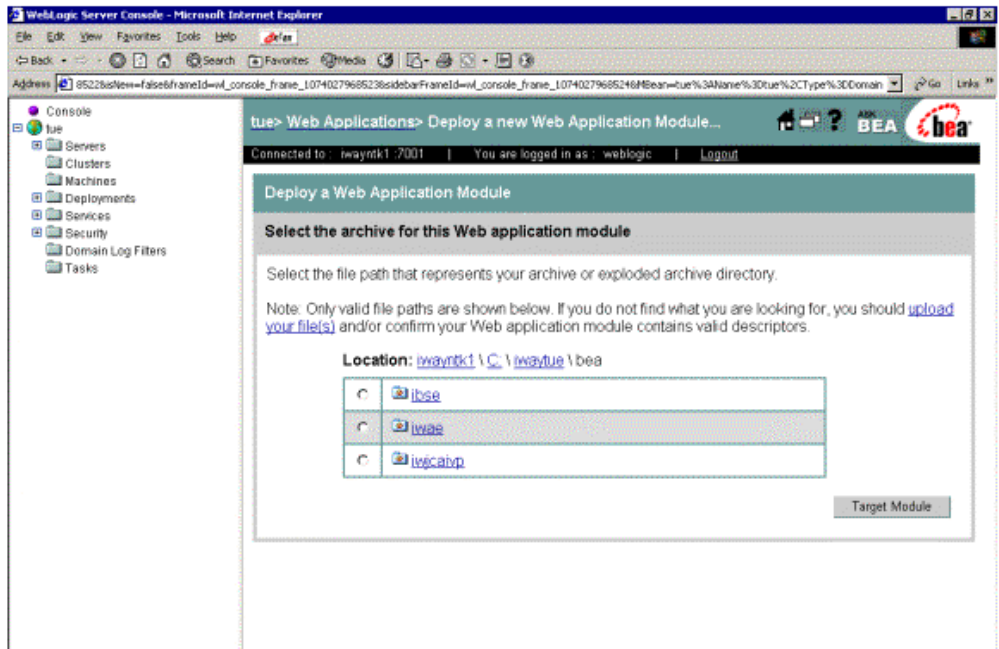
Use the location link to navigate to the location of the exploded WAR file. The WAR file is located in the *iWayHome*\BEA subdirectory.

where:

[*iWayHome*](#)

Is the root directory where the iWay product is installed.

The following window opens.



4. Select *iwjcaivp*.
5. Click *Target Module*.

If deployment is successful, a window similar to the following opens:



Procedure How to Run the Sample Servlet

To run the sample servlet:

1. Open a browser and enter the following URL:

<http://host:port/iwjcaivp>

where:

[host](#)

Is the IP address or DNS name where the application server is installed.

[port](#)

Is the port on which the application server is listening. For a BEA WebLogic Server, the original default is 7001.

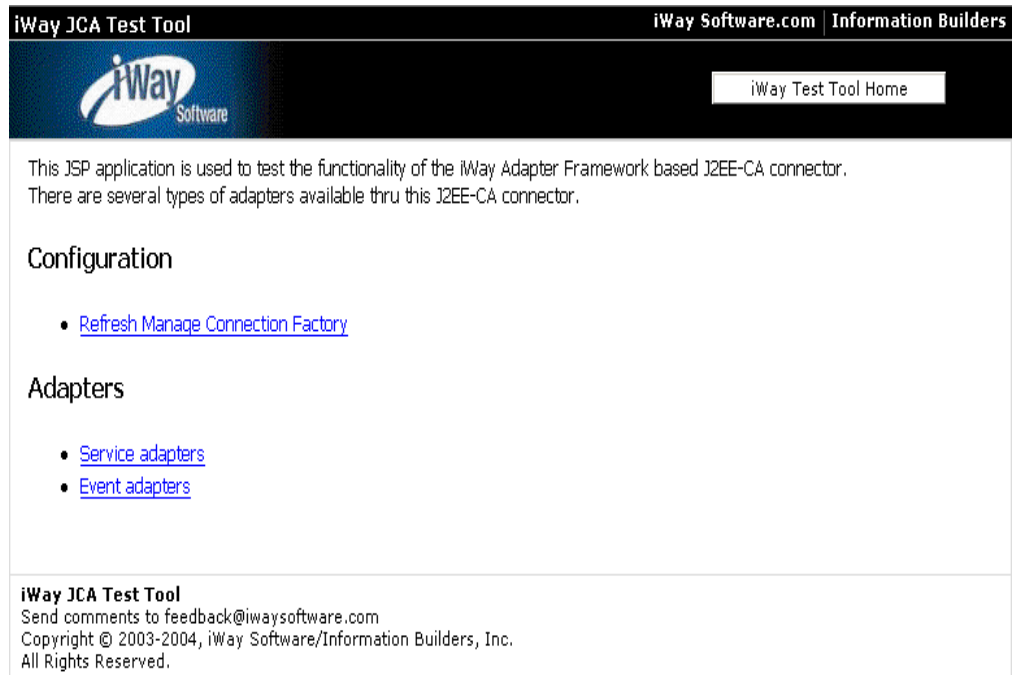
[iwjcaivp](#)

Is the context root to where the Web application is deployed.

For WebLogic, the default URL is

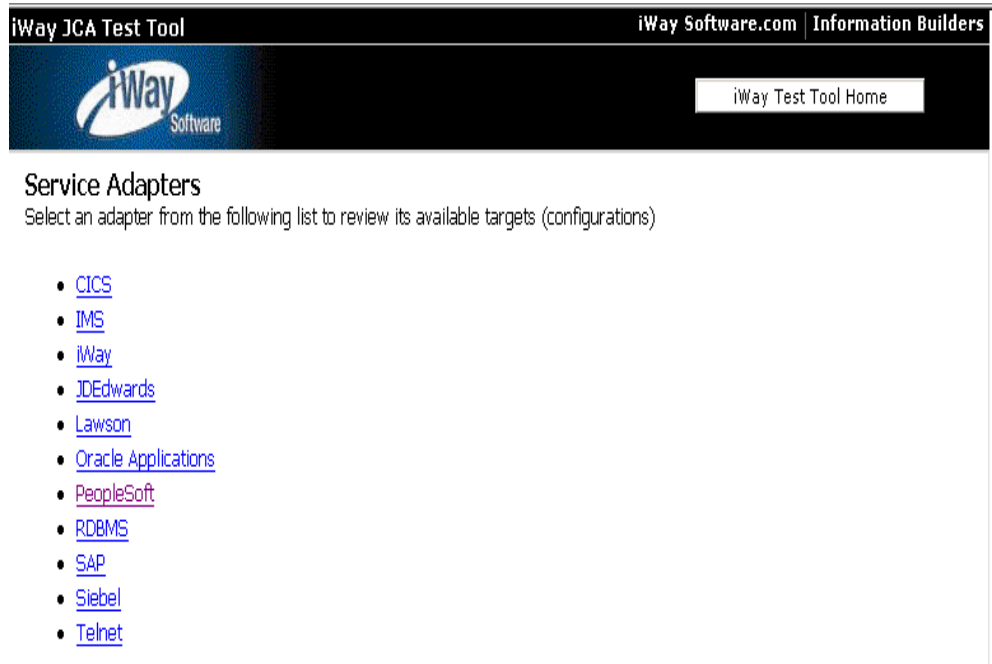
<http://localhost:7001/iwjcaivp>

The iWay JCA Test Tool window opens.



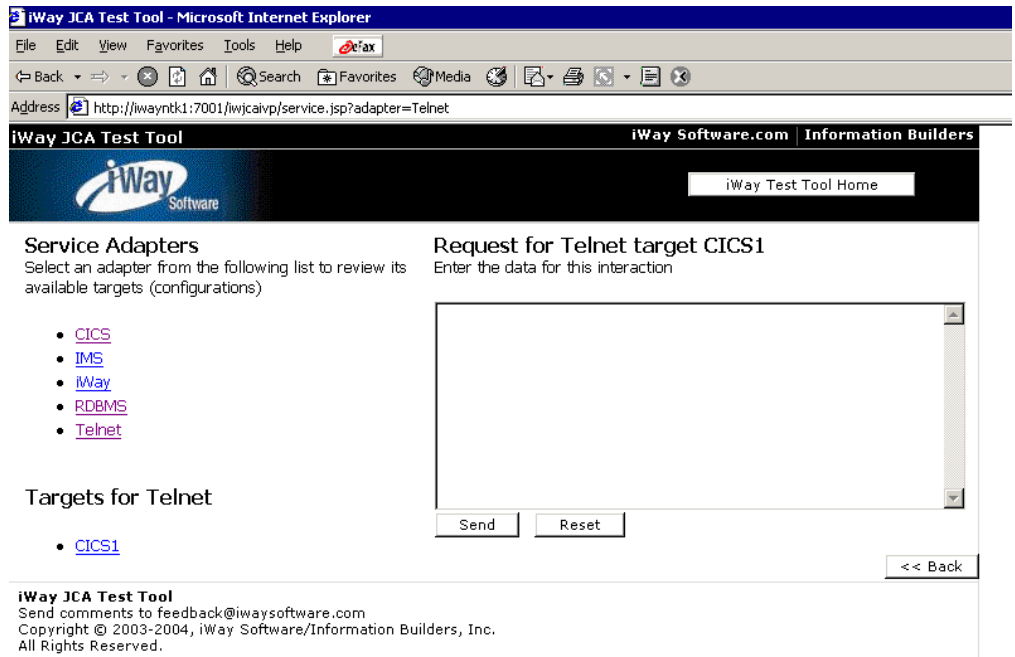
For information about the iWay JCA Test Tool, see Chapter 3, *iWay JCA Installation Verification Program*.

2. Click the *Service adapters* link to display the available adapters.



3. Click *Telnet*.

Note: If you have multiple target adapters defined, you must select CICS1. If you only defined one target, the following window opens.

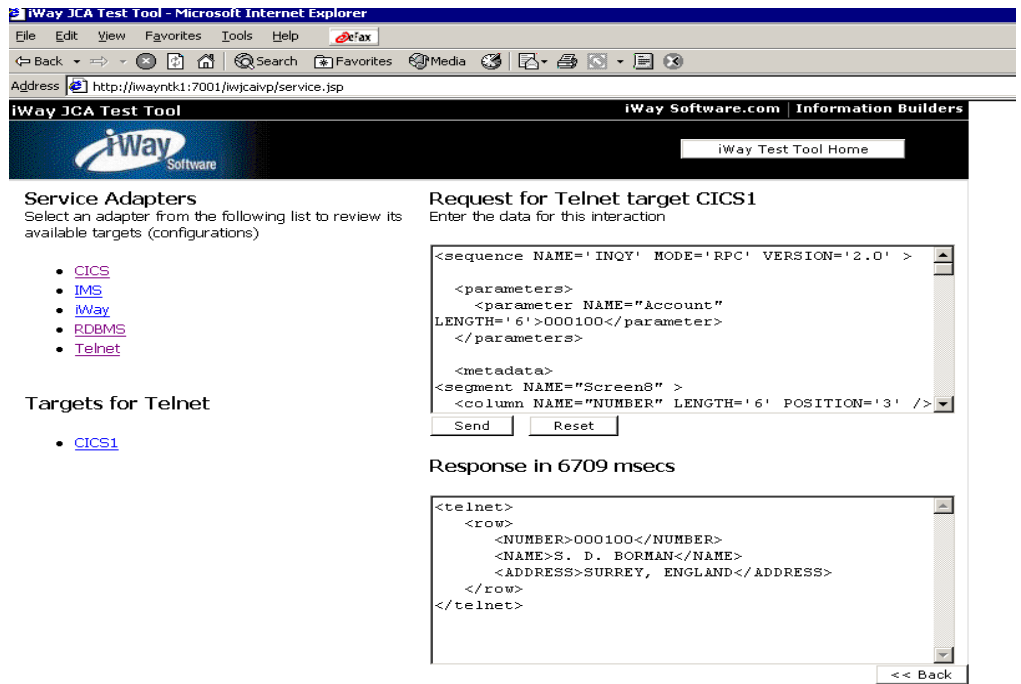


Copy the INQY.xml file from your iway55\tools\telnet\program\system\rpcs folder into the input area provided.

4. Click *Send*.

The iWay Adapter for Telnet connects to the mainframe application and returns the following results:

```
<?xml version="1.0" encoding="UTF-8"?>
<telnet>
  <row>
    <Account>000100</Account>
    <Name>S. D. BORMAN</Name>
    <Address>SURREY, ENGLAND</Address>
    <Balance>$0100.11</Balance>
  </row>
</telnet>
```



Accessing a CICS Transaction From BEA WebLogic Server

The following topics describe how to access a CICS transaction using the iWay Connector for JCA and the iWay Adapter for CICS hosted by BEA WebLogic Application Server.

The following procedure describes how to connect to CICS from iWay Application Explorer (IAE). The procedure uses a sample application servlet supplied with the iWay Connector for JCA.

Procedure How to Connect to CICS From iWay Application Explorer

To connect to CICS from iWay Application Explorer:

1. Follow the installation steps to install the iWay Application Explorer as described in the iWay installation manual.
2. Determine the hostname and port for your application server, and then type the following URL:

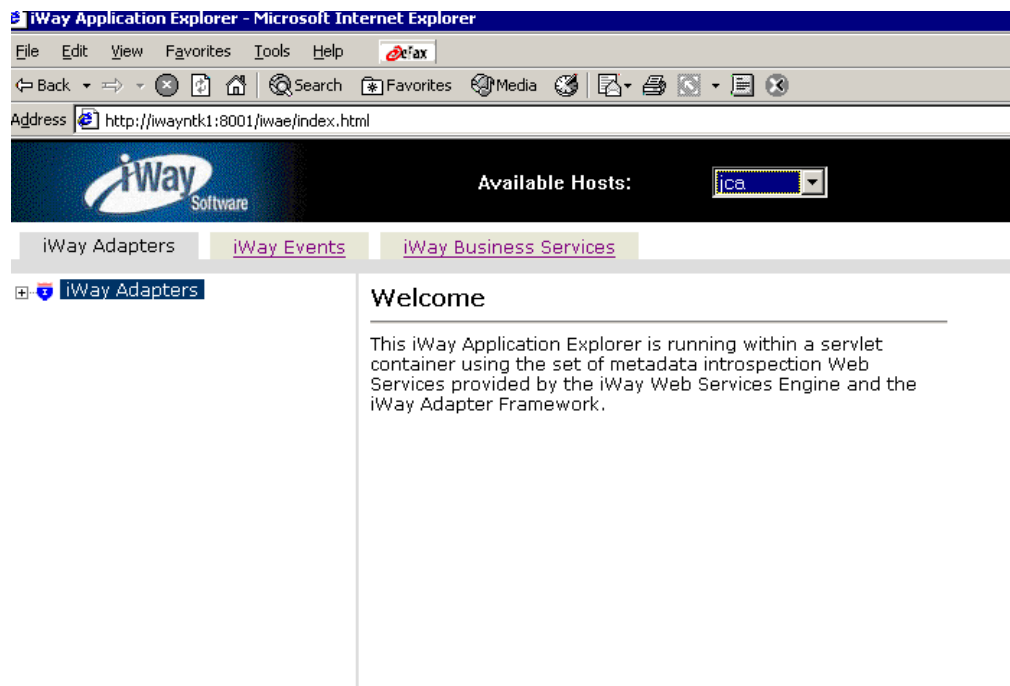
<http://hostname:port/iwae/index.html>

For WebLogic with the default port and domain on your localhost, go to:

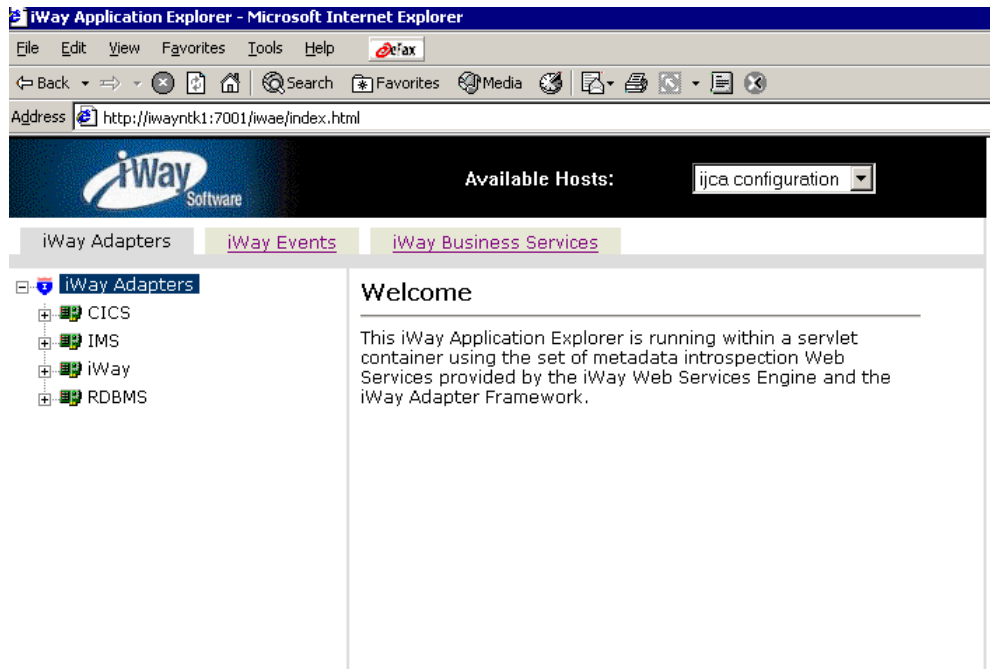
<http://localhost:7001/iwae/index.html>

Application Explorer opens.

3. Select *ijca* configuration from the drop-down list of Available Hosts. For information on how to configure available hosts, see the installation manual.



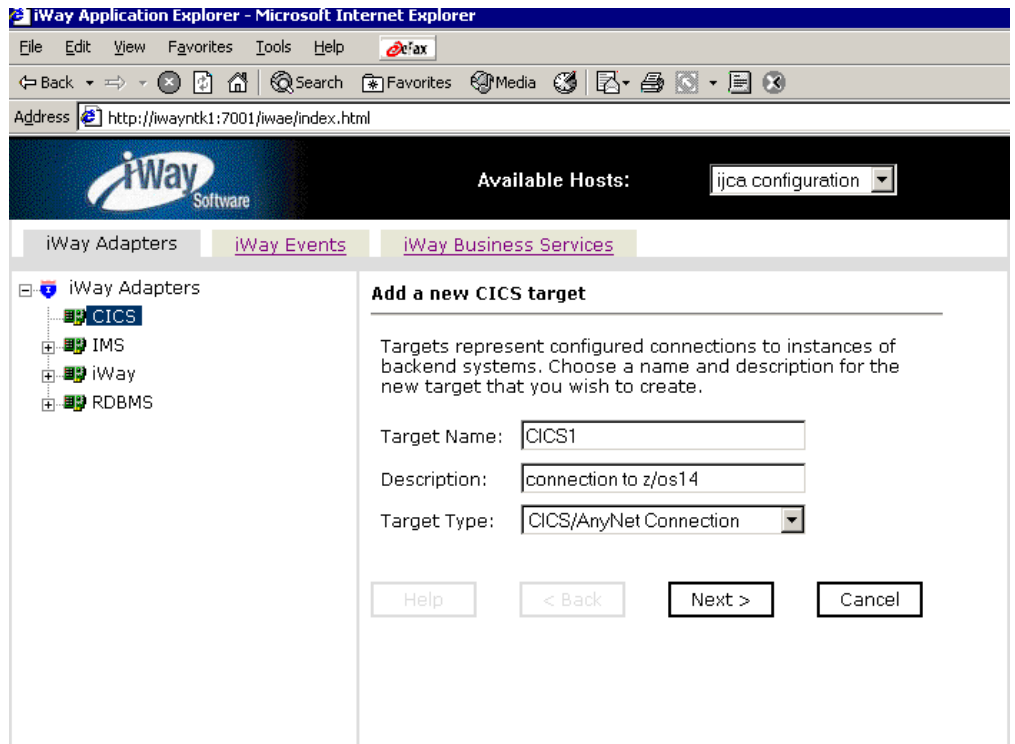
You can expand the iWay Adapter folder to display the list of iWay service adapters currently installed on your system.



Procedure How to Establish a Connection to CICS From iWay Application Explorer

To create a new connection to your CICS system:

1. Expand the iWay Adapters node in Application Explorer.
2. Right-click the *CICS* node.
3. In the left pane, move your pointer over *Operations*, and select *Define a new target*.



4. In the Add a new CICS target dialog box:
 - a. In the Target Name field, type a name for the connection, for example, *CICS1*. The name is used to build a repository entry as well as to identify the connection.
 - b. In the Description field, type a description for the target name you just created. For example, *Connection to zos14*.
 - c. In the Target Type drop-down list, select *CICS/AnyNet Connection*.
5. Click *Next*.

The connection name is verified for the system. If you entered an invalid instance name, a new dialog box opens and prompts you for an instance name again. The Set connection info dialog box appears.
6. Type the connection parameters to create a new connection to CICS.

You can obtain this information from the CICS systems programmer. This information should be the same for all CICS transactions and messages in a single CICS region.

The following table lists the parameters.

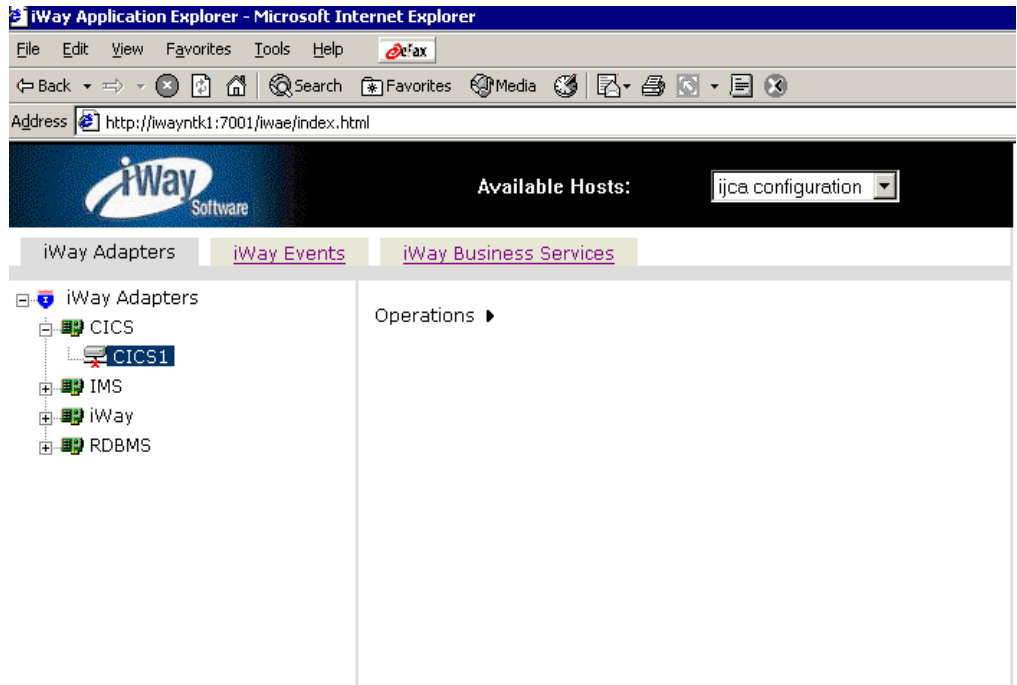
Note: The fields marked with an asterisk (*) are required.

Parameter	Description
Host*	Host name, or IP address, for the computer where CICS is running.
Port AnyNet is listening on*	Port number on which CICS is listening. The AnyNet port, for example 397, is the port which AnyNet (the mainframe TCP to SNA bridge) listens on.
User ID*	Valid user ID for CICS.
Password*	Valid password associated with the CICS user ID.
VTAM Id (VTAM network qualifier)*	VTAM network qualifier.
Partner LU (CICS LU)*	VTAM applied to connect to the CICS system.
Local LU*	LU of the SNA access point to which you have access, for example, SNA server.
LogMode*	Log mode for the connection to CICS, for example, PARALLEL.
Trusted	Security level of VERIFIED (user ID and password are required) is supported unless Trusted is checked. If Trusted is checked, a security level of IDENTIFY (user ID only) is supported. Do not specify LOCAL as the user ID is always sent.
Remote Codepage	CICS system's code page that enables the adapter to correctly translate the incoming data to XML, for example, transforming EBCDIC to ASCII and correctly interpreting binary data.
Number of Sessions	Number of sessions for the AnyNet connection.

For additional information on these parameters, see your documentation for CICS.

7. Click *Finish*.

The newly created connection is added under the CICS service adapter. The configuration information is stored in the repository for the configuration you defined at installation time.



Procedure How to Connect to CICS From iWay Application Explorer

To connect to CICS from iWay Application Explorer:

1. To connect to CICS, move your pointer over *Operations*, and select *Connect*.

The Connect to CICS1 dialog box appears, populated with the values you entered for the connection parameters.

2. Verify your connection parameters and enter a valid password for you system.
3. Click OK.

iWay Application Explorer connects and loads the CICS metadata. Now you can create schemas for Transactions. The schemas are stored in

`c:\iWay55\config\base\CICS\CICS1`

where:

`cics1`

Is the symbolic session name

Viewing Metadata and Creating a Schema

To execute a component interface, a request document is received by a CICS adapter. The adapter processes the request and sends an XML response document indicating the result.

The iWay Application Explorer (iAE) creates the following:

- XML request schema
- XML response schema

The following procedure illustrates how to create request and response schemas for a CICS transaction name FILREAD. The iWay Application Explorer enables you to create XML schemas for this transaction from your COBOL file descriptor (FD).

Procedure How to View Metadata and Create a Schema

1. In the left pane, expand the connection name to display CICS.

You can expand and explore the generic transaction schema or create new schemas to map your CICS common area to XML.

2. Click a component interface to display detailed information about that business function in the right pane.

3. Click *Transactions*, move your pointer over *Operations*, and select *Add Service*.

The Add dialog box opens.

4. Enter the appropriate information for the CICS transaction to map to the COBOL descriptions.

This example creates a node called TEST to execute the FILREAD transaction. The FILREAD transaction has an input and output Cobol FDs that will allow the iWay Adapter for CICS to correctly map the CICS common area to XML.

1. Enter a name to describe the adapter transaction you are creating. This name, for example, TEST, will appear under the Transactions node for the current connection.
2. Enter the name of the program to be called in CICS, for example, FILREAD.
3. Enter the full path to your input COBOL FD. This file must be located on your local disk. File descriptors are used so that the adapter can map byte data to XML. If a file descriptor is not supplied, the data will be returned as one long XML string.
4. Enter the full path to your output COBOL FD. This file must be located on your local disk. File descriptors are used so that the adapter can map byte data to XML. If a file descriptor is not supplied, the data will be returned as one long XML string.
5. Click the new statement (this name matches the table name), move the pointer over *Operations*, and select *Generate Schemas*.

The iWay Application Explorer creates the following:

- XML request schema
- XML response schema

The following XML document was generated from the iWay generated schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
<Transaction location="/CICS/Transaction/TEST">
<NUMB>000100</NUMB>
</Transaction>
</CICS>
```

Procedure How to Disconnect and Persist the Metadata for CICS From iWay Application Explorer

The metadata is persisted to the iWay repository when you disconnect from CICS.

To disconnect to CICS, move your pointer over *Operations*, and select *Disconnect*.

The connection is released and the information is stored to the iWay repository.

Deploying and Running the Sample Servlet

The iWay Connector for JCA includes sample code that enables you to test iWay service and event adapters. This topic describes how to configure and deploy the iWay JCA Test Tool to the BEA WebLogic Server.

Procedure How to Deploy the Sample Servlet

To deploy the sample servlet:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

The following window opens.



WebLogic Server Administration Console

Sign in to work with the WebLogic Server domain **jcawed**

Username:

Password:

- a. Type a user name and password.
- b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from *Deployed Resources*.
3. Click *Deploy a New Web Application*.

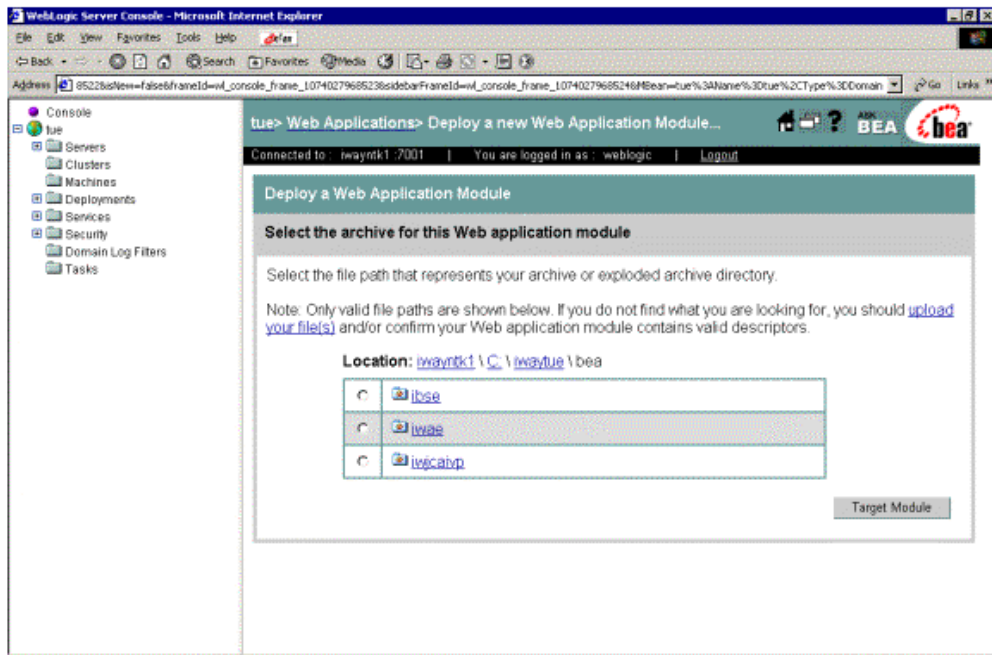
Use the location link to navigate to the location of the exploded WAR file. The WAR file is located in the *iWayHome\BEA* subdirectory.

where:

[*iWayHome*](#)

Is the root directory where the iWay product has been installed.

The following window opens.



4. Select *iwjcaivp*.
5. Click *Target Module*.

If deployment is successful, a window similar to the following opens:



Procedure How to Run the Sample Servlet

To run the sample servlet:

1. Open a browser and enter the following URL:

<http://host:port/iwjcaivp>

where:

[host](#)

Is the IP address or DNS name where the application server is installed.

[port](#)

Is the port on which the application server is listening. For a BEA WebLogic Server, the original default is 7001.

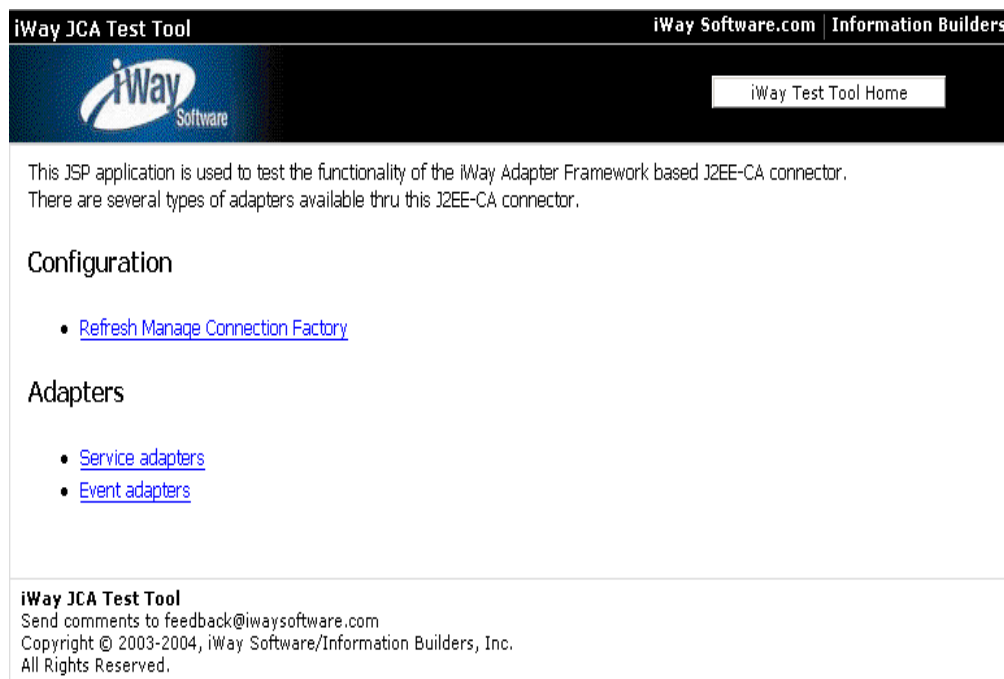
[iwjcaivp](#)

Is the context root to where the Web application is deployed.

For WebLogic, the default URL is

<http://localhost:7001/iwjcaivp>

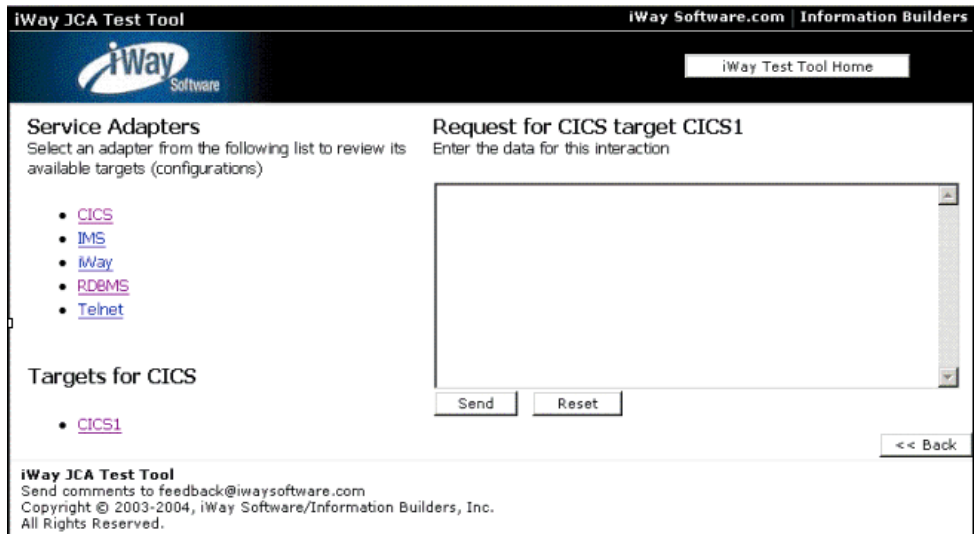
The iWay JCA Test Tool window opens.



For information about the iWay JCA Test Tool, see Chapter 3, *iWay JCA Installation Verification Program*.

2. Click the *Service adapters* link to display the available adapters.
3. Click *CICS*.

Note: If you have multiple target adapters defined, you must select CICS1. If you only defined one target, the following window opens.



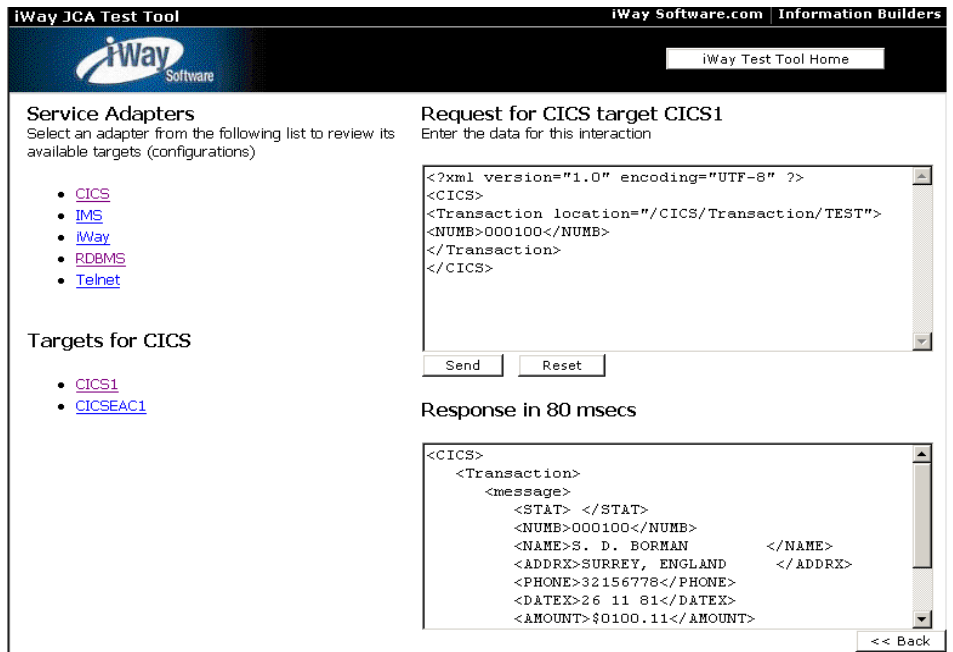
4. Enter the following request to run the FILEREAD transaction with a parameter of '000100':

```
<?xml version="1.0" encoding="UTF-8" ?>
<CICS>
  <Transaction location="/CICS/Transaction/TEST">
    <NUMB>000100</NUMB>
  </Transaction>
</CICS>
```

The following results are returned

