

iWay

iWay XML Adapter for RDBMS for BEA WebLogic
User's Guide
Version 5 Release 5



8.1 VALIDATED

BEA WEBLOGIC PLATFORM

February 11, 2005

DN3501320.0205

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 © 2005, by Information Builders, Inc and iWay Software. All rights reserved. Patent Pending. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Preface

This document is intended for system integrators who develop client-server interfaces between RDBMS and third-party enterprise information system (EIS) applications.

How This Manual Is Organized

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

Chapter/Appendix		Contents
1	Introducing the iWay XML Adapter for RDBMS	Provides an overview of the adapter and how it works.
2	Creating XML Schemas or Business Services	Describes how to create schemas for RDBMS SQL statements and stored procedures for Web services or for JCA deployment.
3	Listening for Database Events	Describes how to configure a listener to listen to a database event.
4	Using Web Services Policy-Based Security	Describes how to configure Web services policy-based security.
5	Management and Monitoring	Describes how to configure and use monitoring tools provided by iBSE and JCA to gauge the performance of your run-time environment.
A	Using Application Explorer in BEA WebLogic Workshop to Create XML Schemas and Web Services	Describes how to use iWay Java Swing Application Explorer running in BEA WebLogic Workshop to create XML schemas and Web services.
B	Using Application Explorer in BEA WebLogic Workshop for Event Handling	Describes how to use the iWay XML Adapter for RDBMS to listen for events in a relational table.

C	Using WebLogic Workshop to Access VSAM	Describes how to produce and access a Web service generated from a VSAM database.
D	JDBC Drivers	Lists the supported JDBC drivers for use with the adapter.

Documentation Conventions

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

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

Related Publications

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

Customer Support

Do you have questions about the iWay XML Adapter for RDBMS?

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

If you bought the product directly from iWay Software, call Information Builders Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your iWay XML Adapter for RDBMS questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code 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.informationbuilders.com>. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of www.informationbuilders.com also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

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

Help Us to Serve You Better

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

The following tables list the specifications our consultants require.

Platform	
Operating System	
OS Version	
Product List	
Adapters	

Platform	
Adapter Deployment	For example, JCA, Business Services Engine, iWay Adapter Manager
Container Version	

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

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

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

Application Explorer Type	Version	Platform
Swing		
Servlet		
ASP		

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

Version	Vendor

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

Request/Question	Error/Problem Details or Information
Provide usage scenarios or summarize the application that produces the problem.	
Did this happen previously?	

Request/Question	Error/Problem Details or Information
Can you reproduce this problem consistently?	
Any change in the application environment : software configuration, EIS/ database configuration, application, and so forth?	
Under what circumstance does the problem <i>not</i> occur?	
Describe the steps to reproduce the problem.	
Describe the problem .	
Specify the error message(s).	

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

XML schema
XML instances
Other input documents (transformation)
Error screen shots
Error output files
Trace and log files
Log transaction

User Feedback

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

Thank you, in advance, for your comments.

iWay Software Training and Professional Services

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

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

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

Contents

1. Introducing the iWay XML Adapter for RDBMS	1-1
Introduction	1-2
Using the Adapter With Relational Databases	1-2
Using the Adapter to Send a Request	1-3
Using the Adapter Listener	1-3
Using the Adapter to Access Non-relational Databases	1-4
Accessing Stored Procedures for Non-relational Data Sources	1-7
Deployment Information for the Adapter	1-7
Deployment Information Roadmap	1-8
The Integration Business Services Engine (IBSE)	1-9
The iWay Enterprise Connector for J2EE Connector Architecture (JCA)	1-9
2. Creating XML Schemas or Business Services	2-1
Generating Schemas and a Business Service	2-2
Starting Servlet Application Explorer	2-3
Creating and Managing a Connection	2-4
Disconnecting From a Defined Target	2-9
Editing a Defined Target	2-9
Deleting a Defined Target	2-10
Viewing Metadata	2-11
Creating a Statement and Generating Schemas	2-14
Creating and Testing a Parameterized SQL Statement	2-16
Creating a Batch Statement	2-20
Generating a Schema for a Prepared Statement	2-21
Generating a Schema for a Stored Procedure	2-22
Request and Response Documents	2-23
Regular SQL Statements	2-24
Parameterized SQL Statements	2-27
Stored Procedure Schemas for an Oracle Database	2-30
Stored Procedure Schemas for a VSAM Database	2-31
Multiple Executions of the Same SQL Statement or Stored Procedure	2-34
Understanding Integration Business Services	2-36
Creating a Business Service	2-36
Testing a Business Service	2-38
Generating WSDL From a Web Service	2-41
Identity Propagation	2-44
3. Listening for Database Events	3-1
Understanding iWay Event Functionality	3-2

Creating, Editing, or Deleting an Event Port	3-2
Creating an Event Port From the Event Adapters Tab	3-2
Editing and Deleting an Event Port	3-16
Creating, Editing, or Deleting an Event Channel	3-17
Creating a Channel	3-17
Editing and Deleting a Channel	3-25
Choosing a Listening Technique	3-25
Standard Event Processing With Row Tracking	3-27
Standard Event Processing With Row Removal	3-33
Trigger-based Event Processing	3-36
4. Using Web Services Policy-Based Security	4-1
Integration Business Services Policy-Based Security	4-2
Configuring Integration Business Services Policy-Based Security	4-3
5. Management and Monitoring	5-1
Managing and Monitoring Services and Events Using iBSE	5-2
Managing and Monitoring Services and Events Using the JCA Test Tool	5-16
Setting Engine Log Levels	5-21
Configuring Connection Pool Sizes	5-22
Migrating Repositories	5-23
File Repositories	5-23
iBSE Repositories	5-23
JCA Repositories	5-28
Migrating Event Handling Configurations	5-28
Exporting or Importing Targets	5-32
Retrieving or Updating Web Service Method Connection Information	5-36
Starting or Stopping a Channel Programmatically	5-40
A. Using Application Explorer in BEA WebLogic Workshop to Create XML Schemas and Web Services A-1	
Generating Schemas and a Business Service	A-2
Starting Application Explorer in WebLogic Workshop	A-3
Creating and Managing a Connection	A-6
Disconnecting From a Defined Target	A-10
Editing a Defined Target	A-10
Deleting a Defined Target	A-11
Viewing Metadata	A-11
Creating a Statement and Generating Schemas	A-15
Creating and Testing a Parameterized SQL Statement	A-17
Creating a Batch Statement	A-22

Request and Response Documents	A-25
Regular SQL Statements	A-26
Parameterized SQL Statements	A-29
Stored Procedure Schemas for an Oracle Database	A-32
Stored Procedure Schemas for a VSAM Database	A-33
Multiple Executions of the Same SQL Statement or Stored Procedure	A-35
Understanding iWay Business Services Engine	A-37
Creating a Business Service	A-38
Testing a Business Service	A-39
Generating WSDL From a Web Service	A-42
Identity Propagation	A-45
Adding a Control for an iWay Resource in BEA WebLogic Workshop	A-45
Adding a Web Service Control to a BEA WebLogic Workshop Application	A-45
Adding an iWay Extensible CCI Control to a BEA WebLogic Workshop Application	A-46
Overview	A-47
Using the Extensible CCI Control	A-52
B. Using Application Explorer in BEA WebLogic Workshop for Event Handling ...	B-2
Starting Application Explorer in WebLogic Workshop	B-3
Understanding iWay Event Functionality	B-4
Creating, Editing, and Deleting an Event Port	B-4
Creating an Event Port From the iWay Event Adapters Tab	B-4
Editing and Deleting an Event Port	B-17
Creating, Editing, and Deleting an Event Channel	B-18
Creating a Channel	B-18
Editing and Deleting a Channel	B-25
Deploying iWay Components in a Clustered BEA WebLogic Environment	B-26
For More Information	B-32
C. Using WebLogic Workshop to Access VSAM	C-1
Using WebLogic Workshop to Access VSAM	C-2
Running the JWSNAME Web Service from WebLogic Workshop	C-8
D. JDBC Drivers	D-1
Copying and Collecting a JDBC File	D-2

CHAPTER 1

Introducing the iWay XML Adapter for RDBMS

Topics:

- Introduction
- Using the Adapter With Relational Databases
- Using the Adapter to Access Non-relational Databases
- Deployment Information for the Adapter

The following topics provide an overview of the iWay XML Adapter for RDBMS and how it works, including descriptions of key features and functionality.

Introduction

Since most custom and packaged applications are built with relational databases, relational database management systems (RDBMS) must be taken into consideration in an enterprise integration strategy. The iWay XML Adapter for RDBMS incorporates in-depth knowledge of relational database query access and transaction, replication, and copy management technologies to optimize the use of databases with enterprise application systems.

You also can use the adapter to gain relational database access to non-relational data sources. To access non-relational data, the adapter works in conjunction with an adapter server component that is installed and runs outside of the target database management system. For the purposes of this manual, the adapter server component is equivalent to an RDBMS.

Using the Adapter With Relational Databases

The adapter enables integration with RDBMS systems by one of the following ways:

- **Sending a Request.** The adapter sends requests to the database, using either SQL queries or stored procedures.
- **Using a Listener.** The adapter listens for database table activity.

In both cases, the query or stored procedure call is expressed to the adapter in the form of an XML document.

Key features of the adapter include:

- Asynchronous, bi-directional message interactions between applications and databases, including IBM DB2, IBM Informix, Microsoft SQL Server, Oracle, IDMS, VSAM, IMS/DB, ADABAS, Sybase RDBMSs, and others.
- iWay Application Explorer, which uses metadata on database servers to build XML schemas for use by adapter requests.
- Integration of requests and table event (outbound) operations in workflows.
- JDBC™ 2.0 standard SQL operations (DELETE, INSERT, SELECT, and UPDATE) and the execution of stored procedures against DB2, Informix, MS SQL Server, Oracle, Sybase, and any database management system accessible by the server component.
- Oracle object-relational extensions, such as processing of nested tables and arrays in accessing PL/SQL stored procedures or supporting outbound database rows on Oracle AQ queues.

Using the Adapter to Send a Request

The adapter can process SQL statements embedded in XML documents and forward them to an RDBMS (or server component) as a request. The RDBMS returns the data to the adapter, which returns the data to the client.

The adapter can:

- Receive a service request from an external client.
- Transform an XML request document into the RDBMS data format.

The request document conforms to the XML schema generated by Application Explorer and based on RDBMS metadata.

- Send the request to the RDBMS and wait for its response.
- Transform the response from the RDBMS data format to an XML document.

The XML document conforms to the XML schema for the response that was generated by Application Explorer and based on RDBMS metadata.

Using the Adapter Listener

The adapter supports the capture of events from applications that write to a database. It captures the data and performs operations based on the content of table rows. The adapter reads one or more rows from the table and creates an XML document representing the column data in each row.

Additional business logic facilities can be applied to the constructed XML document, including transformation, validation, security management, and application processing. Transformations by business logic can include deleting rows or altering column values. The resulting XML is formatted and sent to the adapter for further processing.

The listener can:

- Monitor data changes by repeatedly performing an SQL query.

The SQL listener also supports customized user exits with Java™ classes to define custom operations on the row sets.

- Be configured to operate one row at a time or to operate on sets of data.

You can configure the listener to send events only to a business process workflow when a specified minimum number of rows become available in the source RDBMS.

- Allow the configuration of an optional SQL post-query statement.

The statement performs specific RDBMS operations after the adapter sends the row set (formatted as XML) to a business process workflow.

The default operation is to delete the rows that were transferred to the workflow.

Other options can include moving the rows to an archive table or marking the rows with an SQL UPDATE.

- Support complex configurations.

For example, you may want to extract information periodically from a base table and incorporate reference data from an additional table. Records cannot be deleted from the base table and reference table.

In this case, the adapter uses a temporary table to maintain the sequenced rows in the base table. The temporary table contains a starting value for the sequence. It holds the last value of the sequence field processed by the RDBMS listener, enabling multiple event operations to collect updates while avoiding sending duplicates to a business process workflow.

- Support user-defined exits.

User-defined exits can be implemented to enable more complex programming or external database operations.

For example, an operating system program can be executed to facilitate an import or export process within a custom application.

Using the Adapter to Access Non-relational Databases

iWay Software uses a proprietary metadata management and creation tool that enables all databases on all platforms to look and act as if they were relational databases. This enables a single, uniform approach to data access.

Depending on which databases you have licensed, the following table lists several of the non-relational databases that can be accessed through the adapter.

IBM-Compatible Mainframes (MVS/VM)	OpenVMS	UNIX-Based Computers
ADABAS	ADABAS/C	ADABAS/C
CA-Datcom/DB	DBMS	C-ISAM
DB2	FOCUS	DB2/6000
FOCUS	Ingres	Essbase
CA-IDMS/DB	Oracle	FOCUS
CA-IDMS/SQL	Rdb	Interplex (DMS/RDMS 2200/1100)
IMS/DB	RMS	Informix
ISAM	Sybase	Ingres
Millennium	Progress	Oracle
MODEL 204		Progress
NOMAD		Red Brick
Oracle		Sybase ASE
SQL/DS		Sybase IQ
Supra		Teradata
System 2000		UniVerse (PICK)
Teradata		
TOTAL		
QSAM		
VSAM		

OS/400	Tandem	Windows
FOCUS SQL/400	Enscribe FOCUS NonStop SQL	ADABAS/C DB2/2 Essbase FOCUS Informix Interplex (DMS/RDMS 2200/1100) Oracle Microsoft SQL Server Microsoft Analytical Engine Sybase ASE Sybase IQ Teradata

To enable this access, iWay Software adapter structure is twofold. All Java-based adapters such as the adapters for RDBMS, IMS/TM, and CICS are hosted within an iWay Adapter Framework (iWAF) on a server platform such as BEA WebLogic Server.

A separate iWay server component hosts all of the data adapters that access the underlying non-relational data using select statements. This server component runs outside of the adapter host (for example, BEA WebLogic) environment. The adapter connects to the iWay data adapters hosted in the server component using a Java-based connection.

Because the server component looks as if it were a relational database, the connection string to it is the same as to any relational database, for example, to an Oracle database. Therefore, the RDBMS connections are configured similarly as to a relational database.

After you configure an adapter and create metadata using Application Explorer, you can access the database or file system (such as VSAM) using standard JDBC calls. Therefore, you can access all databases and file systems, whether mainframe, AS400, or UNIX, as if the database were a full JDBC client RDBMS after you configure them on the server.

Read access is supported by all iWay adapters. Write access is supported by all relational adapters such as DB400 and OS390 DB2. Some adapters do not support write access, for example, CA-IDMS/DB, Datacom, and Model 204. Read/write access is supported by ADABAS, VSAM, and IMS via SQL insert and update statements. Depending on the type of database accessed, the server component could have specific platform requirements. For the applicable database in question, see the iWay documentation.

In the usual non-relational database access scenario, the iWay XML Adapter for RDBMS (hosted, for example, by the BEA WebLogic platform) connects to the iWay Adapter for VSAM (hosted by the server component) using JDBC standards. Application Explorer is used to configure this connection. You can then create Web services for SQL and parameterized SQL using Application Explorer. You also can use Application Explorer to create events that occur within the database, such as an insert to a VSAM file or a modification of a VSAM record.

Accessing Stored Procedures for Non-relational Data Sources

The adapter is used when there is a specific requirement to create and execute catalogued iWay stored procedures (remote procedure calls, also referred to as RPCs) on the server component. iWay uses a very powerful fourth generation language that is much more robust than SQL.

iWay stored procedures on the server component can be created to enable complex multi-platform joins, specialized routines, and so forth. iWay stored procedures also enable COBOL or RPG programs to be executed. To use the adapter, an iWay stored procedure must be catalogued before Application Explorer can create the schemas or Web services for that stored procedure. For more information on using the extended functionality within iWay stored procedures, contact iWay Customer Support Services.

When used in conjunction with Application Explorer, the adapter creates Web services that can be used to run the stored procedures from any Web client.

Deployment Information for the Adapter

The adapter works in conjunction with the following components:

- iWay Application Explorer

and

- The iWay server component (for non-relational database access only)

with either

- Integration Business Services Engine (iBSE)

or

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

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

Both iBSE and the iWay Connector for JCA are deployed to the BEA WebLogic environment with Application Explorer and the adapters. The iWay server component is deployed outside of the adapter host (for example, BEA WebLogic Server) and outside of the target database management system.

Deployment Information Roadmap

The following table lists the location of deployment information for the adapter. A description of the Integration Business Services Engine (iBSE) and the iWay Enterprise Connector for J2EE Connector Architecture (JCA) follow the table.

Deployed Component	For more information, see
iWay Application Explorer	Chapters 2 and 3, and Appendices A and B of this guide <i>iWay Installation and Configuration for BEA WebLogic</i> <i>iWay Servlet Application Explorer for BEA WebLogic User's Guide</i>
Integration Business Services Engine (iBSE)	<i>iWay Installation and Configuration for BEA WebLogic</i>
iWay Enterprise Connector for J2EE Connector Architecture (JCA)	<i>iWay Connector for JCA for BEA WebLogic User's Guide</i> <i>iWay Installation and Configuration for BEA WebLogic</i>
iWay server component	<i>iWay Server Installation</i> <i>iWay Server Administration for UNIX, Windows, OpenVMS, OS/400, OS/390 and z/OS</i> <i>iWay Data Adapter Administration for UNIX, Windows, OpenVMS, OS/400, OS/390 and z/OS</i>

The Integration Business Services Engine (iBSE)

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

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

- Custom and legacy applications
- Database queries and stored procedures
- Packaged applications
- Terminal emulation and screen-based systems
- Transactional systems

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

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

The iWay Enterprise Connector for J2EE Connector Architecture (JCA)

The iWay Enterprise Connector for J2EE Connector Architecture (JCA) enables developers of JCA-compliant applications to deploy iWay adapters as JCA resources. The connector is supported on BEA WebLogic Enterprise.

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

CHAPTER 2

Creating XML Schemas or Business Services

Topics:

- Generating Schemas and a Business Service
- Starting Servlet Application Explorer
- Creating and Managing a Connection
- Viewing Metadata
- Creating a Statement and Generating Schemas
- Request and Response Documents
- Multiple Executions of the Same SQL Statement or Stored Procedure
- Understanding Integration Business Services

This section describes how to use Application Explorer to:

- View metadata that describes your SQL statements and stored procedures.
- Create SQL statements and generate XML schemas that define request and response documents.
- Create business services (also known as Web services) for your SQL statements and stored procedures.

Generating Schemas and a Business Service

You can use Application Explorer to connect to relational databases, such as Oracle or Informix and to non-relational databases, such as an IMS database.

When connected to non-relational databases, the adapter uses an iWay server component which enables you to take advantage of all the integration capabilities offered by iWay Software for mainframe database access. For example, when using the iWay server component, you gain relational database access to non-relational data. Stored procedures for non-relational databases can be created and stored using the server component.

Procedure How to Generate Request and Response Document Schemas or a Business Service

1. Start Servlet Application Explorer.

You can open a new or existing connection to a relational or non-relational database management system, as described in *Creating and Managing a Connection* on page 2-4.

After you connect to a system, you can expand the Service Adapters node to view the list of adapters installed on your system. After you finish using a connection, you can close it. If you will not need the connection in the future, you can delete it.

Note: When you close Application Explorer, it automatically closes all open connections.

2. Generate XML schemas.

The schemas define request and response documents for your SQL statements and stored procedures, as described in *Creating a Statement and Generating Schemas* on page 2-14.

You can use the schemas when you create request documents and when you develop logic to process responses.

3. Create request documents.

You create documents for each operation against each table and for each stored procedure. You can use a third-party XML tool to generate a request document from the XML schema.

You also may want to examine the metadata describing your SQL statements and procedures, as described in *Viewing Metadata* on page 2-11. For information about request and response document formats, see *Request and Response Documents* on page 2-23.

4. Generate a business service (also known as a Web service) for an SQL statement or stored procedure. For more information on Web services, see *Understanding Integration Business Services* on page 2-36.

Starting Servlet Application Explorer

The server must be started where Servlet Application Explorer is running.

Procedure How to Start Application Explorer

1. Ensure the server is started where Application Explorer is running.
2. Enter the following URL in your browser window:

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

where:

`hostname`

Is the machine where Application Explorer is installed.

`port`

Is the port number for iBSE. The default port is 7001.

Application Explorer opens.

The Available Hosts drop-down list appears in the upper-right corner. Three tabs appear near the top of the Application Explorer screen. From left to right they are:

- Service Adapters, where you create and manage connections to RDBMS.
- Event Adapters, where you configure RDBMS event listening.
- Integration Business Services, where you create and view business services.

The left pane of the window contains an expandable list of adapter nodes (based on the adapters installed), events, or business services, depending on the tab that is selected. The right pane provides the details of the selected adapter, event, or service, and is the work area where you will define and modify adapter functions and services.

The Available Hosts drop-down list specifies to which Servlet iBSE instance or JCA instance you connect.

For more information on accessing different instances of a JCA installation or a Servlet iBSE, see the *iWay 5.5 Installation and Configuration* documentation.

Creating and Managing a Connection

To access an adapter, you must define a target that connects to the adapter. After the defined target is created, it automatically is saved. You must establish a connection to the defined target every time you start Application Explorer or after disconnecting.

Procedure How to Define a New Target

1. In the left pane of Application Explorer, expand the *Service Adapters* node.
2. Click the *RDBMS* node.
3. In the right pane, move the pointer over *Operations* and select *Define a new target*.

The Add a new RDBMS target dialog box opens in the right pane:

Add a new RDBMS target

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

Target Name:

Description:

Target Type:

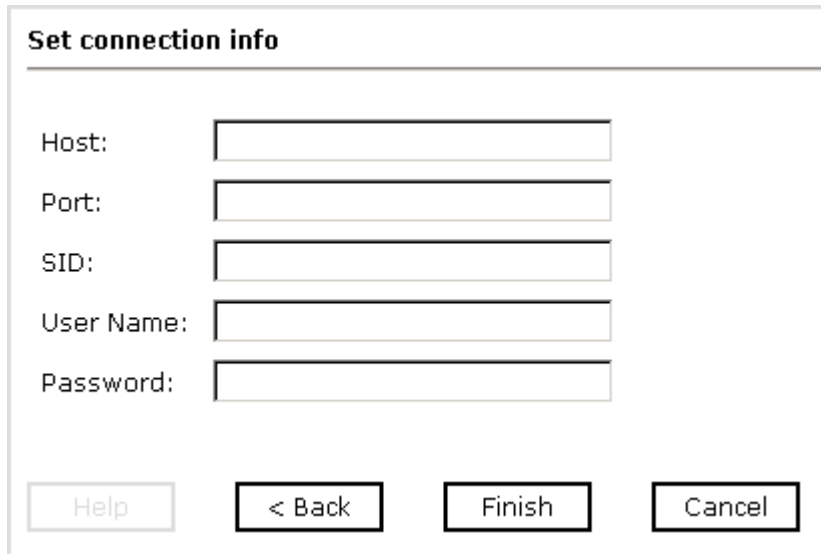
- a. In the Target Name field, type a descriptive name for the target, for example, NewTarget.
- b. In the Description field, type a brief description for the connection.
- c. From the Target Type drop-down list, select the type of target to connect to.

Important: Oracle is the default value. If you are connecting to a non-relational database, select *iWay Server*. This establishes a connection to the iWay server component. The server component is used to provide relational database access to non-relational databases. For more information on the server component, see Chapter 1, *Introducing the iWay XML Adapter for RDBMS*.

Note: The EDA Server and WebFOCUS Server target options are for connecting to Information Builders resources. The types of databases that display in the Target Type drop down list depends on the jar files you have installed in your lib directory.

4. Click *Next*.

The Set connection info dialog box opens in the right pane:



The dialog box titled "Set connection info" contains five text input fields labeled "Host:", "Port:", "SID:", "User Name:", and "Password:". Below the fields are four buttons: "Help", "< Back", "Finish", and "Cancel".

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

The fields that appear in the Set connection info dialog box are specific to the type of database to which you are connecting.

5. Enter connection information that is specific to the database to which you want to connect. The following table lists and defines connection parameters.

Parameter	Definition
Host	DNS or IP name of the server where the database instance resides.
Port	Port number on which the database is listening.
Data Source Name	Specific name of the database or data source to which you connect.
Use driver type	<p>For DB2 connections, select one of the following JDBC drivers:</p> <ul style="list-style-type: none"> • app driver. Ensure you have the DB2 client installed on the machine from which you are accessing Application Explorer. • net driver. Ensure you started the DB2 applet server. <p>For more information, see your DB2 documentation.</p>
File Name	Name of the database that you are trying to access. For the Domino database, the file name ends with the .nsf extension. One of the sample databases provided is JdbcDemo.nsf.
Server Name	For an iWay server component, the name of the service node to which you are connecting.
Service Name	For connecting to a non-relational database to access stored procedures, this parameter defaults to the DEFAULT service name of the iWay server component. Can be left blank.
Informix Server Name	Equivalent to the Informix variable name INFORMIXSERVER.
Driver	Name of the driver used to access the database you want to connect to. For more information, see your database documentation.
Initial Context Factory	JNDI context.INITIAL_CONTEXT_FACTORY that is provided by the JNDI service provider. For WebLogic Server, the WebLogic factory is: <code>weblogic.jndi.WLInitialContextFactory.</code>

Parameter	Definition
URL	<ul style="list-style-type: none"> For a JDBC connection, the JDBC driver-specific URL used to connect to the RDBMS. For information on using <code>selectMethod=Cursor</code>, see <i>Using selectMethod=Cursor</i> on page 2-8. For a data source connection, the URL to use to contact the JNDI provider. The syntax of this URL depends on the JNDI provider being used. This value corresponds to the standard JNDI property, <code>java.naming.provider.url</code>. The URL of the WebLogic Server is <code>t3://host:port</code> where: <code>host</code> Is the machine name where WebLogic Server is installed. <code>port</code> Is the port on which WebLogic Server is listening. The default port, if not changed at installation, is 7001.
JNDI Name	JNDI name of a queue to which events are emitted.
Database	Oracle, SQL Server, Sybase, DB2, Informix, EDA Server, or other.
SID	For an Oracle database, the unique name of the database service selected by the database administrator or the person who installed Oracle.
User Name	Database user ID to access the database. The user ID must have database access to the interface tables that are accessed.
Password	Password associated with the specified user name.

Note: When connecting to the iWay server component for access to non-relational databases, the connection information must be the same for all databases in the server component system.

6. Click *Finish*.

In the left pane, the target name appears under the node where you created the new target. You have finished creating the new target.

For information on connecting to the target, see *How to Connect to a Defined Target* on page 2-8.

Reference **Using selectMethod=Cursor**

To avoid some exceptions when using the iWay XML Adapter for RDBMS with Microsoft SQL Server 2000 Driver for JDBC, you must add `selectMethod=cursor` to the JDBC URL specification. For example,

```
jdbc:microsoft:sqlserver://PMSNJC:1433;DatabaseName=dbname;selectMethod=cursor;
```

This statement determines whether Microsoft SQL Server "server cursors" are used for SQL queries. Because the adapter is not limited to a single active statement while executing a set of queries within a transaction, adding this statement to the JDBC URL allows you to specify multiple queries within a transaction. This helps to prevent errors because it addresses default settings in the adapter and in the driver.

The benefit of specifying this statement is that it enables you to have multiple concurrent statements open from a given connection, which is required for pooled connections.

Procedure **How to Connect to a Defined Target**

1. In the left pane of Application Explorer, expand the *Service Adapters* node.
2. Expand the *RDBMS* node and select the connection you want to open (for example, *NewTarget*):



3. In the right pane, move the pointer over *Operations* and select *Connect*.
The connection dialog box opens displaying the connection information.
4. Verify your connection parameters. If required, provide the password and then click *OK*.

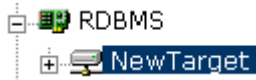
If the parameters are correct and the RDBMS component is available, the node under the RDBMS node displays a plus sign indicating that you are connected to the defined target. Otherwise, an error message appears in the right pane.

Disconnecting From a Defined Target

Although you can maintain multiple open connections, iWay Software recommends disconnecting from targets that are not in use.

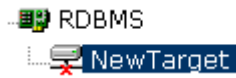
Procedure How to Disconnect From a Defined Target

1. In the left pane of Application Explorer, expand the *Service Adapters* node.
2. Expand the *RDBMS* node.
3. Click the connection you want to close (for example, *NewTarget*):



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

Disconnecting from the application closes the connection, but the connection still appears in the left pane so that you can re-open it. The connection node now has an x icon, indicating that it is closed, as shown in the following figure:



When you want to re-establish a connection, *Connect* is available from the pop-up menu.

Editing a Defined Target

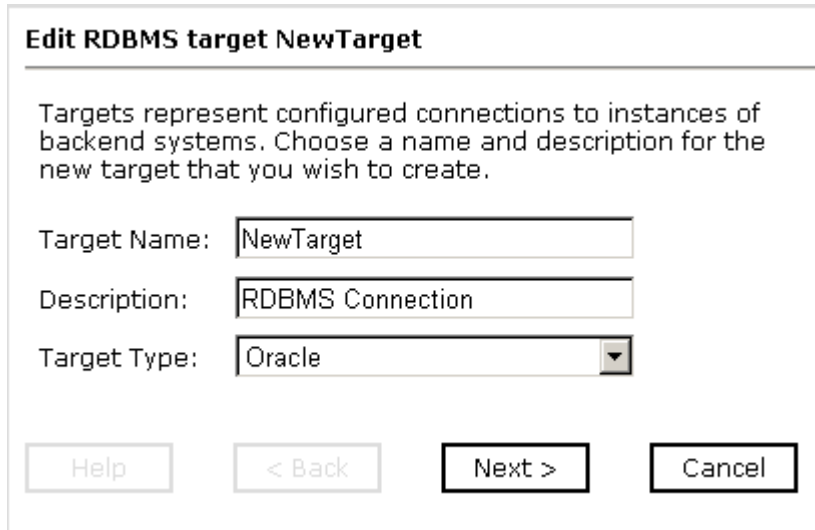
After you create a defined target using Application Explorer, you can edit any information that you provided during the creation process.

Procedure How to Edit a Defined Target

1. In the left pane of Application Explorer, expand the *Service Adapters* node.
2. Expand the *RDBMS* node.
3. Click the connection you want to edit (for example, *NewTarget*):

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

The Edit dialog box opens in the right pane containing three fields (Target Name, Description, and Target Type) and two action buttons (Next and Cancel).



Edit RDBMS target NewTarget

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

Target Name:

Description:

Target Type:

5. Modify the target information as needed and then click *Next*.

The Set connection info dialog box opens in the right pane.

6. Modify the information as needed and then click *Finish*.

Deleting a Defined Target

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

Procedure How to Delete a Defined Target

1. In the left pane of Application Explorer, expand the *Service Adapters* node.
2. Expand the *RDBMS* node to view the list of connections.
3. Click the connection you want to delete.
4. In the right pane, move the pointer over *Operations* and select *Delete*.

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

5. Click *OK*.

The node disappears from the list of available connections.

Viewing Metadata

Viewing metadata is useful when creating request documents. You can view:

- Table metadata, as described in *How to View Table Metadata* on page 2-11.
- Stored procedure metadata for a relational or a non-relational database, as described in *How to View Stored Procedure Metadata* on page 2-12.

Procedure How to View Table Metadata

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 2-8.
2. Expand the *Schemas* node under the desired connection.
3. Select a database.
4. Expand the *Tables* node.
5. Scroll down and select a table to view.

Note: Although the list of tables includes all tables in the RDBMS, the user ID you specified for the connection may not have access to the table you selected. If this is the case, the creation of schemas fails.

When you select a table, a metadata summary table appears in the right pane containing Property and Value columns. The Value column contains the description of the table and ellipsis symbols enabling you to access database properties and columns:

Properties for empdata

Property	Value
iwaf.description	
Database Properties	...
Columns	...

- a. Depending on which properties you want to view, click the ellipsis symbol in the Database Properties or the Columns row.

The properties (for example, data type and column size) appear in the right pane:

Details for collection property Columns

column name	data type	type name	column size	buffer length	decimal digits	num prec radix	nullable
PIN	1	CHAR	9		null	null	0
LASTNAME	1	CHAR	15		null	null	0
FIRSTNAME	1	CHAR	10		null	null	0
MIDINITIAL	1	CHAR	1		null	null	0
DIV	1	CHAR	4		null	null	0
DEPT	1	CHAR	20		null	null	0
JOBCLASS	1	CHAR	8		null	null	0
TITLE	1	CHAR	20		null	null	0
SALARY	8	DOUBLE	15		2	10	0
HIREDATE	9	DATE	10		null	null	0
AREA	1	CHAR	13		null	null	0

Close

- b. Use this information to determine the table (or tables) and fields to use when you are ready to create a schema.

Procedure How to View Stored Procedure Metadata

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 2-8.
2. Expand the *Schemas* node under the desired connection.
3. Select a database.
4. Expand the *Procedures* node.
5. Scroll down and select the procedure to view.

Note: Although the list of procedures includes all procedures in the database, the user ID specified for the connection may not have access to the specified procedure. If this is the case, the creation of schemas fails.

When you select a procedure, a metadata summary table appears in the right pane containing Property and Value columns. The Value column contains the description of the procedure and ellipsis symbols enabling you to access database properties and columns:

Properties for empdata

Property	Value
iwaf.description	
Database Properties	...
Columns	...

- Depending on which properties you want to view, click the ellipsis symbol in the Database Properties or the Columns row.

The properties (for example, data type and column size) appear in the right pane:

Details for collection property Columns

column name	data type	type name	column size	buffer length	decimal digits	num prec radix	nullable
PIN	1	CHAR	9		null	null	0
LASTNAME	1	CHAR	15		null	null	0
FIRSTNAME	1	CHAR	10		null	null	0
MIDINITIAL	1	CHAR	1		null	null	0
DIV	1	CHAR	4		null	null	0
DEPT	1	CHAR	20		null	null	0
JOBCLASS	1	CHAR	8		null	null	0
TITLE	1	CHAR	20		null	null	0
SALARY	8	DOUBLE	15		2	10	0
HIREDATE	9	DATE	10		null	null	0
AREA	1	CHAR	13		null	null	0

Close

- b. Use this information to determine the procedure (or procedures) and fields to use when you are ready to create a schema.

Creating a Statement and Generating Schemas

You can create an SQL statement even when using the adapter for non-relational databases. After you create the statement, you can generate schemas that define request and response documents. The metadata is stored in the iWay Repository, which can be implemented in an RDBMS (such as Oracle or Microsoft SQL Server), a file system, or a specialized XML database. You can generate the following types of statements:

- Regular SQL Statements, as described in *How to Create a Regular SQL Statement* on page 2-14.
- Parameterized SQL statements, as described in *How to Create a Parameterized SQL Statement* on page 2-16.
- Batch statements, as described in *How to a Create Batch Statement* on page 2-20.

You can generate request and response schemas for:

- Regular (non-parameterized) SQL statements and parameterized SQL statements, as described in *How to Generate a Schema for a Prepared Statement* on page 2-21.
- Stored procedures for relational databases, and iWay stored procedures for non-relational databases, as described in *Generating a Schema for a Stored Procedure* on page 2-22.

If you plan to create business services, you are not required to generate a schema. For more information, see *Understanding Integration Business Services* on page 2-36.

Procedure How to Create a Regular SQL Statement

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 2-8.
2. Click the *Statements* node.
3. In the right pane, move the pointer over *Operations* and select *Create Prepared Statement*.

The Create Prepared Statement input area opens in the right pane:

Create Prepared Statement

Name :

Enter SQL Statement :

- a. Type a name for the statement.

iWay Software recommends that you specify a name that describes the service. For example, a name of CustomerIntField could represent a request against the Customer Interface table returning a Field format response document.

- b. In the Enter SQL Statement field, type the SQL statement for the adapter to use.

Note: If you are not the owner of the table(s), the table name must be fully qualified.

4. Click *Create*.

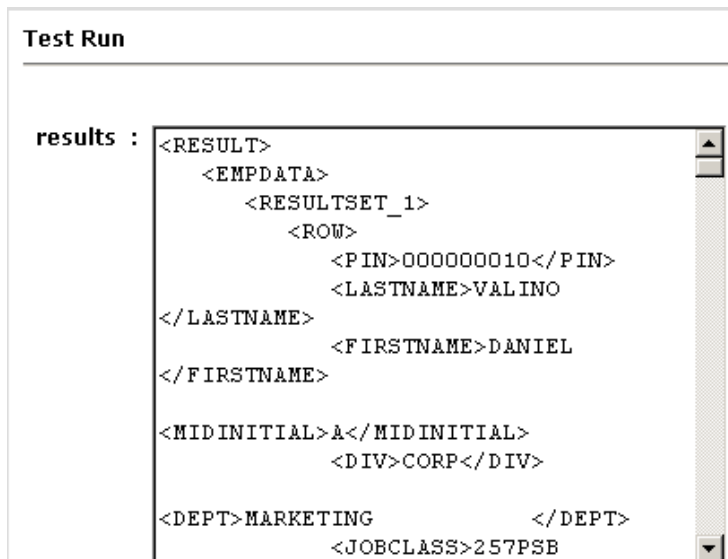
After the SQL statement node is built, you are ready to test the statement.

- For information on testing a regular SQL statement, see *How to Test an SQL Statement* on page 2-15.
- For information on creating schemas for both parameterized and regular SQL statements, see *How to Generate a Schema for a Prepared Statement* on page 2-21.

Procedure How to Test an SQL Statement

1. Select the SQL statement node you want to test.
2. In the right pane, move the pointer over *Operations* and select *Test Run*.
The Test Run dialog box opens in the right pane.
3. Click *Test*.

The results appear in the Test Run results window:



Creating and Testing a Parameterized SQL Statement

Parameterized SQL allows an SQL statement to be stored within the repository system with parameters imbedded within it. These parameters can be retrieved from XML documents at run time and executed against the SQL statements specified at design time. Application Explorer creates and maps parameters for the parameterized SQL at design time.

Procedure How to Create a Parameterized SQL Statement

If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 2-8.

1. Click the *Statements* node.
2. In the right pane, move the pointer over *Operations* and select *Create Prepared Statement*.

The Create Prepared Statement input area appears in the right pane:

Create Prepared Statement

Name :

Enter SQL Statement :

- a. Type a name for the statement.
 - b. In the Enter SQL Statement field, type the parameterized SQL statement.
- Note:** If you are not the owner of the table(s), the table name must be fully qualified.
3. Click *Create*.