```
<CICS>
  <Transaction>
    <message>
      <STAT> </STAT>
      <NUMB>000100</NUMB>
      <NAME>S. D. BORMAN          </NAME>
      <ADDRX>SURREY, ENGLAND     </ADDRX>
      <PHONE>32156778</PHONE>
      <DATEX>26 11 81</DATEX>
      <AMOUNT>$0100.11</AMOUNT>
      <COMMENTS>*****</COMMENTS>
    </message>
  </Transaction>
</CICS>
```

The resulting window should look similar to the following:



Accessing an IMS Transaction From BEA WebLogic Server

The following topics describe how to access an IMS transaction using the iWay Connector for JCA and the iWay Adapter for IMS hosted by BEA WebLogic Application Server.

The following procedure describes how to connect to IMS from iWay Application Explorer (IAE). The procedure uses a sample application servlet supplied with the iWay Connector for JCA.

Procedure How to Connect to IMS From iWay Application Explorer

To connect to IMS from iWay Application Explorer:

1. Follow the installation steps to install the iWay Application Explorer as described in the iWay installation manual.
2. Determine the hostname and port for your application server, and then type the following URL:

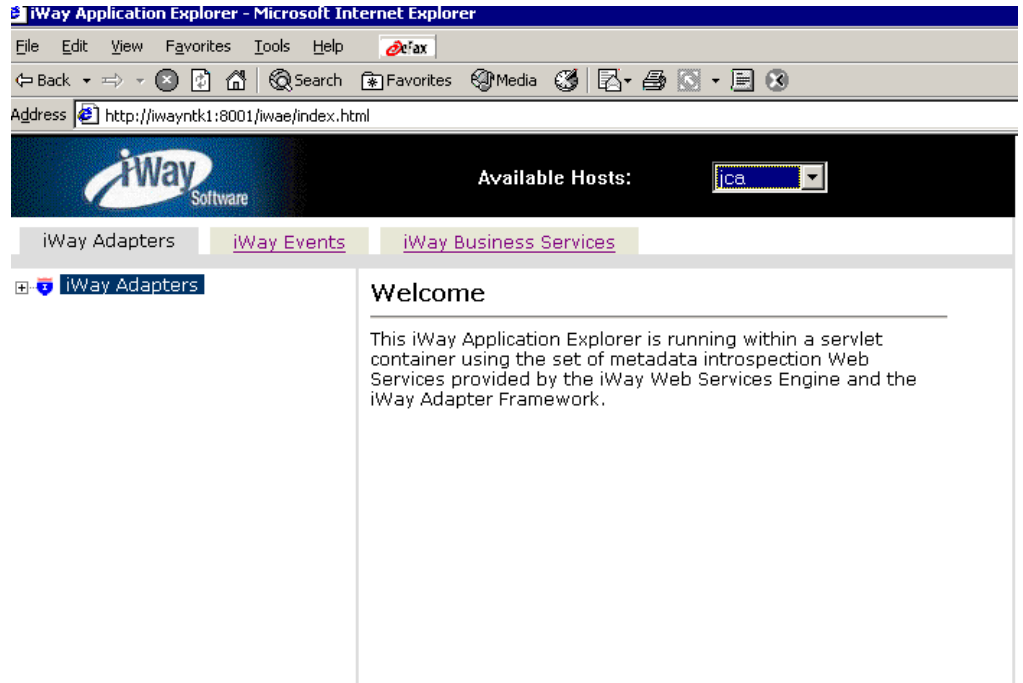
<http://hostname:port/iwae/index.html>

For WebLogic with the default port and domain on your localhost, go to:

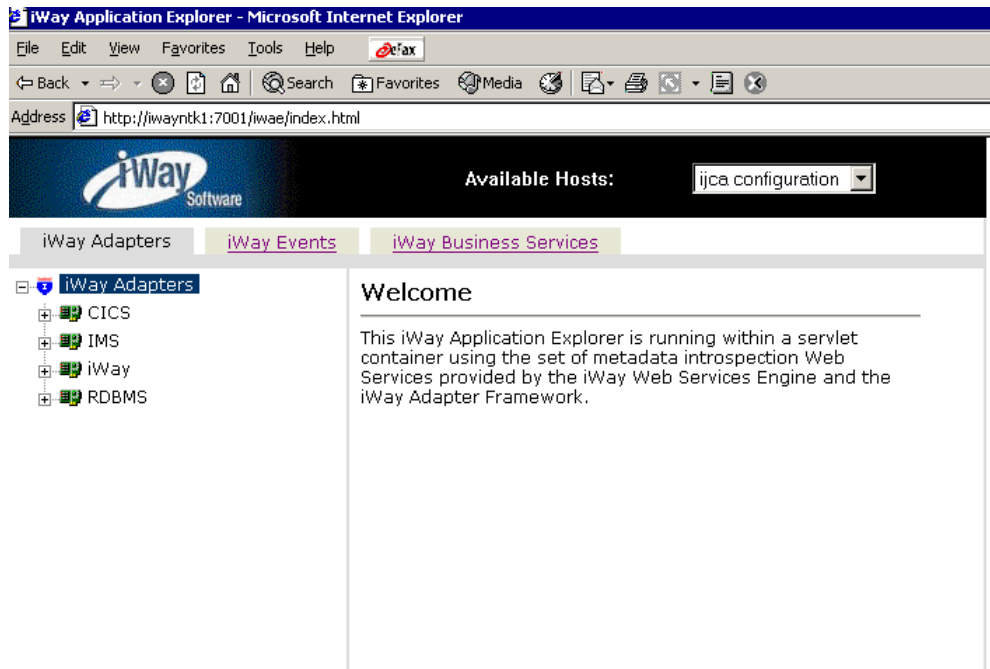
<http://localhost:7001/iwae/index.html>

Application Explorer opens.

3. Select *ijca* configuration from the drop-down list of Available Hosts. For information on how to configure available hosts, see the installation manual.



You can expand the iWay Adapter folder to display the list of iWay service adapters currently installed on your system.



Procedure How to Establish a Connection to IMS From iWay Application Explorer

To create a new connection to your IMS system:

1. Expand the iWay Adapters node in Application Explorer.
2. Right-click the *IMS* node.
3. In the left pane, move your pointer over *Operations*, and select *Define a new target*.
4. In the Add a new IMS target dialog box:
 - a. In the Target Name field, type a name for the connection, for example, *IMS1*. The name is used to build a repository entry as well as to identify the connection.
 - b. In the Description field, type a description for the target name you just created. For example, *Connection to zos14*.
 - c. In the Target Type drop-down list, select *IMS/AnyNet Connection*.
5. Click *Next*.

The connection name is verified for the system. If you entered an invalid instance name, a new dialog box opens and prompts you for an instance name again. The Set connection info dialog box appears.

6. Type the connection parameters to create a new connection to IMS.

You can obtain this information from the IMS systems programmer. This information should be the same for all IMS transactions and messages in a single IMS region.

The following table lists the parameters.

Note: The fields marked with an asterisk (*) are required.

Parameter	Description
Host*	Host name, or IP address, for the computer where IMS is running.
Port AnyNet is listening on*	Port number on which IMS is listening. The AnyNet port, for example 397, is the port which AnyNet (the mainframe TCP to SNA bridge) listens on.
User ID*	Valid user ID for IMS.
Password*	Valid password associated with the IMS user ID.
VTAM Id (VTAM network qualifier)*	VTAM network qualifier.
Partner LU (IMS LU)*	VTAM applied to connect to the IMS system.
Local LU*	LU of the SNA access point to which you have access, for example, SNA server.
LogMode*	Log mode for the connection to IMS, for example, PARALLEL.
Trusted	Security level of VERIFIED (user ID and password are required) is supported unless Trusted is checked. If Trusted is checked, a security level of IDENTIFY (user ID only) is supported. Do not specify LOCAL as the user ID is always sent.
Remote Codepage	IMS system's code page that enables the adapter to correctly translate the incoming data to XML, for example, transforming EBCDIC to ASCII and correctly interpreting binary data.

Parameter	Description
Number of Sessions	Number of sessions for the AnyNet connection.

For additional information on these parameters, see your documentation for IMS.

7. Click *Finish*.

The newly created connection is added under the IMS service adapter. The configuration information is stored in the repository for the configuration you defined at installation time.

Procedure How to Connect to IMS From iWay Application Explorer

To connect to IMS from iWay Application Explorer:

1. To connect to IMS, move your pointer over *Operations*, and select *Connect*.

The Connect to IMS1 dialog box appears, populated with the values you entered for the connection parameters.

2. Verify your connection parameters and enter a valid password for you system.

3. Click *OK*.

iWay Application Explorer connects and loads the IMS metadata. Now you can create schemas for Transactions. The schemas are stored in

`c:\iWay55\config\base\IMS\IMS1`

where:

`IMS1`

Is the symbolic session name

Viewing Metadata and Creating a Schema

To execute a component interface, a request document is received by an IMS adapter. The adapter processes the request and sends an XML response document indicating the result.

The iWay Application Explorer (iAE) creates the following:

- XML request schema
- XML response schema

The following procedure illustrates how to create request and response schemas for an IMS transaction name FILREAD. The iWay Application Explorer enables you to create XML schemas for this transaction from your COBOL file descriptor (FD).

Procedure How to View Metadata and Create a Schema

1. In the left pane, expand the connection name to display IMS.

You can expand and explore the generic transaction schema or create new schemas to map your IMS common area to XML.

2. Click a component interface to display detailed information about that business function in the right pane.

3. Click *Transactions*, move your pointer over *Operations*, and select *Add Service*.

The Add dialog box opens.

4. Enter the appropriate information for the IMS transaction to map to the COBOL descriptions.

This example creates a node called TEST to execute the FILREAD transaction. The FILREAD transaction has an input and output Cobol FDs that will allow the iWay Adapter for IMS to correctly map the IMS common area to XML.

1. Enter a name to describe the adapter transaction you are creating. This name, for example, TEST, will appear under the Transactions node for the current connection.
2. Enter the name of the program to be called in IMS, for example, FILREAD.
3. Enter the full path to your input COBOL FD. This file must be located on your local disk. File descriptors are used so that the adapter can map byte data to XML. If a file descriptor is not supplied, the data will be returned as one long XML string.
4. Enter the full path to your output COBOL FD. This file must be located on your local disk. File descriptors are used so that the adapter can map byte data to XML. If a file descriptor is not supplied, the data will be returned as one long XML string.
5. Click the new statement (this name matches the table name), move the pointer over *Operations*, and select *Generate Schemas*.

The iWay Application Explorer creates the following:

- XML request schema
- XML response schema

The following XML document was generated from the iWay generated schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<IMS>
  <Transaction location="/IMS/Transaction/TEST">
    <NUMB>000100</NUMB>
  </Transaction>
</IMS>
```

Procedure How to Disconnect and Persist the Metadata for IMS From iWay Application Explorer

The metadata is persisted to the iWay repository when you disconnect from IMS.

To disconnect to IMS, move your pointer over *Operations*, and select *Disconnect*.

The connection is released and the information is stored to the iWay repository.

Deploying and Running the Sample Servlet

The iWay Connector for JCA includes sample code that enables you to test iWay service and event adapters. This topic describes how to configure and deploy the iWay JCA Test Tool to the BEA WebLogic Server.

Procedure How to Deploy the Sample Servlet

To deploy the sample servlet:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

The following window opens.



The screenshot shows the BEA WebLogic Server Administration Console login interface. At the top, a teal header bar contains the text "Administration Console" and "BEA WebLogic Server 8.1". Below this, the main content area has a light gray background. The title "WebLogic Server Administration Console" is displayed in a large, bold, teal font. Underneath the title, the text "Sign in to work with the WebLogic Server domain jcwed" is shown in a smaller, gray font. The login form consists of two input fields: "Username:" with the text "weblogic" entered, and "Password:" which is empty. To the right of the password field is a "Sign In" button with a gray background and white text.

- a. Type a user name and password.
 - b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from *Deployed Resources*.
3. Click *Deploy a New Web Application*.

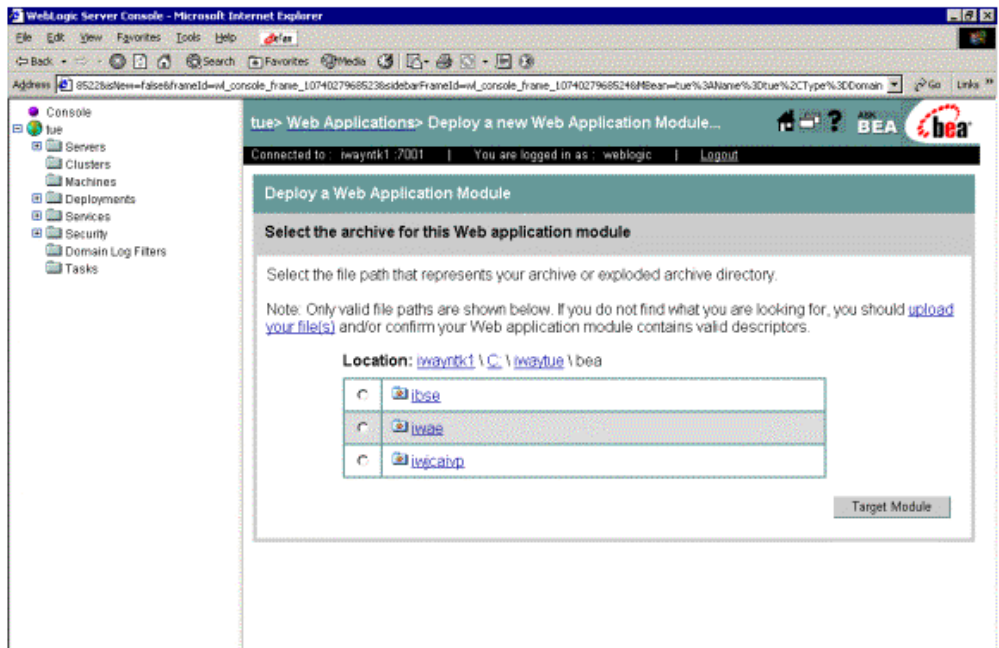
Use the location link to navigate to the location of the exploded WAR file. The WAR file is located in the *iWayHome\BEA* subdirectory.

where:

iWayHome

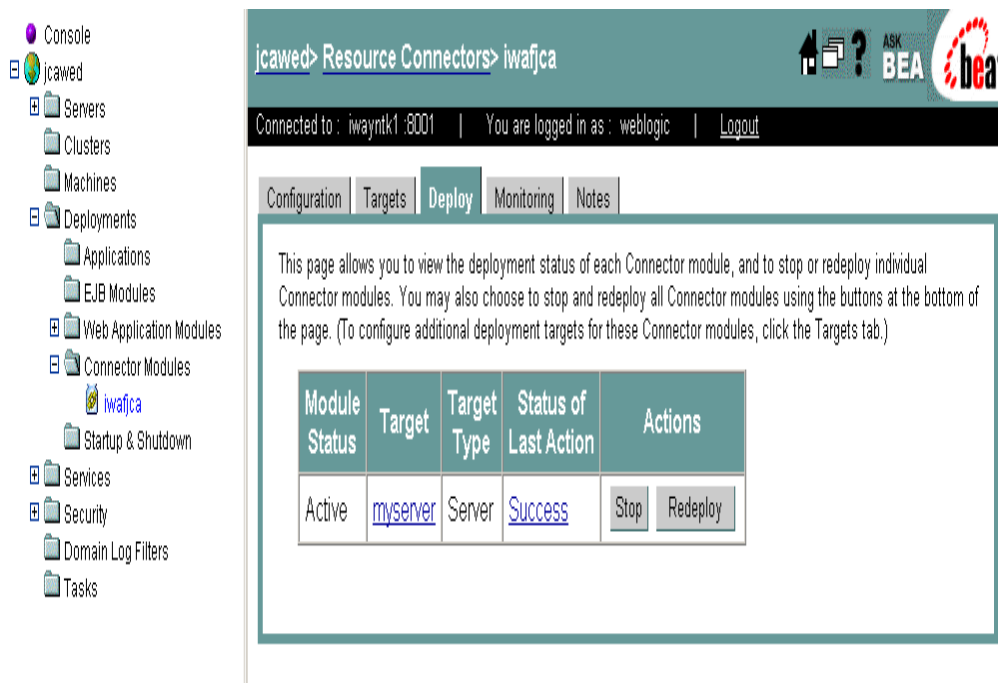
Is the root directory where the iWay product has been installed.

The following window opens.



4. Select *iwjcaivp*.
5. Click *Target Module*.

If deployment is successful, a window similar to the following opens:



Procedure How to Run the Sample Servlet

To run the sample servlet:

1. Open a browser and enter the following URL:

<http://host:port/iwjcaivp>

where:

[host](#)

Is the IP address or DNS name where the application server is installed.

[port](#)

Is the port on which the application server is listening. For a BEA WebLogic Server, the original default is 7001.

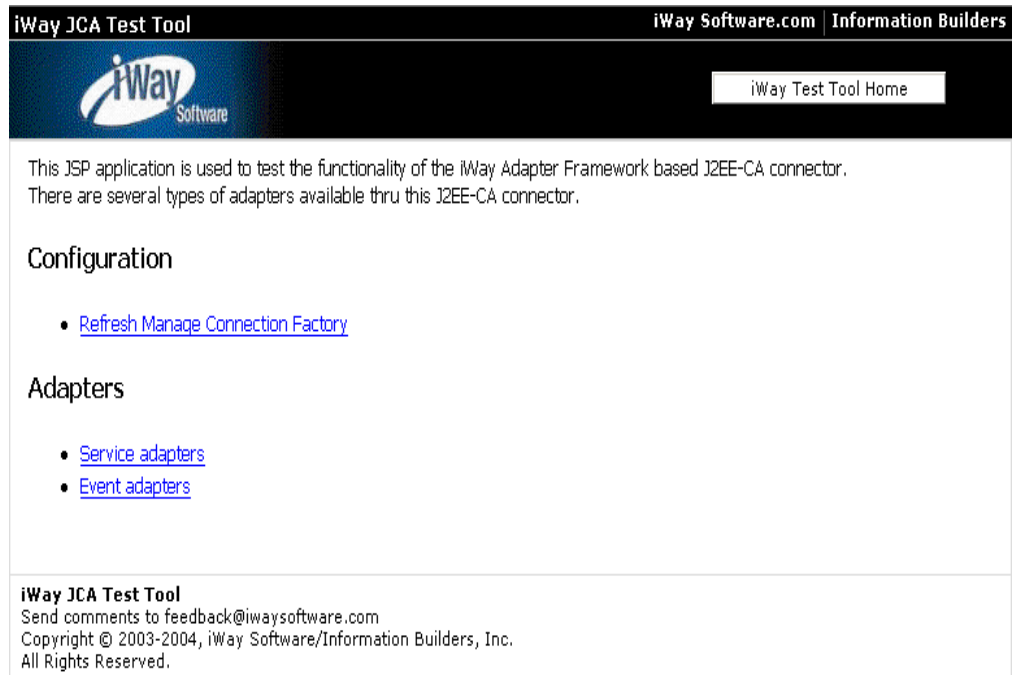
[iwjcaivp](#)

Is the context root to where the Web application is deployed.

For WebLogic, the default URL is

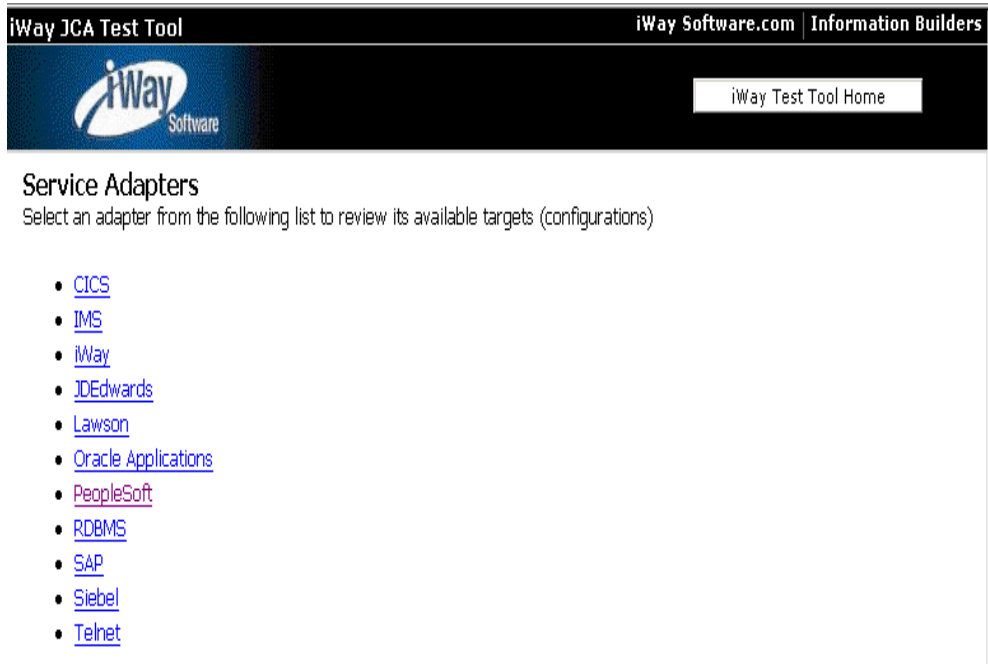
<http://localhost:7001/iwjcaivp>

The iWay JCA Test Tool window opens.



For information about the iWay JCA Test Tool, see Chapter 3, *iWay JCA Installation Verification Program*.

2. Click the *Service adapters* link to display the available adapters.



3. Click *IMS*.

Note: If you have multiple target adapters defined, you must select IMS1.

4. Enter the following request to run the FILEREAD transaction with a parameter of '0000100':

```
<?xml version="1.0" encoding="UTF-8" ?>
<IMS>
  <Transaction location="/IMS/Transaction/TEST">
    <NUMB>000100</NUMB>
  </Transaction>
</IMS>
```

The following results are returned

```

<IMS>
  <Transaction>
    <message>
      <STAT> </STAT>
      <NUMB>000100</NUMB>
      <NAME>S. D. BORMAN          </NAME>
      <ADDRX>SURREY, ENGLAND    </ADDRX>
      <PHONE>32156778</PHONE>
      <DATEX>26 11 81</DATEX>
      <AMOUNT>$0100.11</AMOUNT>
      <COMMENTS>*****</COMMENTS>
    </message>
  </Transaction>
</IMS>

```

Accessing a J.D. Edwards OneWorld Business Function From BEA WebLogic Server

The following topics describe how to access a J. D. Edwards OneWorld business function using the iWay Connector for JCA and the iWay Adapter for J.D. Edwards OneWorld, hosted by BEA WebLogic Application Server.

Preparing to Use J.D. Edwards OneWorld XE

This topic describes the requirements for the iWay Connector for JCA to operate with the iWay Adapter for J.D. Edwards OneWorld.

The following JAR files must be accessible to support the iWay Adapter for J.D. Edwards OneWorld:

[Connector.jar](#)
[Kernel.jar](#).

The jar files should be copied into the iWayHome lib directory

where:

[iWayHome](#)

Is the root directory where the iWay products have been installed. The locations of the JAR files may also be specified in the BEA WebLogic class path as follows:

- <Full path on BEA WebLogic Application Server machine to Kernel.jar>
- <Full path on BEA WebLogic Application Server machine to Connector.jar>

The two JAR files, Kernel.jar and Connector.jar, must be copied from the machine where J.D. Edwards OneWorld is installed on the machine where the BEA WebLogic Application server is installed. You can copy these files to any location, however, this location must appear correctly in the class path.

Connecting to J. D. Edwards OneWorld From iWay Application Explorer

The following procedure describes how to connect to J. D. Edwards OneWorld from iWay Application Explorer (iAE). The procedure uses a sample application servlet supplied with the iWay Connector for JCA.

Procedure How to Connect to J.D. Edwards OneWorld From iWay Application Explorer

To connect to J. D. Edwards OneWorld from iWay Application Explorer:

1. Install the iWay Adapter for J.D. Edwards (which includes the iWay Application Explorer) from the supplied media.

The iWay Application Explorer requires GenJava wrappers. GenJava is supplied as a command line process with several run-time options. The iWay Application Explorer generates schemas against OneWorld GenJava wrappers.

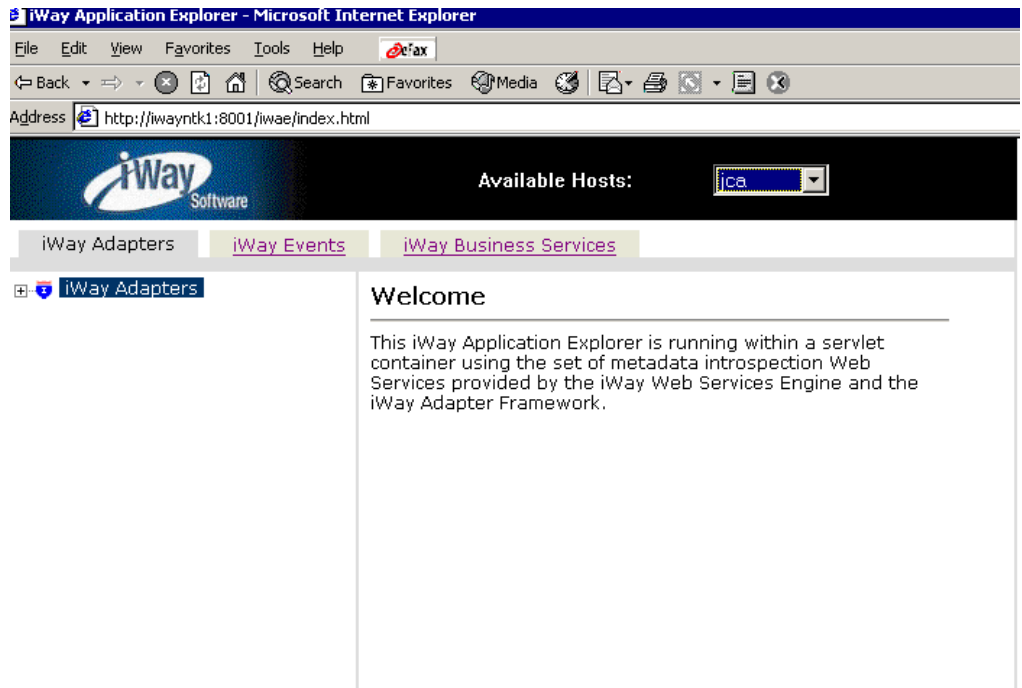
- a. Create the GenJava wrappers using the OneWorld utility called GenJava.
 - b. For more information on GenJava, see the Java section in the *J.D. Edwards Interoperability Guide for OneWorld Xe*.
2. Determine the hostname and port for your application server, and then type the following URL:

<http://hostname:port/iwae/index.html>

For WebLogic with the default port and domain on your localhost, go to:

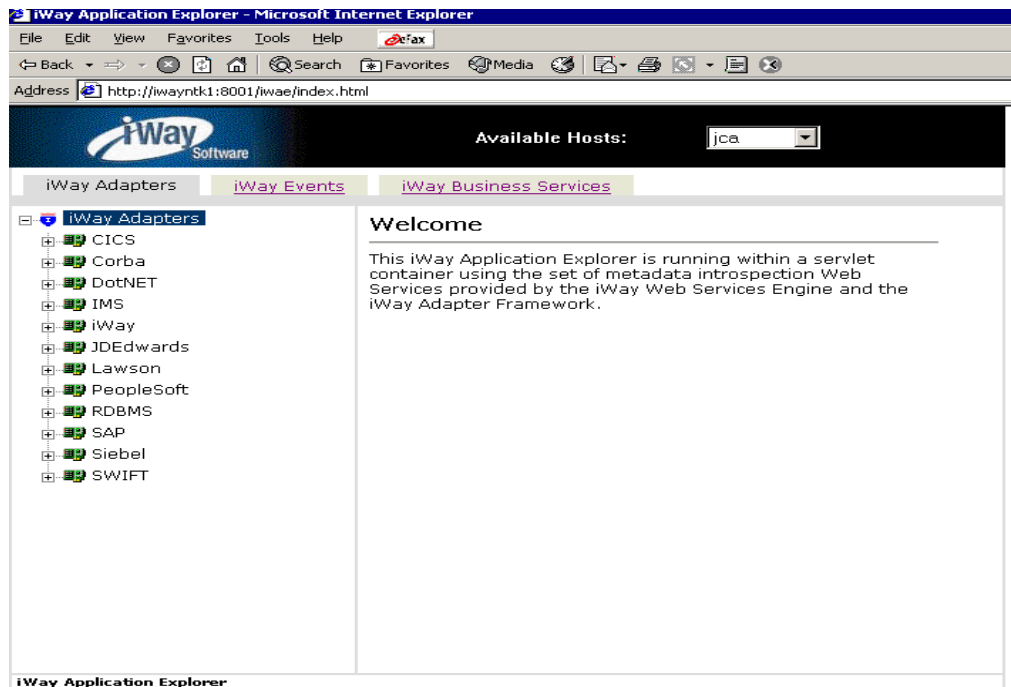
<http://localhost:7001/iwae/index.html>

Application Explorer opens.



3. Select *ijca* configuration from the drop-down list of available hosts.

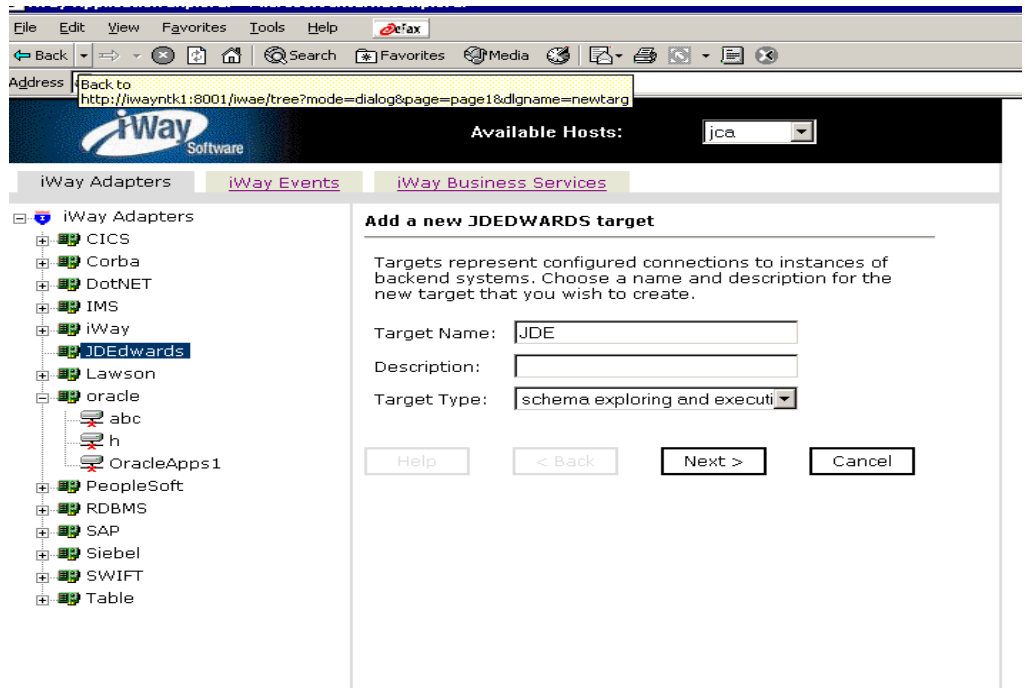
You can expand the iWay Adapter folder to display the list of iWay service adapters currently installed on your system.



Procedure How to Establish a Connection to J.D. Edwards From iWay Application Explorer

To create a new connection to your J.D. Edwards OneWorld system:

1. Expand the iWay Adapters node in Application Explorer.
2. Right-click the J.D. Edwards node.
3. In the left pane, move your pointer over *Operations*, and select *Define a new target*.



4. In the Add a new JDEdwards target dialog box:
 - a. In the Target Name field, type a name for the connection, for example, *JDE*.
The name is used to build a repository entry as well as to identify the connection.
 - b. In the Description field, type a description for the target name you just created. For example, *Connection to JDEdwards*.
 - c. In the Target Type drop-down list, select *Schema exploration only*.
5. Click **Next**.
The connection name is verified for the system. If you entered an invalid instance name, a new dialog box opens, and prompts you for an instance name again.
The Set connection info dialog box appears.
6. Type the connection parameters to create a new connection to J.D. Edwards.
You can obtain this information from the J.D. Edwards systems administrator. This information should be the same for all J.D. Edwards Business Functions in a single J.D. Edwards system.
The following table lists the parameters.

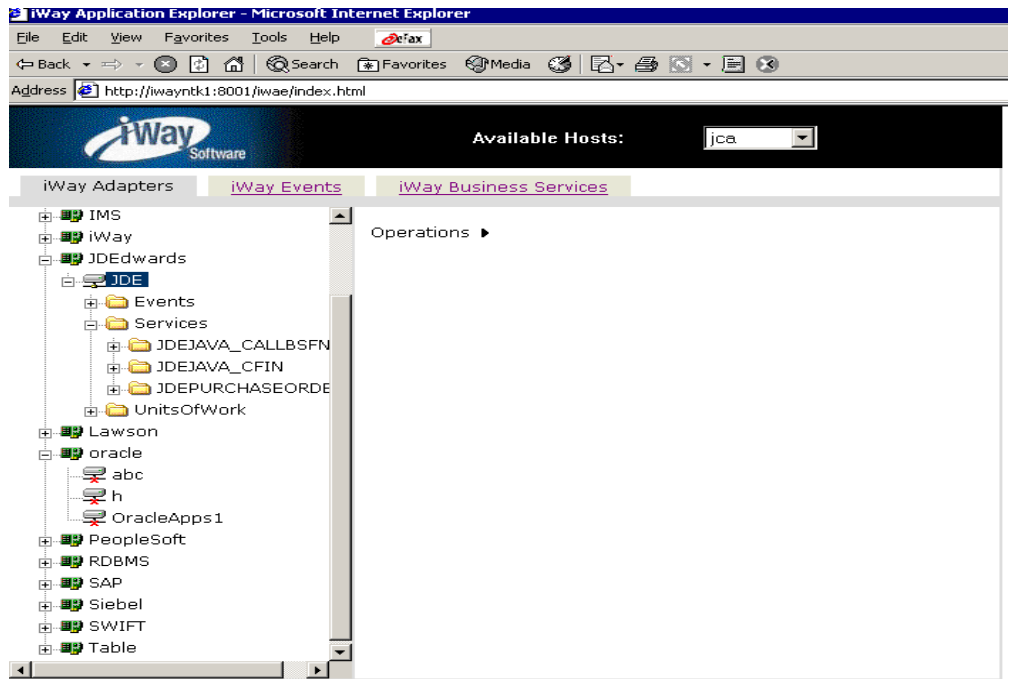
Note: The fields marked with an asterisk (*) are required.

Parameter	Description
Repository Directory	Location of the GenJava-generated Master Business Function wrappers (local), for example, C:\JDEOneWorld.
Host	Name of the server on which J.D. Edwards OneWorld is running, or its IP address, for example, BVISION01.
Port	Number on which the server is listening, for example, 6009.
User	Valid user ID for J.D. Edwards OneWorld, for example, jde.
Password	Password associated with the user ID, for example, adapter.
JDE Environment	J.D. Edwards OneWorld environment, for example, DU733. For more information, see the documentation for J.D. Edwards OneWorld or ask your OneWorld system administrator.

For additional information on these parameters, see your documentation for J.D. Edwards.

7. Click *Finish*.

The newly created connection is added under J.D. Edwards in the iWay Adapters list.



The configuration information is stored in the repository for the configuration you defined at installation time.

Procedure How to Connect to J.D. Edwards From iWay Application Explorer

To connect to J.D. Edwards from iWay Application Explorer:

1. To connect to J.D. Edwards, move your pointer over *Operations*, and select *Connect*.
The Connect to J.D. Edwards dialog box appears, populated with the values you entered for the connection parameters.
2. Verify your connection parameters and enter a valid password for your system.
3. Click *OK*.
4. The iWay Application Explorer connects to J.D. Edwards and loads the J.D. Edwards metadata.

Now you can create schemas for business processes. The schemas are stored in

[C:\iway55\config\base\schemas\JDEdwards\JDE](file:///C:/iway55/config/base/schemas/JDEdwards/JDE)

where:

JDE

Is the symbolic session name.

Viewing Metadata and Creating a Schema

To execute a business function, a request document is received by an agent through the J.D. Edwards OneWorld ThinNet API. The agent processes the request and sends an XML response document indicating the result.

The iWay Application Explorer (iAE) creates the following:

- XML request schema
- XML response schema

The iAE also generates the appropriate entries in the manifest.xml file. This file contains connection and configuration information for the J.D. Edwards OneWorld server where the schemas are created, as well as pointers to the schemas themselves.

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

After you establish a connection to your system, you can display the J.D. Edwards business functions and create schemas.

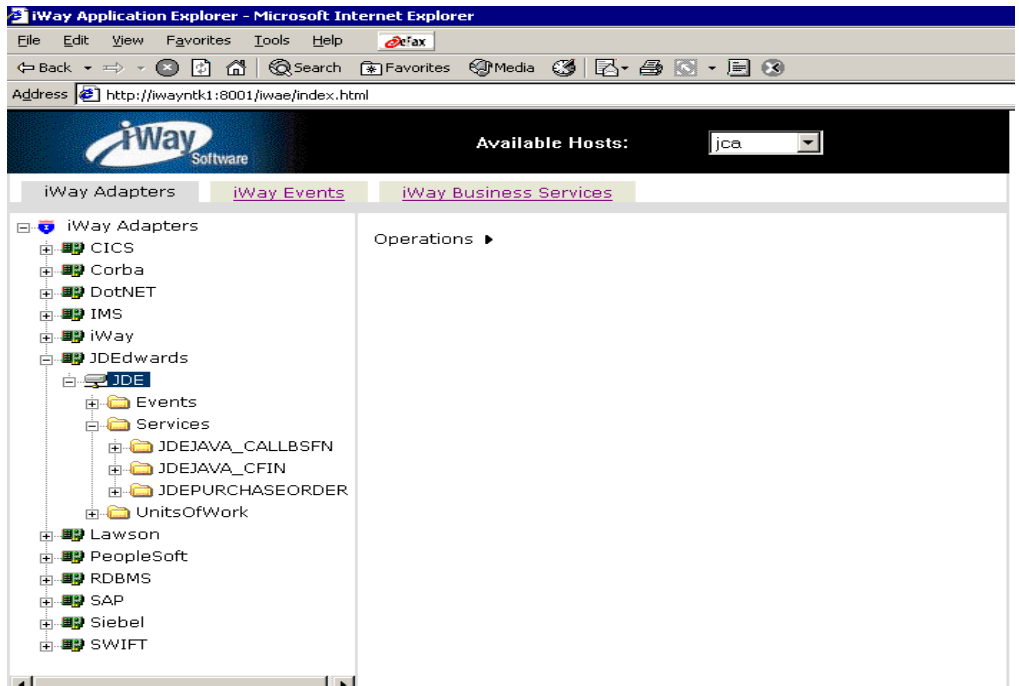
Procedure How to View Metadata and Create a Schema

To display the J.D. Edwards business functions and schemas:

1. In the left pane, expand the connection name to display the J.D. Edwards Business functions.

You can expand and explore all the business functions available.