The Parameter Data Type selection information appears in the right pane:

Create Prepared Statement

params

Parameter Name	Data Type
<input type="text" value="param0"/>	UNKNOWN <input type="button" value="v"/>

- a. In the Parameter Name column, type a name for each parameter.
- b. In the Data Type column, select a data type for each parameter using the drop-down list.

4. Click *Update*.

The properties table for the newly created statement appears in the right pane containing Property and Value columns. The Value column contains a description of the SQL statement, the actual SQL statement, and ellipsis symbols enabling you to access parameters and database properties:

Operations ►
Properties for ParamSQL2

Property	Value
iwaf.description	
Statement	Select * from empdata where LASTNAME=?
Parameters	...
Database Properties	...

- Depending on which properties you want to view, click the ellipsis symbol in the Parameters or Database Properties row.
- For information on testing a parameterized SQL statement, see *How to Test a Parameterized SQL Statement* on page 2-19.
- For information on creating schemas for both parameterized and regular SQL statements, see *How to Generate a Schema for a Prepared Statement* on page 2-21.

Procedure How to Test a Parameterized SQL Statement

1. Select the parameterized SQL statement node you want to test.
2. In the right pane, move the pointer over *Operations* and select *Test Run*.

The Test Run dialog box opens for the SQL statement containing the parameter name, data type, and an input box where you can type the parameter value:

Test Run

params

Parameter Name	Data Type	Value
param0	CHAR	<input type="text"/>

3. For each parameter, type a value in the Value field.

In the case of the following SQL statement:

```
Select * from empdata where LASTNAME=?
```

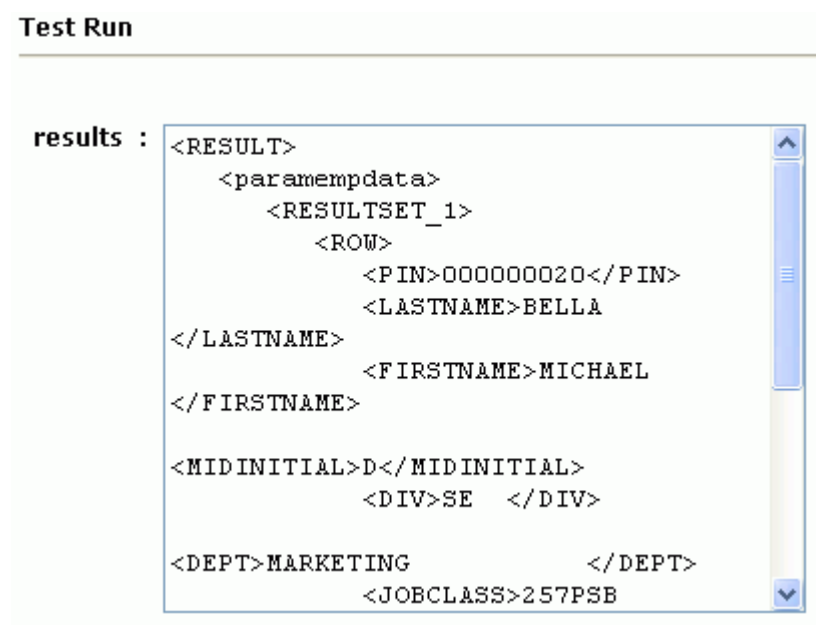
provide a sample character value, for example, BELLA.

In this example, the values correspond to values of fields found in a table.

Parameterized statements may include parameters that are input for SQL functions, for example, the Oracle SQL function TO_DATE(StringParm). In this case, the data type selected is the expected data type of the SQL function. This is why you provide the SQL type when you create the prepared parameterized SQL statement.

4. Click *Test*.

The results appear in the Test Run results window:



5. To exit the results, click *OK*.

Creating a Batch Statement

Batch statements allow you to execute multiple SQL and/or parameterized SQL statements simultaneously.

Procedure How to a Create Batch Statement

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 2-8.
2. In the left pane, select the *Batches* node.
3. In the right pane, move the pointer over *Operations* and select *Create A Batch*.
The Create A Batch dialog box opens in the right pane.

4. Type a name for the new Batch and click *Create*.

The batch properties information appears in the right pane containing Property and Value columns. The Value column contains a description of the batch, the process count, and an ellipsis symbol that enables you to access database properties:

Operations ►
Properties for New Batch

Property	Value
iwaf.description	Runnable Batch
Process Count	0
Database Properties	...

5. Move the pointer over *Operations* and choose whether to add stored procedures or statements.
The Add Statement or Add Procedure drop-down selection appears in the right pane.
6. Select the statement or procedure from the drop-down list and click *Next*.
7. To add more statements or procedures, select the batch node in the left pane, and then select the appropriate option from the Operations menu in the right pane.

Generating a Schema for a Prepared Statement

You must first create the prepared statement before generating a schema for it. For more information on creating prepared statements, see the following procedures:

- *How to Create a Regular SQL Statement* on page 2-14.
- *How to Create a Parameterized SQL Statement* on page 2-16.
- *How to a Create Batch Statement* on page 2-20.

Procedure How to Generate a Schema for a Prepared Statement

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 2-8.
2. Click the *Statements* node.
3. Click the node containing the prepared statement for which you want to generate a schema.
4. In the right pane, move the pointer over *Operations* and select *Generate Schema*.
A table that lists the available schemas appears.
5. To view a schema, click the ellipsis symbol in the Schema column.

The schema is generated and ready to use. You can use the generated request schema to create a sample XML document to be used by the adapter. To add a schema to a business service, see *Understanding Integration Business Services* on page 2-36.

Generating a Schema for a Stored Procedure

The following procedure describes how generate a schema for stored procedures for relational databases, and iWay stored procedures for non-relational databases.

Procedure How to Generate a Schema for a Stored Procedure

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 2-8.
2. Expand the *Schemas* node under the desired connection.
3. Select the database containing the stored procedure for which you want to generate a schema.
4. Expand the *Procedures* node.
5. Select the stored procedure.
6. In the right pane, move the pointer over *Operations* and select *Generate Schema*.

The Schemas table appears in the right pane containing three columns:

Schemas

Part	Root Tag	Schema
Request	RDBMS	...
Response	RESULT	...
Event	N/A	N/A

Help

OK

Cancel

- a. To view the request schema, click the ellipsis symbol that is located in the third column of the Request row.
- b. To view the response schema, click the ellipsis symbol that is located in the third column of the Response row.

The schemas are now ready to use. You can use the generated request schema to create a sample XML document to be used by the adapter.

7. Click OK.

Reference Schema Location

Application Explorer stores the schemas it creates in subdirectories under the iWay home directory of the machine where it is installed. The exact location of the schemas differs depending on whether you deploy Application Explorer with an iBSE or a JCA configuration.

- When using the adapter with an iBSE configuration, the schemas are stored under a \schemas subdirectory of the iWay home directory, for example,

`C:\Program Files\iway55\bea\ibse\wsdl\schemas\service\RDBMS\NewTarget`

where:

NewTarget

Is the name of the connection to the RDBMS system as defined in Application Explorer. Under this directory, Application Explorer creates subdirectories containing schemas.

- When using the adapter with a JCA configuration, the schemas are stored under a \schemas subdirectory of the iWay home directory, for example,

`C:\Program Files\iWay55\config\base\schemas\RDBMS\NewTarget`

where:

NewTarget

Is the name of the connection to the RDBMS system as defined in Application Explorer. Application Explorer stores the schemas in this directory.

Request and Response Documents

You can generate request document schemas using Application Explorer, as described in *Creating a Statement and Generating Schemas* on page 2-14. You can generate request document instances using a third party XML tool and submit those documents to the RDBMS or iWay agent.

The following topics include examples of schemas and instance documents for:

- Regular SQL Statements
- Parameterized SQL Statements
- Stored Procedure Schemas for an Oracle Database
- Stored Procedure Schemas for a VSAM Database

Regular SQL Statements

The following examples are based on schemas created for a regular SQL statement.

Example Regular SQL Request Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:22:33Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RDBMS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="stock_price_select">
          <xsd:complexType>
            <xsd:attribute name="location" type="xsd:string"
use="optional" fixed="RDBMS/Statements/stock price select"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example Regular SQL Request Instance Document

```
<?xml version="1.0" encoding="UTF-8"?>
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\stock price
select_request.xsd">
  <stock_price_select location="RDBMS/Statements/stock price select"/>
</RDBMS>
```

Example Regular SQL Response Schema

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:22:33Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RESULT">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="stock_price_select">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="RESULTSET_1" minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="ROW" minOccurs="0" maxOccurs="unbounded">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="RIC" type="xsd:string"/>
                          <xsd:element name="PRICE" type="xsd:double"/>
                          <xsd:element name="UPDATED" type="xsd:date"/>
                          <xsd:element name="RR" type="xsd:double"/>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

Example Regular SQL Response Instance Document

```
<?xml version="1.0" encoding="UTF-8"?>
<RESULT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\stock price
select_response.xsd">
  <stock_price_select>
    <RESULTSET_1>
      <ROW>
        <RIC>String</RIC>
        <PRICE>3.14159265358979</PRICE>
        <UPDATED>1967-08-13</UPDATED>
        <RR>3.14159265358979</RR>
      </ROW>
      <ROW>
        <RIC>String</RIC>
        <PRICE>3.14159265358979</PRICE>
        <UPDATED>1967-08-13</UPDATED>
        <RR>3.14159265358979</RR>
      </ROW>
      <ROW>
        <RIC>String</RIC>
        <PRICE>3.14159265358979</PRICE>
        <UPDATED>1967-08-13</UPDATED>
        <RR>3.14159265358979</RR>
      </ROW>
    </RESULTSET_1>
  </stock_price_select>
</RESULT>
```


Parameterized SQL Statements

The following examples are based on schemas created for a parameterized SQL statement.

Example Parameterized SQL Request Statement

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T21:57:19Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RDBMS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="paramempdata">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="param0">
                <xsd:complexType>
                  <xsd:simpleContent>
                    <xsd:extension base="xsd:string">
                      <xsd:attribute name="dataType"
type="xsd:string" use="required" fixed="CHAR"/>
                    </xsd:extension>
                  </xsd:simpleContent>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="location" type="xsd:string"
use="optional" fixed="RDBMS/Statements/paramempdata"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example Parameterized SQL Request Instance Document

```
<?xml version="1.0" encoding="UTF-8"?>
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\paramempdata_request.xsd">
  <paramempdata location="RDBMS/Statements/paramempdata">
    <param0 dataType="CHAR">String</param0>
  </paramempdata>
</RDBMS>
```

Example Parameterized SQL Response Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T21:57:20Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RESULT">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="paramempdata">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="RESULTSET_1" minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="ROW" minOccurs="0" maxOccurs="unbounded">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="PIN" type="xsd:string"/>
                          <xsd:element name="LASTNAME" type="xsd:string"/>
                          <xsd:element name="FIRSTNAME" type="xsd:string"/>
                          <xsd:element name="MIDINITIAL" type="xsd:string"/>
                          <xsd:element name="DIV" type="xsd:string"/>
                          <xsd:element name="DEPT" type="xsd:string"/>
                          <xsd:element name="JOBCLASS" type="xsd:string"/>
                          <xsd:element name="TITLE" type="xsd:string"/>
                          <xsd:element name="SALARY" type="xsd:float"/>
                          <xsd:element name="HIREDATE" type="xsd:date"/>
                          <xsd:element name="AREA" type="xsd:string"/>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example Parameterized SQL Response Instance Document

```

<?xml version="1.0" encoding="UTF-8"?>
<RESULT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\paramempdata_response.xsd">
  <paramempdata>
    <RESULTSET_1>
      <ROW>
        <PIN>String</PIN>
        <LASTNAME>String</LASTNAME>
        <FIRSTNAME>String</FIRSTNAME>
        <MIDINITIAL>String</MIDINITIAL>
        <DIV>String</DIV>
        <DEPT>String</DEPT>
        <JOBCLASS>String</JOBCLASS>
        <TITLE>String</TITLE>
        <SALARY>3.14159</SALARY>
        <HIREDATE>1967-08-13</HIREDATE>
        <AREA>String</AREA>
      </ROW>
      <ROW>
        <PIN>String</PIN>
        <LASTNAME>String</LASTNAME>
        <FIRSTNAME>String</FIRSTNAME>
        <MIDINITIAL>String</MIDINITIAL>
        <DIV>String</DIV>
        <DEPT>String</DEPT>
        <JOBCLASS>String</JOBCLASS>
        <TITLE>String</TITLE>
        <SALARY>3.14159</SALARY>
        <HIREDATE>1967-08-13</HIREDATE>
        <AREA>String</AREA>
      </ROW>
      <ROW>
        <PIN>String</PIN>
        <LASTNAME>String</LASTNAME>
        <FIRSTNAME>String</FIRSTNAME>
        <MIDINITIAL>String</MIDINITIAL>
        <DIV>String</DIV>
        <DEPT>String</DEPT>
        <JOBCLASS>String</JOBCLASS>
        <TITLE>String</TITLE>
        <SALARY>3.14159</SALARY>
        <HIREDATE>1967-08-13</HIREDATE>
        <AREA>String</AREA>
      </ROW>
    </RESULTSET_1>
  </paramempdata>
</RESULT>

```

Stored Procedure Schemas for an Oracle Database

The following examples are based on schemas created for a stored procedure for an Oracle database.

Example Stored Procedure Request Schema for an Oracle Database

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:12:21Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RDBMS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="PROCIN">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Y" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="location" type="xsd:string"
use="optional" fixed="RDBMS/Schemas/EDARPK/Procedures/PROCIN"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example Stored Procedure Request Instance Document for an Oracle Database

```
<?xml version="1.0" encoding="UTF-8"?>
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\PROCIN_request.xsd">
  <PROCIN location="RDBMS/Schemas/EDARPK/Procedures/PROCIN">
    <Y>String</Y>
  </PROCIN>
</RDBMS>
```

Example Stored Procedure Response Schema for an Oracle Database

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:18:44Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RESULT">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="PROCIN">
          <xsd:complexType>
            <xsd:sequence/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example Stored Procedure Response Instance Document for an Oracle Database

```
<?xml version="1.0" encoding="UTF-8"?>
<RESULT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\PROCIN_response.xsd">
  <PROCIN/>
</RESULT>
```

Stored Procedure Schemas for a VSAM Database

The following examples are based on schemas created for an iWay stored procedure for a VSAM database.

Example Stored Procedure Request Schema for a VSAM Database

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:05:56Z -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="RPCIn">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="1" type="xs:string"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="optional"
default="RPCVSM"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example **Stored Procedure Request Instance Document for a VSAM Database**

```
<?xml version="1.0" encoding="UTF-8"?>
<RPCIn xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\RPCVSM_request.xsd"
name="RPCVSM">
    <l>String</l>
</RPCIn>
```

Example **Stored Procedure Response Schema for a VSAM Database**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:05:56Z -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
    <xs:element name="RPCOut">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Row" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="COMP_NAME" type="xs:string"/>
                            <xs:element name="EMP_ID" type="xs:string"/>
                            <xs:element name="EMPID" type="xs:string"/>
                            <xs:element name="FIRST_NAME" type="xs:string"/>
                            <xs:element name="LAST_NAME" type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="status" type="xs:string" use="required"/>
            <xs:attribute name="reason" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Example Stored Procedure Response Instance Document for a VSAM Database

```

<?xml version="1.0" encoding="UTF-8"?>
<RPCOut xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\RPCVSM_response.xsd"
status="String" reason="String">
  <Row>
    <COMP_NAME>String</COMP_NAME>
    <EMP_ID>String</EMP_ID>
    <EMPID>String</EMPID>
    <FIRST_NAME>String</FIRST_NAME>
    <LAST_NAME>String</LAST_NAME>
  </Row>
  <Row>
    <COMP_NAME>String</COMP_NAME>
    <EMP_ID>String</EMP_ID>
    <EMPID>String</EMPID>
    <FIRST_NAME>String</FIRST_NAME>
    <LAST_NAME>String</LAST_NAME>
  </Row>
  <Row>
    <COMP_NAME>String</COMP_NAME>
    <EMP_ID>String</EMP_ID>
    <EMPID>String</EMPID>
    <FIRST_NAME>String</FIRST_NAME>
    <LAST_NAME>String</LAST_NAME>
  </Row>
</RPCOut>

```

Multiple Executions of the Same SQL Statement or Stored Procedure

The iWay XML Adapter for RDBMS for BEA WebLogic allows you to execute a prepared SQL statement or stored procedure multiple times with different input parameters each time. The major benefits of this feature are as follows:

- **Connection Resource Utilization.** One connection or thread is established to the backend database. This minimizes resource consumption. Note that the number of cursors for select statements created is based on the number of parameter sets in the submitted request. If there are 3 sets of parameters for a select statement, there will be 3 cursors created and closed sequentially.
- **Logical Unit of Work (LUW).** A logical unit of work (LUW) is established that will roll back all of the updates or inserts that were issued by the SQL statement of prior executions. A transaction starts at the first set of parameters. The sets of parameters are executed in order from top to bottom of the message sequentially. For example, if an insert statement is submitted along with 3 sets of parameters, and the third set of parameters fails to insert, the previous 2 inserts will be rolled back.
- **Integration of Data From an Outside Source.** For integration scenarios where external data is used to "feed" SQL or stored procedures, this feature allows external data to be mapped to one XML execution block for the adapter to execute.

Note: Multiple processes in one message is not supported. Only one SQL statement or stored procedure can be executed in a single XML execution request block.

Example **Stored Procedure Request Instance Document for an SQL Server Database**

The following is a sample XML request document to execute a Microsoft SQL Server stored procedure called CustOrderHist. Note that the PARAMS tag surrounds the parameter(s) to be used as input. The adapter executes the stored procedure twice, with different input values for each execution.

```
<RDBMS>
  <ITERATE location="RDBMS/Schemas/dbo/Procedures/CustOrderHist">
    <PARAMS>
      <parm>ALFKI</parm>
    </PARAMS>
    <PARAMS>
      <parm>NNN*</parm>
    </PARAMS>
  </ITERATE>
</RDBMS>
```


Example Stored Procedure Response Instance Document for an SQL Server Database

The following is a sample XML response document from the stored procedure CustOrderHist. Note the multiple result sets that correspond to the two executions of the stored procedure. The second and last result set contains 0 records found for the input value "NNN*".

```
<?xml version="1.0" encoding="UTF-8" ?>
- <RESULTS>
- <RESULT>
- <CustOrderHist>
- <RESULTSET_1>
- <ROW>
  <ProductName>Aniseed Syrup</ProductName>
  <Total>6</Total>
</ROW>
- <ROW>
  <ProductName>Chartreuse verte</ProductName>
  <Total>21</Total>
</ROW>
- <ROW>
  <ProductName>Escargots de Bourgogne</ProductName>
  <Total>40</Total>
</ROW>
- <ROW>
  <ProductName>Flotemysost</ProductName>
  <Total>20</Total>
</ROW>
- <ROW>
  <ProductName>Grandma's Boysenberry Spread</ProductName>
  <Total>16</Total>
</ROW>
- <ROW>
  <ProductName>Lakkalikööri</ProductName>
  <Total>15</Total>
</ROW>
- <ROW>
  <ProductName>Original Frankfurter grüne Soße</ProductName>
  <Total>2</Total>
</ROW>
- <ROW>
  <ProductName>Raclette Courdavault</ProductName>
  <Total>15</Total>
</ROW>
</RESULTSET_1>
<RETURN_VALUE>0</RETURN_VALUE>
</CustOrderHist>
</RESULT>
```

```
- <RESULT>
- <CustOrderHist>
  <RESULTSET_1 />
  <RETURN_VALUE>0</RETURN_VALUE>
</CustOrderHist>
</RESULT>
</RESULTS>
```

Understanding Integration Business Services

Application Explorer provides Web developers with a simple, consistent mechanism for extending the capabilities of the adapter. The Integration Business Services Engine exposes functionality as Web services. It serves as a gateway to heterogeneous back-end applications and databases.