2. Click a business function to display detailed information about that business function in the right pane.



3. In the left pane, expand JDEPURCHASEORDER, and then the PurchaseOrder business function under the Transactions node.
4. Click *PurchaseOrder*, move your pointer over *Operations*, and select *Create Schemas*.
5. In the right pane, click the *Request Schemas* tab to view the request schema.

You can use the generated request schema to create a sample XML document to be used by the adapter.

The following XML document was generated from the iWay generated schema.

```
<?xml version='1.0' encoding='utf-8' ?>
<!-- testing comments in the beginning of request -->
<jdeRequest type='callmethod'>
<callMethod 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'>17000</param>
<param name='mnShipToNumber'>0.0</param>
<param name='jdOrderDate'>2000/03/02</param>
<param name='cEvaluatedReceiptsFlag'>N</param>
<param name='cCurrencyMode'>D</param>
<param name='szTransactionCurrencyCode'>USD</param>
<param name='mnCurrencyExchangeRate'>0.0</param>
<param name='szOrderedPlacedBy'>SUBSTITUTE</param>
<param name='szProgramID'>EP4310</param>
<param name='szPurchaseOrderPrOptVersion' id=
'Version'>ZJDE0001</param>
<param name='szUserID'>SUBSTITUTE</param>
<param name='mnProcessID' id='processID'></param>
<param name='mnTransactionID' id='transactionID'></param>
</params>
<onError abort='yes'>
<callMethod app='XMLTest' name=
```

```

'F4311ClearWorkFiles' runOnError=
'yes' returnNullData='yes'>
<params>
<param name='szComputerID' idref=
'jobNumber'></param>
<param name=
'mnJobNumber' idref='machineKey'></param>
<param name='cClearHeaderFile'>1</param>
<param name='cClearDetailFile'>1</param>
<param name='mnLineNumber'>0</param>
<param name='cUseWorkFiles'>2</param>
<param name='mnProcessID' idref=
'processID'></param>
<param name='mnTransactionID' idref=
'transactionID'></param>
</params>
</callMethod>
</onError>
</callMethod>
<callMethod app='XMLTest' name='F4311EditLine' runOnError=
'yes' returnNullData='no'>
<params>
<param name='mnJobNumber' idref='jobNumber'></param>
<param name='szComputerID' idref='machineKey'></param>
<param name='cDetailActionCode'>A</param>
<param name='cProcessEdits'>1</param>
<param name='cUpdateOrWriteWorkFile'>2</param>
<param name='cCurrencyProcessingFlag'>Y</param>
<param name='szPurchaseOrderPrOptVersion' idref=
'version'></param>
<param name='szOrderCompany' idref='orderCompany'></param>
<param name='szOrderType'>OP</param>
<param name='szOrderSuffix'>000</param>
<param name='szBranchPlant'> M30</param>
<param name='mnSupplierNumber' idref=
'supplierNumber'></param>
<param name='mnShipToNumber'>0.0</param>
<param name='jdRequestedDate'>2000/03/02</param>
<param name='jdTransactionDate'>2000/03/02</param>
<param name='jdPromisedDate'>2000/03/02</param>
<param name='jdGLDate'>2000/03/02</param>
<param name='szUnformattedItemNumber'>1001</param>
<param name='mnQuantityOrdered'>1</param>
<param name='szDetailLineBranchPlant'> M30</param>
<param name='szLastStatus'>220</param>
<param name='szNextStatus'>230</param>
<param name='cEvaluatedReceipts'>N</param>
<param name='szTransactionCurrencyCode'>USD</param>

```

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

```

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

```

```
'orderNumberAssigned'></param>
</returnParams>
<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>
```

Deploying and Running the Sample Servlet

The iWay Connector for JCA includes sample code that enables you to test iWay service and event adapters. This topic describes how to configure and deploy the iWay JCA Test Tool to the BEA WebLogic Server.

Procedure How to Deploy the Sample Servlet

To deploy the sample servlet:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

The following window opens.

The screenshot shows the BEA WebLogic Server 8.1 Administration Console login interface. At the top, a teal header bar contains the text "Administration Console" and "BEA WebLogic Server 8.1". Below this, the main content area has a title "WebLogic Server Administration Console" and a subtitle "Sign in to work with the WebLogic Server domain **jcawed**". The login form includes a "Username:" label with a text input field containing "weblogic", a "Password:" label with an empty text input field, and a "Sign In" button.

- a. Type a user name and password.
 - b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from *Deployed Resources*.
3. Click *Deploy a New Web Application*.

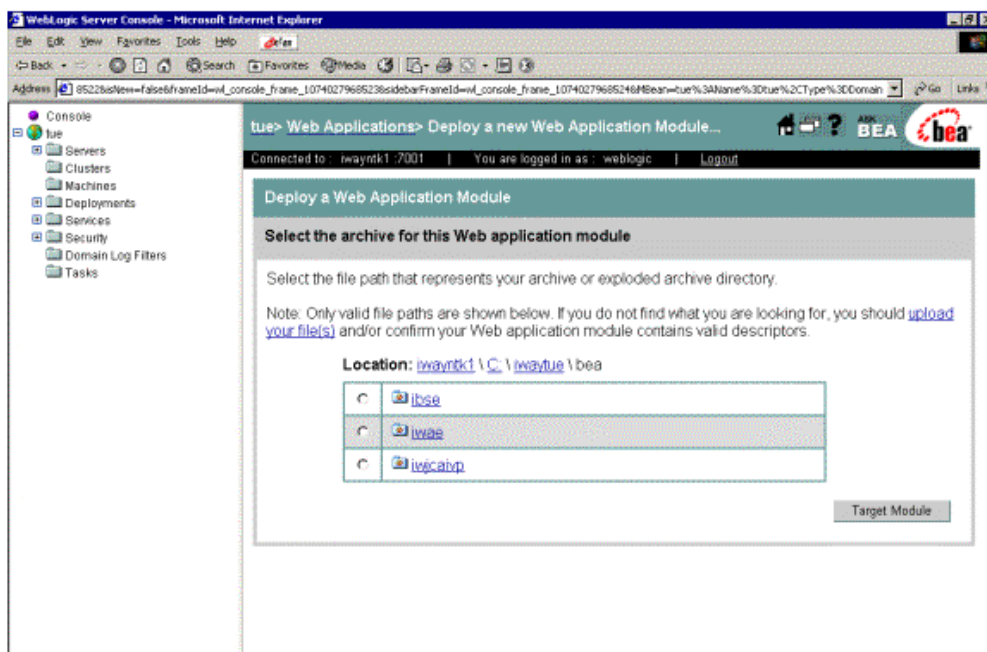
Use the location link to navigate to the location of the exploded WAR file. The WAR file is located in the *iWayHome*\BEA subdirectory.

where:

[*iWayHome*](#)

Is the root directory where the iWay product has been installed.

The following window opens.



4. Select *iwjcaivp*.
5. Click *Target Module*.

If deployment is successful, a window similar to the following opens:



Procedure How to Run the Sample Servlet

To run the sample servlet:

1. Open a browser and enter the following URL:

<http://host:port/iwjcaivp>

where:

[host](#)

Is the IP address or DNS name where the application server is installed.

[port](#)

Is the port on which the application server is listening. For a BEA WebLogic Server, the original default is 7001.

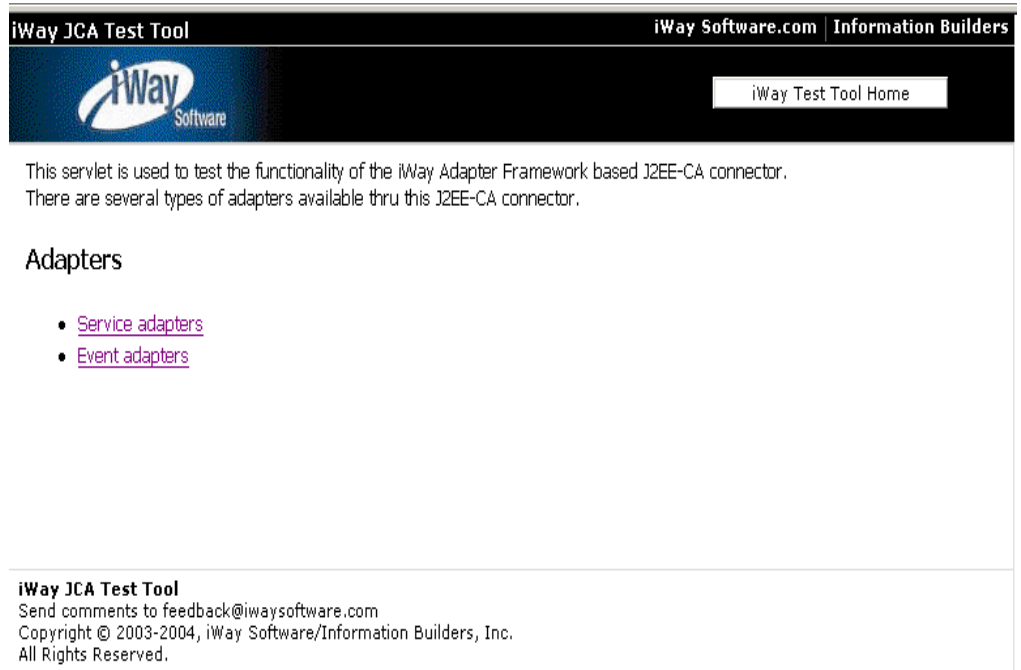
[iwjcaivp](#)

Is the context root to where the Web application is deployed.

For WebLogic, the default URL is

<http://localhost:7001/iwjcaivp>

The iWay JCA Test Tool window opens.

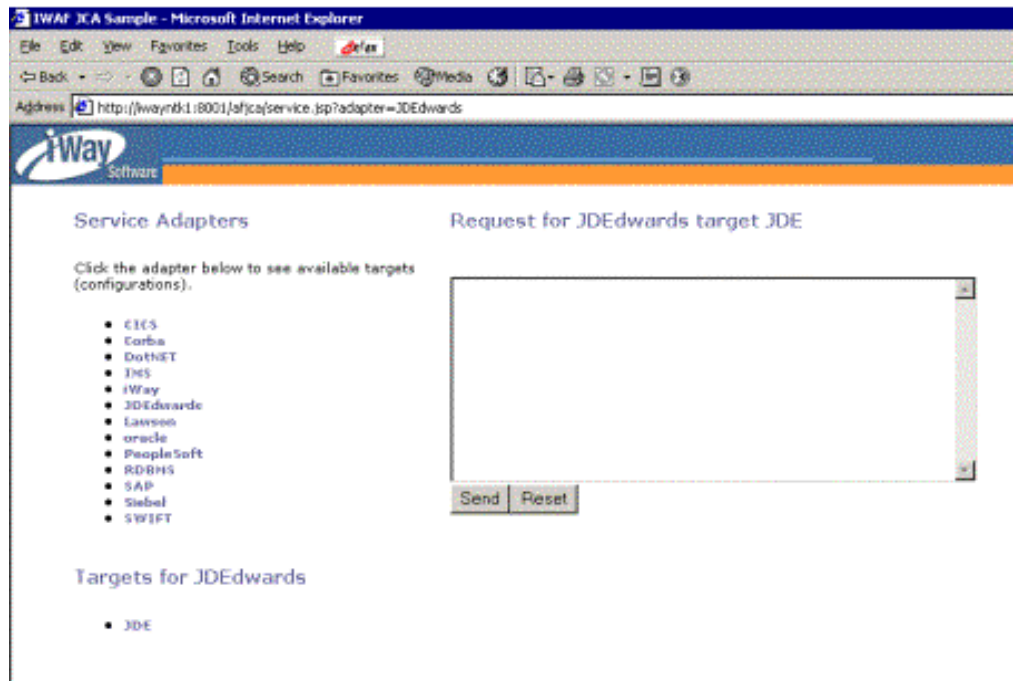


For information about the iWay JCA Test Tool, see Chapter 3, *iWay JCA Installation Verification Program*.

2. Click the *Service adapters* link to display the available adapters.

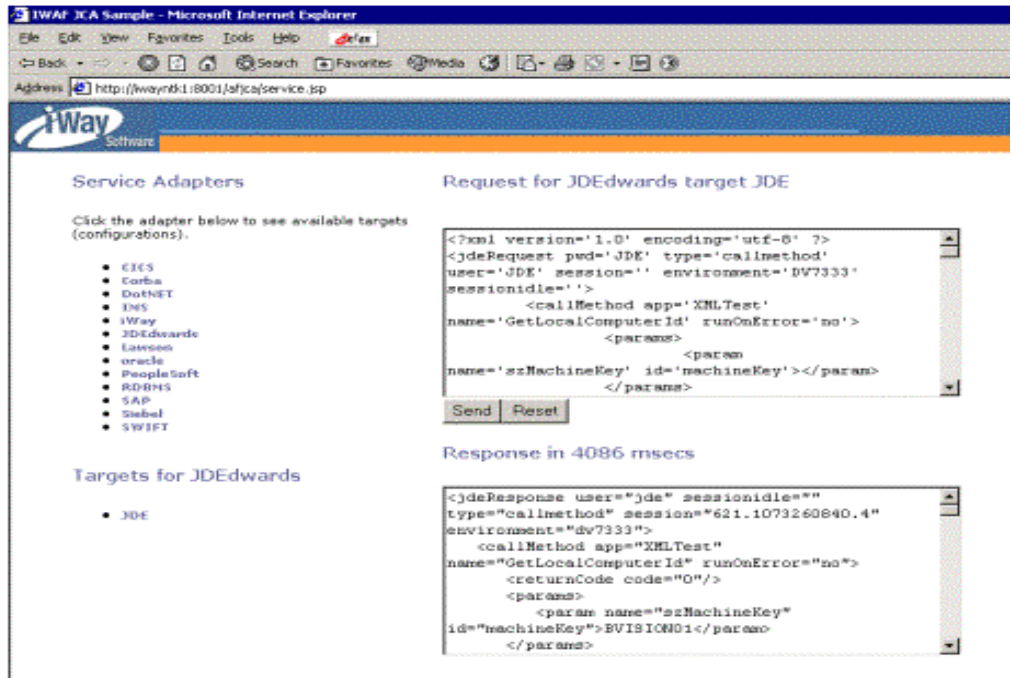


3. Click *JDEdwards*.
The following window opens.



4. Enter the request to run a Purchase Order.

The following results are returned.



Accessing a Lawson Business Function From BEA WebLogic Server

The following topics describe how to access a Lawson Software business function using the iWay Connector for JCA and the iWay Adapter for Lawson, hosted by a BEA WebLogic Server. To illustrate accessing a business function, the procedures use a servlet supplied with the iWay Connector for JCA.

Preparing to Use the iWay Adapter for Lawson

This topic describes the requirements for the iWay Connector for JCA to operate with the iWay Adapter for Lawson.

The iWay Adapter for Lawson Software communicates with the Lawson gateway server over HTTP or HTTPS. These protocols require the Java Secure Socket Extension (JSSE) package.

The JSSE is a set of Java packages that enable secure Internet communications. JSSE implements a Java version of SSL (Secure Sockets Layer) and TLS (Transport Layer Security) protocols. It includes functionality for data encryption, server authentication, message integrity, and optional client authentication. JSSE provides for the secure passage of data between a client and a server running any application protocol (such as HTTP, Telnet, NNTP, and FTP) over TCP/IP.

The following describes the required releases:

- JSSE was integrated into the Java 2 SDK, Standard Edition, v 1.4.
- JSSE 1.0.3_01 is an optional package to the Java 2 SDK, versions 1.2.x and 1.3.x.

If you are using JDK™ 1.3., the following JAR files must be accessible to support the iWay Adapter for Lawson Software: jsse.jar, jcert.jar, and jnet.jar.

You must copy the JSEE JAR files (jsse.jar, jcert.jar, and jnet.jar) to the iWayHome\lib directory

where:

iWayHome

Represents the root directory where iWay55 was installed.

Connecting to Lawson From iWay Application Explorer

The following procedure describes how to connect to Lawson from the iWay Application Explorer (iAE). The procedure uses a sample application servlet supplied with the iWay Connector for JCA.

***Procedure* How to Connect to Lawson From iWay Application Explorer**

To connect Lawson to the iWay Application Explorer:

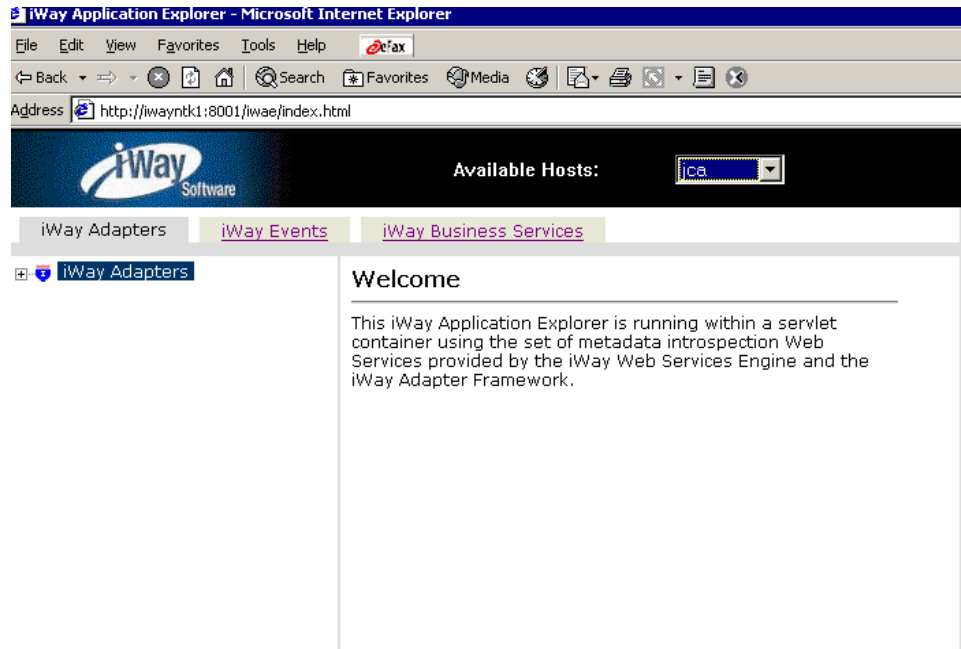
1. Determine the hostname and port for your application server, and then type the following URL:

<http://hostname:port/iwae/index.html>

For WebLogic with the default port and domain on your localhost, go to:

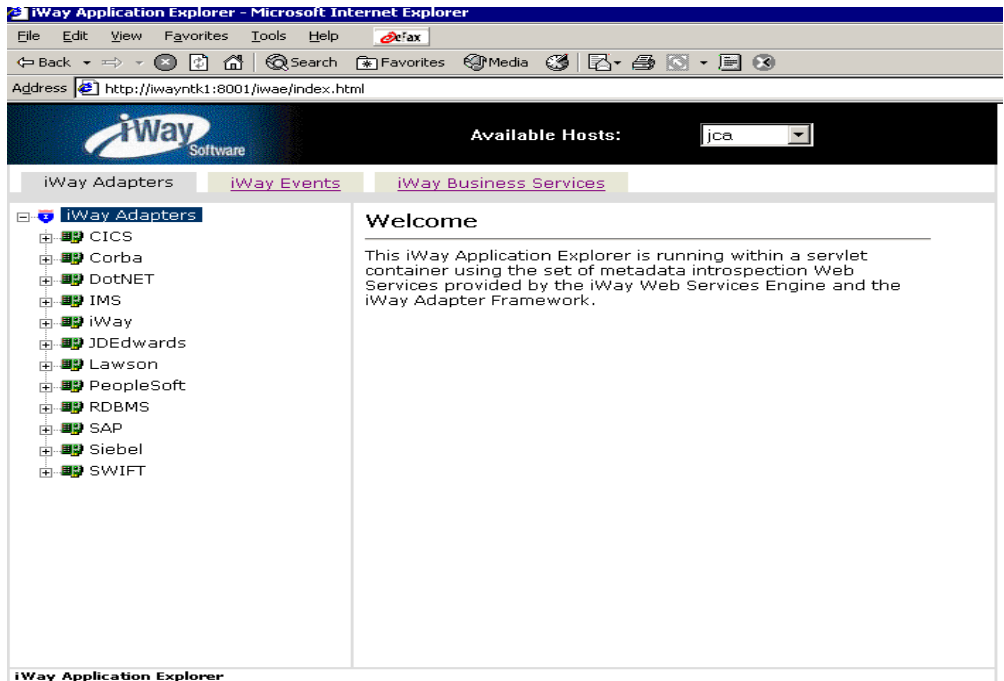
<http://localhost:7001/iwae/index.html>

Application Explorer opens.



2. Select *ijca configuration* from the drop-down list of available hosts.

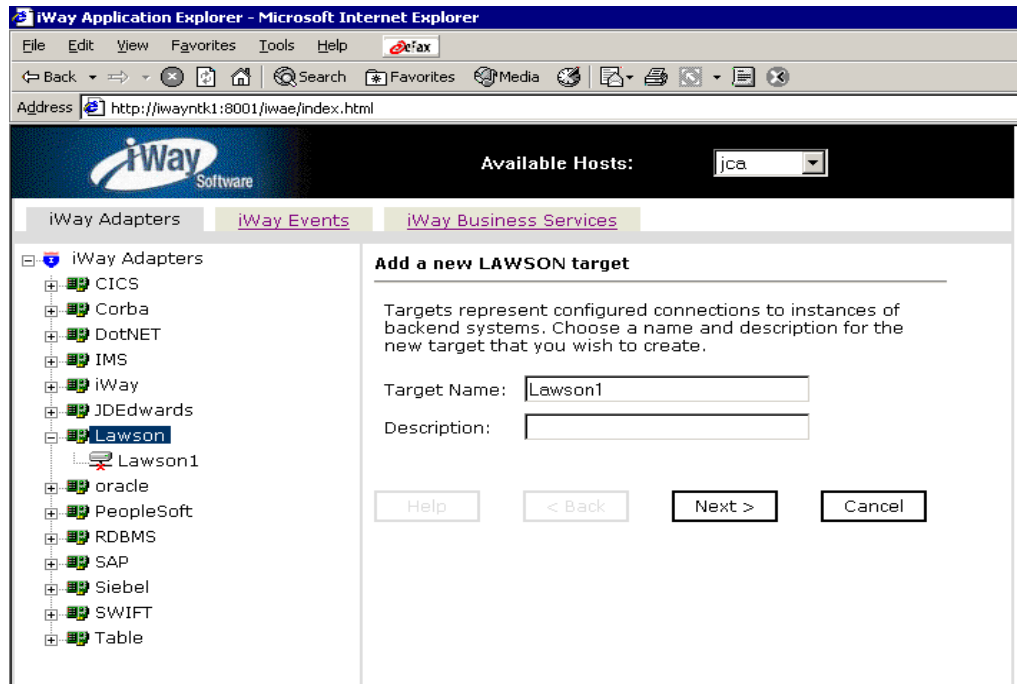
You can expand the iWay Adapter folder to display the list of iWay service adapters currently installed on your system.



Procedure **How to Establish a Connection to Lawson From iWay Application Explorer**

To create a new connection to your Lawson system:

1. Expand the iWay Adapters node in Application Explorer.
2. Expand the Lawson node.
3. In the left pane, move your pointer over *Operations*, and select *Define a new target*.



4. In the Add a new Lawson target dialog box:
 - a. In the Target Name field, type a name for the connection, for example, Lawson1.
The name is used to build a repository entry as well as to identify the connection.
 - b. In the Description field, type a description for the target name you just created. For example, Connection to Lawson.
5. Click Next.

The connection name is verified for the system. If you entered an invalid instance name, a new dialog box opens and prompts you for an instance name again.

The Set connection info dialog box appears.

Type the connection parameters to create a new connection to Lawson. You can obtain this information from the Lawson systems administrator. This information should be the same for all Lawson business functions in a single Lawson system.

The following table lists the parameters.

Note: The fields marked with an asterisk (*) are required.

Parameter	Description
Host	DNS or IP address on which Lawson Open Gateway is running.
Port	Port number on which Lawson Open Gateway is running.
User	Valid user ID defined to the Lawson Software application.
Password	Valid password associated with the specified user ID.
CGI Root	Root directory name for the Lawson CGI.

For additional information, see your documentation for the Lawson system.

Available Hosts:

iWay Adapters | [iWay Events](#) | [iWay Business Services](#)

iWay Adapters

- [-] CICS
- [-] Corba
- [-] DotNET
- [-] IMS
- [-] iWay
- [-] JDEdwards
- Lawson**
 - [-] Lawson1
- [-] oracle
- [-] PeopleSoft
- [-] RDBMS
- [-] SAP
- [-] Siebel
- [-] SWIFT
- [-] Table

Set connection info

Host:

Port:

User:

Password:

CGI Root:

HTTPS: ☐

6. Click *Finish*.

The configuration information is stored in the repository for the configuration you defined at installation time.

Procedure How to Connect to Lawson From iWay Application Explorer

To connect to Lawson from iWay Application Explorer:

1. To connect to Lawson, move your pointer over *Operations*, and select *Connect*.
The Connect to Lawson dialog box appears, populated with the values you entered for the connection parameters.
2. Verify your connection parameters and enter a valid password for your system.
3. Click *OK*.
4. The iWay Application Explorer connects to Lawson and loads the Lawson metadata.

After the iAE successfully connects you to the Lawson system, it accesses the local Lawson files and displays a list of available business objects. The connection to the Lawson system enables you to explore available Lawson business objects.

After the extraction of information is complete, the new connection appears under the Lawson node as illustrated in the following window.

Now you can create schemas for business processes. The schemas are stored in

`C:\iway55\config\base\schemas\Lawson\Lawson1`

where:

`Lawson1`

Is the symbolic session name.

Viewing Metadata and Creating a Schema

The iWay Adapter for Lawson Software enables the processing of Lawson Software screen and batch functions. This topic provides the information you require to create schemas for Lawson Software business objects.

The iWay Application Explorer creates the following:

- XML request schema
- XML response schema

The following procedure illustrates how to create request and response schemas for a Lawson business object named HR 11.1 Employee. The iWay Application Explorer enables you to create XML agent schemas for this function.

Procedure How to View Metadata and Create a Schema

To display the Lawson business functions and schemas:

1. In the left pane, expand the connection name to display the Lawson Business functions.

You can expand and explore all the business functions available.

2. In your Lawson connection in the iWay Application Explorer, expand the Lawson Human Resources node.
3. Expand the HR Human Resources node then the HR 11.1 employee.
4. Click HR 11.1 employee, and move your pointer over *Operations*, and select *Create Schemas*.
5. In the right pane, click the *Request Schemas* tab to view the request schema.

You can use the generated request schema to create a sample XML document to be used by the adapter.

The following XML document was generated from the iWay generated schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSPY v5 rel. 3 U (http://
www.xmlspy.com)-->
<lawson xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="C:\PROGRAM~1\COMMON~1\iway\Adapters
\5.2.104\bin\sessions\default\lawson\Law1\service_HR11.1.xsd">
<productLine>
<token>
<EMP-LAST-NAME-PRE/>
<EMP-LAST-NAME/>
<EMP-NAME-SUFFIX/>
<EMP-FIRST-NAME/>
<EMP-NICK-NAME/>
<EMP-MIDDLE-NAME/>
<EMP-NAME-PREFIX/>
</token>
</productLine>
</lawson>
```

Deploying and Running the Sample Servlet

The iWay Connector for JCA includes sample code that enables you to test iWay service and event adapters. This topic describes how to configure and deploy the iWay JCA Test Tool to the BEA WebLogic Server.

Procedure How to Deploy the Sample Servlet

To deploy the sample servlet:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

The following window opens.



Administration Console
BEA WebLogic Server 8.1

WebLogic Server Administration Console
Sign in to work with the WebLogic Server domain **jcwed**

Username:

Password:

- a. Type a user name and password.
 - b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from *Deployed Resources*.
3. Click *Deploy a New Web Application*.

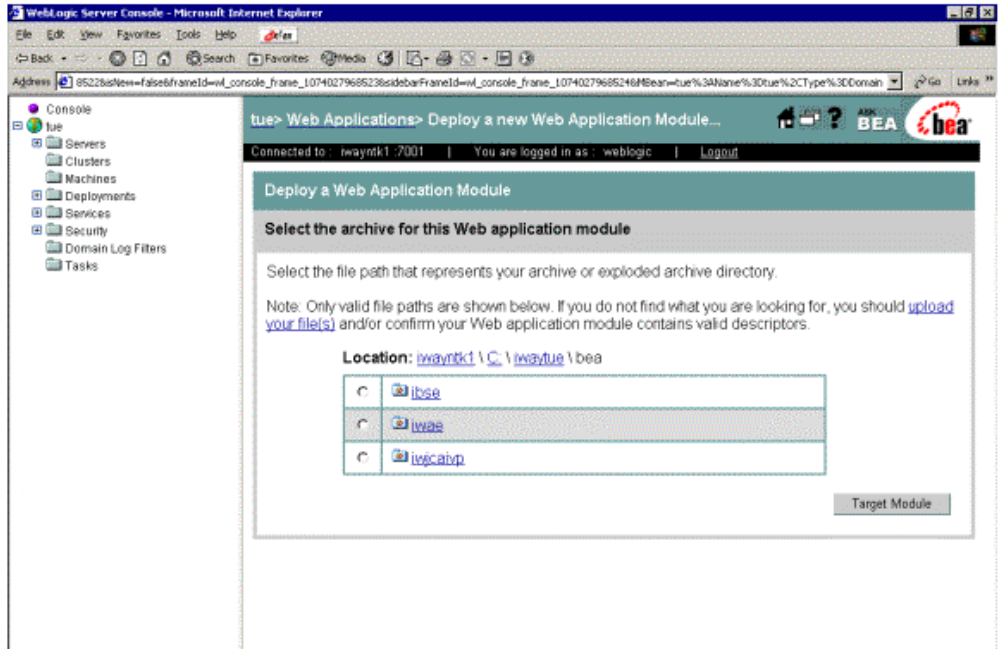
Use the location link to navigate to the location of the exploded WAR file. The WAR file is located in the *iWayHome\BEA* subdirectory.

where:

iWayHome

Is the root directory where the iWay product has been installed.

The following window opens.



4. Select *iwjcaivp*.
5. Click *Target Module*.

If deployment is successful, a window similar to the following opens:



Procedure How to Run the Sample Servlet

To run the sample servlet:

- 1. Open a browser and enter the following URL:

`http://host:port/iwjcaivp`

where:

`host`

Is the IP address or DNS name where the application server is installed.

`port`

Is the port on which the application server is listening. For a BEA WebLogic Server, the original default is 7001.

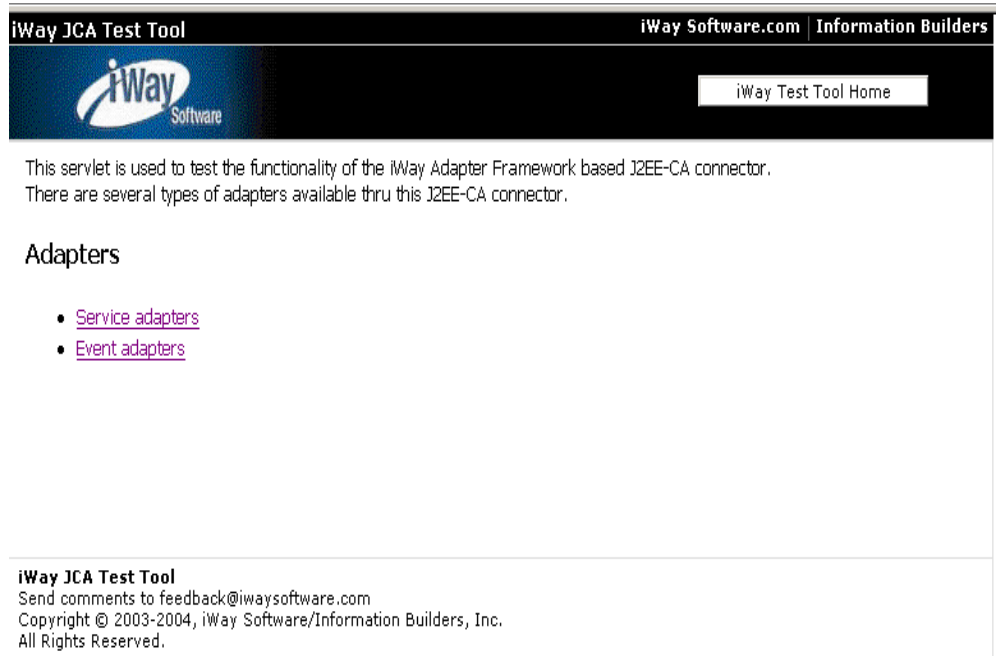
`iwjcaivp`

Is the context root to where the Web application is deployed.

For WebLogic, the default URL is

`http://localhost:7001/iwjcaivp`

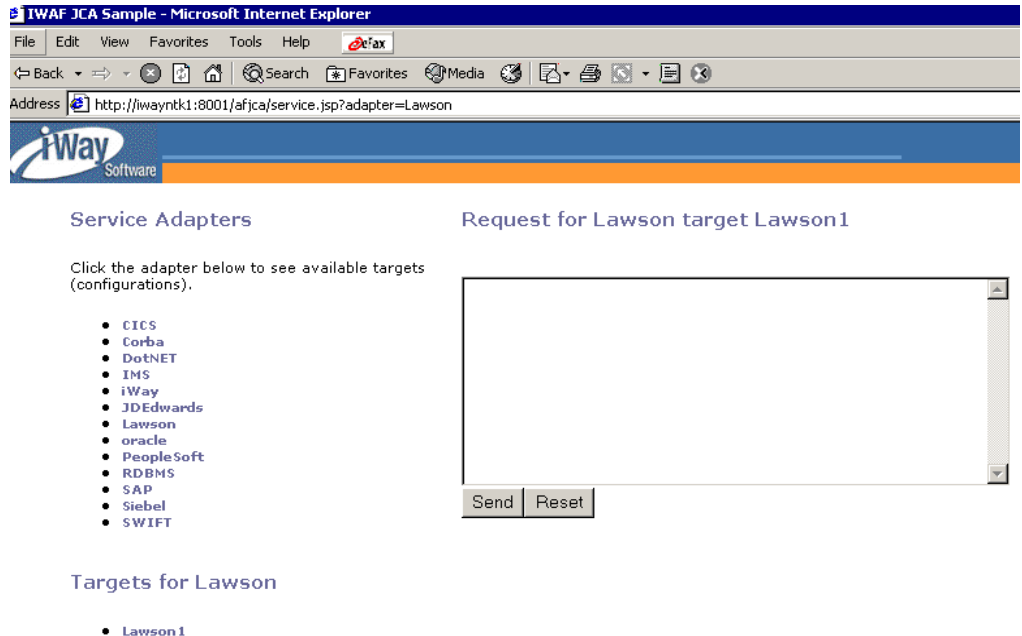
The iWay JCA Test Tool window opens.



For information about the iWay JCA Test Tool, see Chapter 3, *iWay JCA Installation Verification Program*.

2. Click the *Service adapters* link to display the available adapters.
3. Click *Lawson*.

The following window opens.



4. Enter the following to run the employee request.

```
<?xml version="1.0"
" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSPY v5 rel. 3 U (http://
www.xmlspy.com)-->
<lawson xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\PROGRAM~1\COMMON~1\iway\Adapters
\5.2.104\bin\sessions\default\lawson\Law1\service_HR11.1.xsd">
<productLine>
<token>
<EMP-LAST-NAME-PRE/>
<EMP-LAST-NAME/>
<EMP-NAME-SUFFIX/>
<EMP-FIRST-NAME/>
<EMP-NICK-NAME/>
<EMP-MIDDLE-NAME/>
<EMP-NAME-PREFIX/>
</token>
</productLine>
</lawson>
```

The following results are returned:



Service Adapters

Click the adapter below to see available targets (configurations).

- CICS
- Corba
- DotNET
- IMS
- iWay
- JDEdwards
- Lawson
- oracle
- PeopleSoft
- RDBMS
- SAP
- Siebel
- SWIFT

Targets for Lawson

- Lawson1

Request for Lawson target Lawson1

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSPY v5 rel. 3
U (http://
www.xmlspy.com)-->
<lawson
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="C:\PROGRAM~1
\COMMON~1\iway\adapters
\5.2.104\bin\sessions\default\lawson\Law1
Send Reset
```

Response in 220 msec

```
<iwaylawson>
  <productLine name="null">
    <token name="null">
      <message>
        <field> AGS call
      </field>
      <status>0</status>
      <text>Accepted AGS CALL</text>
    </message>
  </token>
</iwaylawson>
```

```
<iwaylaws
on>
  <productLine name="null">
    <token name="null">
      <message>
        <field> AGS call
      </field>
      <status>0</status>
      <text>Accepted AGS CALL</text>
    </message>
  </token>
</productLine>
</iwaylawson>
```

Accessing an Oracle Applications Business Process From a BEA WebLogic Application Server

The following topics describe how to access Oracle business objects using the iWay Connector for JCA and the iWay Adapter for Oracle Applications, hosted by a BEA WebLogic Application Server.

Connecting to Oracle Applications From iWay Application Explorer

The following procedure describes how to connect to Oracle Applications from iWay Application Explorer (iAE).

Procedure How to Connect to Oracle Applications From iWay Application Explorer

To connect to Oracle Applications:

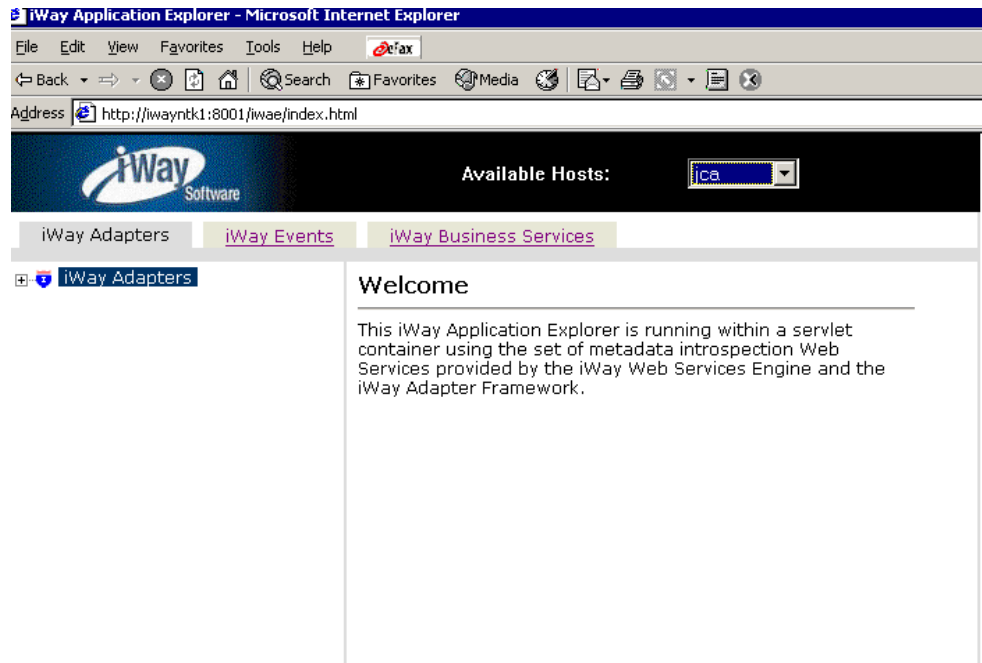
1. Install the iWay Application Explorer (iAE) on the supplied media.
2. Download the Oracle JDBC driver from the Oracle Web site at
<http://technet.oracle.com/software/content.html>
3. Add the Oracle JDBC driver to the system class path.
4. Determine the hostname and port for your application server, and then type the following URL:

<http://hostname:port/iwae/index.html>

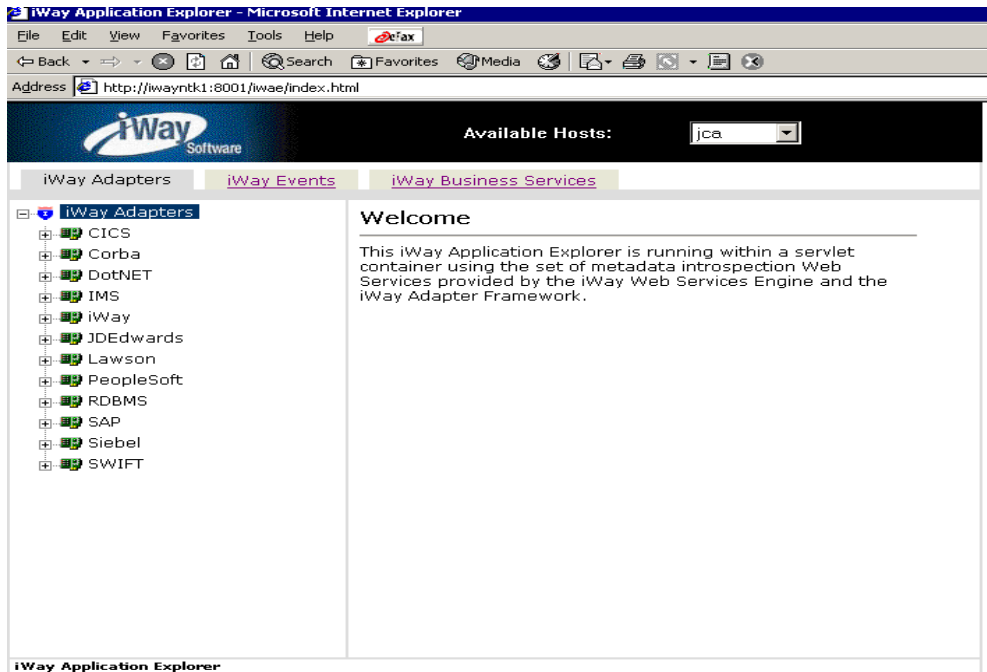
For WebLogic with the default port and domain on your localhost, go to:

<http://localhost:7001/iwae/index.html>

Application Explorer opens.

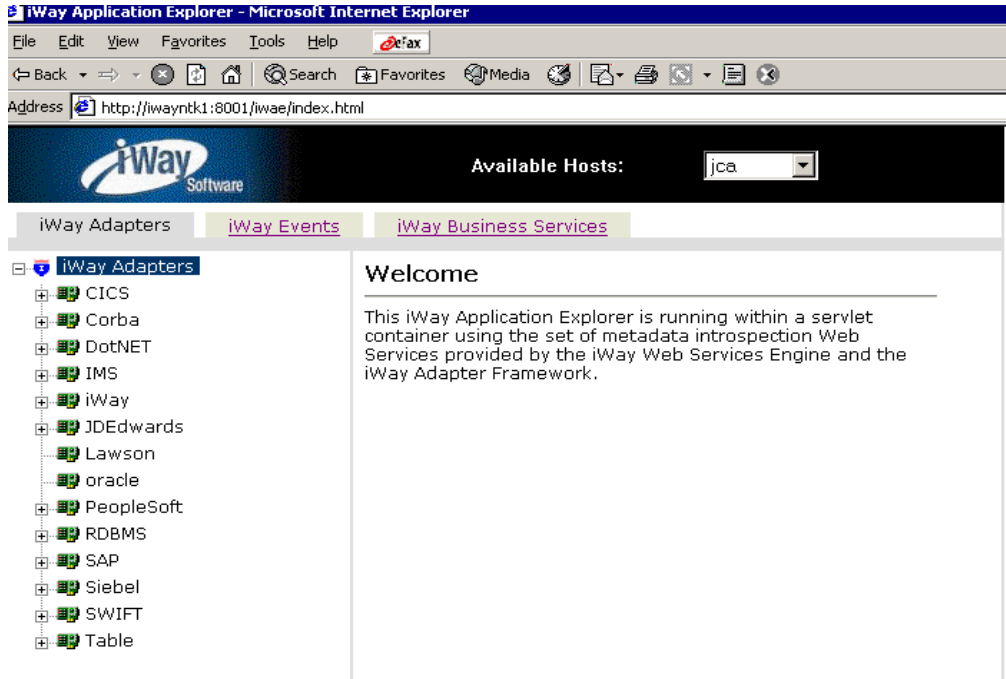


You can expand the iWay Adapter folder to display the list of iWay service adapters currently installed on your system.

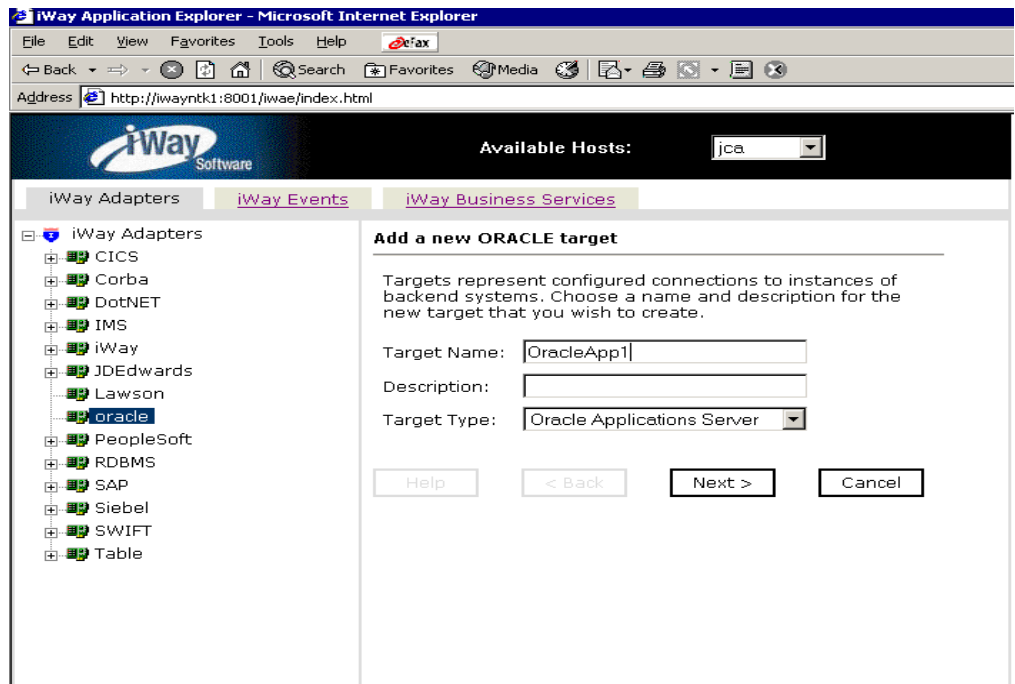


Procedure How to Establish a Connection to Oracle Applications From iWay Application Explorer

To create a new connection to your Oracle Application system:



1. Expand the iWay Adapters node in Application Explorer.
2. Expand the *Oracle* node.
3. In the left pane, move your pointer over *Operations*, and select *Define a new target*.



4. In the Add a new Oracle target dialog box:
 - a. In the Target Name field, type a name for the connection, for example, OracleApp1. The name is used to build a repository entry as well as to identify the connection.
 - b. In the Description field, type a description for the target name you just created. For example, Connection to Oracle.
 - c. In the Target Type drop-down list, select *Oracle Application Server*.
5. Click **Next**.

The connection name is verified for the system. If you entered an invalid instance name, a new dialog box opens and prompts you for an instance name again.

The Set connection info dialog box opens.

6. Type the connection parameters to create a new connection to Oracle.
 You can obtain this information from the Oracle systems administrator. This information should be the same for all Oracle Application Business Processes in a single Oracle system.

The following table lists the parameters required.

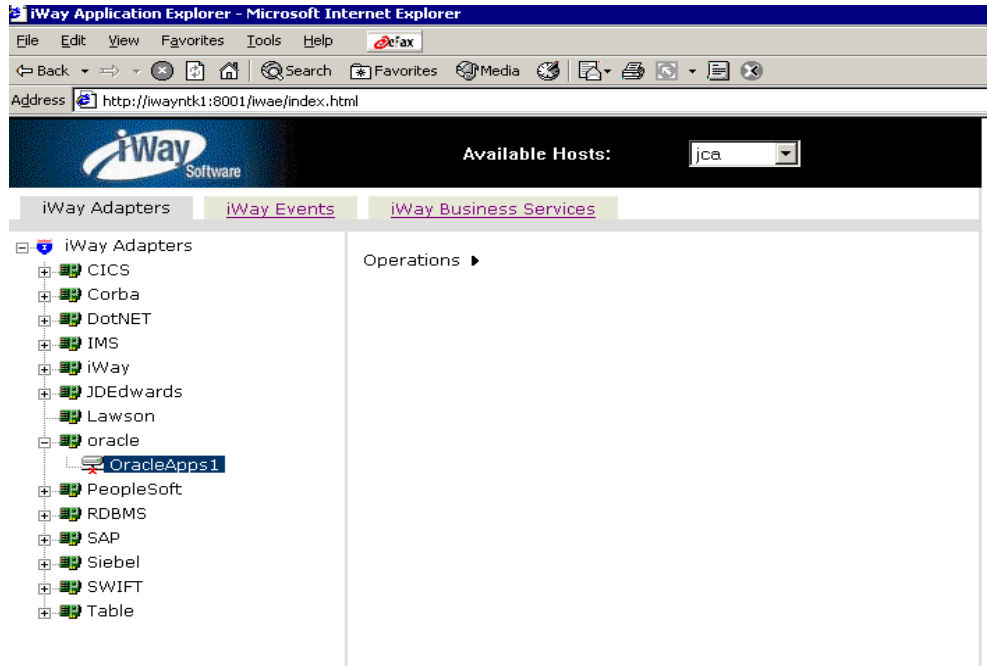
Note: The fields marked with an asterisk (*) are required.

Parameter	Description
Host	DNS or IP address on which Oracle Applications is running.
Port	Port number on which the server is listening.
User	Valid Oracle Applications user ID.
Password	Valid password associated with the user ID.
SID	Database system ID for the instance of Oracle Applications.
Batch	The number of request document records that the adapter will buffer before inserting them into a table. When the adapter reaches the end of the request document, it inserts any remaining buffered records.

For additional information on these parameters, see your Oracle Applications documentation.

7. Click *Finish*.

The newly created connection is added under Oracle in the iWay Adapters list.

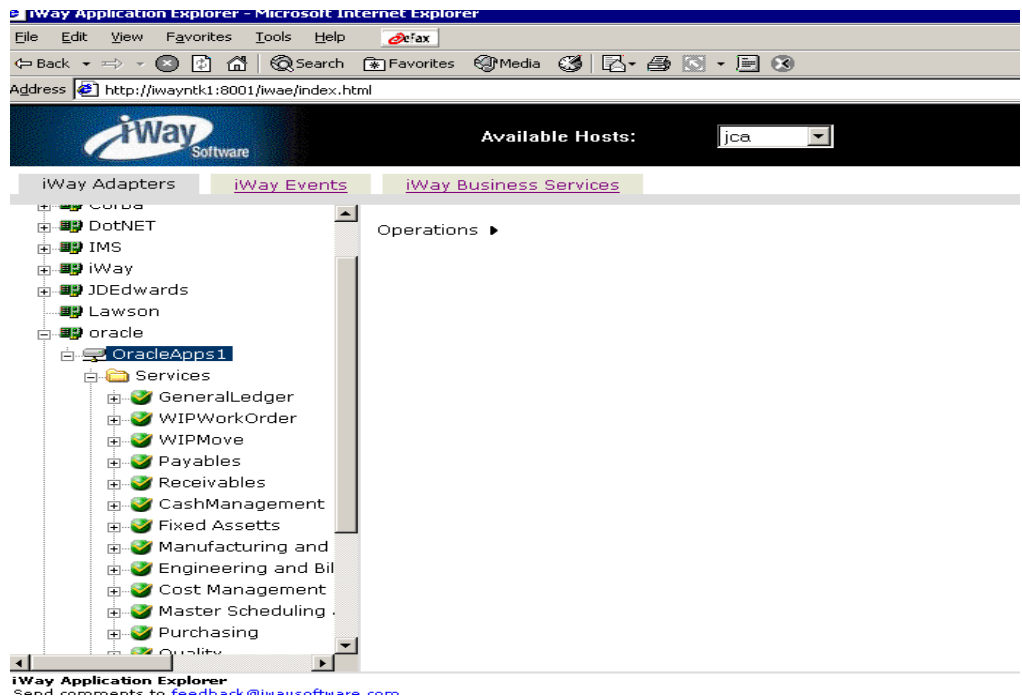


The configuration information is stored in the repository for the configuration you defined at installation time.

Procedure How to Connect to Oracle From iWay Application Explorer

To connect to Oracle from iWay Application Explorer:

1. To connect to OracleApps1, move your pointer over *Operations*, and select *Connect*.
The Connect to OracleApps1 dialog box opens, populated with the values you entered for the connection parameters.
2. Verify your connection parameters and enter a valid password for your system.
3. Click OK.
4. The iWay Application Explorer connects to Oracle and loads the Oracle metadata.



You can now create schemas for business processes. The schemas are stored in

`C:\iway55\config\base\schemas\Oracle Applications\oracleapps1`

where:

`OracleApps1`

Is the symbolic session name.

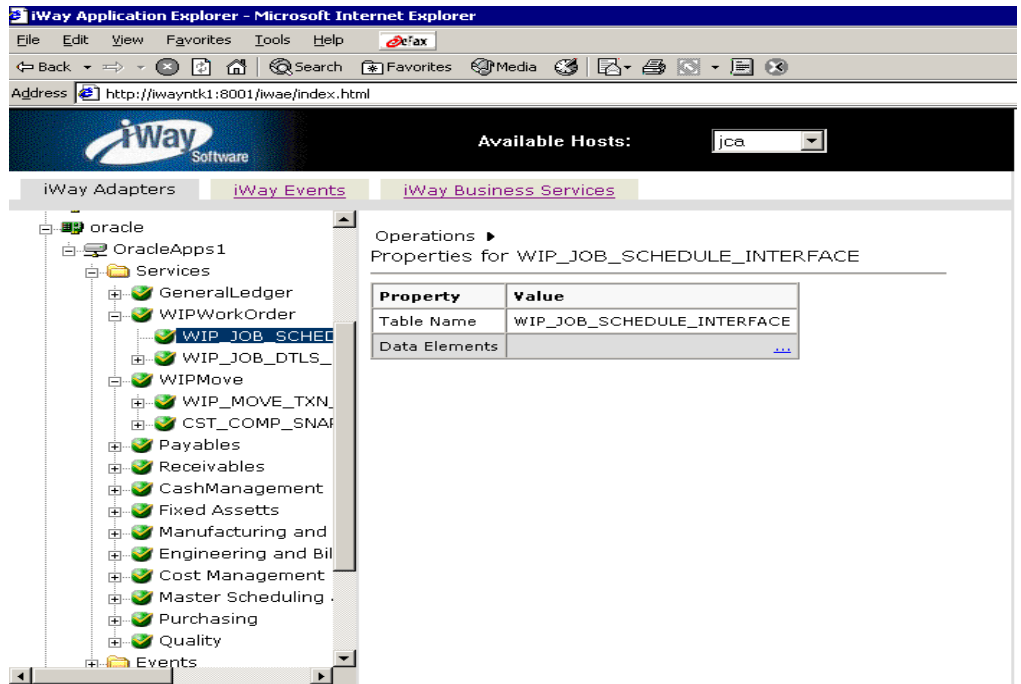
Viewing Metadata and Creating a Schema

After you establish a connection to your system, you can display the Oracle Applications system interface tables and create schemas.

Procedure How to View Metadata and Create a Schema

To display the Oracle Applications system interface tables and schemas:

1. In the left pane, click the icon to the left of the connection name to display the Oracle Applications interface tables.
You can expand and explore all the interface tables available.
2. Click an interface table to display detailed information about that table in the right pane.



3. Expand *Services*, *WIPWorkOrder*, and then *WIP_JOB_Schedule*.
4. Click *WIP_JOB_Schedule*, move your pointer over *Operations*, and select *Create Schemas*.

5. In the right pane, click the *Request Schemas* tab to view the request schema.

You can use the generated request schema to create a sample XML document to be used by the adapter.

The following XML document was generated from the iWay generated schema:

```
<eda>
<ORACLE>
  <WIP_JOB_SCHEDULE_INTERFACE>
    <LAST_UPDATE_DATE>2002-12-02 </LAST_UPDATE_DATE>
    <LAST_UPDATED_BY>1001611</LAST_UPDATED_BY>
    <CREATION_DATE>2002-12-02 </CREATION_DATE>
    <CREATED_BY>1001611</CREATED_BY>
    <LAST_UPDATE_LOGIN/>
    <REQUEST_ID/>
    <PROGRAM_ID/>
    <PROGRAM_APPLICATION_ID/>
    <PROGRAM_UPDATE_DATE/>
    <GROUP_ID>2</GROUP_ID>
    <SOURCE_CODE/>
    <SOURCE_LINE_ID/>
    <PROCESS_TYPE/>
    <ORGANIZATION_ID/>
    <LOAD_TYPE>1</LOAD_TYPE>
    <STATUS_TYPE>3</STATUS_TYPE>
    <OLD_STATUS_TYPE/>
    <LAST_UNIT_COMPLETION_DATE/>
    <OLD_COMPLETION_DATE/>
    <PROCESSING_WORK_DAYS/>
    <DAILY_PRODUCTION_RATE/>
    <LINE_ID/>
    <PRIMARY_ITEM_ID>155</PRIMARY_ITEM_ID>
    <BOM_REFERENCE_ID/>
    <ROUTING_REFERENCE_ID/>
    <BOM_REVISION_DATE/>
    <ROUTING_REVISION_DATE/>
    <WIP_SUPPLY_TYPE>7</WIP_SUPPLY_TYPE>
    <CLASS_CODE>Discrete</CLASS_CODE>
    <LOT_NUMBER/>
    <LOT_CONTROL_CODE/>
    <JOB_NAME/>
    <DESCRIPTION/>
    <FIRM_PLANNED_FLAG/>
    <ALTERNATE_ROUTING_DESIGNATOR/>
    <ALTERNATE_BOM_DESIGNATOR> </ALTERNATE_BOM_DESIGNATOR>
    <DEMAND_CLASS/>
    <START_QUANTITY>100</START_QUANTITY>
    <OLD_START_QUANTITY/>
```

```

<WIP_ENTITY_ID/>
<REPETITIVE_SCHEDULE_ID/>
<ERROR/>
<PARENT_GROUP_ID/>
<ATTRIBUTE_CATEGORY/>
<ATTRIBUTE1/>
<ATTRIBUTE2/>
<ATTRIBUTE3/>
<ATTRIBUTE4/>
<ATTRIBUTE5/>
<ATTRIBUTE6/>
<ATTRIBUTE7/>
<ATTRIBUTE8/>
<ATTRIBUTE9/>
<ATTRIBUTE10/>
<ATTRIBUTE11/>
<ATTRIBUTE12/>
<ATTRIBUTE13/>
<ATTRIBUTE14/>
<ATTRIBUTE15/>
<INTERFACE_ID/>
<LAST_UPDATED_BY_NAME/>
<CREATED_BY_NAME/>
<PROCESS_PHASE>2</PROCESS_PHASE>
<PROCESS_STATUS>1</PROCESS_STATUS>
<ORGANIZATION_CODE>M1</ORGANIZATION_CODE>
<FIRST_UNIT_START_DATE>2002-12-02 </FIRST_UNIT_START_DATE>
<FIRST_UNIT_COMPLETION_DATE>2002-12-02</
FIRST_UNIT_COMPLETION_DATE>
<LAST_UNIT_START_DATE>2002-12-02 </LAST_UNIT_START_DATE>
<SCHEDULING_METHOD/>
<LINE_CODE/>
<PRIMARY_ITEM_SEGMENTS/>
<BOM_REFERENCE_SEGMENTS/>
<ROUTING_REFERENCE_SEGMENTS/>
<ROUTING_REVISION/>
<BOM_REVISION/>
<COMPLETION_SUBINVENTORY/>
<COMPLETION_LOCATOR_ID/>
<COMPLETION_LOCATOR_SEGMENTS/>
<SCHEDULE_GROUP_ID/>
<SCHEDULE_GROUP_NAME/>
<BUILD_SEQUENCE/>
<PROJECT_ID/>
<PROJECT_NAME/>
<TASK_ID/>
<TASK_NAME/>
<NET_QUANTITY>100</NET_QUANTITY>

```

```
<DESCRIPTIVE_FLEX_SEGMENTS/>
<PROJECT_NUMBER/>
<TASK_NUMBER/>
<PROJECT_COSTED/>
<END_ITEM_UNIT_NUMBER/>
<OVERCOMPLETION_TOLERANCE_TYPE/>
<OVERCOMPLETION_TOLERANCE_VALUE/>
<KANBAN_CARD_ID/>
<PRIORITY>2</PRIORITY>
<DUE_DATE>2002-12-02</DUE_DATE>
<ALLOW_EXPLOSION>Y</ALLOW_EXPLOSION>
<HEADER_ID/>
<DELIVERY_ID/>
<COPRODUCTS_SUPPLY/>
<DUE_DATE_PENALTY/>
<DUE_DATE_TOLERANCE/>
</WIP_JOB_SCHEDULE_INTERFACE>

</ORACLE>
```

Deploying and Running the Sample Servlet

The iWay Connector for JCA includes sample code that enables you to test iWay service and event adapters. This topic describes how to configure and deploy the iWay JCA Test Tool to the BEA WebLogic Server.

Procedure How to Deploy the Sample Servlet

To deploy the sample servlet:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

The following window opens.



The screenshot shows the BEA WebLogic Server 8.1 Administration Console login interface. At the top, a teal header bar contains the text "Administration Console" and "BEA WebLogic Server 8.1". Below this, the main content area has a title "WebLogic Server Administration Console" and a subtitle "Sign in to work with the WebLogic Server domain **jcawed**". The login form includes a "Username:" label with a text input field containing "weblogic", a "Password:" label with an empty text input field, and a "Sign In" button.

- a. Type a user name and password.
 - b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from *Deployed Resources*.
3. Click *Deploy a New Web Application*.

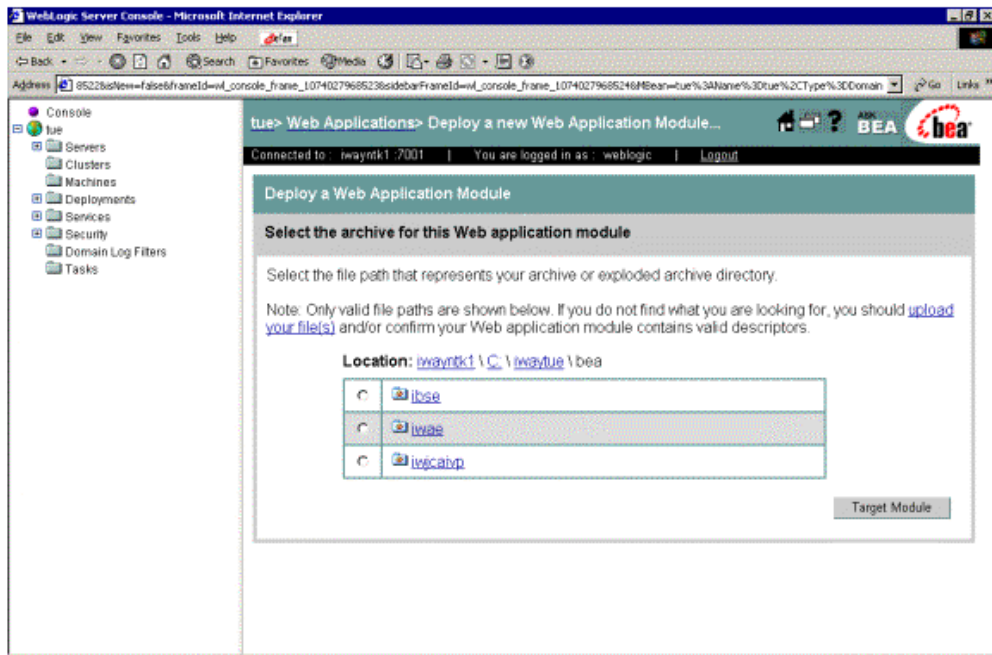
Use the location link to navigate to the location of the exploded WAR file. The WAR file is located in the *iWayHome*\BEA subdirectory.

where:

[*iWayHome*](#)

Is the root directory where the iWay product has been installed.

The following window opens.



4. Select *iwjcaivp*.
5. Click *Target Module*.

If deployment is successful, a window similar to the following opens:



Procedure How to Run the Sample Servlet

To run the sample servlet:

1. Open a browser and enter the following URL:

<http://host:port/iwjcaivp>

where:

host

Is the IP address or DNS name where the application server is installed.

port

Is the port on which the application server is listening. For a BEA WebLogic Server, the original default is 7001.

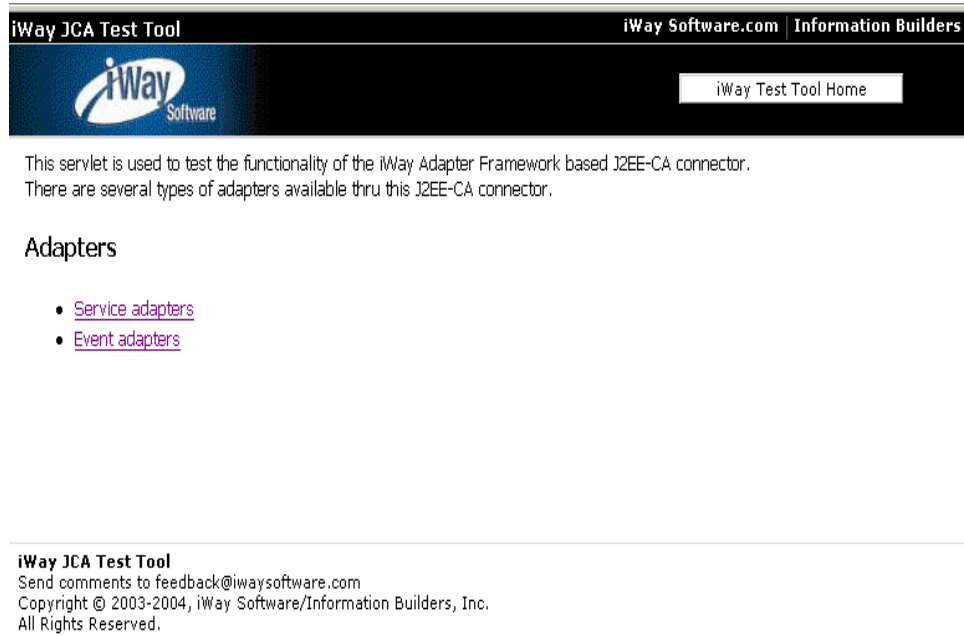
iwjcaivp

Is the context root to where the Web application is deployed.

For WebLogic, the default URL is

<http://localhost:7001/iwjcaivp>

The iWay JCA Test Tool window opens.



For information about the iWay JCA Test Tool, see Chapter 3, *iWay JCA Installation Verification Program*.

2. Click the *Services adapters* link to display the available adapters.

Click the adapter below to see available targets (configurations).

- CICS
- Corba
- DotNET
- IMS
- iWay
- JDEdwards
- Lawson
- PeopleSoft
- RDBMS
- SAP
- Siebel
- SWIFT

3. Click *Oracle*.
4. Enter the following request to run the WIP_JOB_SCHEDULE request:

```

<ORACLE>

<WIP_JOB_SCHEDULE_INTERFACE>
  <LAST_UPDATE_DATE>2002-12-02 </LAST_UPDATE_DATE>
  <LAST_UPDATED_BY>1001611</LAST_UPDATED_BY>
  <CREATION_DATE>2002-12-02 </CREATION_DATE>
  <CREATED_BY>1001611</CREATED_BY>
  <LAST_UPDATE_LOGIN/>
  <REQUEST_ID/>
  <PROGRAM_ID/>
  <PROGRAM_APPLICATION_ID/>
  <PROGRAM_UPDATE_DATE/>
  <GROUP_ID>2</GROUP_ID>
  <SOURCE_CODE/>
  <SOURCE_LINE_ID/>
  <PROCESS_TYPE/>
  <ORGANIZATION_ID/>
  <LOAD_TYPE>1</LOAD_TYPE>
  <STATUS_TYPE>3</STATUS_TYPE>
  <OLD_STATUS_TYPE/>
  <LAST_UNIT_COMPLETION_DATE/>
  <OLD_COMPLETION_DATE/>
  <PROCESSING_WORK_DAYS/>
  <DAILY_PRODUCTION_RATE/>
  <LINE_ID/>
  <PRIMARY_ITEM_ID>155</PRIMARY_ITEM_ID>
  <BOM_REFERENCE_ID/>
  <ROUTING_REFERENCE_ID/>
  <BOM_REVISION_DATE/>
  <ROUTING_REVISION_DATE/>
  <WIP_SUPPLY_TYPE>7</WIP_SUPPLY_TYPE>
  <CLASS_CODE>Discrete</CLASS_CODE>
  <LOT_NUMBER/>
  <LOT_CONTROL_CODE/>
  <JOB_NAME/>
  <DESCRIPTION/>
  <FIRM_PLANNED_FLAG/>
  <ALTERNATE_ROUTING_DESIGNATOR/>
  <ALTERNATE_BOM_DESIGNATOR> </ALTERNATE_BOM_DESIGNATOR>
  <DEMAND_CLASS/>
  <START_QUANTITY>100</START_QUANTITY>
  <OLD_START_QUANTITY/>
  <WIP_ENTITY_ID/>
  <REPETITIVE_SCHEDULE_ID/>
  <ERROR/>

```

```
<PARENT_GROUP_ID/>
<ATTRIBUTE_CATEGORY/>
<ATTRIBUTE1/>
<ATTRIBUTE2/>
<ATTRIBUTE3/>
<ATTRIBUTE4/>
<ATTRIBUTE5/>
<ATTRIBUTE6/>
<ATTRIBUTE7/>
<ATTRIBUTE8/>
<ATTRIBUTE9/>
<ATTRIBUTE10/>
<ATTRIBUTE11/>
<ATTRIBUTE12/>
<ATTRIBUTE13/>
<ATTRIBUTE14/>
<ATTRIBUTE15/>
<INTERFACE_ID/>
<LAST_UPDATED_BY_NAME/>
<CREATED_BY_NAME/>
<PROCESS_PHASE>2</PROCESS_PHASE>
<PROCESS_STATUS>1</PROCESS_STATUS>
<ORGANIZATION_CODE>M1</ORGANIZATION_CODE>
<FIRST_UNIT_START_DATE>2002-12-02 </FIRST_UNIT_START_DATE>
<FIRST_UNIT_COMPLETION_DATE>2002-12-02</
FIRST_UNIT_COMPLETION_DATE>
<LAST_UNIT_START_DATE>2002-12-02 </LAST_UNIT_START_DATE>
<SCHEDULING_METHOD/>
<LINE_CODE/>
<PRIMARY_ITEM_SEGMENTS/>
<BOM_REFERENCE_SEGMENTS/>
<ROUTING_REFERENCE_SEGMENTS/>
<ROUTING_REVISION/>
<BOM_REVISION/>
<COMPLETION_SUBINVENTORY/>
<COMPLETION_LOCATOR_ID/>
<COMPLETION_LOCATOR_SEGMENTS/>
<SCHEDULE_GROUP_ID/>
<SCHEDULE_GROUP_NAME/>
<BUILD_SEQUENCE/>
<PROJECT_ID/>
<PROJECT_NAME/>
<TASK_ID/>
<TASK_NAME/>
<NET_QUANTITY>100</NET_QUANTITY>
<DESCRIPTIVE_FLEX_SEGMENTS/>
<PROJECT_NUMBER/>
<TASK_NUMBER/>
```

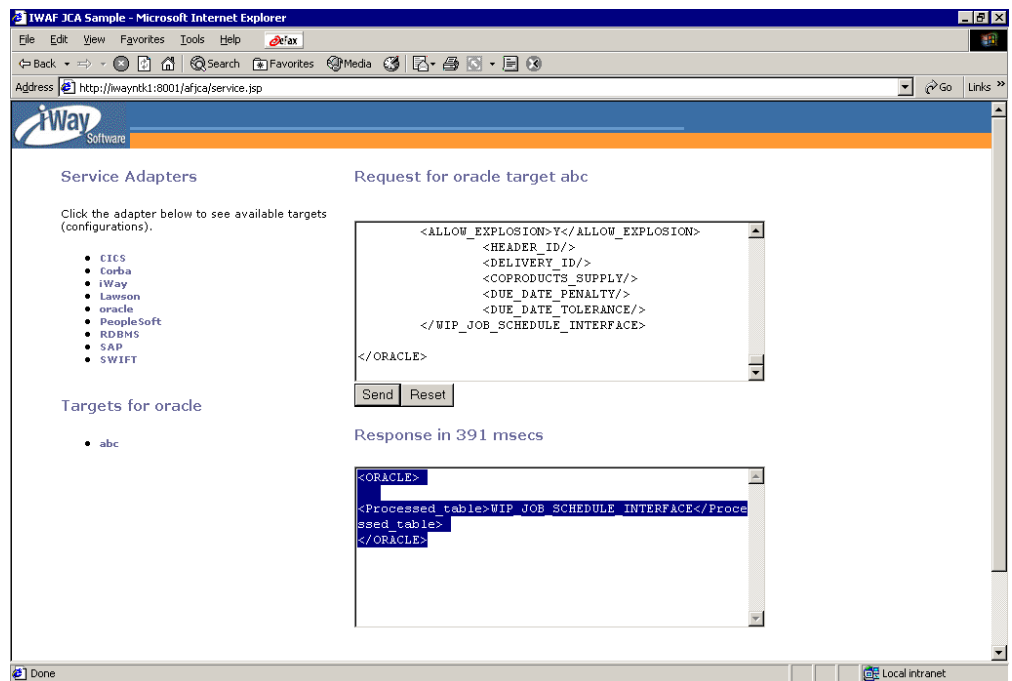
```

<PROJECT_COSTED/>
<END_ITEM_UNIT_NUMBER/>
<OVERCOMPLETION_TOLERANCE_TYPE/>
<OVERCOMPLETION_TOLERANCE_VALUE/>
<KANBAN_CARD_ID/>
<PRIORITY>2</PRIORITY>
<DUE_DATE>2002-12-02</DUE_DATE>
<ALLOW_EXPLOSION>Y</ALLOW_EXPLOSION>
<HEADER_ID/>
<DELIVERY_ID/>
<COPRODUCTS_SUPPLY/>
<DUE_DATE_PENALTY/>
<DUE_DATE_TOLERANCE/>
</WIP_JOB_SCHEDULE_INTERFACE>

</ORACLE> .

```

The results are returned as follows:



```

<ORACLE>
  <Processed_table>WIP_JOB_SCHEDULE_INTERFACE</Processed_table>
</ORACLE>

```

Accessing a PeopleSoft Business Function From BEA WebLogic Server

The following topics describe how to access a PeopleSoft Component Interface using the iWay Connector for JCA and the iWay Adapter for PeopleSoft hosted by BEA WebLogic Application Server.