A Web service is a self-contained, modularized function that can be published and accessed across a network using open standards. It is the implementation of an interface by a component and is an executable entity. For the caller or sender, a Web service can be considered as a “black box” that may require input and delivers a result. A Web service integrates within an enterprise as well as across enterprises on any communication technology stack, whether asynchronous or synchronous, in any format.

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

Creating a Business Service

You can create a business service for an SQL statement or a stored procedure.

Procedure How to Generate a Business Service

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 2-8.
2. Expand the node to display the statements or procedures.
3. Click the SQL statement or stored procedure for which you want to create a business service.
4. In the right pane, move the pointer over *Operations* and select *Create Integration Business Service*.

The Create Web Service information appears in the right pane.

5. Choose whether to create a new service or use an existing service.
 - If you select *Use an existing service*, a drop-down list appears from which you must select the service.
 - If you select *Create a new service*, the Create Web Service dialog box opens in the right pane:

Create Web Service for Param Car

Service Name:

Description:

License:
test

- a. In the Service Name field, type a name to identify the Web service (under the Service node in the left pane of the Integration Business Services tab).
 - b. In the Description field, type a brief description of the Web service.
 - c. In the License field, select the license(s) with which you want to associate this business service. To select more than one, hold down the *Ctrl* key and click the licenses.
6. Click *Next*.
 Another dialog box with the Method Name and Description fields opens.
 - a. In the Method Name field, type a name to specify the name of the SQL statement or stored procedure to be added to the business service.
 - b. In the Description field, type a brief description of the method.
7. Click *Finish*.

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

Testing a Business Service

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

Procedure How to Test a Business Service

1. If you are not on the Integration Business Services tab of Application Explorer, click the tab to access business services.
2. If it is not expanded, expand the list of business services under Integration Business Services.
3. Expand the *Services* node.

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

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

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

The test option appears in the right pane.

- If you are testing a Web service that requires XML input, an input field appears as shown by the following illustration:

Test

To test the operation using the [SOAP protocol](#), click the 'Invoke' button.



The screenshot shows a user interface for testing a business service. At the top, there is a label 'input xml:' followed by a large, empty text area with a vertical scrollbar on the right. Below this text area is a horizontal row of controls: a small, empty text input field, a 'Browse...' button, and three buttons labeled 'Upload', 'More', and 'Invoke'.

- If you are testing a Web service for a parameterized SQL statement, an input area appears where you can enter the parameter value, as shown by the following illustration:

Click [here](#) for a complete list of operations.

paramcardata

Test

To test the operation using the [SOAP protocol](#), click the 'Invoke' button.

Parameter	Value
param0:	<input type="text"/>

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

Application Explorer displays the results in the right pane:

A screenshot of the Application Explorer interface. The right pane displays XML data. The XML is a SOAP response with an envelope, body, and a specific response element containing RPCOut and Row elements. The first Row contains database information for SYSDATABASE, and the second Row contains information for SYSEXTENDED. The XML is color-coded: blue for tags, red for attributes and values, and black for text content.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance">
- <SOAP-ENV:Body>
  - <SPmethResponse
    xmlns="urn:iwaysoftware:ibse:jul2003:SPmeth:respo
    cid="16B82D8AAFC70DF1B04C2C80028A05C8">
    - <RPCOut reason="" status="success">
      - <Row>

        <DBLNAME>SYSDATABASE</DBLNAME>
        <EDAACCESS>R</EDAACCESS>
        <ISOLAT>C</ISOLAT>
        <IDENTIFY>N</IDENTIFY>
        <ENGINE>EDA</ENGINE>
        <DBDESCR>EDA Database
        Information Tables</DBDESCR>
      </Row>
    - <Row>

        <DBLNAME>SYSEXTENDED</DBLNAME>
        <EDAACCESS>R</EDAACCESS>
        <ISOLAT>C</ISOLAT>
        <IDENTIFY>N</IDENTIFY>
```

Generating WSDL From a Web Service

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

Procedure How to Generate WSDL From a Web Service

1. If you are not already in the Integration Business Services tab, click the tab to access business services.
2. In the left pane, expand the list of services to display the Web service for which you want to generate WSDL.
3. Click the Web service.

The link for the service appears in the right pane.

4. Right-click the *Service Description* link and choose *Save Target As*.
5. Choose a location for the file and specify .wsdl for the extension.

Note: The file extension must be .wsdl.

6. Click *Save*.

Example Viewing WSDL Generated from a Web Service

The following is an example of a WSDL file for a Web service called MPS generated from a parameterized SQL statement against a VSAM database.

```
<definitions xmlns:tns="urn:schemas-iwaysoftware-com:iwse"
targetNamespace="urn:schemas-iwaysoftware-com:iwse"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:m11="urn:iwaysoftware:ibse:jul2003:VSAM:response"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:m1="urn:iwaysoftware:ibse:jul2003:VSAM"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"><types><xs:schema
targetNamespace="urn:schemas-iwaysoftware-com:iwse"
elementFormDefault="qualified"><xs:element
name="ibsinfo"><xs:complexType><xs:sequence><xs:element type="xs:string"
name="service"/><xs:element type="xs:string" name="method"/><xs:element
type="xs:string" name="license"/><xs:element type="xs:string"
minOccurs="0" name="disposition"/><xs:element type="xs:string"
minOccurs="0" name="Username"/><xs:element type="xs:string" minOccurs="0"
name="Password"/><xs:element type="xs:string" minOccurs="0"
name="language"/></xs:sequence></xs:complexType></xs:element>
</xs:schema><xs:schema
targetNamespace="urn:schemas-iwaysoftware-com:iwse"
elementFormDefault="qualified"><xs:element
name="adapterexception"><xs:complexType><xs:sequence><xs:element
type="xs:string"
name="error"/></xs:sequence></xs:complexType></xs:element>
</xs:schema><xs:schema
targetNamespace="urn:iwaysoftware:ibse:jul2003:VSAM"
xmlns:m1="urn:iwaysoftware:ibse:jul2003:VSAM"
elementFormDefault="qualified"><xs:element
name="VSAM"><xs:complexType><xs:sequence><xs:element type="xs:string"
name="emp_id"/></xs:sequence></xs:complexType></xs:element>
</xs:schema><xs:schema
targetNamespace="urn:iwaysoftware:ibse:jul2003:VSAM:response"
xmlns:m11="urn:iwaysoftware:ibse:jul2003:VSAM:response"
elementFormDefault="qualified"><xs:element
name="VSAMResponse"><xs:complexType><xs:sequence><xs:element
name="RESULT"><xs:complexType><xs:sequence><xs:element
name="MPSVSAM"><xs:complexType><xs:sequence><xs:element minOccurs="0"
name="RESULTSET_1"><xs:complexType><xs:sequence><xs:element minOccurs="0"
name="ROW" maxOccurs="unbounded"><xs:complexType><xs:sequence><xs:element
type="xs:string" name="EMP_ID"/><xs:element type="xs:string"
name="FIRST_NAME"/><xs:element type="xs:string"
name="LAST_NAME"/><xs:element type="xs:string" name="DEPT"/><xs:element
type="xs:string"
```