Preparing to Use PeopleSoft

This topic describes the requirements for the iWay Connector for JCA to operate with the iWay Adapter for PeopleSoft.

For component interfaces, the adapter connects to the PeopleSoft Application Server by accessing APIs for the component interfaces that correspond to its supported business objects. Every component interface contains the business component's data and business logic, thus alleviating the requirement for the adapter to duplicate the processes defined within the business component.

The adapter executes a component interface by passing an XML request document to execute an instance of the PeopleSoft component interface and its method.

Place the jar file in the iWay Adapters lib directory for the iWay Adapter for PeopleSoft to communicate with the PeopleSoft component interface. For the current version of the product, the default location is:

`C:\Program Files\iWay55\lib`

The PeopleSoft Java Object Adapter, `psjoa.jar`, is also required. You can find this file on the PeopleSoft Application Server under the `PS_HOME\Web\psjoa` directory.

Place the jar file in the iWay Adapters lib directory for the iWay Adapter for PeopleSoft to communicate with the PeopleSoft component interface. For the current version of the product, the default location is:

`C:\Program Files\iWay55\lib`

Connecting to PeopleSoft From iWay Application Explorer

The following procedure describes how to connect to PeopleSoft from the iWay Application Explorer (iAE). The procedure uses a sample application servlet supplied with the iWay Connector for JCA.

Procedure How to Connect to PeopleSoft From iWay Application Explorer

To connect to PeopleSoft from iWay Application Explorer:

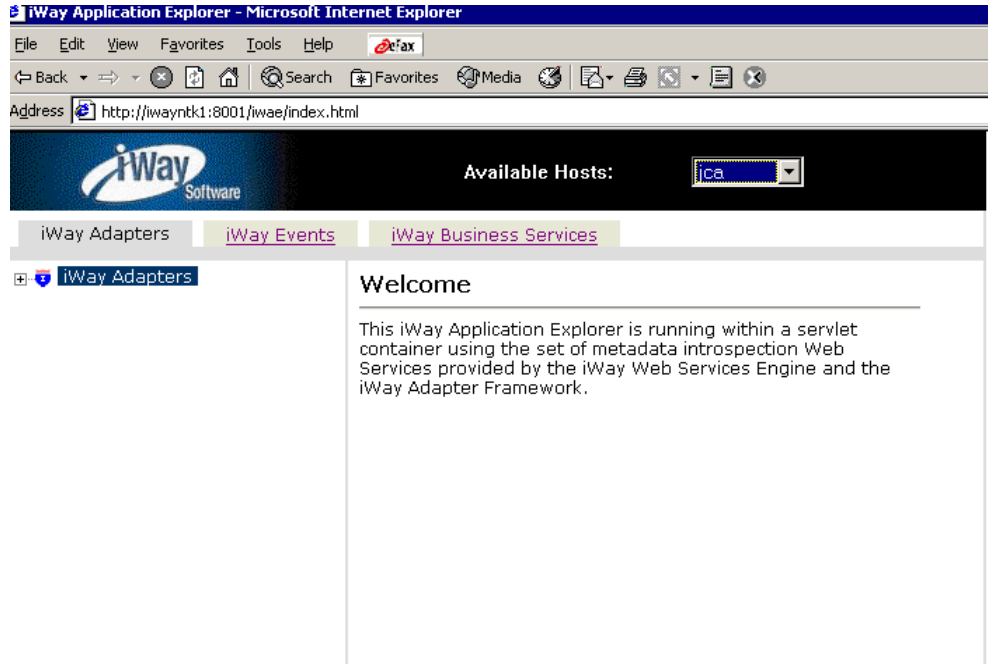
1. Follow the installation steps to install the iWay Application Explorer as described in the iWay installation manual.
2. Determine the hostname and port for your application server, and then type the following URL:

<http://hostname:port/iwae/index.html>

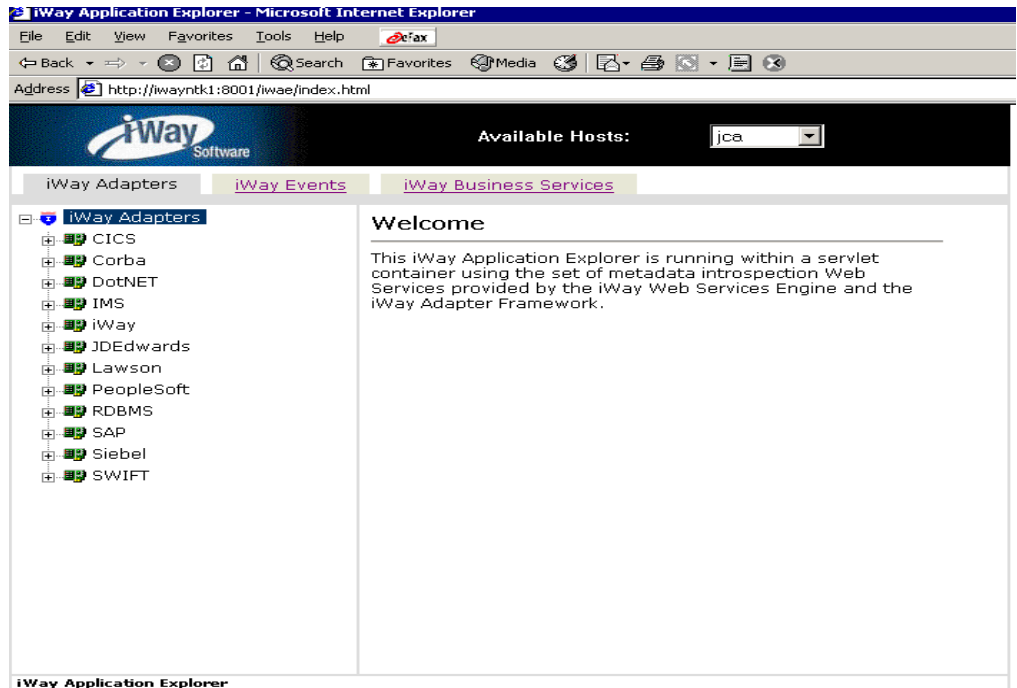
For WebLogic with the default port and domain on your localhost, go to:

<http://localhost:7001/iwae/index.html>

Application Explorer opens.



3. Select *ijca* configuration from the drop-down list of available hosts. For information on how to configure available hosts, see the installation manual.



You can expand the iWay Adapter folder to display the list of iWay service adapters currently installed on your system.

How to Establish a Connection to PeopleSoft From iWay Application Explorer

1. Expand the iWay Adapters node in Application Explorer.
2. Expand the *PeopleSoft* node.
3. In the left pane, move your pointer over *Operations*, and select *Define a new target*.

The screenshot shows the iWay Business Services console. On the left, a tree view under 'iWay Adapters' lists various adapters: CICS, Corba, iWay, JDEdwards, Lawson, oracle, **PeopleSoft**, RDBMS, SAP, SWIFT, and Table. The 'PeopleSoft' adapter is selected. On the right, a dialog box titled 'Add a new PEOPLESOFT target' is open. It contains the following fields:

- Target Name:** isdsrv14
- Description:** (empty)
- Target Type:** Application Server (selected from a dropdown menu)

At the bottom of the dialog are four buttons: Help, < Back, Next >, and Cancel.

4. In the Add a new PeopleSoft target dialog box:
 - a. In the Target Name field, type a name for the connection, for example, isdsrv14. The name is used to build a repository entry as well as to identify the connection.
 - b. In the Description field, type a description for the target name you just created. For example, Connection to psoft on isdsrv14.
 - c. In the Target Type drop-down list, select *Application Server*.
5. Click *Next*.

The connection name is verified for the system. If you entered an invalid instance name, a new dialog box opens and prompts you for an instance name again.

The Set connection info dialog box appears.

6. Type the connection parameters to create a new connection to PeopleSoft.

You can obtain this information from the PeopleSoft systems administrator. This information should be the same for all PeopleSoft component interfaces and messages in a single PeopleSoft system.

The following table lists the parameters required.

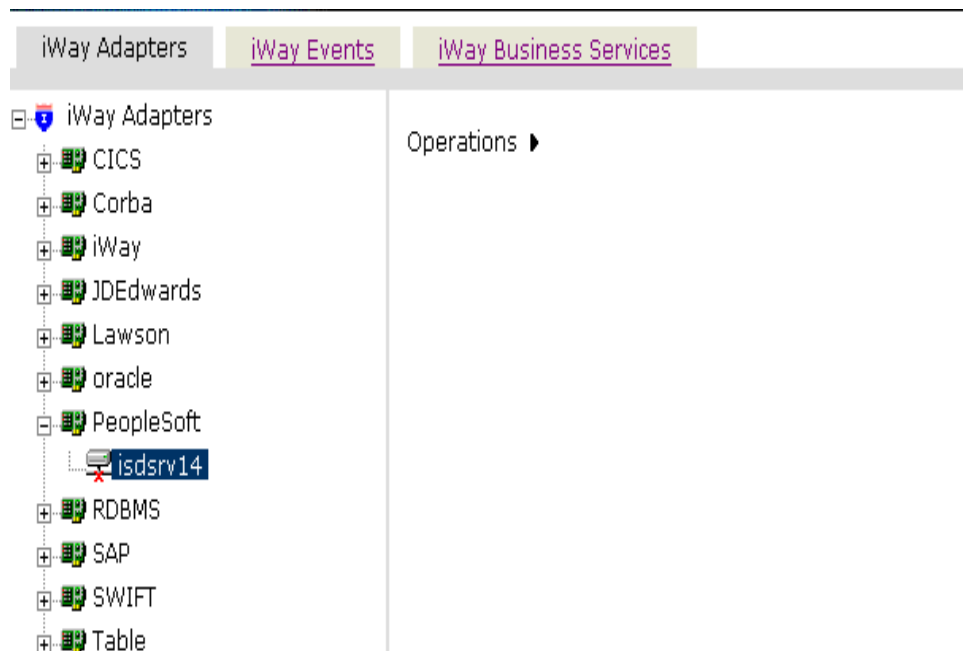
Note: The fields marked with an asterisk (*) are required.

Parameter	Description
Application Server	Name of the server on which PeopleSoft is running or its IP address, for example, isdsrv14.
Port	Port number on which the server is listening, for example, 9000.
User	Valid user ID for PeopleSoft system, for example, VP1.
Password	Password associated with the user ID, for example, adapter.

For additional information regarding these parameters, see your PeopleSoft documentation.

7. Click *Finish*.

The newly created connection is added under the PeopleSoft Service Adapter.



The configuration information is stored in the repository for the configuration you defined at installation time.

Procedure How to Connect to PeopleSoft From iWay Application Explorer

To connect to PeopleSoft from iWay Application Explorer:

1. To connect to PeopleSoft, move your pointer over *Operations*, and select *Connect*.
The Connect to isdsrv14 dialog box opens, populated with the values you entered for the connection parameters.
2. Verify your connection parameters and enter a valid password for your system.
3. Click *OK*.
4. The iWay Application Explorer connects to PeopleSoft and loads the PeopleSoft metadata.

Now you can create schemas for business processes. The schemas are stored in

`C:\iway55\config\base\schemas\PeopleSoft\isdsrv14`

where:

`isdsrv14`

Is the symbolic session name.

Viewing Metadata and Creating a Schema

To execute a component interface, a request document is received by a PeopleSoft Adapter. The adapter processes the request and sends an XML response document indicating the result.

The iWay Application Explorer (iAE) creates the following:

- XML request schema
- XML response schema

The following procedure illustrates how to create request and response schemas for a PeopleSoft Component Interface named Carrier. The iWay Application Explorer enables you to create XML agent schemas for this function.

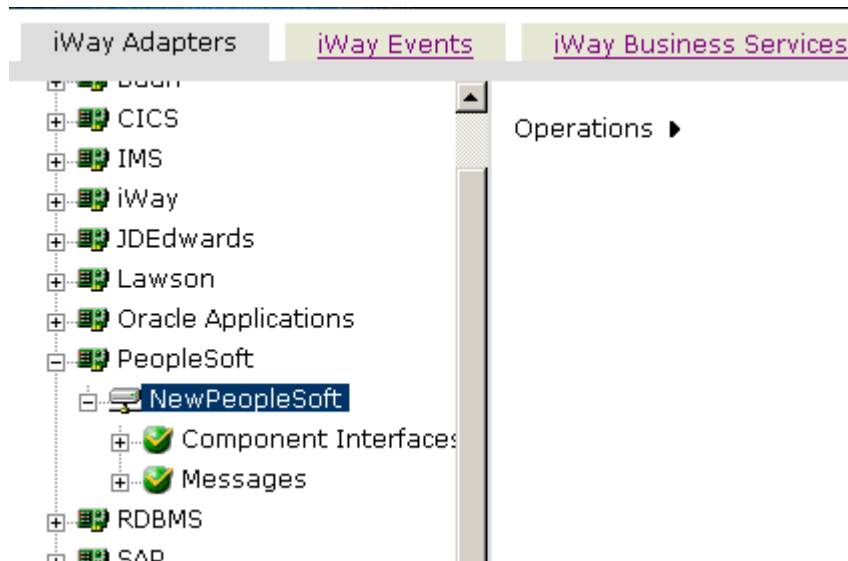
After you establish a connection to your system, you can display the PeopleSoft component interfaces and create schemas.

Procedure How to View Metadata and Create a Schema

To display the PeopleSoft component interfaces and schemas:

1. In the left pane, expand the connection name to display PeopleSoft.
You can expand and explore all the messages and component interfaces available.

2. Click a component interface to display detailed information about that business function in the right pane.



3. In the left pane, expand *Component Interfaces*, and then the *Carrier* function.
4. Click *Carrier*, move your pointer over *Operations*, and select *Create Schemas*.
5. In the right pane, click the *Request Schemas* tab to view the request schema. You can use the generated request schema to create a sample XML document used by the adapter.

The following XML document was generated from the iWay generated schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<PS8>
<component perform="browse">CARRIER</component>
<key name="SETID">SHARE</key>
<key name="CARRIER_ID">IBI</key>
</PS8>
```

Deploying and Running the Sample Servlet

The iWay Connector for JCA includes sample code that enables you to test iWay service and event adapters. This topic describes how to configure and deploy the iWay JCA Test Tool to the BEA WebLogic Server.

Procedure How to Deploy the Sample Servlet

To deploy the sample servlet:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

The following window opens.



WebLogic Server Administration Console

Sign in to work with the WebLogic Server domain **jcawed**

Username:

Password:

- a. Type a user name and password.
 - b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from *Deployed Resources*.
 3. Click *Deploy a New Web Application*.

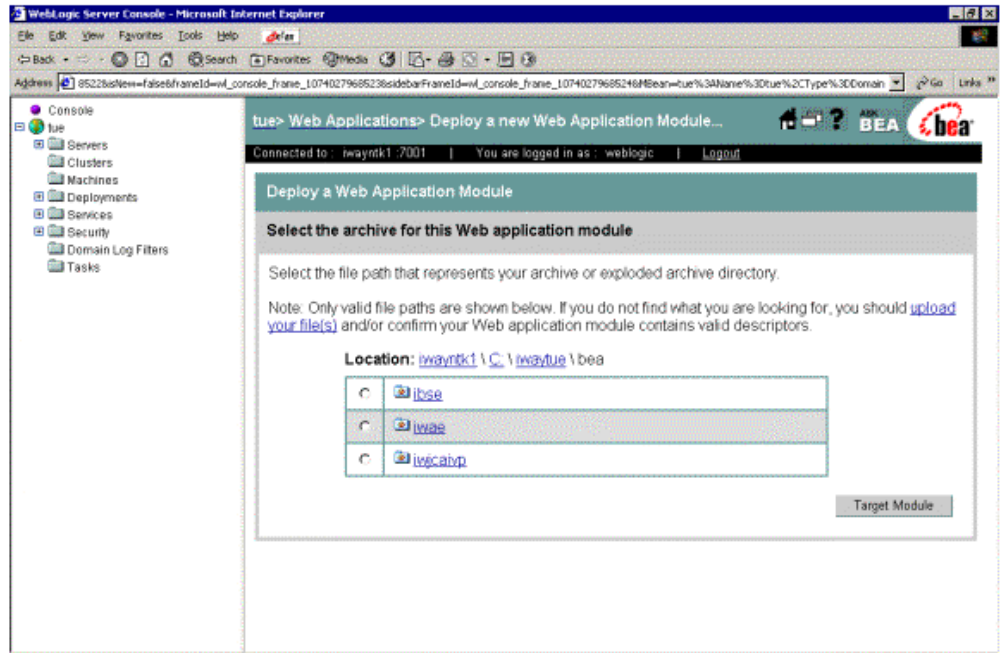
Use the location link to navigate to the location of the exploded WAR file. The WAR file is located in the *iWayHome*\BEA subdirectory.

where:

[*iWayHome*](#)

Is the root directory where the iWay product has been installed.

The following window opens.



4. Select *iwjcaivp*.
5. Click *Target Module*.

If deployment is successful, a window similar to the following opens:



Procedure How to Run the Sample Servlet

To run the sample servlet:

1. Open a browser and enter the following URL:

`http://host:port/iwjcaivp`

where:

`host`

Is the IP address or DNS name where the application server is installed.

`port`

Is the port on which the application server is listening. For a BEA WebLogic Server, the original default is 7001.

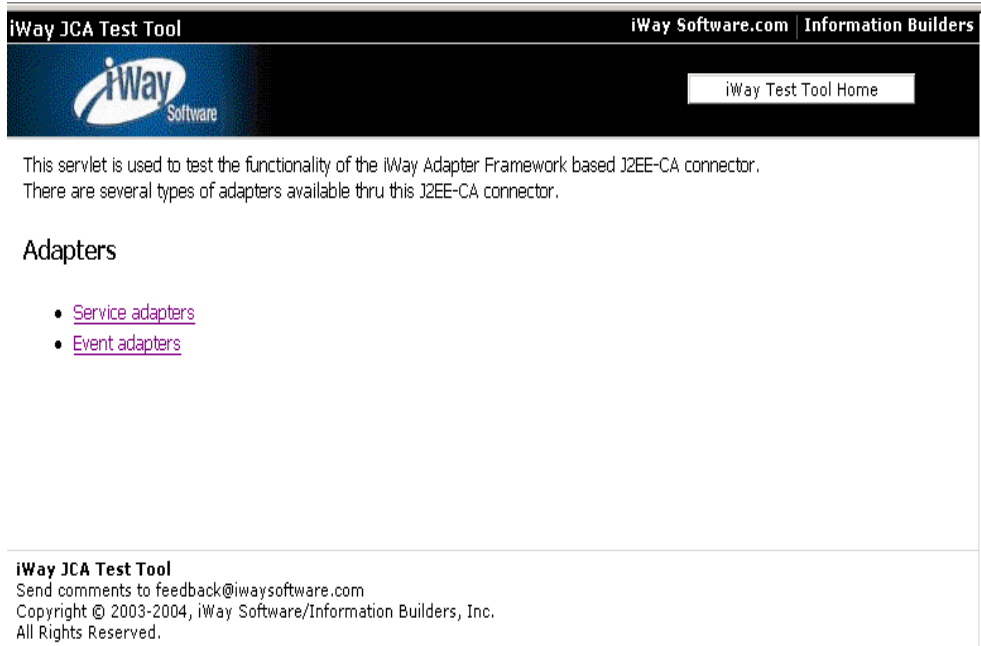
`iwjcaivp`

Is the context root to where the Web application is deployed.

For WebLogic, the default URL is

`http://localhost:7001/iwjcaivp`

The iWay JCA Test Tool window opens.



For information about the iWay JCA Test Tool, see Chapter 3, *iWay JCA Installation Verification Program*.

2. Click the *Service adapters* link to display the available adapters.

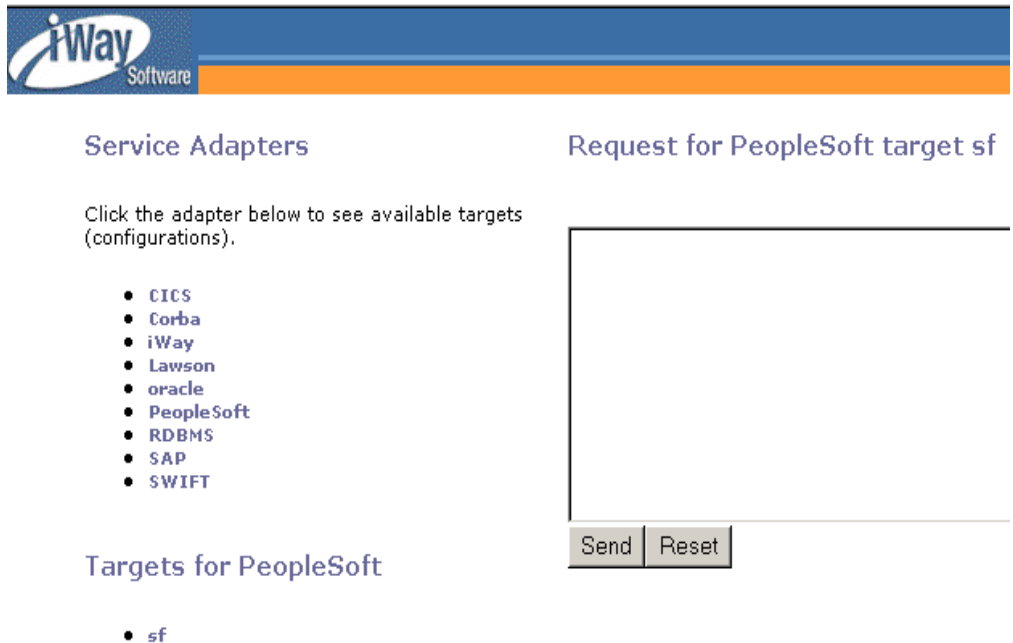
The following window opens:

Click the adapter below to see available targets (configurations).

- CICS
- Corba
- DotNET
- IMS
- iWay
- JDEdwards
- Lawson
- PeopleSoft
- RDBMS
- SAP
- Siebel
- SWIFT

3. Click *PeopleSoft*.

The following window opens.



4. Enter the following request to run a PeopleSoft Carrier request:

```
<?xml version="1.0" encoding="UTF-8"?>
<PS8>
<component perform="browse">CARRIER</component>
<key name="SETID">SHARE</key>
<key name="CARRIER_ID">IBI</key>
</PS8>
```

The following results are returned

Service Adapters

Click the adapter below to see available targets (configurations).

- CICS
- Corba
- iWay
- Lawson
- Oracle
- PeopleSoft
- RDBMS
- SAP
- SWIFT

Targets for PeopleSoft

- sf

Request for PeopleSoft target sf

```
<?xml version="1.0" encoding="UTF-8"?>
<PS8>
  <component
perform="browse">CARRIER</component>
  <key name="SETID">SHARE</key>
  <key name="CARRIER_ID">IBI</key>
</PS8>
```

Send Reset

Response in 8443 msec