```

name="COMP_NAME" /></xs:sequence></xs:complexType></xs:element></xs:sequence></xs:complexType></xs:element></xs:sequence></xs:complexType></xs:element>
</xs:sequence><xs:attribute type="xs:string" use="required"
name="cid" /></xs:complexType></xs:element></xs:schema>    </types><message
name="VSAMIn"><part element="m1:VSAM" name="parameters" />
</message><message name="VSAMOut"><part element="m11:VSAMResponse"
name="parameters" />    </message><message name="MPSHeader"><part
element="tns:ibsinfo" name="header" />    </message><message
name="AdapterException"><part element="tns:adapterexception"
name="fault" />
    </message><portType name="MPSSoap"><operation
name="VSAM"><documentation/><input message="tns:VSAMIn" /><output
message="tns:VSAMOut" /><fault message="tns:AdapterException"
name="AdapterExceptionFault" /></operation>
    </portType><binding type="tns:MPSSoap" name="MPSSoap"><soap:binding
style="document"
transport="http://schemas.xmlsoap.org/soap/http" /><operation
name="VSAM"><soap:operation style="document"
soapAction="MPS.VSAMRequest@production@@"/><input><soap:body
use="literal" /><soap:header part="header" message="tns:MPSHeader"
use="literal" />
    </input><output><soap:body use="literal" />
    </output><fault name="AdapterExceptionFault"><soap:fault
use="literal" name="AdapterExceptionFault" /></fault></operation>
    </binding><service name="MPS"><documentation>MPS</documentation><port
binding="tns:MPSSoap" name="MPSSoap1"><soap:address
location="http://iwayntk1:7001/ibse/IBSEServlet/XDSOAPRouter" /></port>
</service></definitions>

```

For more information on using WSDL in BEA WebLogic Workshop, including an example, see Appendix C, *Using WebLogic Workshop to Access VSAM*.

Identity Propagation

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

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

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

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

CHAPTER 3

Listening for Database Events

Topics:

- Understanding iWay Event Functionality
- Creating, Editing, or Deleting an Event Port
- Creating, Editing, or Deleting an Event Channel
- Choosing a Listening Technique
- Standard Event Processing With Row Tracking
- Standard Event Processing With Row Removal
- Trigger-based Event Processing

This section describes how to use the iWay XML Adapter for RDBMS for BEA WebLogic, deployed to a server such as BEA WebLogic Server, to listen for events in a relational table. Several listening techniques are available, enabling you to choose the technique that best suits your requirements.

Understanding iWay Event Functionality

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

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

- Port

A port associates a particular business object exposed by an adapter with a particular disposition. A disposition defines the protocol and resulting location of the event data. The port defines the end point of the event consumption. For more information, see *Creating, Editing, or Deleting an Event Port* on page 3-2.

- Channel

A channel represents configured connections to particular instances of back-end or other types of systems. A channel binds one or more event ports to a particular listener managed by an adapter. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 3-17.

Creating, Editing, or Deleting an Event Port

The following topics describe how to create, edit, or delete an event port using iWay Servlet Application Explorer.

Creating an Event Port From the Event Adapters Tab

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

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

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

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

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

- File
- JMSQ
- HTTP
- MQ Series

Procedure How to Create an Event Port for File

1. Click the *Event Adapters* tab.
The Event Adapters window opens.
2. In the left pane, expand the *RDBMS* node.
3. Select the *ports* node.

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

The Create New Port dialog box opens in the right pane containing fields to enter a name, description, disposition protocol, and disposition:

Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- a. Type a name and a brief description for the event port.
- b. From the Disposition Protocol drop-down list, select *FILE*.
- c. In the Disposition field, specify a destination file to which the event data is written.

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

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

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

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

Parameter	Description
location	Destination and file name of the document where event data is written. For example, D:\in\x.txt
errorTo	Location to which error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

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

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 3-17.

Procedure How to Create an Event Port for iBSE

1. Click the *Event Adapters* tab.

The Event Adapters window opens.

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

3. Select the *ports* node.

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

The Create New Port dialog box opens in the right pane containing fields to enter a name, description, disposition protocol, and disposition:

Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

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


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

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

Parameter	Description
svcName	Name of the service created with iBSE.
mthName	Name of the method created for the Web service.
responseTo	Location to which responses are posted. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.
errorTo	Location to which error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

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

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 3-17.

Procedure How to Create an Event Port for MSMQ

1. Click the *Event Adapters* tab.
The Event Adapters window opens.
2. In the left pane, expand the *RDBMS* node.
3. Select the *ports* node.

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

The Create New Port dialog box opens in the right pane containing fields to enter a name, description, disposition protocol, and disposition:

Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

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

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

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

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

Parameter	Description
machineName	Machine name where the Microsoft Queuing system is running.
qName	Name of the private queue where messages are placed.
errorTo	Location to which error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click **OK**.

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

You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 3-17.

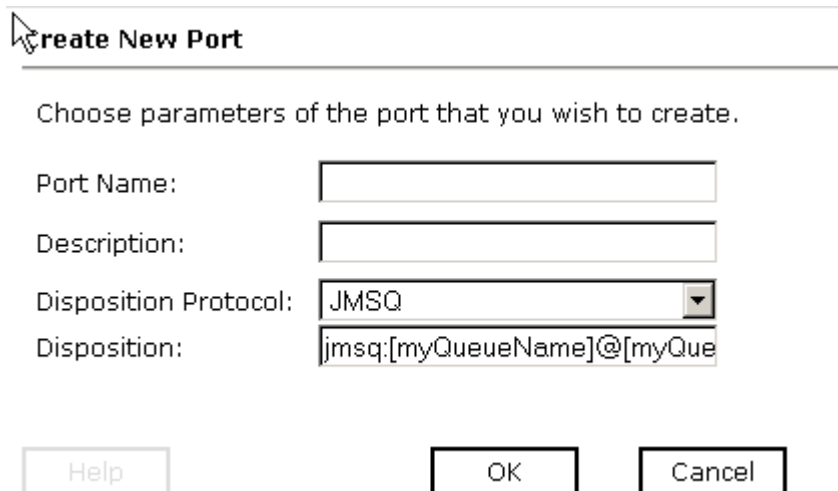
Procedure How to Create an Event Port for JMSQ

1. Click the *Event Adapters* tab.

The Event Adapters window opens.

2. In the left pane, expand the *RDBMS* node.
3. Select the *ports* node.
4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

The Create New Port dialog box opens in the right pane containing fields to enter a name, description, disposition protocol, and disposition:



Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

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

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

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

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

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

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

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

Parameter	Description
myQueueName or jmsqueue	JNDI name of a queue to which events are emitted.
myQueueFac or jmsfactory	Resource that contains information about the JMS Server. The WebLogic connection factory is: <code>javax.jms.QueueConnectionFactory</code>
jndiurl	URL to use to contact the JNDI provider. The syntax of this URL depends on the JNDI provider being used. This value corresponds to the standard JNDI property, <code>java.naming.provider.url.</code> The URL of the WebLogic Server is <code>t3://host:port</code> where: <code>host</code> Is the machine name where WebLogic Server is installed. <code>port</code> Is the port on which WebLogic Server is listening. The default port, if not changed at installation, is 7001.

Parameter	Description
jndifactory	Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider. For WebLogic Server, the WebLogic factory is: <code>weblogic.jndi.WLInitialContextFactory.</code>
user	Valid user name required to access a JMS server.
password	Valid password required to access a JMS server.
errorTo	Location to which error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

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

You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 3-17.

Procedure How to Create a Port for SOAP

1. Click the *Event Adapters* tab.
The Event Adapters window opens.
2. In the left pane, expand the *RDBMS* node.
3. Select the *ports* node.

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

The Create New Port dialog box opens in the right pane containing fields to enter a name, description, disposition protocol, and disposition:

Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

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

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

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

Parameter	Description
wSDL-url	<p>The URL to the WSDL file that is required to create the SOAP message. For example:</p> <p>http://localhost:7001/ibse/IBSEServlet/test/webservice.ibs?wsdl</p> <p>where:</p> <p>webservice</p> <p>Is the name of the Web service you created using Application Explorer.</p> <p>This value can be found by navigating to the Integration Business Services tab and opening the <i>Service Description</i> link in a new window. The WSDL URL appears in the Address field.</p> <p>You can also open the WSDL file in a third party XML editor (for example, XMLSPY) and view the SOAP request settings to find this value.</p>
soapaction	The method that will be called by the SOAP disposition. This value can be found in the WSDL file.
method	Web service method you are using. This value can be found in the WSDL file.
namespace	XML namespace you are using. This value can be found in the WSDL file.
responseTo	<p>Location to which responses are posted. Optional.</p> <p>A predefined port name or another disposition URL. The URL must be complete, including the protocol.</p>
errorTo	<p>Location to which error logs are sent. Optional.</p> <p>A predefined port name or another disposition URL. The URL must be complete, including the protocol.</p>

5. Click *OK*.

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

You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 3-17.

Procedure How to Create an Event Port for HTTP

1. Click the *Event Adapters* tab.
The Event Adapters window opens.
2. In the left pane, expand the *RDBMS* node.
3. Select the *ports* node.
4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

The Create New Port dialog box opens in the right pane containing fields to enter a name, description, disposition protocol, and disposition:

Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- a. Type a name and a brief description for the event port.
- b. From the Disposition Protocol drop-down list, select *HTTP*.
- c. In the Disposition field, type an HTTP destination.

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

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

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

<http://host:port/uri>

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

Parameter	Description
myurl	URL target for the post operation, for example, http://myhost:1234/docroot
responseTo	Location to which responses are posted. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

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

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

5. Click *OK*.

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

You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 3-17.

Procedure How to Create an Event Port for MQ Series

1. Click the *Event Adapters* tab.
The Event Adapters window opens.
2. In the left pane, expand the *RDBMS* node.
3. Select the *ports* node.
4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

The Create New Port dialog box opens in the right pane containing fields to enter a name, description, disposition protocol, and disposition:

Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- a. Type a name and a brief description for the event port.
- b. From the Disposition Protocol drop-down list, select *MQ Series*.
- c. In the Disposition field, type an MQ Series destination.

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

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

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

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

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

Parameter	Description
qManager	Name of the queue manager to which the server must connect.

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

5. Click **OK**.

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

You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 3-17.

Editing and Deleting an Event Port

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

Procedure How to Edit an Event Port

1. In the left pane, select the event port you want to edit.
2. In the right pane, move the pointer over *Operations* and select *Edit*.
The Edit Port dialog box opens.
3. Make the required changes and click **OK**.

Procedure How to Delete an Event Port

1. In the left pane, select the event port you want to delete.
2. In the right pane, move the pointer over *Operations* and select *Delete*.
A confirmation dialog box opens.
3. To delete the event port you selected, click *OK*.
The event port disappears from the list in the left pane.

Creating, Editing, or Deleting an Event Channel

The following topics describe how to create, edit, or delete a channel for your iWay Event. All defined event ports must be associated with a channel.

Creating a Channel

The following procedure describes how to create a channel using iWay Servlet Application Explorer.

Procedure How to Create a Channel

1. Click the *Event Adapters* tab.
The Event Adapters window opens. The adapters that appear in the left pane support events.
2. In the left pane, expand the *RDBMS* node.
The ports and channels nodes appear in the left pane.
3. Click the *channels* node.

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

The Add a new RDBMS channel dialog box opens in the right pane containing fields to enter a name, description, and channel type:

Add a new RDBMS channel

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

Channel Name:

Description:

Channel Type:

- a. Type a name (for example, NewChannel) and a brief description for the channel
 - b. From the drop-down list, select a channel type.
5. Click *Next*.

The Edit channels dialog box opens in the right pane containing listener parameters:

Edit channels

JDBC-ODBC
Bridge
Parameters

Oracle
Parameters

SQL Server
Parameters

EDA Server
Parameters

Host:

Port:

Database Name:

User:

Password:

Polling Interval:

SQL Query:

Post Query:

Delete Keys:

Help

< Back

Next >

Cancel

- a. Select either an Oracle, SQL Server, EDA Server, or JDBC-ODBC Bridge Listener.

Note: If you are configuring listening capabilities for a non-relational database, select the EDA Server Listener.

The following table lists and describes the parameters for all of the listeners.

Parameter	Description
Host	Name or URL of the machine where the database is installed.
Port	Port on which the Host database is listening.
Database Name <ul style="list-style-type: none"> For SQL Server and EDA Server Listener 	<ul style="list-style-type: none"> Database name of the database where the table specified in the SQL statement is located. <p>Note: When you access a non-relational database, and the server component is an SSCTL server component, the database name must be the service name and you must specify it. If the server component is installed on USS, you can leave the database field blank. For more information about the server component, see Chapter 1, <i>Introducing the iWay XML Adapter for RDBMS</i>.</p>
SID <ul style="list-style-type: none"> For Oracle Listener 	
Data Source <ul style="list-style-type: none"> For JDBC-ODBC Bridge Listener 	
User	Database user ID to access the table.
Password	Database password associated with the user ID.
Polling Interval	Interval, in milliseconds, at which to check for new input.

Parameter	Description
SQL Query	<p>SQL SELECT statement that the listener issues to poll the table.</p> <p>If the SQL statement includes a date column or long text column, you must provide a value for the SQL Post-query parameter. The value you provide must not contain a date column or a long text column. This applies whether you provide an SQL statement here or rely upon the default.</p> <p>For example, the following SELECT statement retrieves all unprocessed records from the DISCRETE_JOBS table:</p> <pre>SELECT * FROM WIP_DISCRETE_JOBS D WHERE DJ.WIP_ENTITY_ID > (SELECT WIP_ENTITY_ID FROM WIP.TEMP_NEW_WORK_ORDER_ENTITY_ID)</pre> <p>Important: When a SQL Query joins two or more tables, a SQL Post Query must be used. Also, do not use a semicolon at the end of a SQL statement for a SQL Query or a SQL Post Query.</p>

Parameter	Description
Post Query	<p>A SQL statement that is executed after each new record has been read from the table. This is case sensitive: the case used to specify the column names must match the case used in the SELECT statement that polled the table. If the SQL Query property was omitted so that a default SELECT statement polled the table, the case used to specify the column names must match the case used to define the columns in the DBMS's native schema.</p> <p>If you do not specify a value for SQL Post-query, each record read from the table will be deleted after it has been read. How this happens depends on whether you specify the Delete Keys property. If you:</p> <ul style="list-style-type: none"> Specify the Delete Keys property, by default the adapter issues a DELETE statement with a WHERE clause containing every key column specified for the Delete Keys property. <p>At run-time this will be faster than if you had not specified the Delete Keys property if there is an index on the key, or if there are fewer key columns than there are columns in the SELECT statement that polled the table.</p> Do not specify the Delete Keys property, by default the adapter issues a DELETE statement with a WHERE clause that specifies every column from the SELECT statement that polled the table. <p>You can choose to retain the table's data once it has been read by specifying a value for this parameter, as shown in the examples that follow.</p> <p>Note that the SQL Post-query and Delete Keys parameters are mutually exclusive, as Delete Keys applies to the default DELETE statement, and SQL Post-query overrides the default DELETE statement. You can provide a value for one or the other, but not for both.</p> <p>There are two field operators, ? and ^, that you can use in a post-query SQL statement; for more information, see <i>The Post-query Parameter Operators</i> on page 3-30.</p> <p>Important: When a SQL Query joins two or more tables, a SQL Post Query must be used. Also, do not use a semicolon at the end of a SQL statement for a SQL Query or a SQL Post Query.</p>

Parameter	Description
Delete Keys	<p>Comma-separated list of key columns to be used in the default DELETE statement. DELETE operates on keys, so specify the table's key columns.</p> <p>This is case sensitive: the case used to specify the column names must match the case used in the SELECT statement that polled the table. If the SQL Query property was omitted so that a default SELECT statement polled the table, the case used to specify the column names must match the case used to define the columns in the DBMS's native schema.</p> <p>Note that the Delete Keys and SQL Post Query parameters are mutually exclusive, as Delete Keys applies to the default DELETE statement, and SQL Post Query overrides the default DELETE statement. You can provide a value for one or the other, but not for both. For more information, see the description of the SQL Post-query parameter in this table.</p>

- b. Type the system information that is specific to the database on which you are listening based on the descriptions in the previous table.

6. Click *Next*.

The Select Ports dialog box opens in the right pane containing lists for available and current ports and buttons to enable you to move ports from one list to the other:

Select Ports

Available

rdbms1
rdbms2
rdbms3

Current

<<

<

>

>>

Help

< Back

Finish

Cancel

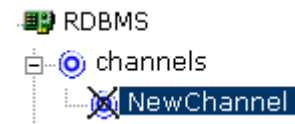
- a. Select an event port from the list of available ports. To select more than one, hold down the *Ctrl* key and click the ports.
 - b. Click the single right (>) arrow button to transfer the selected port(s) to the list of current ports. To transfer all event ports, click the double right (>>) arrow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

Summary information appears in the right pane:

Operations ▶	
Channel Description	NewChannel
Channel Status	Disconnected
Ports	[rdbms1, rdbms2]

The summary information provides the channel description, channel status, and current ports. All the information is associated with the channel you created.

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



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

Procedure How to Start and Stop a Channel

1. Expand the *Event Adapters* node.
2. Expand the *RDBMS* node.
3. Select the channel you want to start or stop.
4. To start the channel, move the pointer over *Operations* and select *Start the channel*.

The channel becomes active and the X over the icon disappears:



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

Editing and Deleting a Channel

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

Procedure How to Edit a Channel

1. Expand the *Event Adapters* node.
2. Expand the *RDBMS* node.
3. In the left pane, select the channel you want to edit.
4. In the right pane, move the pointer over *Operations* and select *Edit*.
The Edit channels dialog box opens.
5. Make the required changes to the channel configuration and click *Finish*.

Procedure How to Delete a Channel

1. Expand the *Event Adapters* node.
2. Expand the *RDBMS* node.
3. In the left pane, select the channel you want to delete.
4. In the right pane, move the pointer over *Operations* and select *Delete*.
A confirmation dialog box opens.
5. To delete the channel you selected, click *OK*.

The channel disappears from the list in the left pane.

Choosing a Listening Technique

You can detect an event in a relational or non-relational table and propagate it to other processes using a Table Listener.

An elaborate polling technology enables the specification of SQL SELECT statements to execute on a periodic basis. After data is polled, it passes through the adapter for further processing.

Note: Event processing may be limited for some non-relational databases due to the functionality of the database and its interaction with the iWay server component. For more information on the iWay server component, see the *iWay Data Adapter Administrator User's Guide* or consult with your DBA.

Choosing a Listening Technique

You can poll a relational or non-relational database directly and send the results to a file or JMS message queue. You also can use the following advanced techniques to listen to a database event.

- Standard event processing with row tracking

The listener polls a table, sends each newly inserted row to a destination you specify (known as the disposition), and uses a control table to track the row that was most recently read. The control table prevents the most recently read row from being read again during the next listening cycle.

You can apply this flexible yet simple technique in most situations.

For more information, see *Standard Event Processing With Row Tracking* on page 3-27.

- Standard event processing with row removal

The listener polls a table, sends each newly inserted row to a destination you specify, and then deletes the new row from the table to prevent it from being read again during the next listening cycle.

You apply this technique when the source table is used to pass data to the adapter, and the table rows are not required to persist. Rows are deleted as they are processed.

For more information, see *Standard Event Processing With Row Removal* on page 3-33.

- Trigger-based event processing

At design time, you assign triggers to a joined group of tables. At run time, the triggers write information about table changes to a common control table. The listener polls the control table and sends information about the table changes to a destination you specify. The listener deletes new rows from the control table to prevent them from being read again during the next listening cycle.

You apply this technique when listening for events in a group of large joined tables, or when you must know whether a row was updated or deleted.

For more information, see *Trigger-based Event Processing* on page 3-36.

Standard Event Processing With Row Tracking

The standard event processing with row tracking technique enables you to listen to the source table without removing its rows. It requires you to create a single-cell control table that tracks the last new row the Table Listener read from the source table.

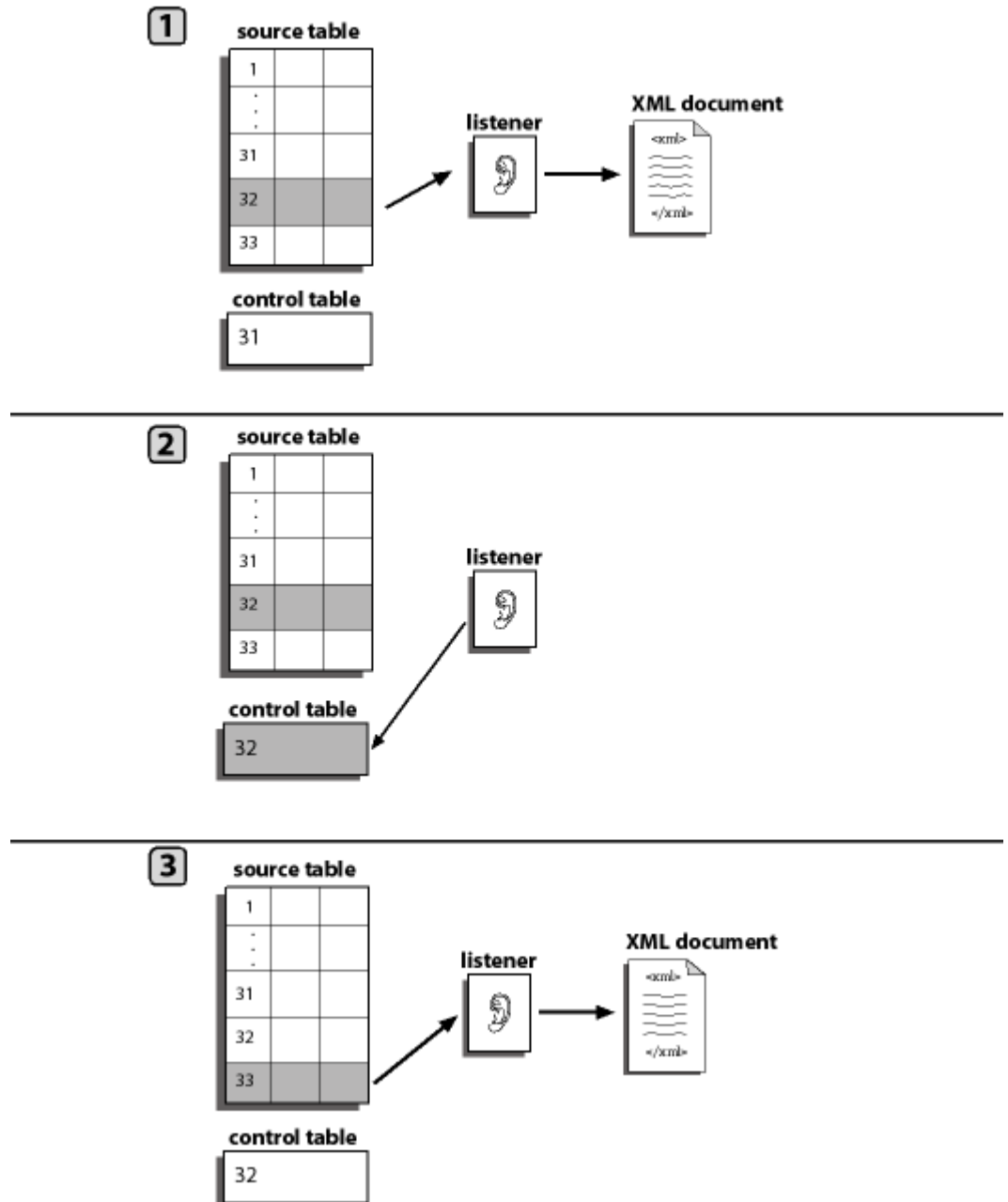
The single column of the control table corresponds to a column (or to a group of columns) in the source table that is unique, sortable, and indicates how recently the row was added to the source table relative to the other rows. For example, the first row added to the source table has the lowest value, and the last row added has the highest value. This value is called the *event key*.

When you create the control table, initialize it to the event key of the row most recently added to the source table. When you specify the listener properties, configure the SQL Post-query property of the listener to automatically update the control table event key.

Each time the listener queries the source table, it looks for rows added since the last query—that is, for rows whose event key is greater than the current value of the field in the control table. It reads each row of this type and returns it to the specific destination using an XML document. To ensure that the row is not read again the next time the listener queries the table, the listener updates the field in the control table to match the value of the row that was just read from the source table.

Note: Event processing may be limited for some non-relational databases due to the functionality of the database and its interaction with the iWay server component. For more information on the iWay server component, see the *iWay Data Adapter Administrator User's Guide* or consult with your DBA.

The following figure illustrates standard event processing with row tracking.



In the previous figure:

1. The listener queries the source table and copies each source table row whose event key is greater than the control table event key. The listener copies the row to an XML document and sends it to the destination defined in the port disposition using the File protocol.
2. The listener updates the event key in the control table to match the row it most recently read.
3. The listener copies the next source table row to an XML document.

The process repeats.

Procedure How to Implement Standard Event Processing With Row Tracking

To implement standard event processing with row tracking:

1. Create a control table. For an example, see *Creating the Control Table for an RDBMS (Oracle) Event* on page 3-29.
2. Configure an RDBMS Table Listener in the iWay Web Console.

In addition to the required listener properties for standard event processing with row tracking, you also must provide values for the following optional properties:

- **SQL Query**, the SQL SELECT statement that identifies the source table to which the adapter listens and with which it queries the table.
- **SQL Post-query**, the SQL statements that maintain the field in the control table.

For detailed instructions about configuring a listener, see *How to Create a Channel* on page 3-17. For information on post query parameters, see *The Post-query Parameter Operators* on page 3-30.

Example Creating the Control Table for an RDBMS (Oracle) Event

This example uses an Oracle E-Business Suite (also known as Oracle Applications) table. You can apply the same technique in a similar way to other types of relational databases.

You can follow the steps in this example to create an Oracle E-Business Suite table named TEMP_NEW_YORK_ORDER_ENTITY that has a single field named WIP_ENTITY_ID. You specify this table when you configure the RDBMS Table Listener, as described in *The Post-query Parameter Operators* on page 3-30.

When discrete jobs are created through the Oracle E-Business Suite graphical interface, an entry is created in the WIP.WIP_DISCRETE_JOBS table. For this example, you configure an event to detect new entries to this table. You use the standard event processing with row tracking technique. (Oracle E-Business Suite processing cannot delete rows from the table.)

You first create a simple table to track the records processed.

1. From within Oracle SQL*PLUS, run the following SQL:

```
CREATE TABLE WIP.TEMP_NEW_WORK_ORDER_ENTITY_ID
(
    WIP_ENTITY_ID    NUMBER
)
```

This creates a single table with a single field.

Note: Oracle SQL*Plus is part of the Oracle client software. If it is not installed, contact your Oracle Database Administrator.

You must be logged in under the APPS schema or a similar ID with access rights to the Oracle E-Business Suite WIP schema.

2. Create a single record in the table and provide it with the highest WIP_ENTITY_ID ID from your system.

You can obtain this ID from the WIP.WIP_DISCRETE_JOBS table.

This sets the value at which to start detecting events as records enter the WIP_DISCRETE_JOBS table.

3. After you create a simple table in Oracle, you must configure the listener.

Reference The Post-query Parameter Operators

When you configure a Table Listener, you can use two special field operators, ? and ^, with the SQL Post-query parameter. Both of these operators dynamically substitute database values in the SQL post-query statement at run time:

- ?fieldname is evaluated at run time as field = value.

The ? operator is useful in UPDATE statements:

```
UPDATE table WHERE ?field
```

For example, the following statement

```
UPDATE Stock_Prices_Temp WHERE ?RIC
```

might be evaluated at run time as:

```
UPDATE Stock_Prices_Temp WHERE RIC = 'PG'
```


- ^fieldname is evaluated at run time as value

The ^ operator is useful in INSERT statements:

```
INSERT INTO table VALUES (^field1, ^field2, ^field3, ... )
```

For example, the following statement

```
INSERT INTO Stock_Prices_Temp VALUES (^RIC, ^Price, ^Updated)
```

might be evaluated at run time as:

```
INSERT INTO Stock_Prices_Temp VALUES ('PG', 88.62, '2003-03-18
16:24:00.0')
```

Example Listening to trans_event Using the Row Tracking Technique

In this example, you listen to the trans_event table using the row tracking technique and use last_trans as the control table that contains the last value of the primary key read from trans_event.

For more information on configuring a listener, see *How to Create a Channel* on page 3-17.

last_trans is to contain a single value in a single row and must be set up prior to configuring the Table Listener. The last_trans column must have the same name as the primary key in the trans_event table. This key must be unique and sortable.

The table schemas for this example are:

```
SQL> describe trans_event
```

Name	Null?	Type

EVENT_ID	NOT NULL	NUMBER(38)
LAST_NAME		VARCHAR2(50)
TRANS_ID		CHAR(2)

```
SQL> describe last_trans
```

Name	Null?	Type

EVENT_ID		NUMBER

The last_trans single field value must contain the starting value of the primary key.

The listener generates XML response documents for each record found in the trans_event table with a primary key greater than the value found in the last_trans table.

1. Using a SQL query/data manipulation tool supplied by the database vendor, insert a record into the trans_event table based on the following information.
 - EVENT_ID=1
 - LAST_NAME='Kaplan'
 - TRANS_ID='03'

When setting up the port, a specific path is configured for a disposition using the File protocol. A response document with the record data is deposited into the directory after the insert is made.

The following is an example of a response document for the listener deposited into a directory specified when the Port is configured.