```
<PS8>
  <result>
    <CARRIER_ID row="1">
      <SETID>SHARE</SETID>
      <CARRIER_ID>IBI</CARRIER_ID>
      <EFFDT>08/08/2002</EFFDT>
      <DESCR>Test</DESCR>
      <EFF_STATUS>A</EFF_STATUS>
      <TAXPAYER_ID/>
      <NETWORK_ID/>
    </CARRIER_ID>
  </result>
</PS8>
```

Accessing an SAP Business Object From BEA WebLogic Server

The following topics describe how to access an SAP business object using the iWay Connector for JCA and the iWay Adapter for SAP, hosted by a BEA WebLogic Server. The SAP Client is required for the iWay Application Explorer and the iWay Adapter for SAP.

Connecting to SAP From iWay Application Explorer

The following procedure describes how to connect to SAP from iWay Application Explorer (iAE). The procedure uses a sample application servlet supplied with the iWay Connector for JCA and a BAPI named BAPI_COMPANYCODE_GETDETAIL.

Procedure How to Connect to SAP From iWay Application Explorer

To connect to SAP from iWay Application Explorer (iAE):

1. Install the iWay Application Explorer on the supplied media.
2. Download the SAP Client from the SAP Web site at
<http://service.sap.com/connectors>
3. Add the SAP Client to the system class path.

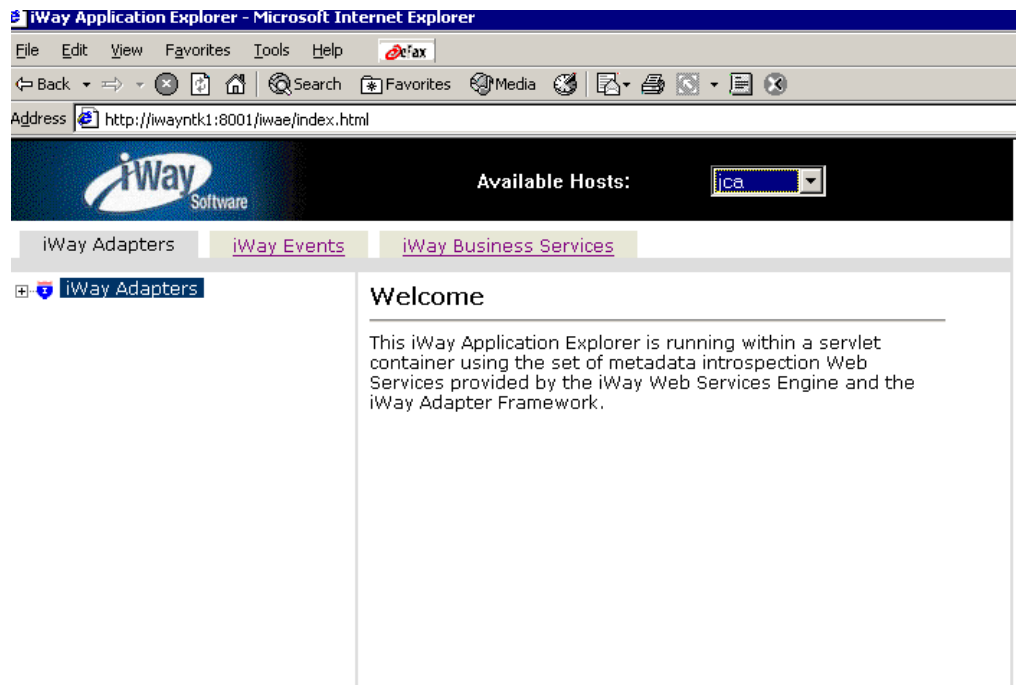
- a. Add *Sapjco.jar* to the class path.
 - b. Add the directory where the DLLs are located to the system path.
4. Determine the hostname and port for your application server, and then type the following URL:

<http://hostname:port/iwae/index.html>

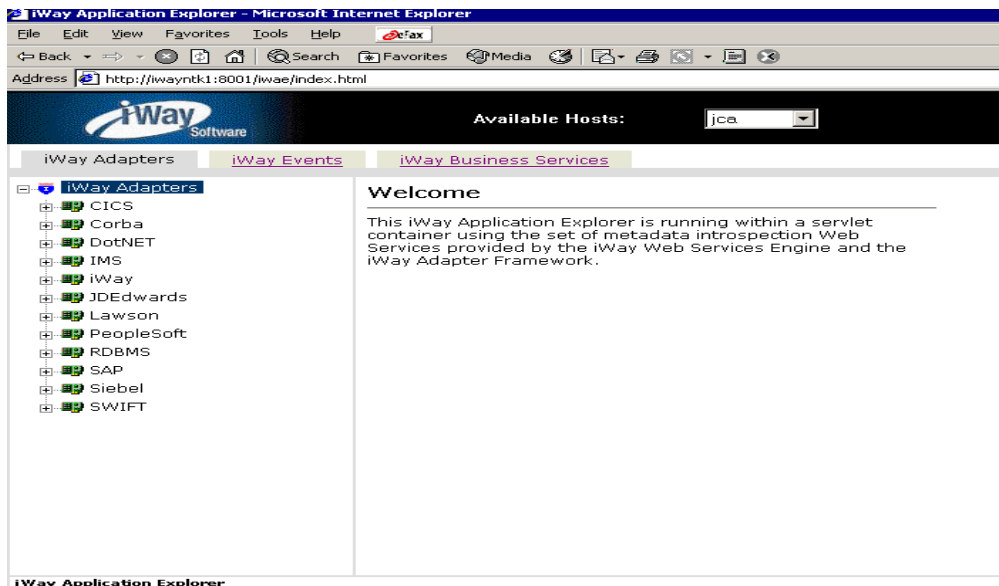
For WebLogic with the default port and domain on your localhost, go to:

<http://localhost:7001/iwae/index.html>

Application Explorer opens.

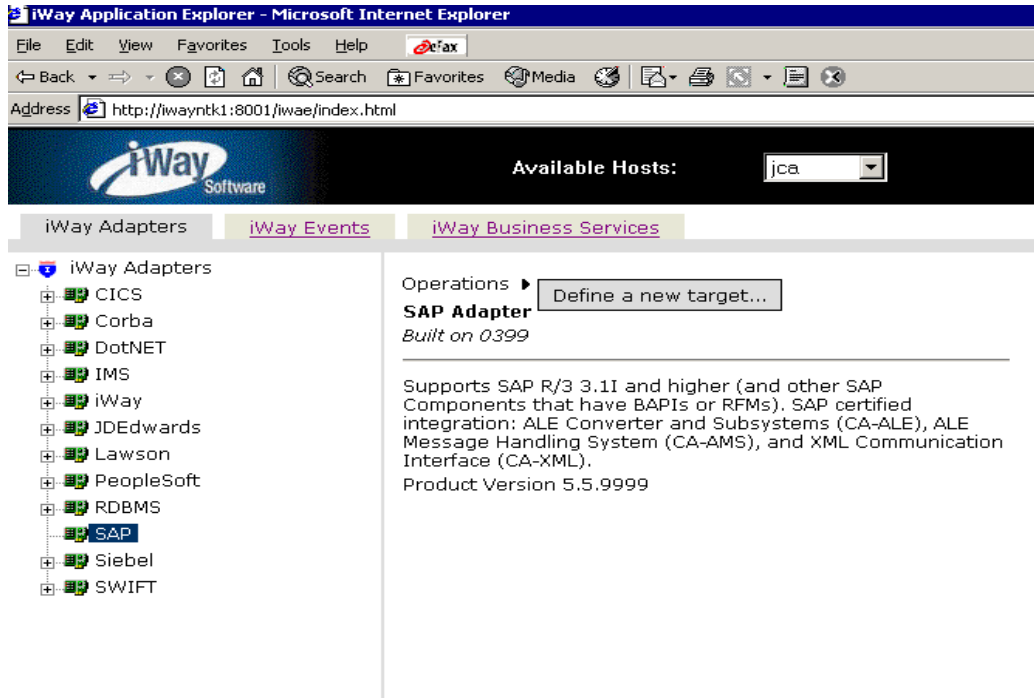


You can expand the iWay Adapter folder to display the list of iWay service adapters currently installed on your system.

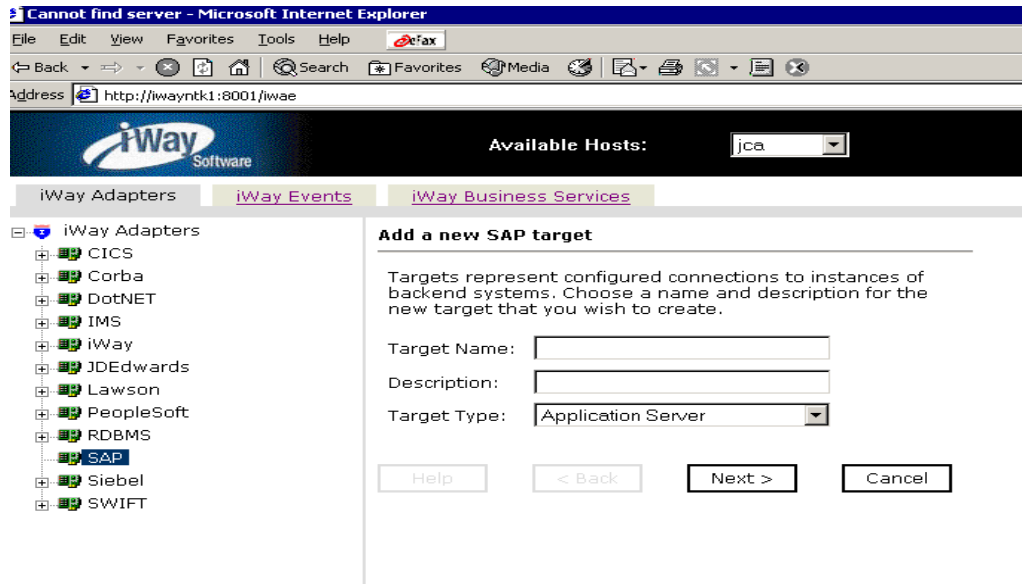


Procedure How to Establish a Connection to SAP From iWay Application Explorer

To create a new connection to your SAP system:



1. Expand the iWay Adapters node in Application Explorer.
2. Expand the SAP node.
3. In the left pane, move your pointer over *Operations*, and select *Define a new target*.



4. In the Add a new SAP target dialog box:
 - a. In the Target Name field, type a name for the connection, for example, SAP1.

The name is used to build a repository entry as well as to identify the connection.
 - b. In the Description field, type a description for the target name you just created. For example, Connection to isdsrv2.
 - c. In the Target Type drop-down list, select *Application Server*.
5. Click *Next*.

The connection name is verified for the system. If you entered an invalid instance name, a new dialog box opens and prompts you for an instance name again.

The Set connection info dialog box opens.
6. Type the connection parameters to create a new connection to SAP.

You can obtain this information from the SAP systems administrator. This information should be the same for all RFC, IDOC, and BAPIs in a single SAP system.

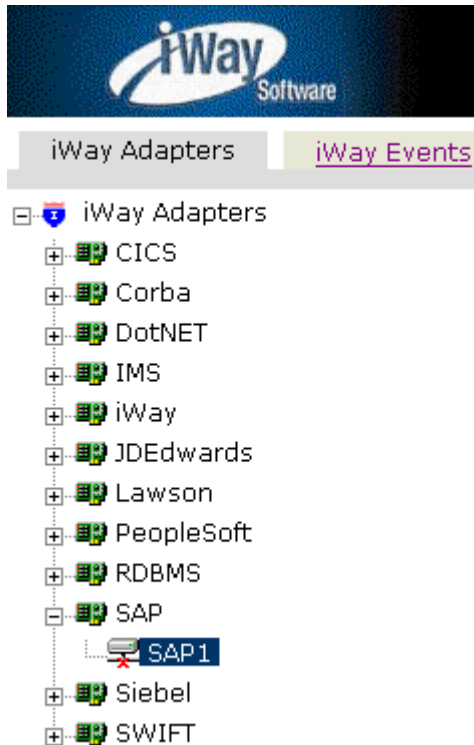
The following table lists the parameters required.

Note: The fields marked with an asterisk (*) are required.

Parameter	Description
Application Server	DNS or IP address on which the SAP Server is running.
System Number	SAP system with the SAP landscape.
EDI Version	Electronic document version.
Client	SAP Client number. This is the logical entity associated with the SAP data tables.
User	Valid SAP user ID.
Password	Valid password associated with the SAP user ID.

7. Click *Finish*.

The newly created connection is added under SAP in the iWay Adapters list.



The configuration information is stored in the repository for the configuration you defined at installation time.

Procedure How to Connect to SAP From iWay Application Explorer

To connect to SAP from iWay Application Explorer:

1. To connect to SAP1, move your pointer over *Operations*, and select *Connect*.

The Connect to SAP1 dialog box opens, populated with the values you entered for the connection parameters.

2. Verify your connection parameters and enter a valid password for your system.
3. Click OK.
4. The iWay Application Explorer connects to SAP and loads the SAP metadata.

Now you can create schemas for RFCs, IDOCs, or BAPIs. The schemas are stored in

<C:\iway55\config\base\schemas\SAP\isdsvr2>

where:

isdsrv2

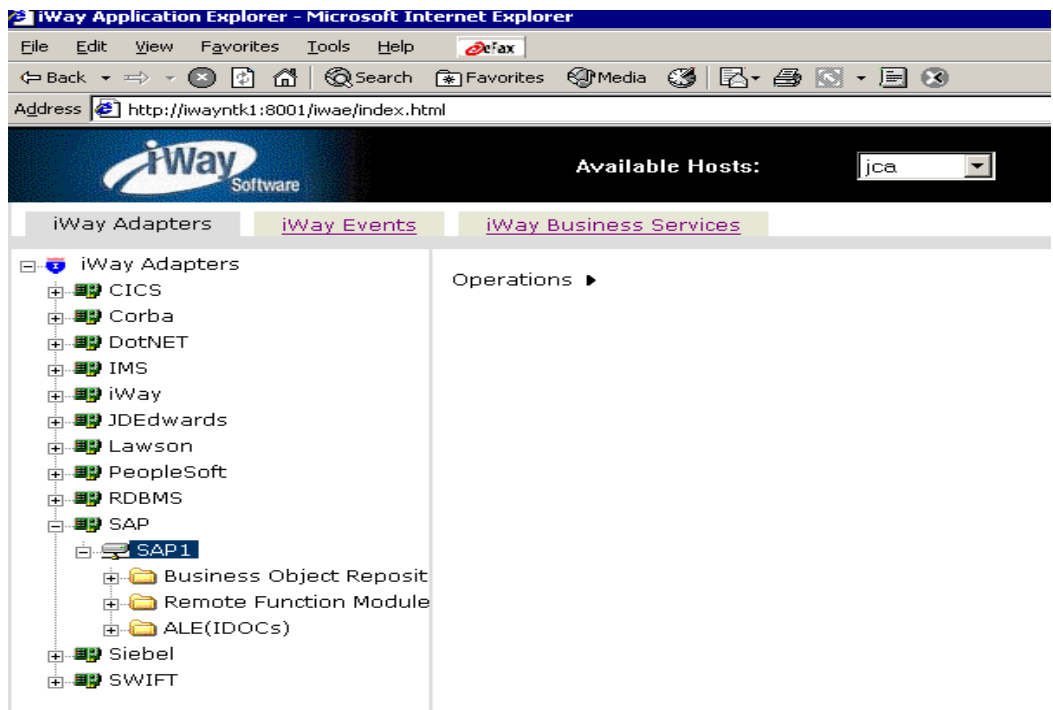
Is the symbolic session name.

Viewing Metadata and Creating a Schema

After you establish a connection to your system, you can view metadata and create schemas for SAP.

Procedure How to View Metadata and Create a Schema for SAP

To display the SAP business objects and schemas:



1. In the left pane, expand the SAP1 connection name to display the SAP business objects.
2. Select *Financial Accounting*, and then, *CompanyCode*.
3. Click *BAPI_COMPANYCODE_GETLIST*, move your pointer over *Operations*, and select *Create Schemas*.
4. In the right pane, click the *Request Schemas* tab to view the request schema.

You can use the generated request schema to create a sample XML document to be used by the adapter.

The following XML document was generated from the iWay generated schema:

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

Deploying and Running the Sample Servlet

The iWay Connector for JCA includes sample code that enables you to test iWay service and event adapters. This topic describes how to configure and deploy the iWay JCA Test Tool to the BEA WebLogic Server.

Procedure How to Deploy the Sample Servlet

To deploy the sample servlet:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

The following window opens.



WebLogic Server Administration Console

Sign in to work with the WebLogic Server domain **jcawed**

Username:

Password:

- a. Type a user name and password.

- b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from *Deployed Resources*.
3. Click *Deploy a New Web Application*.

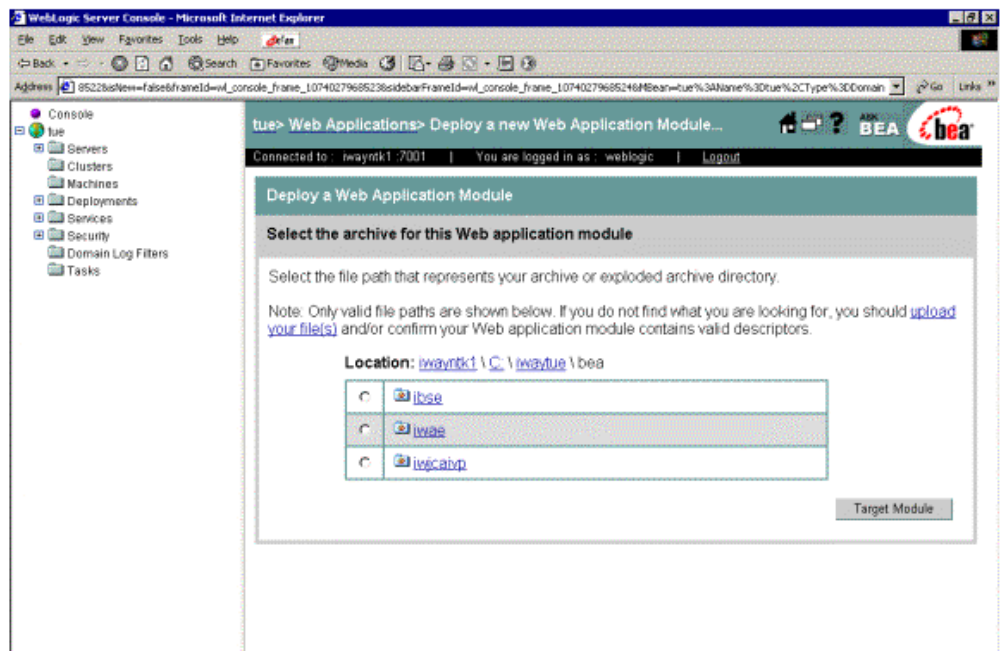
Use the location link to navigate to the location of the exploded WAR file. The WAR file is located in the *iWayHome\BEA* subdirectory.

where:

iWayHome

Is the root directory where the iWay product has been installed.

The following window opens.



4. Select *iwjcaivp*.
5. Click *Target Module*.

If deployment is successful, a window similar to the following opens:



Procedure How to Run the Sample Servlet

To run the sample servlet:

1. Open a browser and enter the following URL:

<http://host:port/iwjcaivp>

where:

[host](#)

Is the IP address or DNS name where the application server is installed.

[port](#)

Is the port on which the application server is listening. For a BEA WebLogic Server, the original default is 7001.

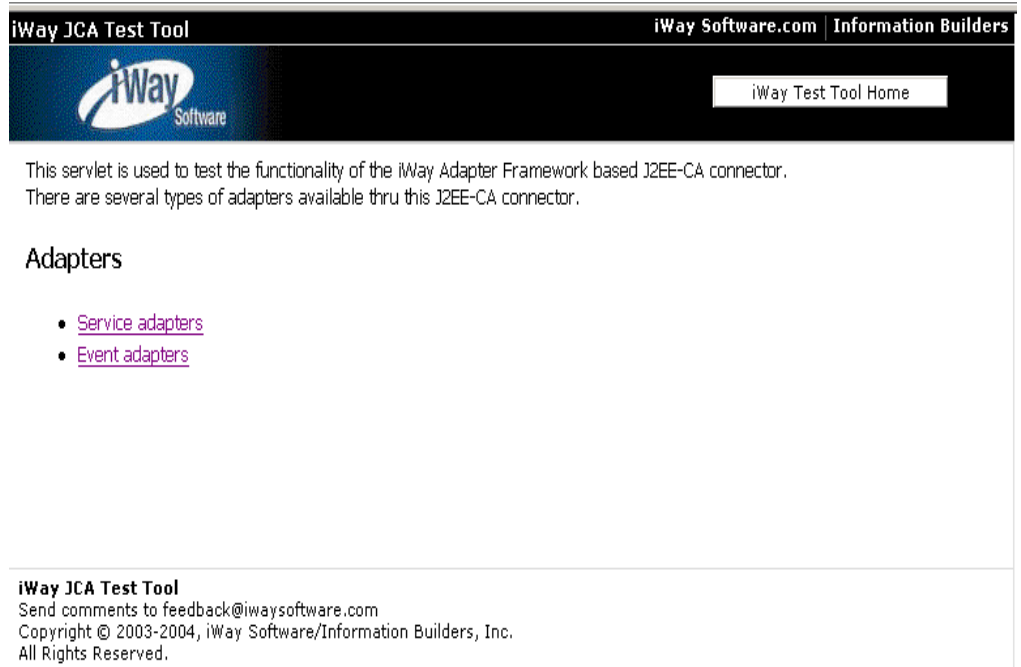
[iwjcaivp](#)

Is the context root to where the Web application is deployed.

For WebLogic, the default URL is

<http://localhost:7001/iwjcaivp>

The iWay JCA Test Tool window opens.



For information about the iWay JCA Test Tool, see Chapter 3, *iWay JCA Installation Verification Program*.

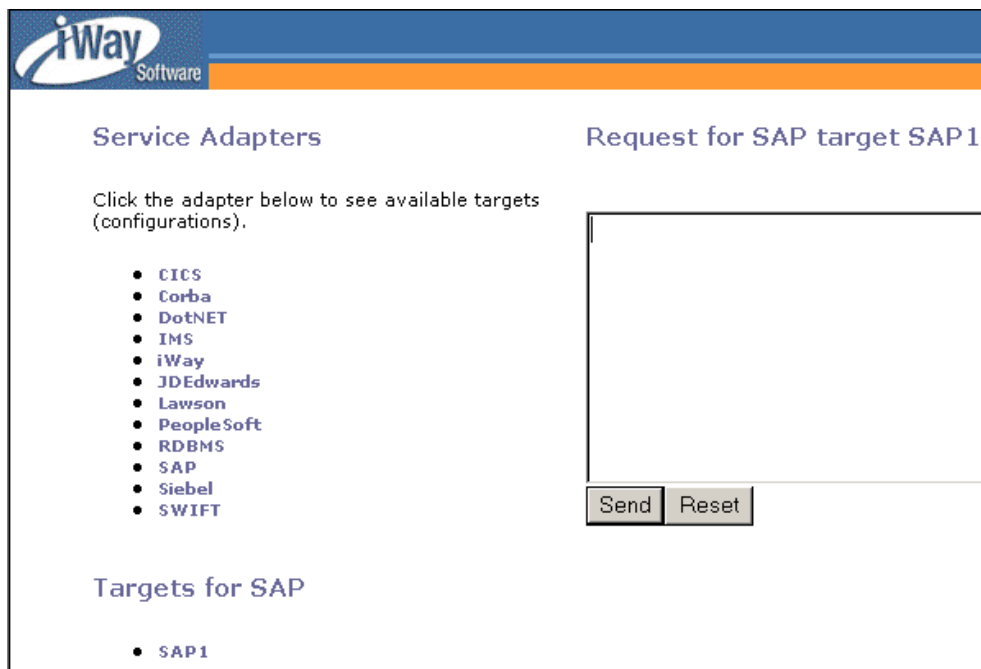
2. Click the *Service adapters* link to display the available adapters.

Click the adapter below to see available targets (configurations).

- CICS
- Corba
- DotNET
- IMS
- iWay
- JDEdwards
- Lawson
- PeopleSoft
- RDBMS
- SAP
- Siebel
- SWIFT

3. Click *SAP*.

The following window opens.



The image shows a web application window titled "Request for SAP target SAP1". The window has a blue header bar with the "iWay Software" logo. Below the header, the page is divided into two main sections. The left section is titled "Service Adapters" and contains a list of adapters: CICS, Corba, DotNET, IMS, iWay, JDEdwards, Lawson, PeopleSoft, RDBMS, SAP, Siebel, and SWIFT. A text prompt says "Click the adapter below to see available targets (configurations)". The right section is titled "Request for SAP target SAP1" and contains a large empty text area for entering a request. Below the text area are "Send" and "Reset" buttons. At the bottom of the window, there is a section titled "Targets for SAP" which contains a single entry: "SAP1".

Service Adapters

Click the adapter below to see available targets (configurations).

- CICS
- Corba
- DotNET
- IMS
- iWay
- JDEdwards
- Lawson
- PeopleSoft
- RDBMS
- SAP
- Siebel
- SWIFT

Request for SAP target SAP1

Send Reset

Targets for SAP

- SAP1

4. Type the following request to run BAPI_COMPANYCODE_GETDETAIL.

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


The following results are returned.

The screenshot shows the iWay Software interface. On the left, under 'Service Adapters', there is a list of adapters: CICS, Corba, DotNET, IMS, iWay, JDEdwards, Lawson, PeopleSoft, RDBMS, SAP, Siebel, and SWIFT. Below this, under 'Targets for SAP', there is a single target: SAP1. On the right, under 'Request for SAP target SAP1', there is a text area containing an XML request: `<?xml version="1.0" encoding="UTF-8"?><BAPI_COMPANYCODE_GETLIST></BAPI_COMPANYCODE_GETLIST>`. Below the text area are 'Send' and 'Reset' buttons. Under 'Response in 5588 msec', there is a text area containing an XML response: `<BAPI_COMPANYCODE_GETLIST.Response><RETURN><TYPE/><CODE/><MESSAGE/><LOG_NO/><LOG_MSG_NO>000000</LOG_MSG_NO><MESSAGE_V1/><MESSAGE_V2/><MESSAGE_V3/>`.

Accessing a Siebel Business Object From BEA WebLogic Server

The following topics describe how to access a Siebel Business Object using the iWay Connector for JCA and the iWay Adapter for Siebel hosted by BEA WebLogic Application Server.

Preparing to Use Siebel

This topic describes the requirements for the iWay Connector for JCA to operate with the iWay Adapter for Siebel.

The adapter uses Siebel's Java Data Bean API. The Java Data Bean API is distributed as jar files (libraries) with the Siebel Enterprise Server or Thin Client installation media. These libraries vary by Siebel release in both content and name so the files should always be taken from the target Siebel system. Refer to your Siebel documentation or administrator for the location of these files. Once located these files should be placed in the IWAY55/lib directory. The following list provides the library names supporting the Java Data Bean API for several Siebel release levels:

Siebel 6.3.x	SiebelTcOM.jar, SiebelTcCommon.jar, SiebelTC_enu.jar, SiebelDataBean.jar
--------------	--

Siebel 7.0.3	SiebelJI_Common.jar, SiebelJI_enu.jar
Siebel 7.5.2	SiebelJI_Common.jar, SiebelJI_enu.jar, SiebelJI.jar

For complete instructions on installing and configuring the Siebel Adapter, see the *iWay Installation and Configuration* manual.

Connecting to Siebel From iWay Application Explorer

The following procedure describes how to connect to Siebel from the iWay Application Explorer (iAE). The procedure uses a sample application servlet supplied with the iWay Connector for JCA.

Procedure How to Connect to Siebel From iWay Application Explorer

To connect to Siebel from iWay Application Explorer:

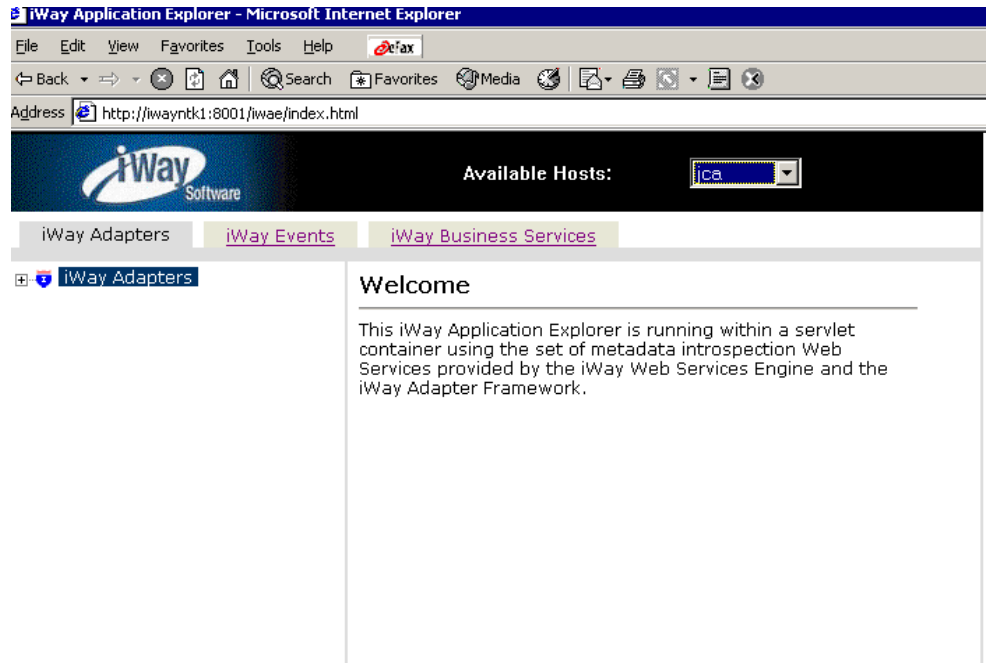
1. Follow the installation steps to install the iWay Application Explorer as described in the iWay installation manual.
2. Determine the hostname and port for your application server, and then type the following URL:

<http://hostname:port/iwae/index.html>

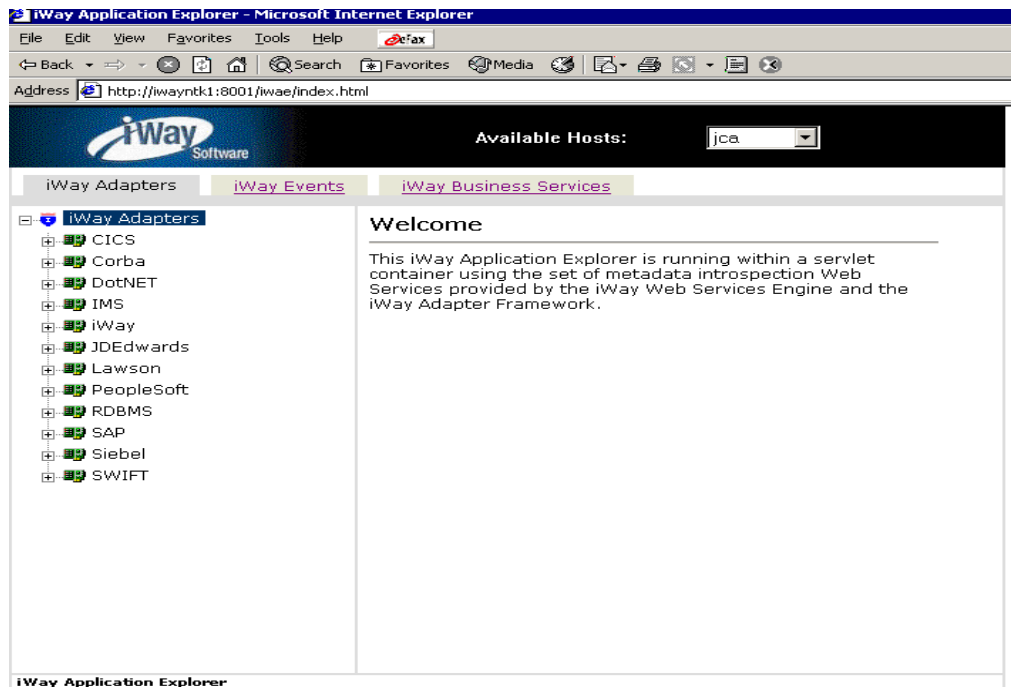
For WebLogic with the default port and domain on your localhost, go to:

<http://localhost:7001/iwae/index.html>

Application Explorer opens.



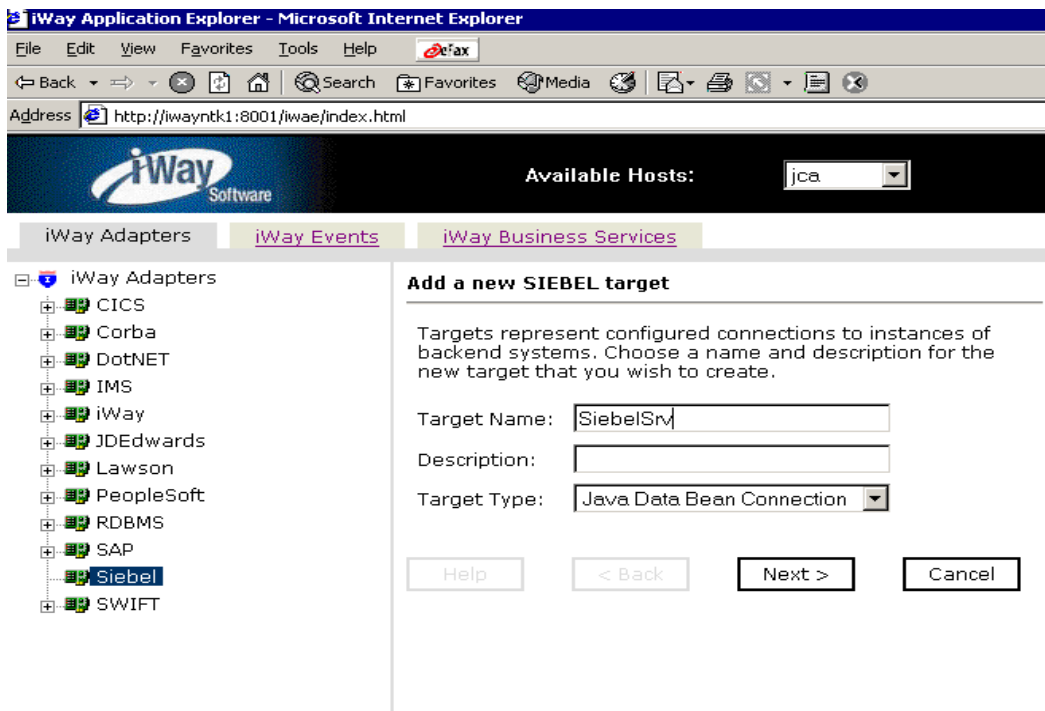
3. Select *ijca configuration* from the drop-down list of available hosts. For information on how to configure available hosts, see the installation manual.



You can expand the iWay Adapter folder to display the list of iWay service adapters currently installed on your system.

Procedure How to Establish a Connection to Siebel From iWay Application Explorer

To create a new connection to your Siebel system:



1. Expand the iWay Adapters node in Application Explorer.
2. Expand the *Siebel* node.
3. In the left pane, move your pointer over *Operations*, and select *Define a new target*.
4. In the Add a new Siebel target dialog box:
 - a. In the Target Name field, type a name for the connection, for example, SiebelSrv. The name is used to build a repository entry as well as to identify the connection.
 - b. In the Description field, type a description for the target name you just created. For example, Connection to Siebel on SiebelSrv.
 - c. In the Target Type drop-down list, select *Java Data Bean Connection*.
5. Click Next.

The connection name is verified for the system. If you entered an invalid instance name, a new dialog box opens and prompts you for an instance name again.

The Set connection info dialog box appears.

6. Type the connection parameters to create a new connection to Siebel.

You can obtain this information from the Siebel systems administrator. This information should be the same for all Siebel Business Objects, Business Components, and Integration Objects in a single system.

The following table lists the parameters required.

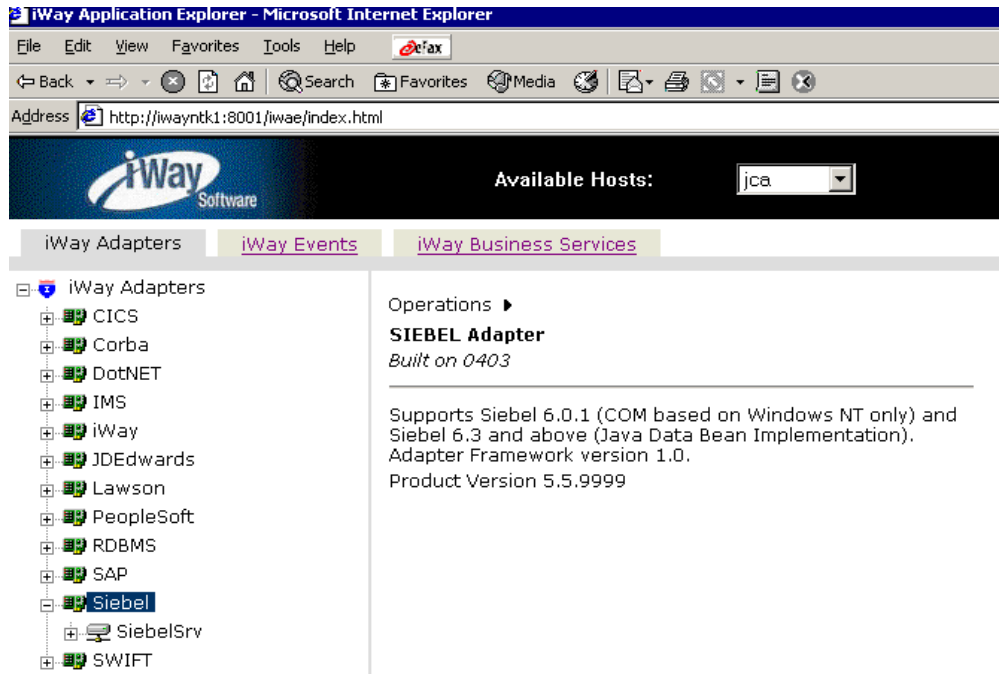
Note: The fields marked with an asterisk (*) are required.

Parameter	Description
Gateway Server	DNS name or IP address of the Siebel Gateway server. The Gateway Server is a logical grouping that contains the Name Server and the optional Resonate Central Dispatch server, and handles connections between clients and servers in a Siebel environment. The Gateway Server also performs a certain amount of load balancing by managing session logons.
Enterprise Server	DNS or IP address of the Siebel Server domain. The Siebel Enterprise Server is a logical grouping of Siebel application servers. The Siebel Enterprise Server comprises: <ul style="list-style-type: none">• One or more Siebel Servers• One Gateway Server• One file system
Siebel Server	Individual Siebel Server in the Siebel domain.
User	Valid user ID for Siebel system for example, SADMIN.
Password	Valid password associated with the user ID, for example, adapter.

For additional information regarding these parameters, see your Siebel documentation.

7. Click *Finish*.

The newly created connection is added under Siebel in the iWay Adapters list.



The configuration information is stored in the repository for the configuration you defined at installation time.

Procedure How to Connect to Siebel From iWay Application Explorer

To connect to Siebel from iWay Application Explorer (iAE):

1. To connect to Siebel, move your pointer over *Operations*, and select *Connect*.

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

2. Verify your connection parameters and enter a valid password for your system.
3. Click OK.
4. The iWay Application Explorer connects to Siebel and loads the Siebel metadata.

Now you can create schemas for business services, business objects, business components, and integration objects. The schemas are stored in:

`C:\iway55\config\base\schemas\Siebel\SiebelSrv`

where:

SiebelSrv

Is the symbolic session name.

Viewing Metadata and Creating a Schema

To execute a component interface, a request document is received by a Siebel Adapter. The adapter processes the request and sends an XML response document indicating the result.

The iWay Application Explorer (iAE) creates the following:

- XML request schema
- XML response schema

The following procedure illustrates how to create request and response schemas for a Siebel business object named Account. The iWay Application Explorer enables you to create XML agent schemas for this function.

After you establish a connection to your system, you can display the Siebel business objects and create schemas.

Procedure How to View Metadata and Create a Schema

To display the Siebel business objects and schemas:

1. In the left pane, expand the SiebelSrv connection name to display the business objects, business services, and integration objects.
2. In the left pane, expand *Business Object*, and then, *Account*.
3. Click *Account*, move your pointer over *Operations*, and select *Create Schemas*.
4. In the right pane, click the *Request Schemas* tab to view the request schema.

You can use the generated request schema to create a sample XML document to be used by the adapter.

The following XML document was generated from the iWay generated schema:

```
<Siebel location="S/BO/Account/Account/insert">
  <insert>
    <Name>1234Dave</Name>
    <Street_Address>yyyyy</Street_Address>
    <City>GRIND_2</City>
  </insert>
</Siebel>
```


Deploying and Running the Sample Servlet

The iWay Connector for JCA includes sample code that enables you to test iWay service and event adapters. This topic describes how to configure and deploy the iWay JCA Test Tool to the BEA WebLogic Server.

Procedure How to Deploy the Sample Servlet

To deploy the sample servlet:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

The following window opens.



The screenshot shows the BEA WebLogic Server Administration Console login interface. At the top, a teal header bar contains the text "Administration Console" and "BEA WebLogic Server 8.1". Below this, the main heading is "WebLogic Server Administration Console" in a teal font, followed by the instruction "Sign in to work with the WebLogic Server domain **jcawed**". The login form consists of two input fields: "Username:" with the text "weblogic" entered, and "Password:" which is empty. To the right of the password field is a "Sign In" button.

- a. Type a user name and password.
 - b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from *Deployed Resources*.
 3. Click *Deploy a New Web Application*.

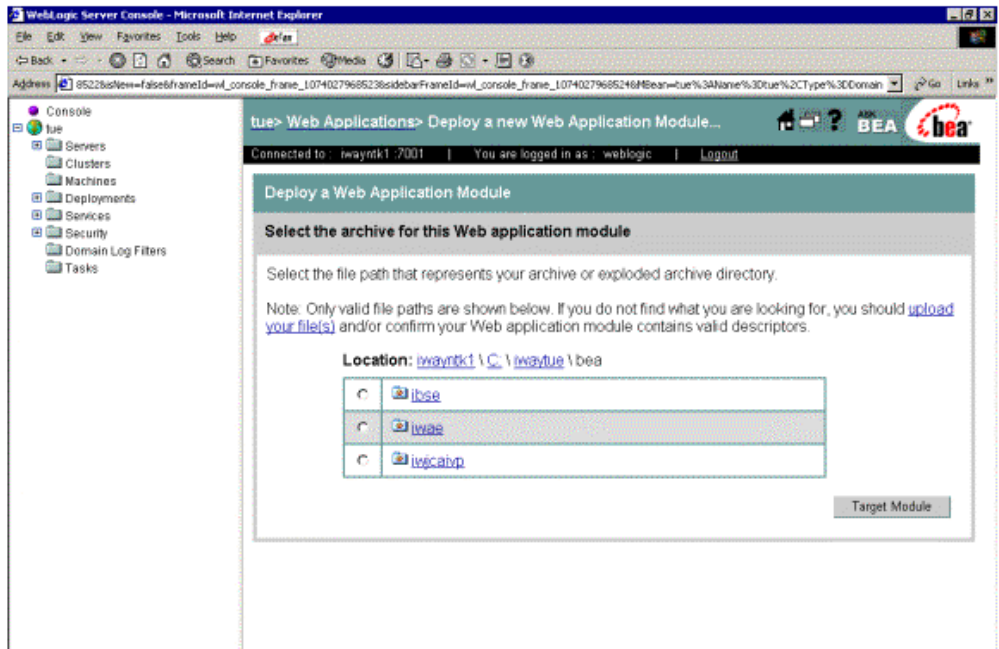
Use the location link to navigate to the location of the exploded WAR file. The WAR file is located in the *iWayHome*\BEA subdirectory.

where:

[*iWayHome*](#)

Is the root directory where the iWay product has been installed.

The following window opens.



4. Select *iwjcaivp*.
5. Click *Target Module*.

If deployment is successful, a window similar to the following opens:

The screenshot shows the BEA WebLogic Server Console interface. On the left is a tree view with the following nodes: Console, jcwed, Servers, Clusters, Machines, Deployments, Applications, EJB Modules, Web Application Modules, Connector Modules (expanded), iwayfca (selected), Startup & Shutdown, Services, Security, Domain Log Filters, and Tasks. The main content area displays the 'Deploy' tab for the 'iwayfca' connector. The page title is 'jcwed> Resource Connectors> iwayfca'. The status bar shows 'Connected to: iwayntk1:8001 | You are logged in as: weblogic | Logout'. The 'Deploy' tab is active, showing a message: 'This page allows you to view the deployment status of each Connector module, and to stop or redeploy individual Connector modules. You may also choose to stop and redeploy all Connector modules using the buttons at the bottom of the page. (To configure additional deployment targets for these Connector modules, click the Targets tab.)' Below the message is a table with the following data:

Module Status	Target	Target Type	Status of Last Action	Actions
Active	myserver	Server	Success	<input type="button" value="Stop"/> <input type="button" value="Redeploy"/>

Procedure How to Run the Sample Servlet

To deploy the sample servlet:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

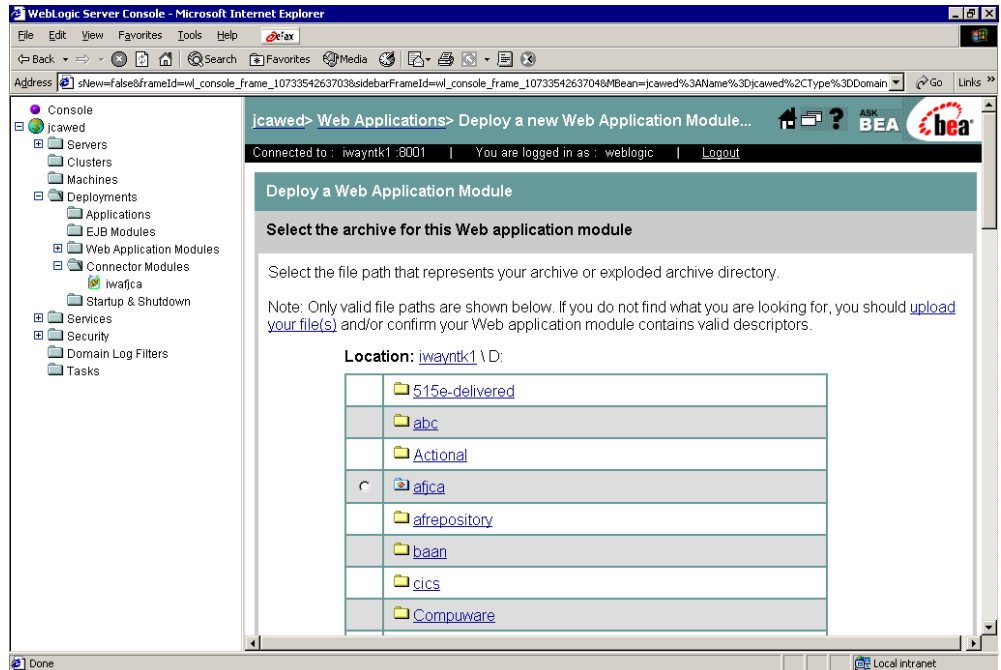
The following window opens.



The screenshot shows the BEA WebLogic Server 8.1 Administration Console login interface. At the top, a teal header bar contains the text "Administration Console" and "BEA WebLogic Server 8.1". Below this, the main content area has a title "WebLogic Server Administration Console" and a subtitle "Sign in to work with the WebLogic Server domain **jcawed**". The login form includes a "Username:" label next to a text input field containing "weblogic", a "Password:" label next to an empty text input field, and a "Sign In" button located below the password field.

- a. Type a user name and password.
 - b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from *Deployed Resources*.
 - a. Click *Deploy a New Web Application*.
 - b. Use the location link to navigate to the location of the exploded WAR file.

The following window opens.



3. Select *iwaycaivp*.
4. Click *Target Module*.

If deployment is successful, the following window opens.



Procedure How to Run the Sample Servlet

To run the sample servlet:

1. Open a browser and enter the following URL:

<http://host:port/iwjcaivp>

where:

[host](#)

Is the IP address or DNS name where the application server is installed.

[port](#)

Is the port on which the application server is listening. For a BEA WebLogic Server, the original default is 7001.

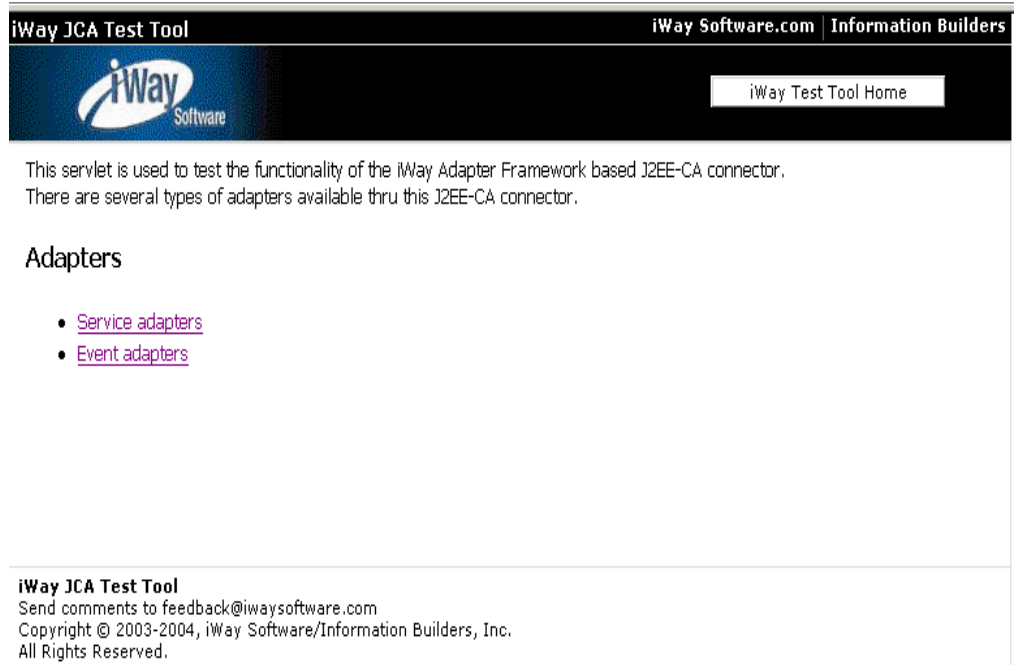
[iwjcaivp](#)

Is the context root to where the Web application is deployed.

For WebLogic, the default URL is

<http://localhost:7001/iwjcaivp>

The iWay JCA Test Tool window opens.



For information about the iWay JCA Test Tool, see Chapter 3, *iWay JCA Installation Verification Program*.


2. Click the *Service adapters* link to display the available adapters.

Click the adapter below to see available targets (configurations).

- [CICS](#)
- [Corba](#)
- [DotNET](#)
- [IMS](#)
- [iWay](#)
- [JDEdwards](#)
- [Lawson](#)
- [PeopleSoft](#)
- [RDBMS](#)
- [SAP](#)
- [Siebel](#)
- [SWIFT](#)

3. Click *Siebel*.

The following window opens.



Service Adapters

Click the adapter below to see available targets (configurations).

- CICS
- Corba
- DotNET
- IMS
- iWay
- JDEdwards
- Lawson
- PeopleSoft
- RDBMS
- SAP
- Siebel
- SWIFT

Request for Siebel target SiebelSrv

Send Reset

Targets for Siebel

- SiebelSrv

4. Enter the following request to run a Siebel account request.