```
<Oracle>
  <row>
    <EVENT_ID>1</EVENT_ID>
    <LAST_NAME>Kaplan</LAST_NAME>
    <TRANS_ID>03</TRANS_ID>
  </row>
</Oracle>
```

2. Configure the listener by specifying the following properties when creating the channel.

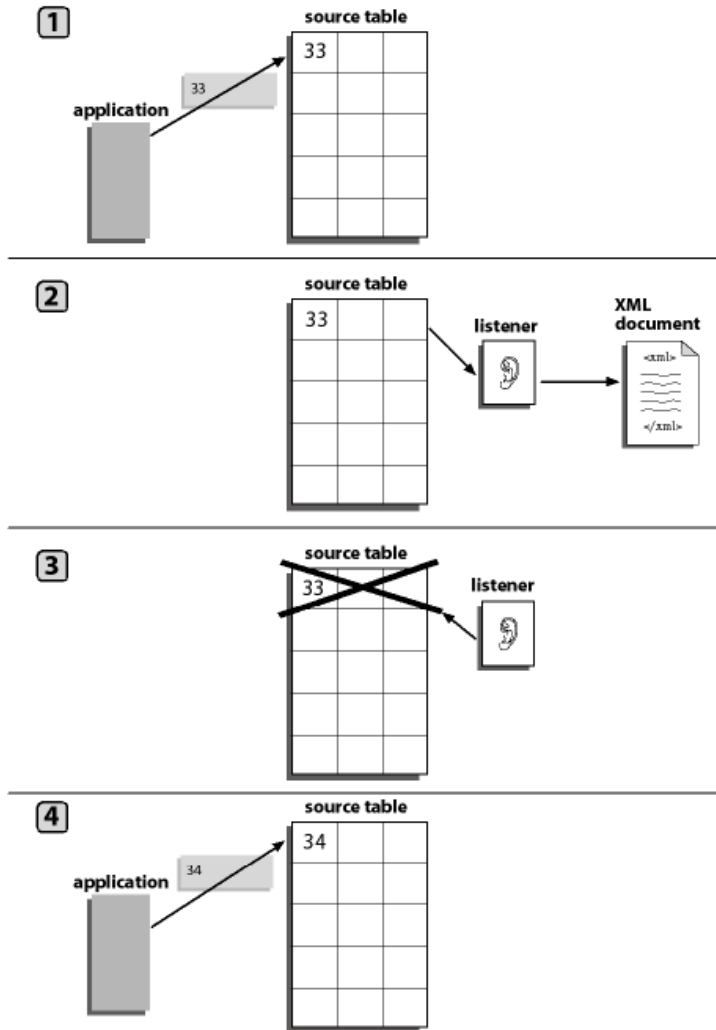
Parameter	Description
Host	Name or URL of the machine on which the database is installed.
Port	Port on which the Host database is listening.
User Name	User name that is registered with the back-end RDBMS.
Password	Password associated with the user name.
SQL Query	<code>SELECT * FROM TRANS_EVENT WHERE EVENT_ID>(select EVENT_ID from LAST_TRANS)</code>
Post Query	<code>UPDATE LAST_TRANS SET ?EVENT_ID</code>
Polling Interval	Interval in seconds.

Standard Event Processing With Row Removal

The standard event processing with row removal technique assumes that the source table is used to pass the data to the adapter and that the table rows are not required to persist. The Table Listener periodically queries the source table. When it finds a row, it reads it and returns it to the file disposition specified when the port is configured via an XML document. To ensure that the row is not read again when the listener next queries the table, the listener deletes the row from the table.

Note: Event processing may be limited for some non-relational databases due to the functionality of the database and its interaction with the iWay server component. For more information on the iWay server component, see the *iWay Data Adapter Administrator User's Guide* or consult with your DBA.

The following figure illustrates standard event processing with row removal.



In the previous figure:

1. Your application inserts a new row into the source table.
2. The listener queries the source table and copies the new row to an XML document and sends it to the destination defined in the port disposition using the File protocol.
3. The listener deletes the source table row to ensure that the row is not read again when the listener next queries the table.
4. The application inserts a new row into the source table.

The process repeats itself.

Procedure How to Implement Standard Event Processing With Row Removal

To implement the standard event processing with row removal technique:

1. Configure a Table Listener.
2. In addition to the required listener properties, provide values for the following optional properties:
 - SQL Query: the SQL SELECT statement that identifies the source table to which the adapter listens and with which it queries the table.
 - Post query: to identify the rows that the adapter automatically deletes from the table.

For detailed instructions about configuring a listener, see *How to Create a Channel* on page 3-17. For information on Post query parameters, see *The Post-query Parameter Operators* on page 3-30.

Example Listening to stock_prices Using the Row Removal Technique

In this example, you listen to the stock_prices table using the row removal technique.

```
SQL> describe stock_prices
```

Name	Null?	Type
-----	-----	-----
RIC	NOT NULL	VARCHAR2 (6)
PRICE		NUMBER (7, 2)
UPDATED		DATE

When a record is added to stock_prices, an XML document is generated with the contents of the record.

The location to which the document is saved is specified in the configuration of the port disposition property (using the File protocol) associated with this Table Listener.

After generating the document the record is deleted from the table.

1. Configure the listener by specifying the following properties when creating the channel.
 - a. In the Host field, provide the name or URL of the machine on which the database is installed.
 - b. In the Port field, provide the name of the port on which the Host database is listening.
 - c. In the User Name field, provide the user name that is registered with the back-end RDBMS.

- d. In the Password field, provide the Oracle Applications user ID authorized to access the Oracle Applications system.
- e. For the SQL Query, use `select * from stock_prices`.
- f. For the Post Query, use `delete from stock_price where ?RIC`.
- g. For Polling Interval, specify an interval in seconds.

For a description of these properties, see *The Post-query Parameter Operators* on page 3-30.

- 2. For more information on configuring a listener, see *How to Create a Channel* on page 3-17.

Trigger-based Event Processing

Trigger-based event processing is a technique for listening to multiple joined relational tables. You also can use it to detect when a row was deleted or updated.

The trigger-based technique provides the following benefits:

- Improves performance when listening for events in a group of large joined tables.

When processing joined tables, the database system creates a Cartesian product working table. When the joined tables are large, the interim working table is very large. The standard technique of processing database events, in which the adapter periodically listens to the entire structure of joined tables, can consume a significant amount of computing resources.

The trigger-based technique avoids this overhead by requiring the Table Listener to query a single small control table and by writing to the control table only when an event actually occurs.

- Increases the number of event types that the adapter recognizes.

Using the trigger-based technique, you can tell when a row was updated, deleted, or inserted. Using the standard technique, you can tell only when a row was inserted.

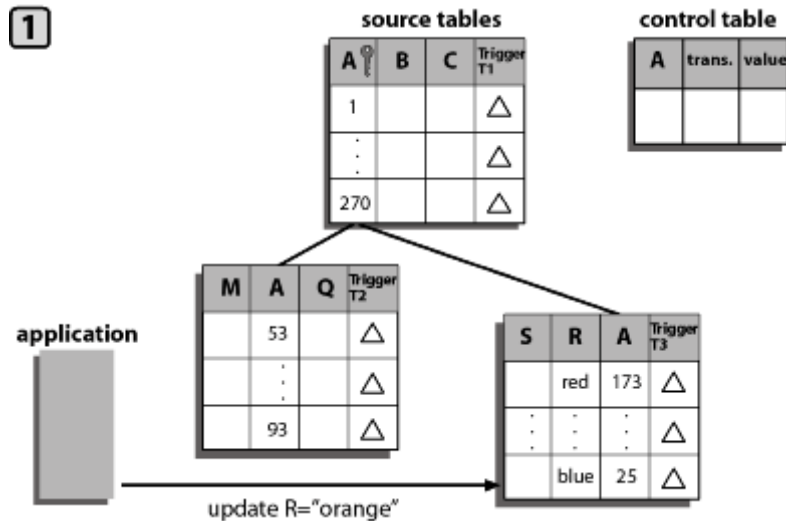
To use the trigger-based technique, you assign a trigger to each table that you want to monitor. When a value changes, it fires the corresponding trigger that writes data to a control table. The iWay XML Adapter for RDBMS for BEA WebLogic listens to the control table by running a query against it. When it finds a row in the control table, it reads it and returns it to the port disposition created when the port is configured via an XML document. To ensure the row is not read again when the listener next queries the table, the listener deletes the row from the table.

The trigger-based technique enables you to recognize changes to an entity. For the purposes of this discussion, an entity is a real-world object that is represented in the database by a hierarchical set of tables.

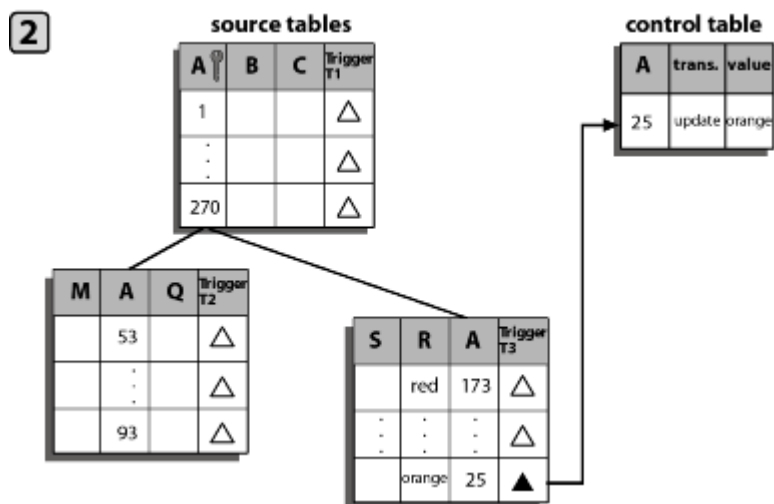
You manage the triggers using a native RDBMS tool (such as SQL*Plus for Oracle tables) and configure the listener using the iWay Web Console.

Note: Event processing may be limited for some non-relational databases due to the functionality of the database and its interaction with the iWay server component. For more information on the iWay server component, see the *iWay Data Adapter Administrator User's Guide* or consult with your DBA.

The following figures illustrate trigger-based event processing:

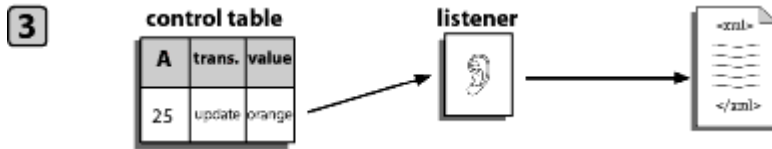


1. Your application updates a row in a group of related source tables. The update causes a row trigger to fire in the changed table.

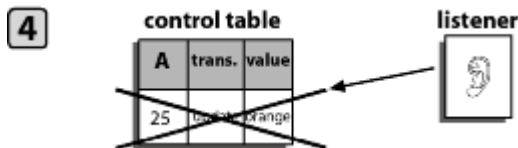


- The trigger inserts a row into the control table.

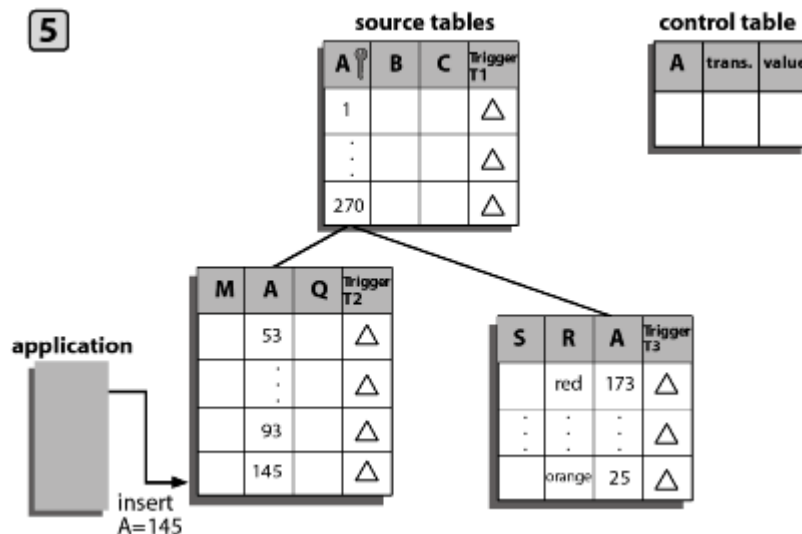
The new control table row includes the key value (25), the type of transaction (update), and the new cell value (orange).



- The listener queries the control table and copies the new row to an XML document. It sends the document to the Reply_to destination.



- The listener deletes the control table row to ensure that the row is not read again when the listener next queries the table.



- The application inserts a new row into one of the source tables. The process repeats itself.

Procedure How to Implement Trigger-based Event Processing

To implement the trigger-based event processing technique:

1. Create the control table.

The purpose of the control table is to capture the key of each entity that changed, regardless of which entity table changed.

You can store a variety of information in the control table, including the key of the entity that was inserted, updated, or deleted and the name of the table and field that was updated.

The design of the control table is a function of the business logic of your application. For example, you can choose between creating one control table for a group of joined source tables or one control table per source table. Among the issues to consider are the kinds of events to monitor (insertions, deletions, or updates), and whether you want to monitor only the highest-level table in a group of joined tables or all of the tables in the group.

2. Assign triggers to the source tables.

The triggers you assign, and to which tables you assign them, is determined by what kind of change you want to monitor. The triggers implement event-processing logic. For a sample trigger, see *Trigger on WIP_ENTITY_NAME Column in an Oracle Table* on page 3-41.

For example, consider a bill of materials scenario. (A bill of materials is a list of all the parts required to manufacture an item, the subparts required for the parts, and so on. The complete item/parts/subparts relationship can extend to several levels, creating a data structure like a tree with the finished item as the root.) In a bill of materials, each level in the parts hierarchy is represented by a separate table. You might assign a trigger to only the highest-level table (the finished product), or you might assign triggers to all tables (the finished product and its parts and subparts).

If multiple changes are made to the same row during one listener cycle, you could configure the event adapter to record all the changes. If a row was inserted and then updated, both changes are logged.

3. Configure the listener when creating a channel in the Application Explorer console.

In addition to the required listener properties, for trigger-based event processing you also must provide values for the following optional properties:

- SQL Query: the SQL SELECT statement that identifies the control table to which the adapter listens and with which it queries the table to determine changes in the source tables.
- Post query: to identify the rows that the adapter automatically deletes from the control table.

For detailed instructions about configuring a listener, see *How to Create a Channel* on page 3-17. For information about Post query operators, see *The Post-query Parameter Operators* on page 3-30.

Example Trigger on WIP_ENTITY_NAME Column in an Oracle Table

The following trigger fires when a change is made to the WIP_ENTITY_NAME column of the WIP.WIP_ENTITIES Oracle E-Business Suite table. When it fires, the trigger writes the relevant values to the control table IWAY.IWAY_PO_CDC.

```
CREATE OR REPLACE TRIGGER IWAY.IWAY_PO_CDC_WE_TRG

AFTER INSERT OR DELETE OR UPDATE OF WIP_ENTITY_NAME
ON WIP.WIP_ENTITIES
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO IWAY.IWAY_PO_CDC
      VALUES (
        :NEW.WIP_ENTITY_ID,
        :NEW.ORGANIZATION_ID,
        'UPDATE' );
  ELSE
    INSERT INTO IWAY.IWAY_PO_CDC
      VALUES (
        :OLD.WIP_ENTITY_ID,
        :OLD.ORGANIZATION_ID,
        'UPDATE' );
  END IF;

  EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
      NULL;          -- Record already exists