```
<Siebel location="S/BO/Account/Account/insert">
  <insert>
    <Name>1234Dave</Name>
    <Street_Address>yyyyy</Street_Address>
    <City>GRIND_2</City>
  </insert>
</Siebel>.
```


The following results are returned:

```
<SiebelResponse status="success"/>
```

The screenshot shows the iWay Software interface. On the left, under "Service Adapters", there is a list of adapters: CICS, Corba, DotNET, IMS, iWay, JDEdwards, Lawson, PeopleSoft, RDBMS, SAP, Siebel, and SWIFT. Below this list, it says "Click the adapter below to see available targets (configurations)". Under "Targets for Siebel", there is a single entry: SiebelSrv.

On the right, under "Request for Siebel target SiebelSrv", there is a text area containing the following XML:


```
<Name>1234Dave</Name>
<Street_Address>yyyyy</Street_Address>
<City>GRIND_2</City>
</insert>
</Siebel>
```

 Below the text area are "Send" and "Reset" buttons.

Below the "Targets for Siebel" section, it says "Response in 1042 msec". To the right of this, there is a text area containing the following XML:


```
<SiebelResponse status="success"/>
```

Issuing a VSAM Parameterized Query From BEA WebLogic Server

This topic describes how to issue VSAM parameterized queries using the iWay Connector for JCA and the iWay Adapter for RDBMS, hosted by a BEA WebLogic Server.

Connecting to VSAM From iWay Application Explorer

The following procedure describes how to connect to VSAM from the iWay Application Explorer (IAE). The procedure uses a sample application servlet supplied with the iWay Connector for JCA.

Procedure How to Connect to VSAM From iWay Application Explorer

To connect to iWay Data Server for VSAM from iWay Application Explorer:

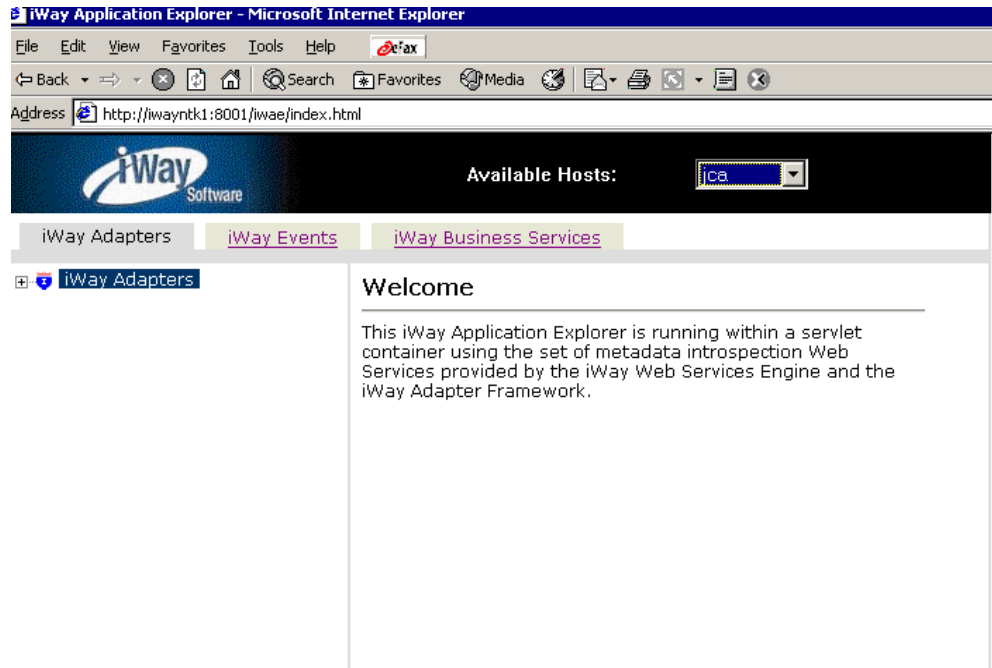
1. Follow the installation steps to install the iWay Application Explorer as described in the iWay installation manual.
2. Determine the hostname and port for your application server, and then type the following URL:

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

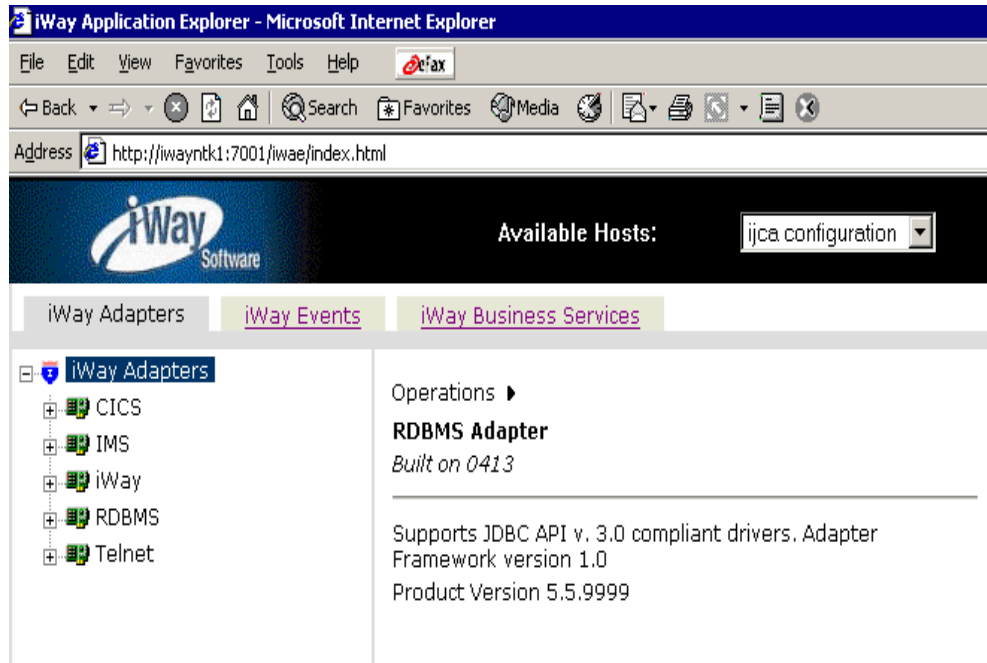
For WebLogic with the default port and domain on your localhost, go to:

<http://localhost:7001/iwae/index.html>

Application Explorer opens.



3. Select *ijca configuration* from the drop-down list of Available Hosts. For information on how to configure available hosts, see the installation manual.

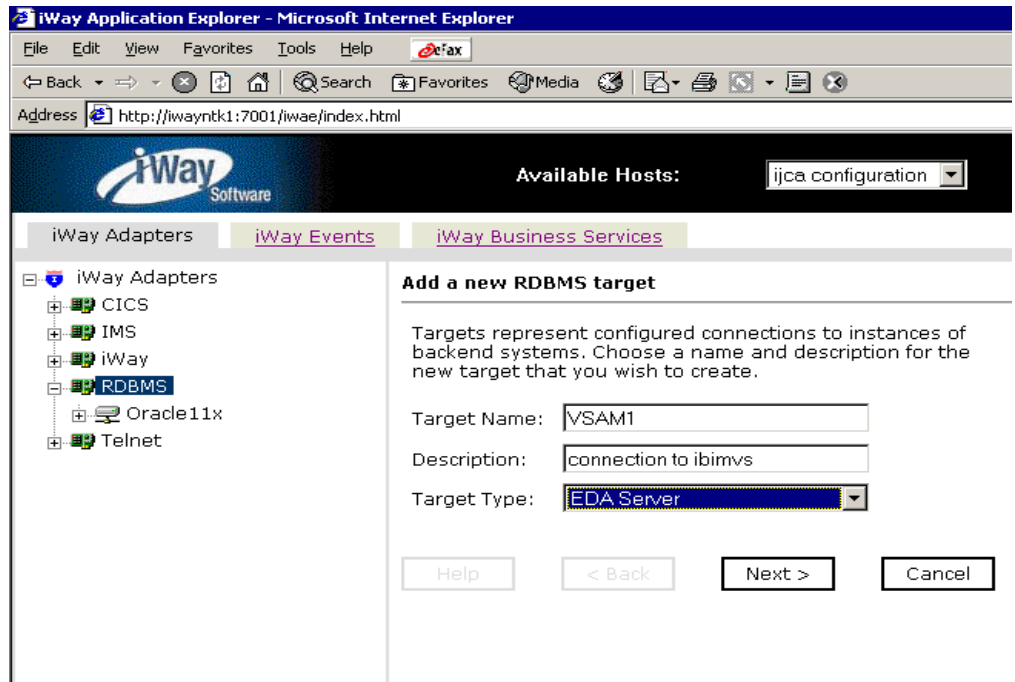


You can expand the iWay Adapter folder to display the list of iWay service adapters currently installed on your system.

Procedure How to Establish a Connection to iWay Adapter for VSAM From iWay Application Explorer

To create a new connection to your iWay Data Server for VSAM system:

1. Expand the iWay Adapters node in Application Explorer.
2. Right-click the *VSAM1* node.
3. In the left pane, move your pointer over *Operations*, and select *Define a new target*.



4. In the Add a new RDBMS target dialog box:
 - a. In the Target Name field, type a name for the connection, for example, *VSAM1*.

The name is used to build a repository entry as well as to identify the connection.
 - b. In the Description field, type a description for the target name you just created. For example, Connection to VSAM on ibimvs.
 - c. In the Target Type drop-down list, select *EDA Server*.
5. Click *Next*.

The connection name is verified for the system. If you entered an invalid instance name, a new dialog box opens, and prompts you for an instance name again.

The Set connection info dialog box appears.
6. Type the connection parameters to create a new connection to the iWay Data Server. You can obtain this information from the iWay Server administrator. This information should be the same for all iWay SQL statements and stored procedures in a single iWay Data Server environment.

The following table lists the parameters.

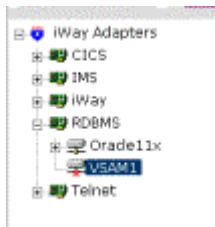
Note: The fields marked with an asterisk (*) are required.

Parameter	Description
Host	DNS name or IP address where iWay Data Server is running.
Port	port number on which the server is listening. The default port is 8100.
User	Valid user ID on the target system.
Password	Password associated with the user ID.

For additional information regarding these parameters, see your RDBMS documentation.

7. Click *Finish*.

The newly created connection is added under the RDBMS Service Adapter.



The configuration information is stored in the repository for the configuration you defined at installation time.

Procedure How to Connect to the iWay Data Server for VSAM From iWay Application Explorer

To connect to the VSAM Server from iWay Application Explorer:

1. To connect to the iWay Data Server for VSAM, move your pointer over *Operations*, and select *Connect*.

The Connect to VSAM1 dialog box opens, populated with the values you entered for the connection parameters.

2. Verify your connection parameters and enter a valid password for your system.
3. Click *OK*.
4. The iWay Application Explorer connects and loads the iWay metadata.

Now you can create schemas for. Stored procedures and SQL statements. The schemas are stored in

`C:\iway55\config\base\schemas\RDBMS\VSAM1`

where:

`VSAM1`

Is the symbolic session name.

Viewing Metadata and Creating a Schema

The iWay Application Explorer enables you to view the tables or stored procedures available in an RDBMS system.

***Procedure* How to View Tables or Stored Procedures**

To display the VSAM tables and statements:

1. In the left pane, expand the connection name to display VSAM1.
2. Expand the s% folder to display all available iWay data server tables.
3. Expand the tables folder to display all available tables.

Creating a Parameterized Request

You can generate service schemas for parameterized SQL statements using the iWay Application Explorer. Using a third party XML tool, instances (or XML templates) can be generated and used to submit against the RDBMS adapter agent. Parameterized SQL allows SQL statements to be stored within the repository system within which parameters imbedded within it. These parameters can then be taken from XML documents at run time and executed against the SQL statements specified at design time. The iWay Application Explorer is used to create and map parameters for the parameterized SQL at design time.

To execute a component interface, a request document is received by an RDBMS adapter. The adapter processes the request and sends an XML response document indicating the result.

The iWay Application Explorer (iAE) creates the following:

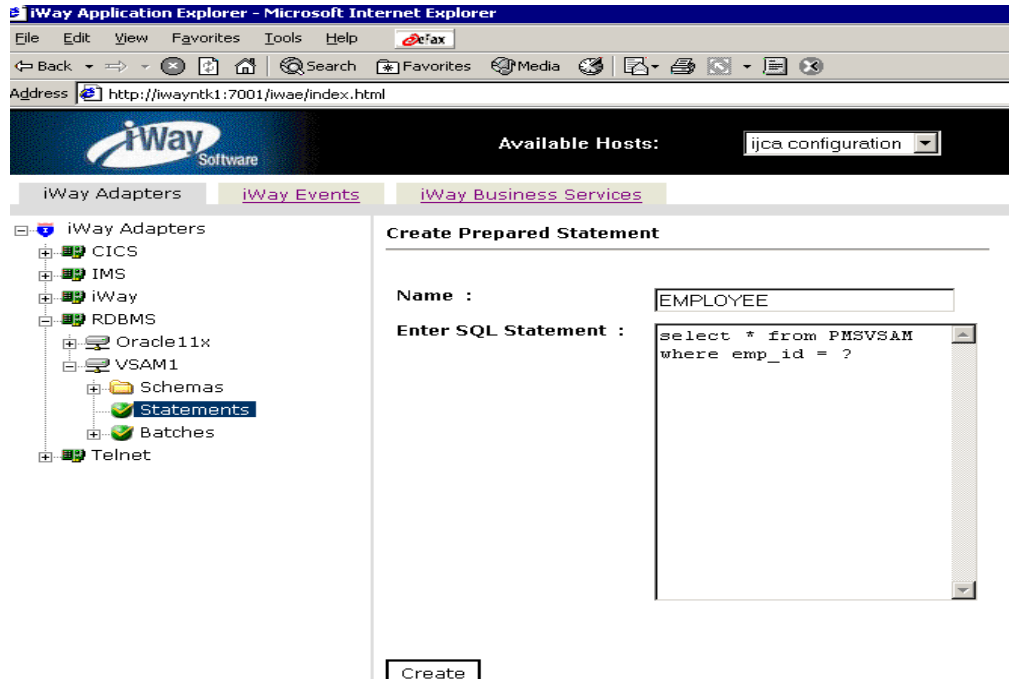
- XML request schema
- XML response schema

The following procedure illustrates how to create request and response schemas for an VSAM parameterized SQL request. The iWay Application Explorer enables you to create XML schemas for this function.

After you establish a connection to your system, you can display the iWay data server tables, statements, and batches, and create schemas for the SQL prepared statements.

Procedure How to View Metadata and Create a Schema

1. In the left pane, expand the connection name to display the VSAM1.
2. After browsing the tables in the step above, click the statements folder and move your pointer over *Operations*, and select *Create Prepared Statements*.



3. In the right pane, type a name for your prepared statement.
4. In the SQL statement box, type the SQL statement to be used by the adapter.
Question marks can be used if values are substituted at run time, for example:

```
select * from pmsvsam where emp_id = ?
```

Note: The table names must be fully qualified if you are not the owner of the table(s).

5. Click *Create*.

A window opens, prompting you to enter parameter names for each value to be substituted at run time. You must also select a data type for each parameter.

Enter a name for each parameter that you defined in your SQL query. Associate a data type with each parameter.

6. Click the new statement (this name matches the table name) and move the pointer over *Operations*, and select *Generate Schemas*.

The iWay Application Explorer creates the following:

- XML request schema
- XML response schema

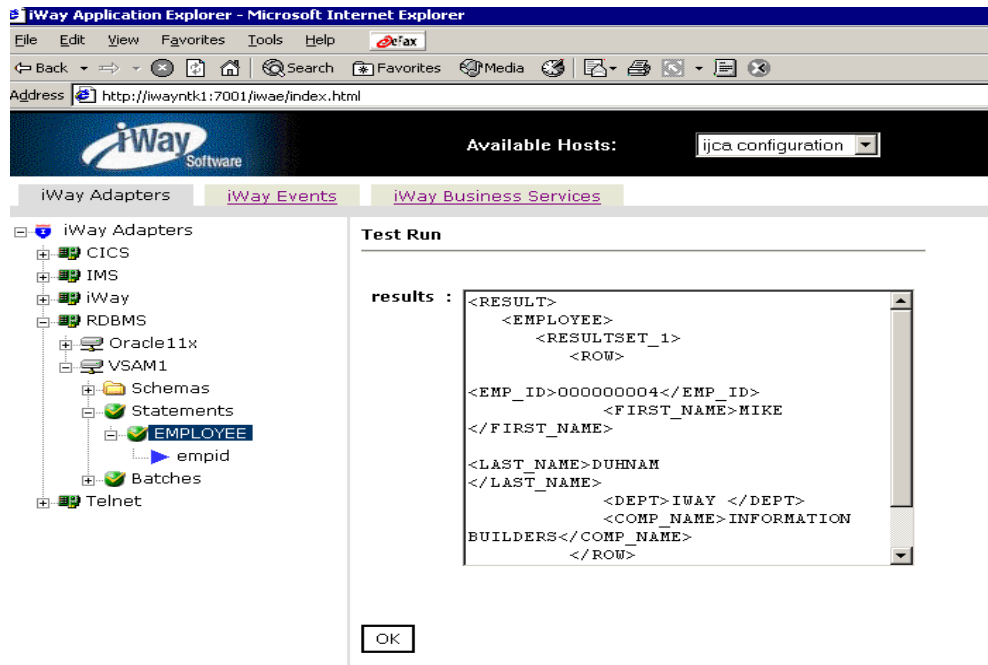
The following document is generate from the XML request schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSPY v5 rel. 4 U (http://
www.xmlspy.com)-->
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\iwaytue\config\base\schemas\RDBMS\VS
AM1\EMPLOYEE_request-335580736.xml">
  <EMPLOYEE location="RDBMS/Statements/EMPLOYEE">
    <empid dataType="CHAR">000000004</empid>
  </EMPLOYEE>
</RDBMS>
```

Note: You can also use the Test Run feature and verify that you have correctly built your prepared statement.

7. Click the new statement (this name matches the table name), move the pointer over *Operations*, and select *Test Run*.
8. Enter an appropriate value for the parameterized SQL, for example, '000000004'. This runs a request of select * from PMSVSAM where emp_id='000000004'.
9. Click *Test*.

The following result is returned:



10. Click OK.

Procedure How to Disconnect and Persist the Metadata for VSAM From iWay Application Explorer

The metadata is persisted to the iWay repository when you disconnect from VSAM.

To disconnect to iWay Data Server for VSAM, move your pointer over *Operations*, and select *Disconnect*.

The connection is released and the information is stored in the iWay repository.

Deploying and Running the Sample Servlet

The iWay Connector for JCA includes sample code that enables you to test iWay service and event adapters. This topic describes how to configure and deploy the iWay JCA Test Tool to the BEA WebLogic Server.

Procedure How to Deploy the Sample Servlet

To deploy the sample servlet:

1. Log on to the BEA WebLogic Server Console.

The default URL is

<http://localhost:7001/console>

The following window opens.



The screenshot shows the BEA WebLogic Server 8.1 Administration Console login page. At the top, there is a teal header bar with the text "Administration Console" and "BEA WebLogic Server 8.1". Below this, the main heading is "WebLogic Server Administration Console" in a large, bold, teal font. Underneath the heading, it says "Sign in to work with the WebLogic Server domain **jcawed**". There are two input fields: "Username:" with the text "weblogic" entered, and "Password:" which is empty. To the right of the password field is a "Sign In" button.

- a. Type a user name and password.
 - b. Click *Sign In*.
2. When the WebLogic Server console opens, select *Web Application Modules* from *Deployed Resources*.
3. Click *Deploy a New Web Application*.

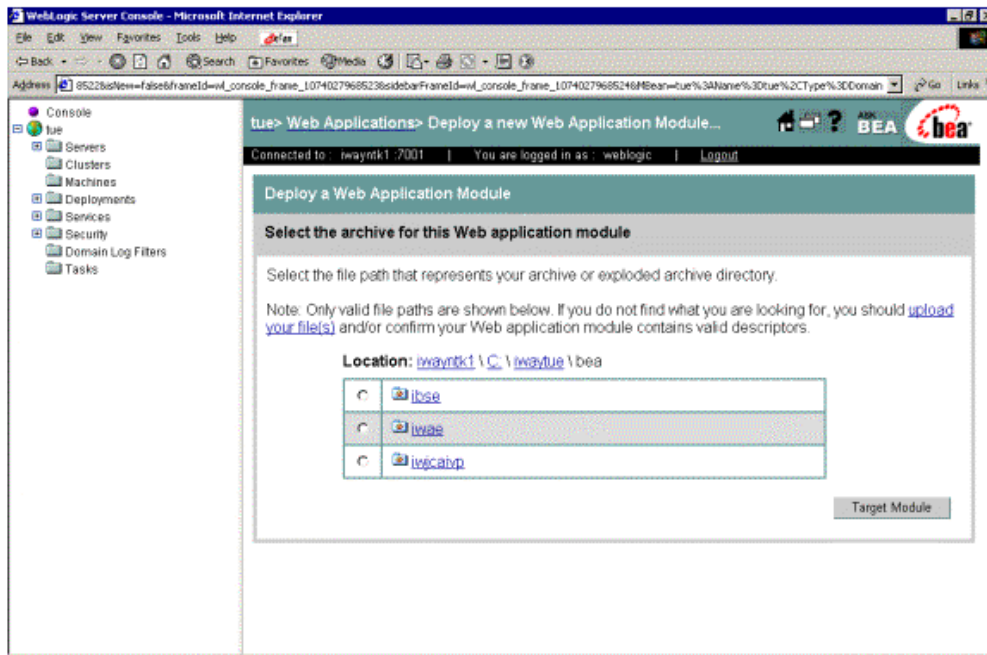
Use the location link to navigate to the location of the exploded WAR file. The WAR file is located in the *iWayHome*\BEA subdirectory.

where:

[*iWayHome*](#)

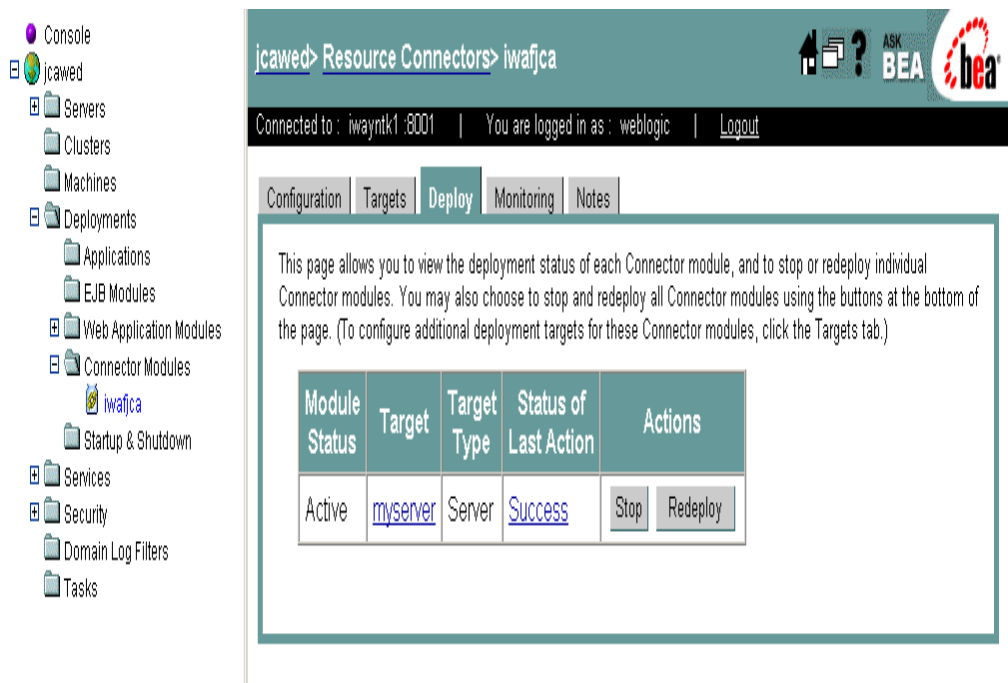
Is the root directory where the iWay product has been installed.

The following window opens.



4. Select *iwjcaivp*.
5. Click *Target Module*.

If deployment is successful, a window similar to the following opens:



Procedure How to Run the Sample Servlet

To run the sample servlet:

1. Open a browser and enter the following URL:

<http://host:port/iwjcaivp>

where:

[host](#)

Is the IP address or DNS name where the application server is installed.

[port](#)

Is the port on which the application server is listening. For a BEA WebLogic Server, the original default is 7001.

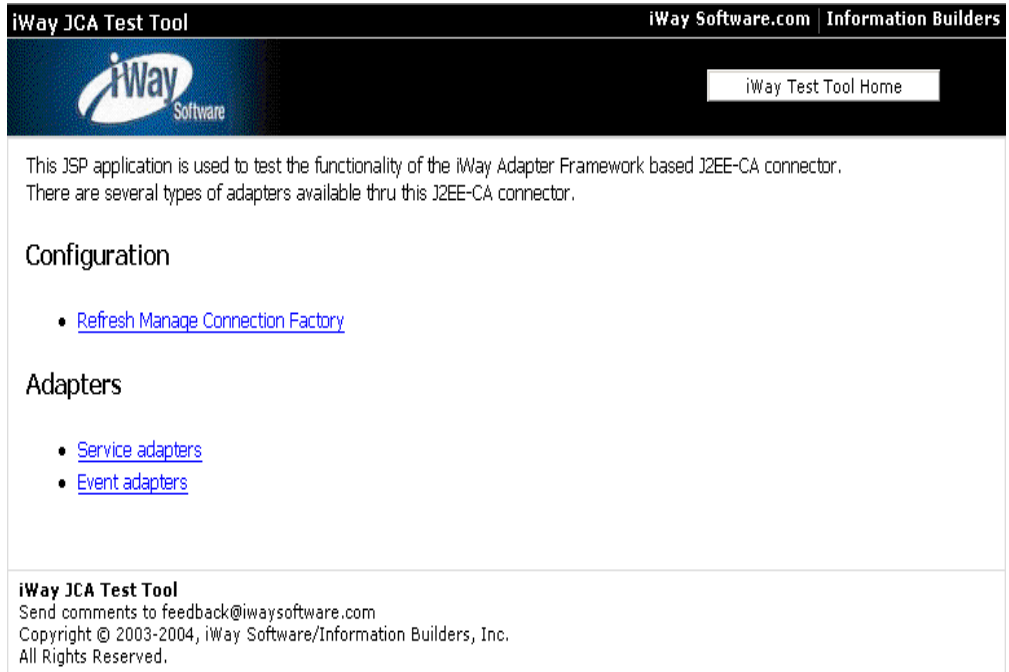
[iwjcaivp](#)

Is the context root to where the Web application is deployed.

For WebLogic, the default URL is

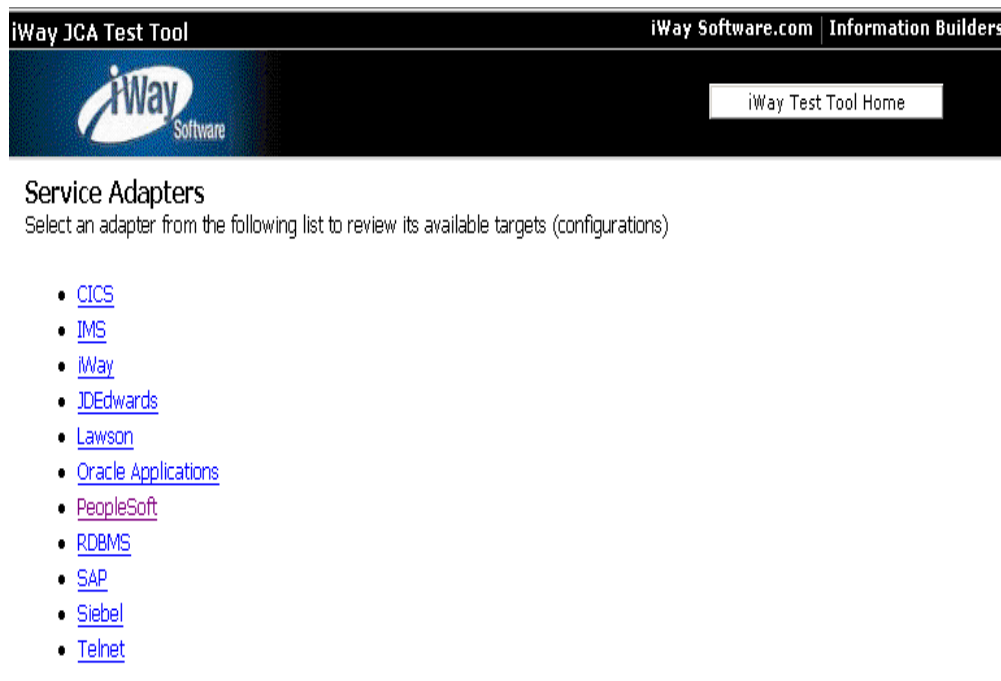
<http://localhost:7001/iwjcaivp>

The iWay JCA Test Tool window opens.



For information about the iWay JCA Test Tool, see Chapter 3, *iWay JCA Installation Verification Program*.

2. Click the *Service adapters* link to display the available adapters.



3. Click *RDBMS*.

If you have multiple target adapters defined, you must select VSAM1. If you only defined one target, the following window opens.

iWay JCA Test Tool iWay Software.com | Information Builders

Service Adapters
Select an adapter from the following list to review its available targets (configurations)

- [CICS](#)
- [IMS](#)
- [iWay](#)
- [RDBMS](#)
- [Telnet](#)

Targets for RDBMS

- [Oracle11x](#)
- [VSAM1](#)

Request for RDBMS target VSAM1
Enter the data for this interaction

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

4. Enter the following request to run a select all employees with an employee ID of '000000004' from the VSAM table:

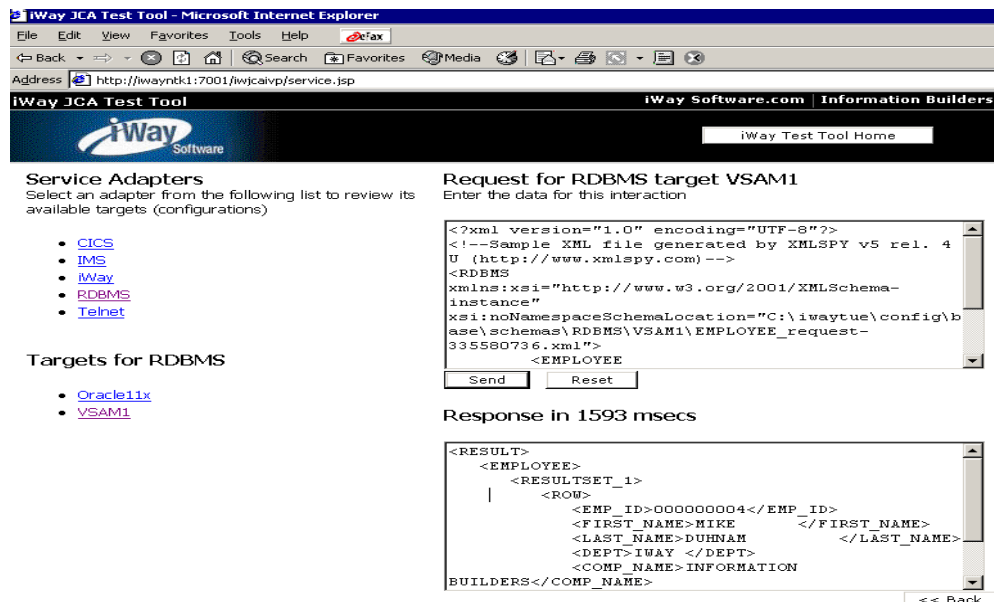
```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSPY v5 rel. 4 U (http://
www.xmlspy.com)-->
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\iwaytue\config\base\schemas\RDBMS\VS
AM1\EMPLOYEE_request-335580736.xml">
  <EMPLOYEE location="RDBMS/Statements/EMPLOYEE">
    <empid dataType="CHAR">000000004</CHAR>
  </EMPLOYEE>
</RDBMS>
```

The following results are returned

Issuing a VSAM Parameterized Query From BEA WebLogic Server

```
<RESULT>
  <EMPLOYEE>
    <RESULTSET_1>
      <ROW>
        <EMP_ID>000000004</EMP_ID>
        <FIRST_NAME>MIKE          </FIRST_NAME>
        <LAST_NAME>DUHNAM        </LAST_NAME>
        <DEPT>IWAY </DEPT>
        <COMP_NAME>INFORMATION BUILDERS</COMP_NAME>
      </ROW>
    </RESULTSET_1>
  </EMPLOYEE>
</RESULT>
```

The resulting window should look similar to the following:



APPENDIX A

Servlet Sample Code

Topic:

- iWay Servlet Sample Code

This section contains iWay servlet sample code.

iWay Servlet Sample Code

The following are examples of iWay servlet code.

Example **Error.jsp**

The following is an example of Error.jsp code.

```
<%@ page contentType="text/html" isErrorPage="true" %>

<%@ page import="java.util.*, java.io.*" %>
<%@ page import="com.ibi.adapters.*" %>
<%@ page import="javax.resource.*" %>

<%!
    // CallStack to String
    public static String getStackTrace(Throwable t) {

        if(t == null) {
            return "Callstack not available.";
        }

        String stackTrace = null;

        try {
            StringWriter sw = new StringWriter();
            PrintWriter pw = new PrintWriter(sw);
            t.printStackTrace(pw);
            pw.close();
            sw.close();
            stackTrace = sw.getBuffer().toString();
        }
        catch(Exception ex) {}
        return stackTrace;
    }
%>

<!-- ===== -->
<!--                HEADER                -->
<!-- ===== -->

<html>
    <head>
        <title>IWAF JCA Sample </title>
        <link href="default.css" rel="stylesheet" media="screen">
    </head>

    <body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0"
```

```

text="#000000">

<%@ include file="header.html" %>

<TABLE WIDTH="100%" BORDER="0" CELSPACING="0" CELLPADDING="0">
  <TR>

    <!-- -----Begin Side ----- --%>
    <TD WIDTH="130" VALIGN="top" ALIGN="RIGHT"/>
    <TD WIDTH="5%">&nbsp;   </TD>

    <!-- -----Begin Center ----- --%>
    <TD WIDTH="85%" VALIGN="top">
      <br>
      <H1 CLASS="orange">ERROR PAGE</H1>

      <span class="body">
        <%
          String message =
            (String)request.getAttribute("iway.message");
          if (message != null) {
            out.print(message);
          } else {
            %>
            Notify your administrator with the information below.
            <% } %>
          </span>

          <h2><%= new Date() %></h2>

          <%
            Exception ae = null; // AdapterException
            Throwable vt = null; // VendorException
            if (exception != null) {
              if (exception instanceof ResourceException) {
                ae = ((ResourceException)exception).getLinkedException();
                if (ae instanceof AdapterException) {
                  vt = ((AdapterException)ae).getVendorThrowable();
                }
              }
              if (ae != null) {
                out.println("<h2> AdapterException: " + ae.getMessage() +
"</h2>");
                out.println("<p>" + getStackTrace(ae) + "</p>");
                if (vt != null) {
                  out.println("<h2> VendorException: " +
vt.getMessage() + "</h2>");

```

```
                out.println("<p>" + getStackTrace(vt) + "</p>");
            }
        }
    }
    %>
    <h2>IWAFJCAException: <%= exception.getMessage() %></h2>
    <p><%= getStackTrace(exception) %></p>
    <% } %>

    </td>
</tr>
</table>

<!-- ===== -->
<!--          FOOTER          -->
<!-- ===== -->

    <%@ include file="footer.html" %>

    </body>
</html>
```

Example Event.jsp

The following is an example of Event.jsp code.

```
<%@ page error
Page="error.jsp" %>
<%@ page contentType="text/html" %>

<%@ include file="includes/include_util.jsp" %>
<%@ include file="includes/include_jca.jsp" %>
<%@ include file="includes/include_ae.jsp" %>

<%

////////////////////////////////////
ConnectionFactory cf = (ConnectionFactory)application.getAttribute("cf");
if (cf == null) {
    log(application, "service.jsp: Obtaining IWAF JCA Connection
factory,");
    cf = getConnectionFactory(application);
}
////////////////////////////////////

// Only do it once
String[] adapterNames = (String[])session.getAttribute("e.adapterNames");
if (adapterNames == null) {
    IDocument doc = aeCall(cf, "<GETADAPTERINFO/>");
    adapterNames = getEventAdapterNames(doc);
    session.setAttribute("e.adapterNames", adapterNames);
}

// Parsing request parameters
// Parsing request parameters
String adapter = request.getParameter("adapter");
if (adapter == null && adapterNames.length > 0) {
    adapter = adapterNames[0];
}

// Only do it if it is not a session
String[] channelNames =
(String[])session.getAttribute("event.channelNames");
String sessionAdapter = (String)session.getAttribute("event.adapter");
if (channelNames == null || (!adapter.equals(sessionAdapter))) {
    IDocument doc = aeCall(cf, "<GETCHANNELINFO><target>" + adapter + "</
target></GETCHANNELINFO>");
    channelNames = getChannelNames(doc);
    session.setAttribute("event.adapter", adapter);
    session.setAttribute("event.channelNamesNames", channelNames);
}
```

```

String channel = request.getParameter("channel");
if (channel == null && channelNames.length > 0) {
    channel = channelNames[0];
}

String action = request.getParameter("action");
if (channel != null && action != null) {
    if (action.equals("start")) {
        aeCall(cf, "<STARTCHANNEL><target>" + adapter + "</target><name>" +
channel + "</name></STARTCHANNEL>");
    } else {
        aeCall(cf, "<STOPCHANNEL><target>" + adapter + "</target><name>" +
channel + "</name></STOPCHANNEL>");
    }
}

IDocument channelDoc = null;
if (channel != null) {
    channelDoc = aeCall(cf, "<GETCHANNELSTATUS><target>" + adapter + "</
target><name>" + channel + "</name></GETCHANNELSTATUS>");
}

%>

<!-- ===== -->
<!--          HEADER          -->
<!-- ===== -->

<html>
  <head>
    <title>IWAF JCA Sample </title>
    <link href="default.css" rel="stylesheet" media="screen">
  </head>

  <body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0"
text="#000000">

    <%@ include file="header.html" %>

<!-- ===== -->
<!--          BODY          -->
<!-- ===== -->

    <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
      <TR>
        <TD WIDTH="5%" VALIGN="top" ALIGN="RIGHT" />