END;
```

CHAPTER 4

Using Web Services Policy-Based Security

Topics:

- Integration Business Services Policy-Based Security
- Configuring Integration Business Services Policy-Based Security

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

Note: For the iWay 5.5 RG2 Release, it is recommended that policy-based security not be enabled.

Integration Business Services Policy-Based Security

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

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

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

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

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

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

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

Configuring Integration Business Services Policy-Based Security

Before you create instances of policies, you must have a minimum of one user or one group to associate to an instance. You can create users and groups using Servlet Application Explorer. For more information, see *How to Create a User to Associate With a Policy* on page 4-3 or *How to Create a Group to Associate With a Policy* on page 4-5.

An execution policy governs who can execute the business service to which the policy is applied. For more information, see *How to Create an Execution Policy* on page 4-7.

You configure the IP and Domain Restriction policy type slightly differently from other policy types. The IP and Domain Restriction policy type controls connection access to iBSE and therefore, need not be applied to an individual business service. You need not create a policy, however, you must enable the Security Policy option in Servlet Application Explorer. For more information, see *How to Configure IP and Domain Restrictions* on page 4-10.

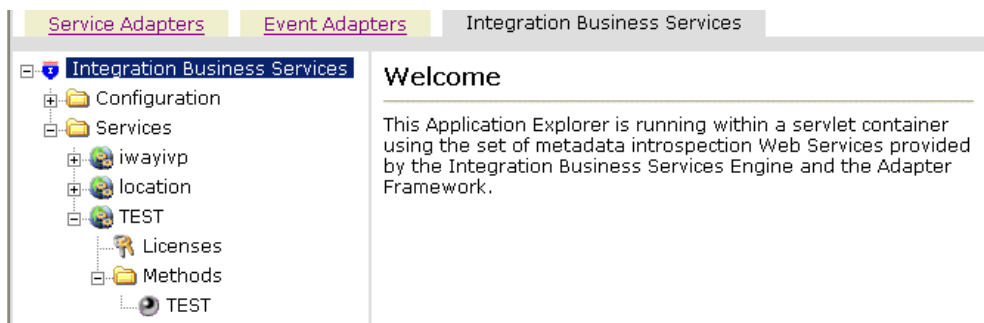
Note: For the iWay 5.5 RG2 Release, it is recommended that policy-based security not be enabled.

Procedure How to Create a User to Associate With a Policy

To create a user to associate with a policy:

1. Open *Servlet Application Explorer*.

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



- a. Click the *Integration Business Services* tab.
- b. Expand the *Configuration* node.
- c. Expand the *Security* node.

- d. Expand the *Users and Groups* node.
 - e. Select *Users*.
 2. In the right pane, move the pointer over *Operations* and select *Add*.

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

Add a new user

Name:

Password:

Description:

- a. In the Name field, type a user ID.
 - b. In the Password field, type the password associated with the user ID.
 - c. In the Description field, type a description of the user (optional).
 3. Click *OK*.

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

Operations ►



Users

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

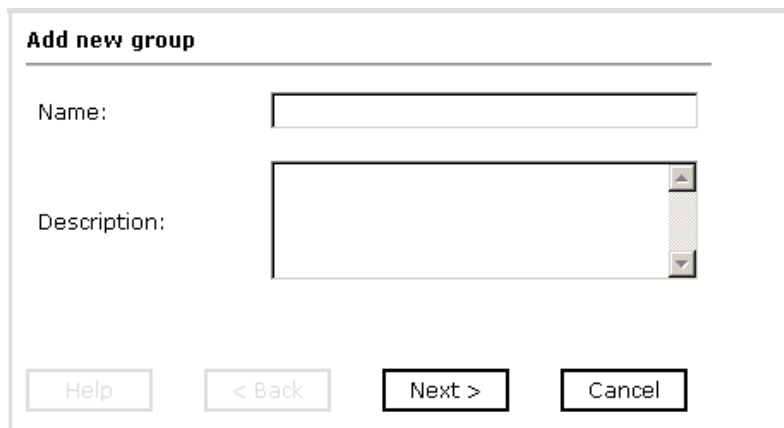
User Id	Description
<input type="checkbox"/> bse1	

Procedure How to Create a Group to Associate With a Policy

To create a group to associate with a policy:

1. Open *Servlet Application Explorer*.
 - a. Click the *Integration Business Services* tab.
 - b. Expand the *Configuration* node.
 - c. Expand the *Security* node.
 - d. Expand the *Users and Groups* node.
 - e. Select *Groups*.
2. In the right pane, move the pointer over *Operations* and click *Add*.

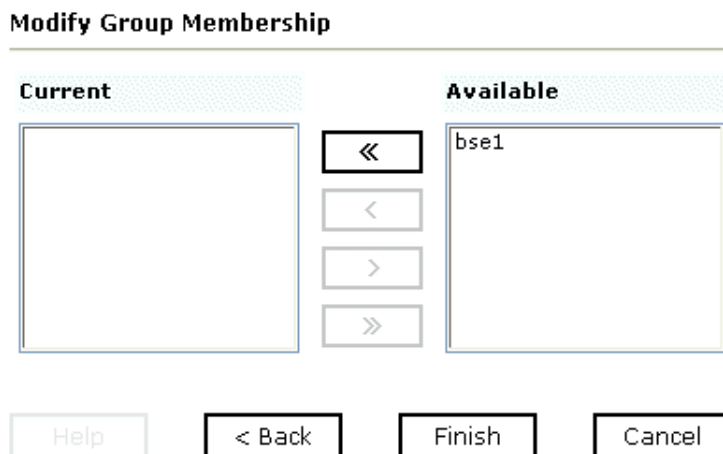
The following image shows the Add new group pane that opens with fields where you enter a name and a description for the group. To continue after typing inputs, click the *Next* button. The pane also includes a *Help* button, a *Back* button to return to the previous screen, and a *Cancel* button to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



The image shows a window titled "Add new group". It contains two input fields: "Name:" with a single-line text box, and "Description:" with a multi-line text box. At the bottom, there are four buttons: "Help", "< Back", "Next >", and "Cancel".

- a. In the Name field, type a name for the group.
 - b. In the Description field, type a description for the group (optional).
3. Click *Next*.

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



The image shows a window titled "Modify Group Membership". It features two list boxes: "Current" on the left and "Available" on the right. The "Available" list box contains the text "bse1". Between the two list boxes are four arrow buttons: a double left arrow (<<), a single left arrow (<), a single right arrow (>), and a double right arrow (>>). At the bottom, there are four buttons: "Help", "< Back", "Finish", and "Cancel".

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

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

The new group is added.

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

Operations ►



Groups

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

Group name	Description
<input type="checkbox"/> newgroup	

Procedure How to Create an Execution Policy

To create an execution policy:

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

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



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

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

The form is titled 'Add a new policy'. It contains three input fields: 'Name:' with a text box, 'Type:' with a drop-down menu showing 'Execution', and 'Description:' with a text area. At the bottom, there are four buttons: 'Help', '< Back', 'Next >', and 'Cancel'.

- a. In the Name field, type a a name for the policy.
- b. From the Type drop-down list, select *Execution*.
- c. In the Description field, type a description for the policy (optional).

3. Click *Next*.

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

4. Select a minimum of one user or group from the Available pane.

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

5. Click *Next*.

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

Member Id	Permission
user.ibse1	Deny
group.ibse_group	Deny

Buttons: Help, < Back, Finish, Cancel

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

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

Operations ▶

Policies

You can configure policies for the Integration Business Services Engine to manage resource execution, service routing, data restrictions and failover/recovery actions.

Name	Type	Description
<input type="checkbox"/> ibse_policy	Execution	

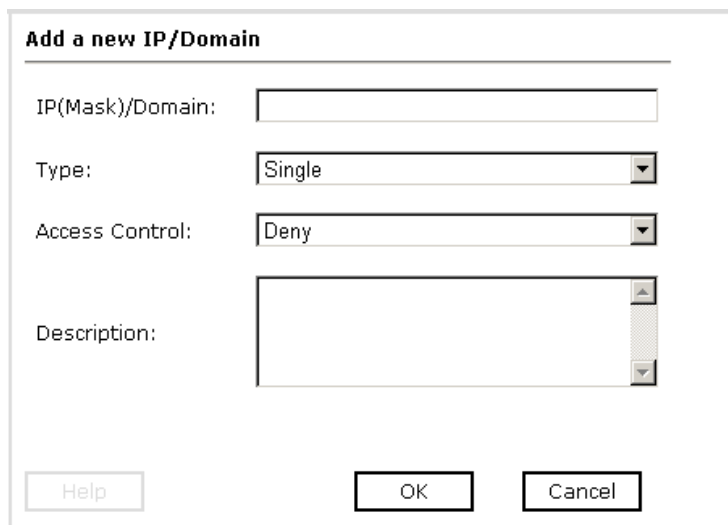
Procedure How to Configure IP and Domain Restrictions

To configure IP and domain restrictions:

1. Open *Servlet Application Explorer*.

- a. Select the *Integration Business Services* tab.
 - b. Expand the *Configuration* node.
 - c. Expand the *Security* node.
 - d. Select *IP and Domain*.
2. In the right pane, move the pointer over *Operations* and click *Add*.

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



Add a new IP/Domain

IP(Mask)/Domain:

Type:

Access Control:

Description:

- a. From the Type drop-down list, select the type of restriction.
- b. In the IP(Mask)/Domain field, type the IP or domain name using the following guidelines.
 - If you select Single (Computer) from the Type drop-down list, you must provide the IP address for that computer. If you only know the DNS name for the computer, click *DNS Lookup* to obtain the IP Address based on the DNS name.
 - If you select Group (of Computers), you must provide the IP address and subnet mask for the computer group.
 - If you select Domain, you must provide the domain name, for example, yahoo.com.

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

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

Operations ►



IP and Domain

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

IP(Mask) / Domain	Access	Description
<input type="checkbox"/> test	Deny	

CHAPTER 5

Management and Monitoring

Topics:

- Managing and Monitoring Services and Events Using iBSE
- Managing and Monitoring Services and Events Using the JCA Test Tool
- Setting Engine Log Levels
- Configuring Connection Pool Sizes
- Migrating Repositories
- Exporting or Importing Targets
- Retrieving or Updating Web Service Method Connection Information
- Starting or Stopping a Channel Programmatically

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

Managing and Monitoring Services and Events Using iBSE

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

The following monitoring levels are available for services:

- System
- Service
- Method

The following monitoring levels are available for events:

- System
- Channel
- Port

Procedure: How to Configure Monitoring Settings

To configure monitoring settings:

1. Ensure that your BEA WebLogic Server is started.
2. To access the monitoring console, enter the following URL in your Web browser:

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

where:

[localhost](#)

Is the machine where the application server is running.

[port](#)

Is the HTTP port for the application server.

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

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

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

iBSE Settings:		Save
Property Name	Property Value	
System		
Language	English ▾	
Adapter Lib Directory	C:\Program Files\iWay55\lib	
Encoding	UTF-8 ▾	
Debug Level	NONE ▾	
Number of Async. Processors	0 ▾	
Security		
Admin User	iway	
Admin Password	****	
Policy	<input type="checkbox"/>	
Repository		
Repository Type	File System ▾	
Repository Url	file://C:\Program Files\iWay55\bea\ibse	
Repository Driver		
Repository User		
Repository Password		
Repository Pooling	<input type="checkbox"/>	
More configuration...		
		Save

3. Click *More configuration*.

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

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

where:

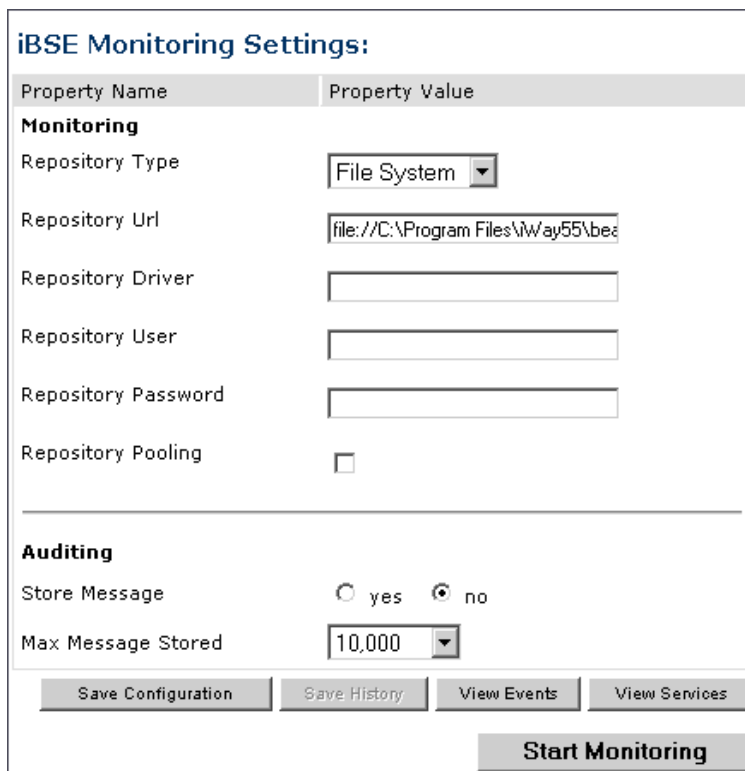
localhost

Is the machine where the application server is running.

port

Is the HTTP port for the application server.

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



The image shows a window titled "iBSE Monitoring Settings:". It is divided into two main sections: "Monitoring" and "Auditing".

Monitoring Section:

- Property Name:** Repository Type
- Property Value:** File System (selected in a dropdown)
- Property Name:** Repository Url
- Property Value:** file:///C:/Program Files/iWay55/bes
- Property Name:** Repository Driver
- Property Value:** (empty text field)
- Property Name:** Repository User
- Property Value:** (empty text field)
- Property Name:** Repository Password
- Property Value:** (empty text field)
- Property Name:** Repository Pooling
- Property Value:** ☐

Auditing Section:

- Property Name:** Store Message
- Property Value:** ☐ yes ☒ no
- Property Name:** Max Message Stored
- Property Value:** 10,000 (selected in a dropdown)

Buttons:

- Save Configuration
- Save History (disabled)
- View Events
- View Services
- Start Monitoring

- a. In the Monitoring pane, from the Repository Type drop-down list, select the type of repository you are using.
- b. To connect to the database in the Repository Url field, type a JDBC URL.
- c. To connect to the database in the Repository Driver field, type a JDBC Class.
- d. To access the monitoring repository database, type a user ID and password.
- e. To enable pooling, click the *Repository Pooling* check box.
- f. In the Auditing pane, select *yes* if you want to store messages.

This option is disabled by default.

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

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

By default, 10,000 is selected.

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

- h. Click *Save Configuration*.

4. Click *Start Monitoring*.

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

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

Procedure: How to Monitor Services

To monitor services:

1. Ensure that your BEA WebLogic Server is started.
2. From the iBSE Monitoring Settings window, click *Start Monitoring*.
3. Click *View Services*.

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

The screenshot shows a window titled "Service Statistics". It is divided into two main sections: "Web Service Methods" and "Statistics".

Web Service Methods

Service	Method
all	

Statistics

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

At the bottom right of the window is a button labeled "< home".

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

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

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

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

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

The screenshot shows a window titled "Service Statistics". It is divided into two main sections: "Web Service Methods" and "Statistics".

Web Service Methods: This section contains two drop-down lists. The "Service" list on the left has "B0100033" selected. The "Method" list on the right has "all methods" selected.

Statistics: This section contains a table with the following data:

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

At the bottom right of the window, there are two buttons: "Suspend Service" and "< home".

- a. To stop a service at any time, click *Suspend Service*.
- b. To restart the service, click *Resume Service*.
5. Select a method for the service from the Method drop-down list.

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

Service Statistics

Web Service Methods

Service
Method

B0100033
GetEffectiveAddress

Statistics

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

Suspend Service
< home

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

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

The screenshot shows a web application window titled "Invocation Statistics". It contains three main sections: "Message Information", "Client Information", and "Detail".

Message Information

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

Client Information

Client IP	127.0.0.1
Client Host Name	127.0.0.1
User Name	

Detail

Message	Size
Request Message	409 bytes
Response Message	665 bytes

In the bottom right corner, there is a button labeled "< home".

7. To view the XML request document in your Web browser, click *Request Message*.
You can also view the XML response document for the service.
8. To return to the iBSE Monitoring Settings window, click *home*.

Procedure: How to Monitor Events

To monitor events:

1. Ensure that your BEA WebLogic Server is started.
2. In the iBSE Monitoring Settings window, click *Start Monitoring*.
3. Click *View Events*.

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

Channel Statistics

Channels

Channels: Ports:

Statistics

Total Event Count	4
Total Success Count	3
Total Error Count	1
Average Event Size	337.0 bytes
Average Event Reply Size	na
Average Delivery Time	1274.0 ms
Last Delivery Time	250 ms
Successful Events	<input type="text" value="select a correlation id"/>
Failed Events	<input type="text" value="select a correlation id"/>

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

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

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

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

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

Channel Statistics

Channels

Channels: TestChan Ports: all

Statistics

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

Suspend Channel Start Channel

< home

- a. To stop a channel at any time, click *Suspend Channel*.
 - b. To start the channel, click *Start Channel*.
5. From the Ports drop-down list, select a port for the channel.

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

The image shows a software window titled "Channel Statistics". It is divided into two main sections: "Channels" and "Statistics".

Channels Section:

- Contains two labels: "Channels" and "Ports".
- Under "Channels" is a drop-down menu showing "TestChan".
- Under "Ports" is a drop-down menu showing "TestPort".

Statistics Section:

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

At the bottom right of the window, there are two buttons: "Suspend Channel" and "Start Channel". Below these buttons is a button labeled "< home".

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

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

Message Statistics

Message Information

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

Messages

Detail	size
Event Message	446 bytes
Reply Message	na

< home

- a. To view the XML event document in your Web browser, click *Event Message*.
- b. To return to the iBSE Monitoring Settings window, click *home*.

Managing and Monitoring Services and Events Using the JCA Test Tool

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

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

To manage and monitor services using the JCA Test Tool:

1. Open a Web browser to:

<http://localhost:port/iwjcaivp>

where:

[localhost](#)

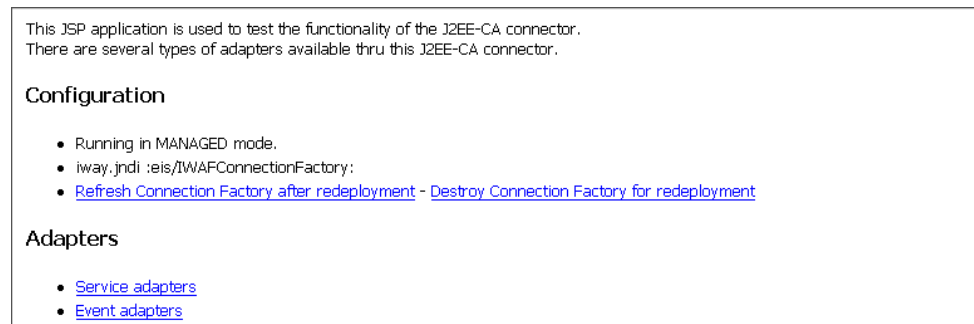
Is the name of the machine where your application server is running.

[port](#)

Is the port for the domain you are using. The port for the default domain is 7001.,for example:

<http://localhost:7001/iwjcaivp>

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



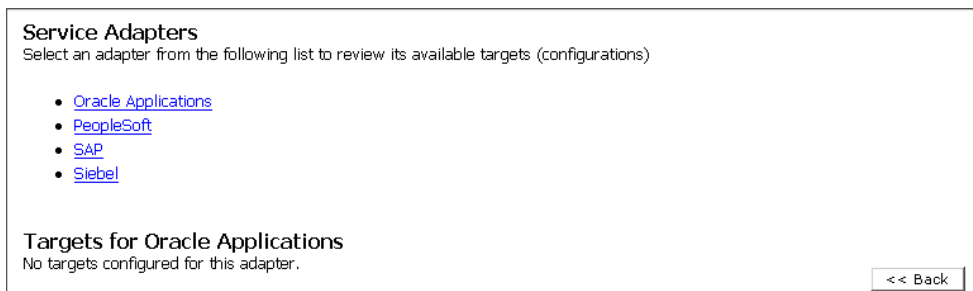
The JCA Test Tool runs in managed mode by default.

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

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

- a. Click *Destroy Connection Factory* for redeployment.
 - b. Redeploy the JCA connector module using the BEA WebLogic Server console.
 - c. In the JCA Test Tool, click *Refresh Connection Factory after redeployment*.
3. Click *Service adapters*.

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



4. Select a service adapter to monitor.

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

The screenshot displays the JCA Test Tool interface with the following sections:

- Service Adapters**
Select an adapter from the following list to review its available targets (configurations)
 - [Oracle Applications](#)
 - [PeopleSoft](#)
 - [SAP](#)
 - [Siebel](#)
- Targets for Siebel**
 - [TestService](#)
- Statistics for Siebel target TestService**

TotalRequestCount	: 0
TotalSuccessCount	: 0
TotalErrorCount	: 0
AverageExecutionTime	: 0 msec.
LastExecutionTime	: 0 msec.
- Request for Siebel target TestService**

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

User:

Password:

Input Doc:

- a. Click the desired target for your service adapter.
 - b. In the Request area, enter a user name and password.
 - c. In the Input Doc area, enter a request document that was created from the request schema for your service.
5. Click *Send*.

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

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

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

To manage and monitor events using the JCA Test Tool:

1. Open a Web browser to:

<http://localhost:port/iwjcaivp>

where:

[localhost](#)

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

[port](#)

Is the port for the domain you are using. The port for the default domain is 7001, for example:

<http://localhost:7001/iwjcaivp>

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

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

Configuration

- Running in MANAGED mode.
- `iway.jndi :eis/IWAFConnectionFactory`:
- [Refresh Connection Factory after redeployment](#) - [Destroy Connection Factory for redeployment](#)

Adapters

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

The JCA Test Tool runs in managed mode by default.

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

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

- a. Click *Destroy Connection Factory for redeployment*.
 - b. Redeploy the JCA connector module using the BEA WebLogic Server console.
 - c. In the JCA Test Tool, click *Refresh Connection Factory after redeployment*.
3. Click *Event adapters*.

The Event Adapters page opens.

4. Select the event adapter to monitor.
5. Click the desired channel for your event adapter.
6. Click *start*.

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

Current channel Statistics	
Commands: start refresh	
Active: false	
TotalRequestCount	: 0
TotalSuccessCount	: 0
TotalErrorCount	: 0
AverageExcecutionTime	: 0 msec.
LastExcecutionTime	: 0 msec.
Statistics for port 'fileIN'	
TotalRequestCount	: 0
TotalSuccessCount	: 0
TotalErrorCount	: 0
AverageExcecutionTime	: 0 msec.
LastExcecutionTime	: 0 msec.

Setting Engine Log Levels

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

Procedure: How to Enable Tracing for Servlet iBSE

To enable tracing for Servlet iBSE:

1. Open the Servlet iBSE configuration page at:

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

where:

`localhost`

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

`port`

Is the port for the domain you are using. The port for the default domain is 7001, for example:

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

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

The default location for the trace information on Windows is:

`C:\Program Files\bea\ibse\ibselogs`

Procedure: How to Enable Tracing for JCA

To enable tracing for JCA:

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

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

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

For example:

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

A directory in the configuration directory contains the logs.

- a. Review the logs generated by your application server.
 - b. Leave the remainder of the previous file unchanged.
3. Save the file and exit the editor.
4. Redeploy the connector.

Configuring Connection Pool Sizes

The following topic describes how to configure connection pool sizes for the JCA connector.

Procedure: How to Configure Connection Pool Sizes

To configure connection pool sizes:

1. Open the extracted ra.xml file in a text editor.
2. Locate and change the following setting:
pool-params. The JCA Resource Connector has an initial capacity value of 0 by default and cannot be changed. The maximum capacity value is 10 by default and can be changed to a higher value.

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

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

Migrating Repositories

During design time, a repository is used to store metadata created when using Application Explorer to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. For more information on configuring repositories, see the *iWay Installation and Configuration for BEA WebLogic* documentation.

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

File Repositories

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

```
drive:\Program Files\iWay55\bea\ibse\ibserrepo.xml
```

where:

drive

Is the location of your iWay 5.5 installation.

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