```

```

<TD WIDTH="30%" VALIGN="top">

    <!-- Adapters -->
    <br/>
    <h4>Event Adapters</h4>
    <p>Click the adapter below to see available channels
(configuration).</p>
    <ul>
        <%
            for (int i = 0; i < adapterNames.length; i++) {
        %>
        <li>
            <a href="event.jsp?adapter=<%= adapterNames[i] %>"
class="top_menu"><%= adapterNames[i] %></a>
        </li>
        <%
            }
            // When adapter not provided, pick the first one.
            if (adapter == null && adapterNames.length > 0) {
                adapter = adapterNames[0];
            }
        %>
    </ul>

    <!-- Channels for adapter -->
    <br/>
    <h4>Channels for <b><%= adapter %></b></h4>
    <%
        if (channelNames.length <= 0) {
            out.print("<p>No Channels configured for this
adapter.</p>");
        } else {
    %>
    <ul>
        <% for (int i = 0; i < channelNames.length; i++) { %>
        <li>
            <a href="event.jsp?adapter=<%= adapter %>&channel=<%=
channelNames[i] %>" class="top_menu"><%= channelNames[i] %></a>
            <a href="event.jsp?adapter=<%= adapter %>&channel=<%=
channelNames[i] %>&action=start" class="top_menu"> (start)</a>
            <a href="event.jsp?adapter=<%= adapter %>&channel=<%=
channelNames[i] %>&action=stop" class="top_menu"> (stop)</a>
        </li>
        <% } /* for */
        %>
    </ul>
    <% } /* else */ %>

```

```

        </TD>

        <% if (channel != null) { %>
            <TD WIDTH="65%" VALIGN="top">
                <br/>
                <h4>Adapter Status for <%= adapter+" channel "+channel
%></h4>

                <% if (channelDoc != null) { %>
                    <h4>Status</h4>
                    <table>
                        <%
                            INode statusNode =
channelDoc.getRootTree().getFirstChildNode();
                            String isactive =
statusNode.findChildByName("isactive").getValue();
                            out.println("<tr><td>Active:</td><td>");
                            out.println(isactive);
                            out.println("</td></tr><tr><td>Init. time:</
td><td>");

                            out.println(long2Date(statusNode.findChildByName("inittime").getValue()))
                            ;

                                if (isactive.equalsIgnoreCase("true")) {
                                    out.println("</td></tr><tr><td>Activate time:</
td><td>");

                                        String actTime =
statusNode.findChildByName("activatetime").getValue();
                                        out.println(long2Date(actTime));
                                        out.println("</td></tr><tr><td>Elapsed time:</
td><td>");

                                            out.println(elapseString(actTime));
                                            out.println("</td></tr><tr><td>Service count:</
td><td>");

                                                out.println(statusNode.findChildByName("servicecount").getValue());
                                                out.println("</td></tr><tr><td>Error count:</
td><td>");

                                                    out.println(statusNode.findChildByName("errorcount").getValue());
                                                    out.println("</td></tr><tr><td>Event count:</
td><td>");

                                                        out.println(statusNode.findChildByName("eventcount").getValue());
                                                        out.println("</td></tr><tr><td>Avg. service time
(msec):</td><td>");

```



```

out.println(statusNode.findChildByName("avgservicetime").getValue());
                                out.println("</td></tr><tr><td>Last service time
(msec):</td><td>");

out.println(statusNode.findChildByName("lastservicetime").getValue());
    }
    %>
    <% } %>
    </td></tr></table>
    <br/>
    </TD>
    <% } %>

    </TR>
    </table>

<!-- ===== -->
<!--          FOOTER          -->
<!-- ===== -->

    <%@ include file="footer.html" %>

    </body>
</html>

```

Example Include_ae.jsp

The following is an example of Include_ae.jsp code.

```
<%@ page import="java.util.*" %>

<%!

/*
 * Static helper methods for parsing IWAE messages
 */

// Based on GETADAPTERINFOResponse document
// @return array of available service adapters
private static String[] getAdapterNames(IDocument doc)
{
    List adapterNamesList = new ArrayList();

    INode rootNode      = doc.getRootTree();
    INode adapterNode = rootNode.findNodeByPath("/GETADAPTERINFOResponse/
adapter");

    while(adapterNode != null) {

        // Getting adapters
        adapterNode.snipNode(); // Not to mess up next find.
        INode targetNode = adapterNode.findChildByName("target");

        // Add the adapters with design descriptor to the list
        List descriptors = adapterNode.getAllChildren("descriptor");
        for (Iterator i = descriptors.iterator(); i.hasNext(); ) {
            INode descriptorNode = (INode)i.next();
            String attrFormat = descriptorNode.getAttribute("format");
            if (attrFormat != null && attrFormat.equals("design")) {
                adapterNamesList.add(targetNode.getValue());
                break;
            }
        } // for

        // Finding next adapter
        adapterNode = rootNode.findNodeByPath("/GETADAPTERINFOResponse/
adapter");
    }

    String[] names = new String[adapterNamesList.size()];
    names = (String[])adapterNamesList.toArray(names);
}
```

```

        return names;
    }

    // Based on GETADAPTERINFOResponse document
    // @return array of available service adapters
    private static String[] getEventAdapterNames(IDocument doc)
    {

        List adapterNamesList = new ArrayList();

        INode rootNode      = doc.getRootTree();
        INode adapterNode = rootNode.findNodeByPath("/GETADAPTERINFOResponse/
adapter");

        while(adapterNode != null) {

            // Getting adapters
            adapterNode.snipNode(); // Not to mess up next find.
            INode targetNode = adapterNode.findChildByName("target");

            // Add the adapters with design descriptor to the list
            List descriptors = adapterNode.getAllChildren("descriptor");
            for (Iterator i = descriptors.iterator(); i.hasNext(); ) {
                INode descriptorNode = (INode)i.next();
                String attrFormat = descriptorNode.getAttribute("format");
                if (attrFormat != null && attrFormat.equals("event")) {
                    adapterNamesList.add(targetNode.getValue());
                    break;
                }
            } // for

            // Finding next adapter
            adapterNode = rootNode.findNodeByPath("/GETADAPTERINFOResponse/
adapter");
        }

        String[] names = new String[adapterNamesList.size()];
        names = (String[])adapterNamesList.toArray(names);

        return names;
    }

    // Based on GETTARGETINFOResponse
    // @return array of available configurations for adapter
    private static String[] getTargetNames(IDocument doc)
    {

        List targetNamesList = new ArrayList();

```

```
    INode rootNode    = doc.getRootTree();
    INode targetNode = rootNode.findNodeByPath("/GETTARGETINFOResponse/
target");

    while(targetNode != null) {

        // Getting adapters
        targetNode.snipNode(); // Not to mess up next find.
        INode nameNode = targetNode.findChildByName("name");
        targetNamesList.add(nameNode.getValue());

        // Finding next adapter
        targetNode = rootNode.findNodeByPath("/GETTARGETINFOResponse/
target");
    }

    String[] names = new String[targetNamesList.size()];
    names = (String[])targetNamesList.toArray(names);

    return names;
}

// Based on GETTARGETINFOResponse
// @return array of available configurations for adapter
private static String[] getChannelNames(IDocument doc)
{

    List targetNamesList = new ArrayList();

    INode rootNode    = doc.getRootTree();
    INode targetNode = rootNode.findNodeByPath("/GETCHANNELINFOResponse/
channel");

    while(targetNode != null) {

        // Getting adapters
        targetNode.snipNode(); // Not to mess up next find.
        INode nameNode = targetNode.findChildByName("name");
        targetNamesList.add(nameNode.getValue());

        // Finding next adapter
        targetNode = rootNode.findNodeByPath("/GETCHANNELINFOResponse/
channel");
    }

    String[] names = new String[targetNamesList.size()];
    names = (String[])targetNamesList.toArray(names);
}
```

```
        return names;
    }

    %>
```

Example Include_jca.jsp

The following is an example of Include_jca.jsp code.

```
<%@ page import="com.ibi.common.*, com.ibi.afjca.cci.*,
com.ibi.afjca.spi.*" %>
<%@ page import="javax.resource.*,
javax.resource.cci.*, javax.resource.spi.*" %>
<%@ page import="javax.naming.*" %>

<%!

////////////////////////////////////
/
//JCA
////////////////////////////////////
/

// Obtain ConnectionFactory from JNDI
private ConnectionFactory getConnectionFactory(ServletContext
application)
    throws ResourceException, NamingException
{
    ConnectionFactory cf = null;

    String iwayHome      = application.getInitParameter("iway.home");
    String iwayConfig    = application.getInitParameter("iway.config");
    String iwayLogLevel  = application.getInitParameter("iway.loglevel");

    String iwayJndi      = application.getInitParameter("iway.jndi");

    application.log("iway.home      : " + iwayHome);
    application.log("iway.config    : " + iwayConfig);
    application.log("iway.loglevel: " + iwayLogLevel);
    application.log("iway.jndi     : " + iwayJndi);

    if (iwayJndi != null && iwayJndi.trim().length() > 0) {
        InitialContext context = new InitialContext();
        cf = (ConnectionFactory)context.lookup(iwayJndi);
        application.setAttribute("non-managed", Boolean.FALSE);
    } else {
        IWAManagedConnectionFactory mcf = new
IWAManagedConnectionFactory();
        mcf.setIWayHome(iwayHome);
        mcf.setIWayConfig(iwayConfig);
        mcf.setLogLevel(iwayLogLevel);
        cf = (ConnectionFactory)mcf.createConnectionFactory();
        application.setAttribute("non-managed", Boolean.TRUE);
    }
}
```

```

        application.setAttribute("cf", cf);
        return cf;
    }

    // Obtain Connection for design time
    private Connection getDesignTimeConnection(ConnectionFactory cf)
        throws ResourceException
    {
        // Create connectionSpec
        IWAConnectionSpec cs = new IWAConnectionSpec();
        cs.setAdapterName("IAEAdapter"); // Special Adapter

        return cf.getConnection(cs);
    }

    // Process design time message
    private IWARecord executeDesignInteraction(Connection c, Record r)
        throws ResourceException
    {
        // Create interaction
        Interaction i = c.createInteraction();

        // Create interactionSpec for DESIGNTIME
        IWAInteractionSpec is = new IWAInteractionSpec();
        is.setFunctionName("IWAE"); // AE Function, only available for
        IAEAdapter

        // Execute
        return (IWARecord)i.execute(is, r);
    }

    ///////////////////////////////////////////////////
    /

    private static Connection getRuntimeConnection(ConnectionFactory cf,
        String adapterName, String configuration)
        throws ResourceException
    {
        // Create connectionSpec
        // Aidong - Maybe we need a designTime connection spec.
        IWAConnectionSpec cs = new IWAConnectionSpec();
        cs.setAdapterName(adapterName);
        cs.setConfig(configuration);

        return cf.getConnection(cs);
    }

```

```
// Execute Service Call
private static Record executeRunInteraction(Connection c, Record r)
    throws ResourceException
{
    // Create interaction
    Interaction i = c.createInteraction();

    // Create interactionSpec for RUNTIME
    IWAFFInteractionSpec is = new IWAFFInteractionSpec();
    is.setFunctionName("PROCESS");

    // Execute
    return i.execute(is, r);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//

// Execute AE Call
private IDocument aeCall(ConnectionFactory cf, String message) throws
Exception {

    //log("Processing design time message: " + message);
    Connection c = getDesignTimeConnection(cf);

    IWAFFRecord rIn = new IWAFFRecord("input");
    rIn.setRootXML(message);
    IWAFFRecord resRec = executeDesignInteraction(c, rIn);

    c.close();
    return resRec.getIDocument();
}

// Execute Service Call
private String serviceCall(ConnectionFactory cf, String message, String
adapter, String config)
    throws Exception
{
    //log("Processing service message: " + message);
    Connection c = getRuntimeConnection(cf, adapter, config);

    /*
    IWAFFRecord rIn = new IWAFFRecord("input");
    rIn.setRootXML(message);
    IWAFFRecord docRec = executeRunInteraction(c, rIn);
    */
}
```



```
*/

IndexedRecord rIn =
cf.getRecordFactory().createIndexedRecord("input");
rIn.add(message);
IndexedRecord response = (IndexedRecord) executeRunInteraction(c,
rIn);

/*
// Using IWAFFRecord to get IDocument. Aidong is fixing this limitation
IWAFFRecord docRec = new IWAFFRecord();
docRec.setRootXML((String)response.get(0));
*/
c.close();
return (String)response.get(0);
}

%>
```

Example **Include_util.jsp**

The following is an example of Include_util.jsp code.

```
<%@ page import="java.io
.*, java.util.*" %>

<%!

// CallStack to String
public static String getStackTrace(Throwable t) {

    String stackTrace = null;

    try {
        StringWriter sw = new StringWriter();
        PrintWriter pw = new PrintWriter(sw);
        t.printStackTrace(pw);
        pw.close();
        sw.close();
        stackTrace = sw.getBuffer().toString();
    }
    catch(Exception ex) {}
    return stackTrace;
}

// Web Server logging
public static void log(ServletContext ctx, String msg) {
    ctx.log(msg);
}

// long string to date string
private static String long2Date(String millis) {
    Date date = new Date(Long.parseLong(millis));
    return date.toString();
}

// TODO: It got to be a better way of doing it.
private static String elapseString(String aMillis) {

    StringBuffer sb = new StringBuffer();

    // In seconds
    long runningTime = System.currentTimeMillis() -
Long.parseLong(aMillis);
    runningTime = runningTime / 1000;    // seconds

    long days, hours, minutes, seconds;
```

```
days = runningTime / (86400);
if (days > 0) {
    sb.append(days);
    sb.append(" days ");
    runningTime = runningTime - (days * 86400);
}

hours = runningTime / (3600);
if (hours > 0) {
    sb.append(hours);
    sb.append(" hours ");
    runningTime = runningTime - (hours * 3600);
}

minutes = runningTime / 60;
if (minutes > 0) {
    sb.append(minutes);
    sb.append(" min(s) and ");
    runningTime = runningTime - (minutes * 60);
}

seconds = runningTime;
sb.append(seconds);
sb.append(" sec(s)");

return sb.toString();
}

%>
```

Example Index.jsp

The following is an example of Index.jsp code.

```
<%@ page content
Type="text/html" %>

<!-- ===== -->
<!--          HEADER          -->
<!-- ===== -->

<html>
  <head>
    <title>IWAJ JCA Sample </title>
    <link href="default.css" rel="stylesheet" media="screen">
  </head>

  <body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0"
text="#000000">

    <%@ include file="header.html" %>

    <!-- ===== -->
    <!--          BODY          -->
    <!-- ===== -->

    <TABLE WIDTH="100%" BORDER="0" CELSPACING="0" CELLPADDING="0">
      <TR>
        <TD WIDTH="130" VALIGN="top" ALIGN="RIGHT" />

        <TD WIDTH="85%" VALIGN="top">

<!--
          <br/>
          <h4>Remote management</h4>
          <p>This allows the remote management of the JCA Adapter
via
          IWAE using a SOAP listener.
          <br><b>IMPORTANT: Security is currently disabled</b>
          </p>
          <ul>
            <li><a href="#" class="top_menu">Configure</a></li>
            <li><a href="#" class="top_menu">Monitor</a></li>
            <li><a href="#" class="top_menu">Start/Stop</a></li>
          </ul>
-->
          <br/>
          <h4>Testing tools</b></h4>
          <ul>
```

```

        <li><a href="service.jsp" class="top_menu">Service
adapters</a></li>
        <li><a href="event.jsp" class="top_menu">Event
adapters</a></li>
        </ul>
    </TD>

</TR>
</table>

<!-- ===== -->
<!-- FOOTER -->
<!-- ===== -->

    <%@ include file="footer.html" %>

</body>
</html>

```

Example Service.jsp

The following is an example of Service.jsp code.

```
<%@ page errorPage= "error.jsp" %>

<%@ page contentType="text/html" %>

<%@ include file="includes/include_util.jsp" %>
<%@ include file="includes/include_jca.jsp" %>
<%@ include file="includes/include_ae.jsp" %>

<%

////////////////////////////////////
ConnectionFactory cf = (ConnectionFactory)application.getAttribute("cf");
if (cf == null) {
    log(application, "service.jsp: Obtaining IWAF JCA Connection
factory,");
    cf = getConnectionFactory(application);
}
////////////////////////////////////

// Only do it once
String[] adapterNames = (String[])session.getAttribute("e.adapterNames");
if (adapterNames == null) {
    IDocument doc = aeCall(cf, "<GETADAPTERINFO/>");
    adapterNames = getAdapterNames(doc);
    session.setAttribute("e.adapterNames", adapterNames);
}

// Parsing request parameters
String adapter = request.getParameter("adapter");
if (adapter == null && adapterNames.length > 0) {
    adapter = adapterNames[0];
}

String target = request.getParameter("target");

// Only do it if it is not a session
String[] targetNames =
(String[])session.getAttribute("service.targetNames");
String sessionAdapter = (String)session.getAttribute("service.adapter");
if (targetNames == null || (!adapter.equals(sessionAdapter))) {
    IDocument doc = aeCall(cf, "<GETTARGETINFO><target>" + adapter + "</
target></GETTARGETINFO>");
    targetNames = getTargetNames(doc);
    session.setAttribute("service.adapter", adapter);
    session.setAttribute("service.targetNames", targetNames);
}
```

```

    }

    //////////////////////////////////////
    // Service call
    //////////////////////////////////////
    String input      = request.getParameter("input");
    String output     = null;
    long   elapsedTime = 0;

    if (input != null) {
        long stime = System.currentTimeMillis();
        output = serviceCall(cf, input, adapter, target);
        elapsedTime = System.currentTimeMillis() - stime;
    }

    %>

<!-- ===== -->
<!--          HEADER          -->
<!-- ===== -->

<html>
  <head>
    <title>IWAF JCA Sample </title>
    <link href="default.css" rel="stylesheet" media="screen">
  </head>

  <body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0"
text="#000000">

    <%@ include file="header.html" %>

<!-- ===== -->
<!--          BODY          -->
<!-- ===== -->

    <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
      <TR>
        <TD WIDTH="5%" VALIGN="top" ALIGN="RIGHT" />

        <TD WIDTH="30%" VALIGN="top">

          <!-- Adapters -->
          <br/>
          <h4>Service Adapters</h4>
          <p>Click the adapter below to see available targets
(configuration).</p>
          <ul>

```

```

        <%
            for (int i = 0; i < adapterNames.length; i++) {
        %>
        <li>
            <a href="service.jsp?adapter=<%= adapterNames[i] %>"
class="top_menu"><%= adapterNames[i] %></a>
        </li>
        <%
            }
        %>
    </ul>

    <!-- Targets for adapter -->
    <br/>
    <h4>Targets for <b><%= adapter %></b></h4>
    <%
        if (targetNames.length <= 0) {
            out.print("<p>No targets configured for this
adapter.</p>");
        } else {
    %>
    <ul>
        <% for (int i = 0; i < targetNames.length; i++) { %>
            <li>
                <a href="service.jsp?adapter=<%= adapter
%>&target=<%= targetNames[i] %>" class="top_menu"><%= targetNames[i]
%></a>

                </li>
            <% } /* for */

            // When target not provided, pick the first one.
            if (target == null && targetNames.length > 0) {
                target = targetNames[0];
            }
        %>
    </ul>
    <% } /* else */ %>
</TD>

<% if (target != null) { %>
    <TD WIDTH="65%" VALIGN="top">
        <br/>
        <h4>Request for <%= adapter+" target "+target %></h4>
        <FORM ACTION="service.jsp" METHOD="POST">
            <input type='hidden' name='adapter' value='<%=
adapter %>' />

            <input type='hidden' name='target' value='<%= target
%>' />

```



```

        <br/><textarea name="input" rows="10" cols="50"><%=
(input==null?"":input) %></textarea>
        <br/><input type="submit" value="Send"/><input
type="reset"/>
        <% if (output != null) { %>
        <h4>Response in <%= elapsedTime %> msecs</h4>
        <textarea name="output" rows="10" cols="50"
disable="true"><%= output %></textarea>
        <% } %>
        </FORM>
        <br/>
        </TD>
    <% } %>

</TR>
</table>

<!-- ===== -->
<!-- FOOTER -->
<!-- ===== -->

    <%@ include file="footer.html" %>

</body>
</html>

```

Example IWAFJCATestTool.java

The following is an example of IWAFJCATestTool.java code.

```

import javax.resource.*;
import javax.resource.cci.*;
import java.io.FileInputStream;
import java.io.IOException;

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

```

```
/**
 * The purpose of this sample is to illustrate how to use the IWAF
Universal
 * JCA Adapter and its enhancements to the JCA 1.0 specification.
 *
 * All the enhancements are done via JCA standard interfaces,
ConnectionSpec
 * and InteractionSpec. In addition, we've created our own Record
 * implementation to support XML, BLOB and CLOB.
 *
 * The designtime capabilities are done via our XML AE message interface.
Below * is a list of supported messages.
 *
 * - GETADAPTERINFO - List of available adapters
 *
 * - GSETTARGETINFO - List of available
configuration for an adapter
 * - SETTARGETINFO - Maintain target (adapter)
configuration
 * - GETTARGET - Connect to an
adapter configuration
 * - GETCOMPONENT - Browse components
 * - GETSCHEMA - Obtain component
schemas
 * - ADDPORT - Create port
definition with component schema
 * - GETINTERACTION - Obtain component
interaction
 * - RUNTINTERACTION - Run component interaction
 * - RELEASETARGET - Disconnect from an adapter
configuration
 *
 * - GETDISPOSITIONINFO - List of available port
disposition
 * - GETPORTINFO - List of available
port for an adapter
 * - SETPORTINFO - Maintain port
information
 * - REMOVEPORT - Remove port
 * - GETCHANNELINFO - List of available channels
for an adapter
 * - SETCHANNELINFO - Maintain channel
configuration
 * - STARTCHANNEL - Start channel
configuration
 * - STOPCHANNEL - Stop channel
configuration
 * - GETCHANNELSTATUS - Obtain channel configuration status
```

```

*
* Inbound (Event) support is done with our IWAFFInboundInteractionSpec
and,
* optionally, with our MessageListener interface.
*
* IMPORTANT!!! THIS INTERFACE IS A COPY OF JCA 1.5 INTERFACE AND WE MAY
NOT
* MAKE IT AVAILABLE IN OUR JCA 1.0 IMPLEMENTATION.
*
* Author: Marcelo Borges
* Date: July, 2003
*/
public class IWAFFJCATestTool {

    private static String HOME =
System.getProperty("iway.home");
    private static String CONFIG = System.getProperty("iway.config");
    private static String LOG_LEVEL = "FATAL";

    private static String MSG_DESIGN;

    private static String ADAPTER;
    private static String TARGET;
    private static String MSG_RUN;

    private static String CHANNEL;
    private static int CCI;

    //////////////////////////////////////
    ////

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

        processArgs(args);

        // TODO: Aidong, our IWAFFManagedConnectionFactory must check and
        //         proper exception (at least runtime exception)
        if(HOME == null) {
            throw new IllegalArgumentException("-home MUST be set");
        }
        // Aidong, likewise
        if(CONFIG == null) {
            throw new IllegalArgumentException("-config MUST be set");
        }
    }
}

```

```

////////////////////////////////////
//////////
// JCA WORK
////////////////////////////////////
//////////

    log("Running iWay JCA Test Tool (For syntax use '-help'
argument)");
    log("Using home '" + HOME + "'");
    log("Using config '" + CONFIG + "'");

    ConnectionFactory cf = getConnectionFactory();

    try {

        //////////////////////////////////////
        //////////
        // Process Design time message
        //////////////////////////////////////
        //////////

        // Design time will be processed if the -ae file argument was
passed.
        if (MSG_DESIGN != null) {
            log("Processing designtime message.");
            Connection c = getDesigntimeConnection(cf);

            // We are using our own JCA Record implementation. This record
is a
            // wrapper to our IDocument interface and allows us to
represent
            // XML, BLOB and CLOB.

            // Aidong - Do we use the record name?
            IWAFRecord rIn = new IWAFRecord("input");
            rIn.setRootXML(MSG_DESIGN);

            IWAFRecord response = executeDesignInteraction(c, rIn);
            log("***** D E S I G N    T I M E    I N P U T *****
");

            log(MSG_DESIGN);
            log("***** D E S I G N    T I M E    O U T P U T *****
");

            log(response.getRootXML());
        }

        //////////////////////////////////////

```

```

//////////
// Process Run time message
////////////////////////////////////
//////////

// Design time will be processed if the -ae file argument was passed.
if (ADAPTER != null && TARGET != null && MSG_RUN != null) {
    log("Processing runtime message for adapter '" + ADAPTER + "'
target '" + TARGET + "'.");
    Connection c = getRuntimeConnection(cf, ADAPTER, TARGET);

    // Aidong - We should also support IWAF for RUNTIME.
    //IWAFRecord rIn = new IWAFRecord("input");
    //rIn.setRootXML(MSG_RUN);
    //IWAFRecord response = executeRunInteraction(c, rIn);

    // Use JCA IndexRecord, named "input" for runtime processing.
    IndexedRecord rIn =
cf.getRecordFactory().createIndexedRecord("input");
    rIn.add(MSG_RUN);

    IndexedRecord response = (IndexedRecord)
executeRunInteraction(c, rIn);

    log("***** R U N    T I M E    I N P U T ***** ");
    log(MSG_RUN);
    log("***** R U N    T I M E    O U T P U T ***** ");

    // The response document will be the first element in the
returned
    // indexed record
    log((String)response.get(0));
}

////////////////////////////////////
//////////
// Event support
////////////////////////////////////
//////////

// Design time will be processed if the -ae file argument was
passed.
if (ADAPTER != null && CHANNEL != null) {
    log("Initializing channel for adapter '" + ADAPTER + "'
channel name '" + CHANNEL + "'.");
    Connection c = getInboundConnection(cf, ADAPTER, CHANNEL);

    // Create CCI based event message listener. The Listener class

```

```
is
    // declare as an Inner class in this source. When using this
    // feature our JCA will act as a PORT and handle the event
    // broadcasting.
    if (CCI > 0) {
        for (int i=0; i < CCI; i++) {
            Listener l = new Listener("Listener #" + i);
            ((IWAFFInboundConnection)c).subscribe(l);
            log("CCI Message listener #" + i + " subscribed to
channel");
        }
    }

    log("Starting... ");
    executeInboundInteraction(c, "START");

    // Wait until 'Q' typed in to terminate the program.
    log("Started. Type 'Q' to stop the channel.");
    int typeIn = 'A';
    for(;typeIn != 'Q';) {
        typeIn = System.in.read();
    }

    log("Stopping...");
    executeInboundInteraction(c, "STOP");
    log("Stopped");
}

} finally {
    // Required because of our Event threading capabilities
    // If not called, the application will never end.
    // TODO: Aidong, Maybe we can only start our threading when
the
    //          event capability is used.
    ((IWAFFConnectionFactory)cf).destroy();
    log("***** D O N E ***** ");
}

}

////////////////////////////////////
//////
// STEP 1 - OBTAIN CONNECTION FACTORY
////////////////////////////////////
//////

/*
```

```

    * This is an unmanage environment example where we create the
    * Manage connection factory ourselves. In a manage (J2EE)
environment
    * the AS will created based on the RAR XML information. The
    * ManagedConnectionFactory will then be available via JNDI
    *
    * Our ManageConnectionFactory has 3 parameters:
    *
    * - IWayHome          - Points to iWay instalation directory. The
Resource
    *                                adapter use this
information to find the 'lib' directory
    *                                to load adapters
dynamicaly.
    * - IWayConfig      - Configuration name. With IWayHome, it will find a
    *                                configuration
directory for logging and adapter
    *                                instance (target)
configuration.
    * - LogLevel          - IWAF JCA system LogLevel. It can take the
following:
    *                                (DEBUG < INFO < WARN
< ERROR < FATAL).
    *
    */
    private static ConnectionFactory getConnectionFactory()
        throws ResourceException
    {
        IWAFManagedConnectionFactory mcf = new
IWAFManagedConnectionFactory();
        mcf.setIWayHome(HOME);
        mcf.setIWayConfig(CONFIG);
        mcf.setLogLevel(LOG_LEVEL);

        return (ConnectionFactory)mcf.createConnectionFactory();
    }

    ///////////////////////////////////////////////////
    // STEP 2 - OBTAIN CONNECTION FOR AN ADAPTER
    ///////////////////////////////////////////////////

    /* IWAFConnectionSpec is a implementation of ConnectionSpec used for
    * creating a designtime or runtime service adapter connection.
    *
    * Our ConnectionSpec has 7 parameters:
    *

```

```
* - adapterName - Name of the adapter. There is a special adapter
*               called "IAEAdapter". This adapter is used for
*               design time.
* - config - Adapter configuration name. NOT
REQUIRED FOR IWAAdapter.
* - language - Default is en
* - country - Default is us
*
* - userName - user name. If provided, it
overwrites configuration.
* - password - password. If provided, it
overwrites configuration.
*
* - logLevel - logLevel. It overwrites the level set by
the
*               ManagedConnectionFactory property.
*
* The connection pooling is fully supported and done based on these
* properties, excluding logLevel.
*
* Currently our JCA adapter support only basic security mapping. The
DEBUG
* log will provide detail information of the mapping behaviour. Here
is how
* it works.
*
* 1 If userName and password was NOT set, and no security was
provided by
* the application server, the JCA will still let it pass and rely
on the
* adapter configuration security information.
* 2 If userName and password was set, these values will overwrite
the adapter
* configuration. JCA will compare with the security provided by
the AS,
* and log in case the values didn't match, but still let it go
through.
*
* IWAAdapterInboundConnectionSpec provides access to an event adapter
channel. Its
* properties are:
*
* - adapterName
* - logLevel
* - language
* - country
* - channel - Name of the adapter channel
configuration
```



```

        * - disposition - "JCA" for JCA 1.5 type of event, otherwise all
done via
        *
        *           our channel port configuration.
        */

private static Connection getDesigntimeConnection(ConnectionFactory cf)
    throws ResourceException
{
    // Create connectionSpec
    // Aidong - Maybe we need a designtime connection spec.
    IWAFFConnectionSpec cs = new IWAFFConnectionSpec();
    cs.setAdapterName("IAEAdapter"); // Special Adapter

    return cf.getConnection(cs);
}

private static Connection getRuntimeConnection(ConnectionFactory cf,
    String adapterName, String configuration)
    throws ResourceException
{
    // Create connectionSpec
    // Aidong - Maybe we need a designtime connection spec.
    IWAFFConnectionSpec cs = new IWAFFConnectionSpec();
    cs.setAdapterName(adapterName);
    cs.setConfig(configuration);

    return cf.getConnection(cs);
}

private static Connection getInboundConnection(ConnectionFactory cf,
    String adapterName, String channelName)
    throws ResourceException
{
    // Create InboundConnectionSpec
    IWAFFInboundConnectionSpec ics = new IWAFFInboundConnectionSpec();
    ics.setAdapterName(adapterName);
    ics.setChannel(channelName);

    // disposition as "JCA", means CCI application wants to get
    // message delivered. When set, our JCA will register itself
    // as a port to the channel, and pass the events along to its
    // registers MessageListeners.
    if (CCI > 0) {
        ics.setDisposition("JCA");
    }
}

```

```

        // IWAFFInboundConnection object to work with
        return cf.getConnection(ics);
    }

    ///////////////////////////////////////////////////////////////////
    // STEP 3A - EXECUTE DESIGNTIME INTERACTION
    ///////////////////////////////////////////////////////////////////

    /* There are 2 InteractionSpecs available.
    *
    * - IWAFFInteractionSpec: Used for both designtime and runtime. The
    *   designtime is executed via the IWAFF functionName and the runtime
    *   via "PROCESS". The "IWAFF" functionName can only be used by the
    *   special adapter IAEAdapter. The "PROCESS" can be used by any
    adapter.
    *
    * - IWAFFInboundInteraction: Used to start and stop the channel. Done
    via
    *   2 defined functionNames: "START" and "STOP"
    *
    * The "IWAFF" functionName of IWAFFInteractionSpec will required the use
    * of IWAFFRecord with the IWAFFInteraction. "PROCESS" uses standard
    * IndexedRecord.
    */
    // @return ADAPTER, CONFIGURATION ConnectionSpec
    private static IWAFFRecord executeDesignInteraction(Connection c, Record
    r)
        throws ResourceException
    {
        // Create interaction
        Interaction i = c.createInteraction();

        // Create interactionSpec for DESIGNTIME
        IWAFFInteractionSpec is = new IWAFFInteractionSpec();
        is.setFunctionName("IWAFF"); // AE Function, only available for
    IAEAdapter

        // Execute
        return (IWAFFRecord)i.execute(is, r);
    }

```

```

    // @return ADAPTER, CONFIGURATION ConnectionSpec
    private static Record executeRunInteraction(Connection c, Record r)
        throws ResourceException
    {
        // Create interaction
        Interaction i = c.createInteraction();

        // Create interactionSpec for RUNTIME
        IWAFFInteractionSpec is = new IWAFFInteractionSpec();
        is.setFunctionName("PROCESS");

        // Execute
        return i.execute(is, r);
    }

    // @return ADAPTER, CONFIGURATION ConnectionSpec
    private static void executeInboundInteraction(Connection c, String
function)
        throws ResourceException
    {
        // Get Interaction
        Interaction i = c.createInteraction();

        // Create InboundInteractionSpec
        IWAFFInboundInteractionSpec is = new IWAFFInboundInteractionSpec();
        is.setFunctionName(function);

        i.execute(is, null);
    }

    ///////////////////////////////////////////////////////////////////
    // STEP 4 - JCA 1.1 like 5 feature. Event can be handled by CCI
    application
    ///////////////////////////////////////////////////////////////////

```

```
/**
 * Inner class: listener
 *
 * Any listener class has to implement
com.ibi.afjca.cci.MessageListener
 * interface in order to hook up IWAF correctly.
 */
public static class Listener implements MessageListener {
    private final String whoAmI;

    private Listener (String who) {
        whoAmI = who;
    }

    // Interfac method called by the Channel
    public Record onMessage(Record record) {
        if(record instanceof IWAFRecord) {
            IWAFRecord iwafRec = (IWAFRecord)record;
            if (iwafRec.getRootXML() != null) {
                log(whoAmI + " got XML message:
            } else {
                log(whoAmI + " got NON XML message.");
            }
        } else {
            log(whoAmI + " got an unknown Record: " + record);
        }
        // Aidong: We need to handle the reply.
        return null;
    }
}
```

```

////////////////////////////////////
////
// Syntax and parameter parsing
////////////////////////////////////
////

private static void printSyntax() {
    System.out.println("-----");
    System.out.println(" Syntax: IWAFJCATestTool [args]");
    System.out.println("-----");

    System.out.println(" -help          : prints this message.");
    System.out.println("");
    System.out.println("*** Required Configuration");
    System.out.println(" -home      dir      : iWay installation
directory");
    System.out.println(" -config   name      : iWay configuration
name");
    System.out.println(" -loglevel level    :
[DEBUG|INFO|WARN|ERROR|FATAL]. Default is ERROR.");
    System.out.println("");
    System.out.println("*** Designtime capability");
    System.out.println(" -ae        file     : File with input message
for design time");
    System.out.println("");
    System.out.println("*** Runtime capability");
    System.out.println(" -adapter  name      : Adapter name");
    System.out.println(" -target   name      : Adapter configuration
name (Service)");
    System.out.println(" -input    file      : File with input message
for runtime");
    System.out.println(" -channel  name      : Adapter channel name
(Event)");
    System.out.println(" -cci      #         : Number of CCI message
listeners(Event)");
    System.out.println("");
    System.out.println("Examples: ");
    System.out.println("");
    System.out.println("IWAFJCATestTool -home c:\iway55 -config test
-ae c:\adapterinfo.xml");
    System.out.println(" - Executes the adapterInfo.xml AE
message.");
    System.out.println("");
    System.out.println("IWAFJCATestTool -home c:\iway55 -config test
-adapter IMS -config ims7tcp -input c:\test1.xml");
    System.out.println(" - Executes the test1.xml message against
ims7tcp IMS.");
}

```

```

        System.out.println("");
        System.out.println("IWAFCATestTool -home c:\\iway55 -config test
-adapter IMS -channel filein");
        System.out.println(" - Brings up the channel configuration
filein from IMS adapter.");
        System.out.println("-----
-----");
        System.exit(0);
    }

    private static void processArgs(String[] args) throws Exception {
        for (int i=0; i < args.length; i++) {
            try {
                if (args[i].equals("-help")) {
                    printSyntax();
                } else if (args[i].equals("-home")) {
                    i++;
                    HOME = args[i];
                } else if (args[i].equals("-config")) {
                    i++;
                    CONFIG = args[i];
                } else if (args[i].equals("-loglevel")) {
                    i++;
                    LOG_LEVEL = args[i];
                } else if (args[i].equals("-ae")) {
                    i++;
                    MSG_DESIGN = readFile(args[i]);
                } else if (args[i].equals("-input")) {
                    i++;
                    MSG_RUN = readFile(args[i]);
                } else if (args[i].equals("-adapter")) {
                    i++;
                    ADAPTER = args[i];
                } else if (args[i].equals("-target")) {
                    i++;
                    TARGET = args[i];
                } else if (args[i].equals("-channel")) {
                    i++;
                    CHANNEL = args[i];
                } else if (args[i].equals("-cci")) {
                    i++;
                    CCI = Integer.valueOf(args[i]).intValue();
                }
            } catch (IndexOutOfBoundsException iobe) {
                System.err.println("Problem parsing argument '" + args[i-1] + "'. Check your command line.");
                printSyntax();
            } catch (NumberFormatException nfe) {

```

```

        System.err.println("Argument -cci receive '" + args[i] +
        "'. It must be a number.");
    }
    } // for
}

////////////////////////////////////
/////
// Helper methods
////////////////////////////////////
/////

private static void log(Object msg) {
    log(msg, null);
}
private static void log(Object msg, Throwable t) {
    System.out.println(msg);
    if (t != null) {
        t.printStackTrace();
    }
}

/**
 * Helper method to read file into String. Assumes the file is in the
 * machines codepage.
 */
private static String readFile(String requestFile) throws IOException {
    FileInputStream fileIn = new FileInputStream(requestFile);
    byte[] inBytes = new byte[fileIn.available()];

    int offset = 0, counter = 0, len = inBytes.length;
    do {
        counter = fileIn.read(inBytes, offset, len);
        offset += counter;
    } while(counter != -1 && offset < len);

    return new String(inBytes);
}

}

```

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