```
drive:\ProductionConfig\bea\ibse\ibserrepo.xml
```

iBSE Repositories

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

Note: The following procedure allows you to migrate only Web services. If migrating event handling information is one of your requirements, you must migrate at the database level. For more information, see *Migrating Event Handling Configurations* on page 5-28.

Procedure: How to Migrate an iBSE Repository Configured for Oracle

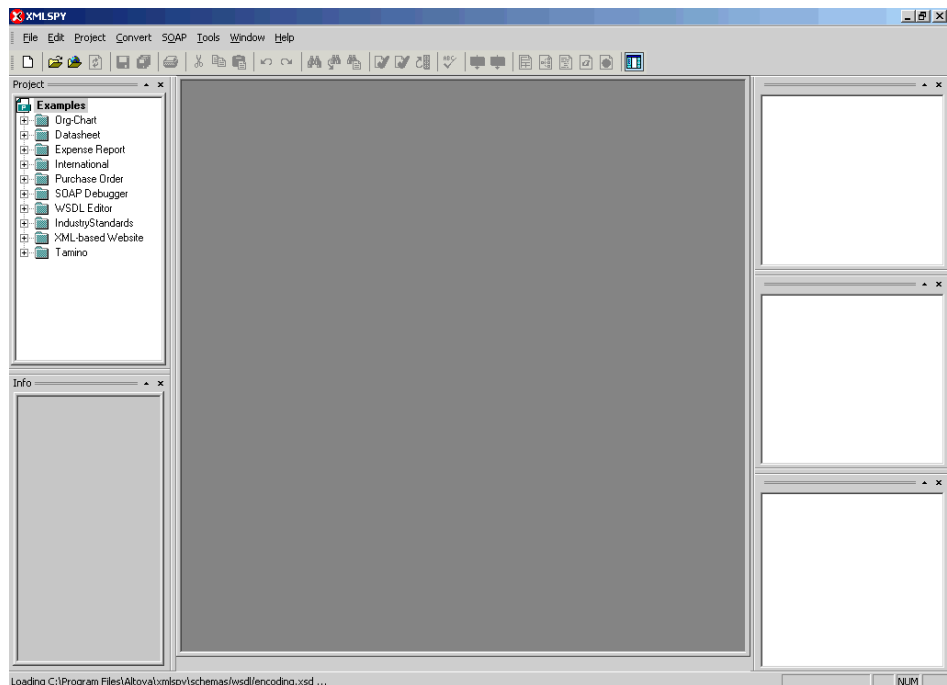
To migrate an iBSE repository that is configured for Oracle:

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

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

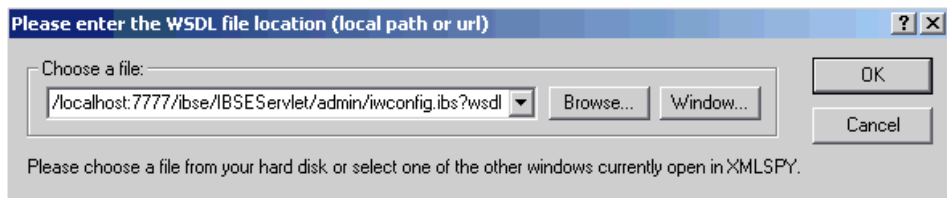
2. Open a third party XML editor, for example, XMLSPY.

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



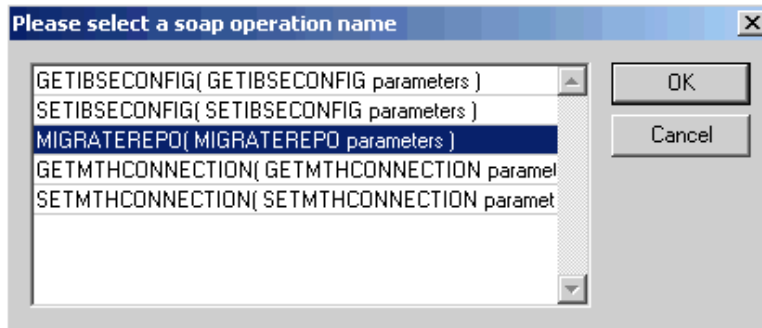
3. From the SOAP menu, select *Create new SOAP request*.

The following image shows the WSDL file location dialog box that opens, where you enter a local path or URL. The dialog includes Browse, Window, OK, and Cancel buttons.



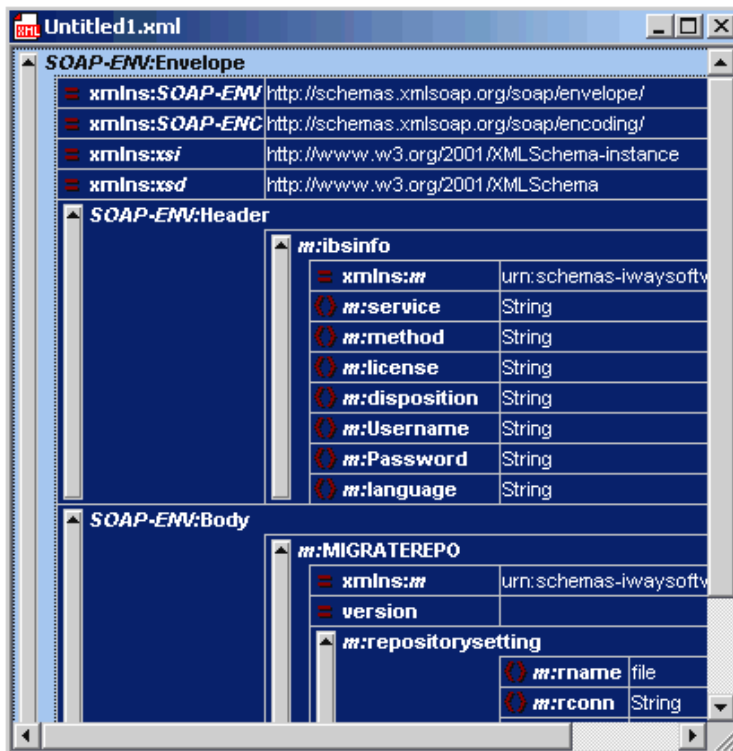
4. In the Choose a file field, paste the iBSE configuration service URL.
5. Click OK.

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



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

The following image shows a portion of the window that opens with the structure of the SOAP envelope. It includes information about location and schemas.



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

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



8. To display the structure of the SOAP envelope as text, click the *Text view* icon.
The <SOAP-ENV:Header> tag is not required and can be deleted from the SOAP envelope.

9. Locate the following section:

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

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

For example, the Oracle repository URL has the following format:

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

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

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

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

10. Perform one of the following migration options.

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

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

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

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

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

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

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

JCA Repositories

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

Procedure: How to Migrate a JCA Repository

To migrate a JCA repository:

1. Navigate to the location of your JCA configuration directory where the repository schemas and other information is stored, for example:
`C:\Program Files\iWay55\config\base`
2. Locate and copy the *repository.xml* file.
3. Place this file in a new JCA configuration directory to migrate the existing repository.

Your JCA repository migrates to the new JCA configuration directory.

Migrating Event Handling Configurations

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

Procedure How to Migrate a Microsoft SQL Server 2000 Repository

To migrate a Microsoft SQL Server 2000 repository:

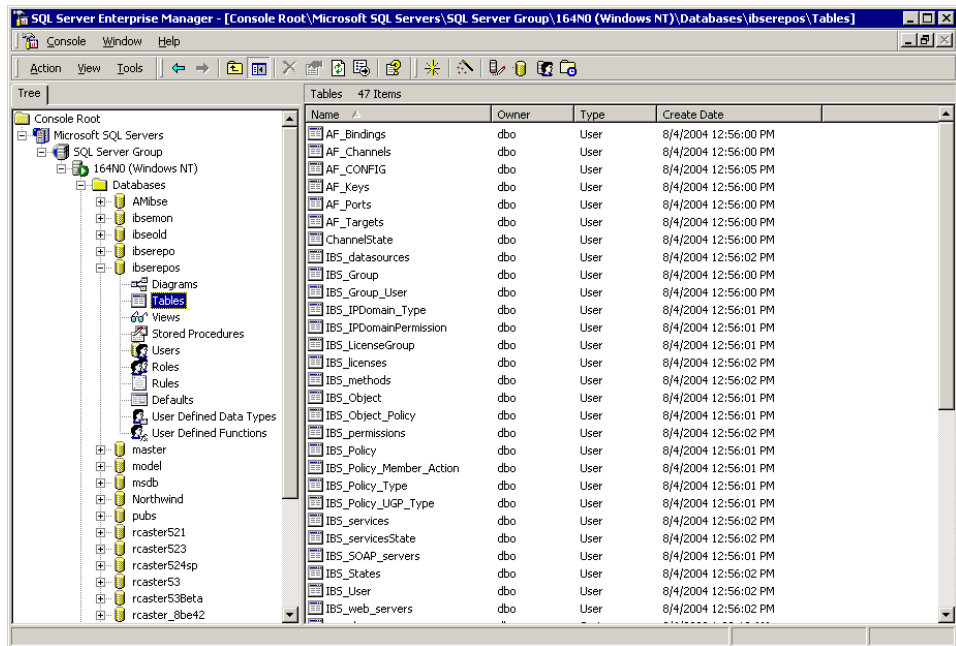
1. Open a command prompt and navigate to the iWay setup directory. The default location on Windows is:

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

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

`iwse.sql`

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



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

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

Procedure How to Migrate an Oracle Repository

To migrate an Oracle repository:

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

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

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

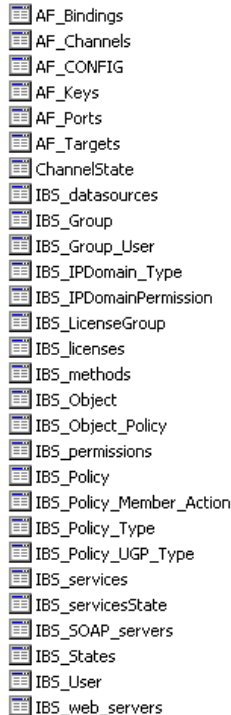
For Oracle 8:

```
iwse.ora
```

For Oracle 9:

[iwse.ora9](#)

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



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

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

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

Procedure How to Migrate a Sybase Repository

To migrate a Sybase repository:

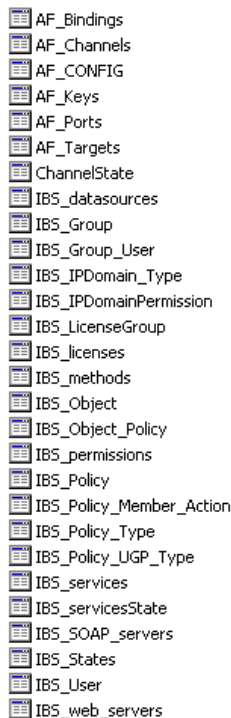
1. Open a command prompt and navigate to the iWay setup directory. The default location on Windows is:

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

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

[sybase-iwse.sql](#)

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



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

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

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

Procedure How to Migrate a DB2 Repository

To migrate a DB2 repository:

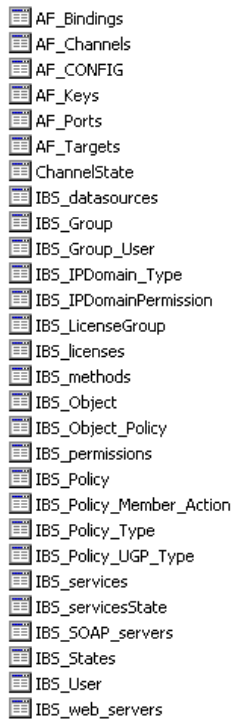
1. Open a command prompt and navigate to the iWay setup directory. The default location on Windows is:

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

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

`db2-iwse.sql`

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



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

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

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

Exporting or Importing Targets

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

Procedure: How to Export a Target

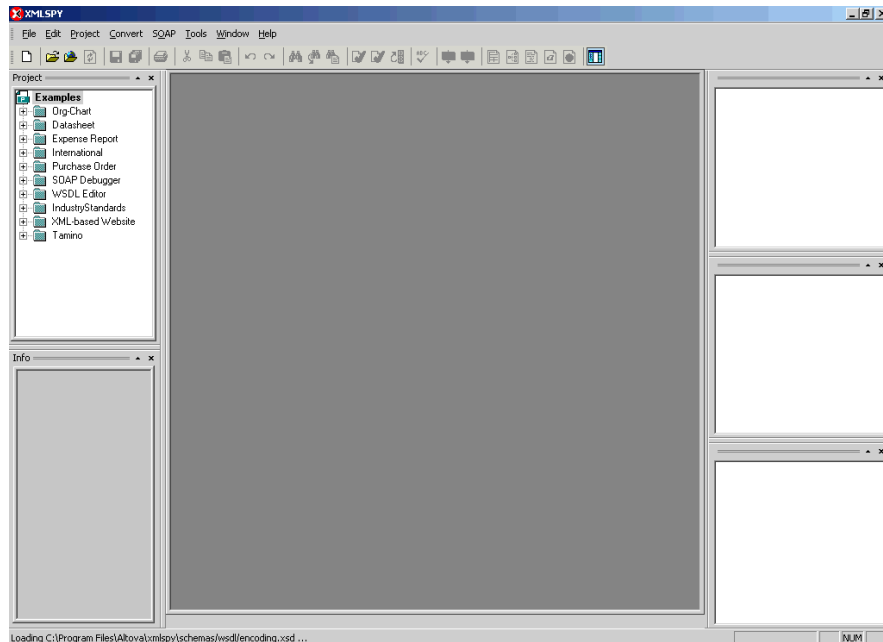
To export a target:

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

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

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

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



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

The WSDL file location dialog box opens.

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

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

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

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

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

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



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

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

9. Locate the following section:

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

- a. For the <m:target> tag, replace the String placeholder with the EIS target system name as it appears in Application Explorer and verify whether this value is case sensitive.
 - b. For the <m:name> tag, replace the String placeholder with the name of the target you want to export.
10. From the SOAP menu, select *Send request to server*.

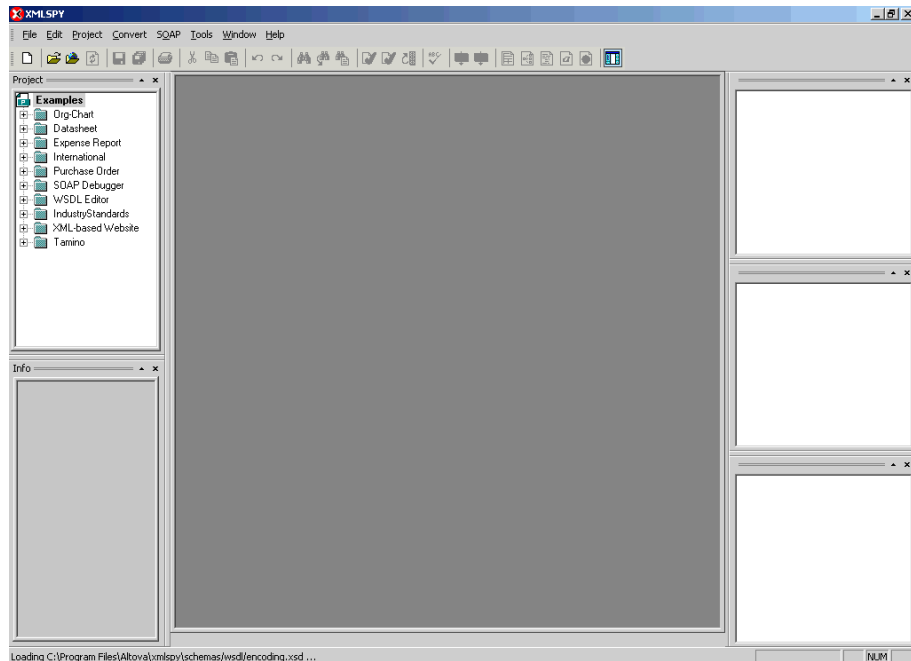
A response is returned that contains the <m: exporttime> and <m: contents> elements. You must use these elements when importing your target.

Procedure: How to Import a Target

To import a target:

1. Copy the iBSE administrative services for Application Explorer URL, for example:
<http://localhost:7777/ibse/IBSEServlet/admin/iwae.ibs?wsdl>
2. Open a third party XML editor, for example, XMLSPY.

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



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

The WSDL file location dialog box opens.

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

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

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

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

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

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



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

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

8. Locate the following section:

```
<m:IMPORTTARGET
xmlns:m="urn:schemas-iwaysoftware-com:dec2002:iwse:af">
<m:targetinstance>
<m:target>String</m:target>
<m:name>String</m:name>
<m:description>String</m:description>
<m:repositoryid>String</m:repositoryid>
<m:exporttime>2001-12-17T09:30:47-05:00</m:exporttime>
<m:contents>R01GODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</m:contents>
</m:targetinstance>
</m:IMPORTTARGET>
```

- a. For the <m:target> tag, replace the String placeholder with the EIS target system name.
 - b. For the <m:name> tag, replace the String placeholder with the new name of the target you want to import.
 - c. For the <m:description> tag, replace the String placeholder with a description of the target.
 - d. For the <m:repositoryid> tag, copy and paste the contents of the <m:repositoryid> tag that was returned when you exported your target.
 - e. For the <m: exporttime> tag, copy and paste the contents of the <m: exporttime> tag that was returned when you exported your target.
 - f. For the <m: contents> tag, copy and paste the contents of the <m: contents> tag that was returned when you exported your target.
- 9.** From the SOAP menu, select *Send request to server*.

Retrieving or Updating Web Service Method Connection Information

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

Procedure: How to Retrieve Web Service Method Connection Information

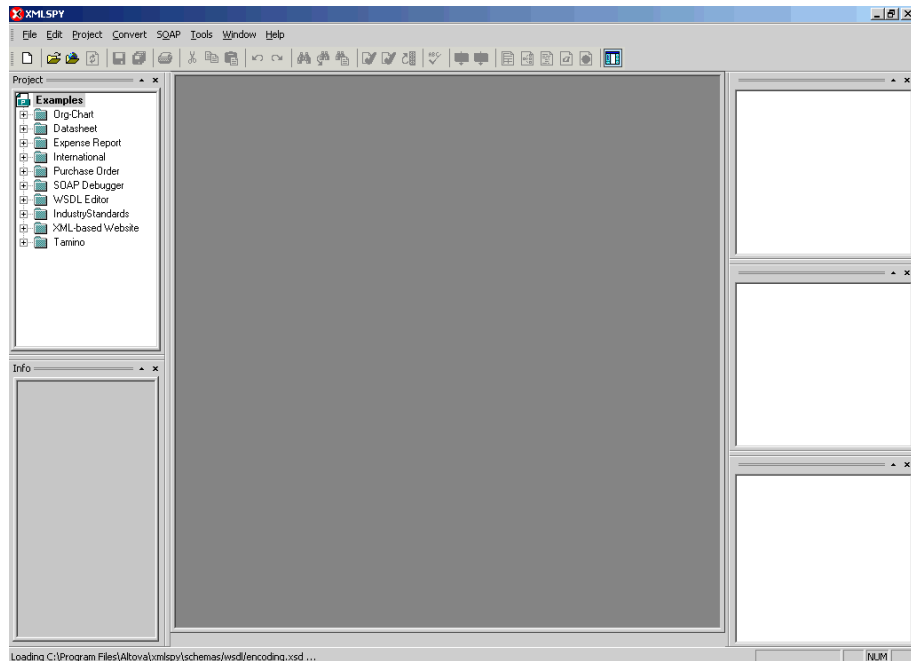
To retrieve Web service method connection information:

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

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

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

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



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

The WSDL file location dialog box opens.

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

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

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

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

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

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



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

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

8. Locate the following section:

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

- a. For the <m:serviceName> tag, replace the String placeholder with the name of the Web service.
 - b. For the <m:methodName> tag, replace the String placeholder with name of the Web service method.
9. From the SOAP menu, select *Send request to server*.

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

Procedure: How to Update Web Service Method Connection Information

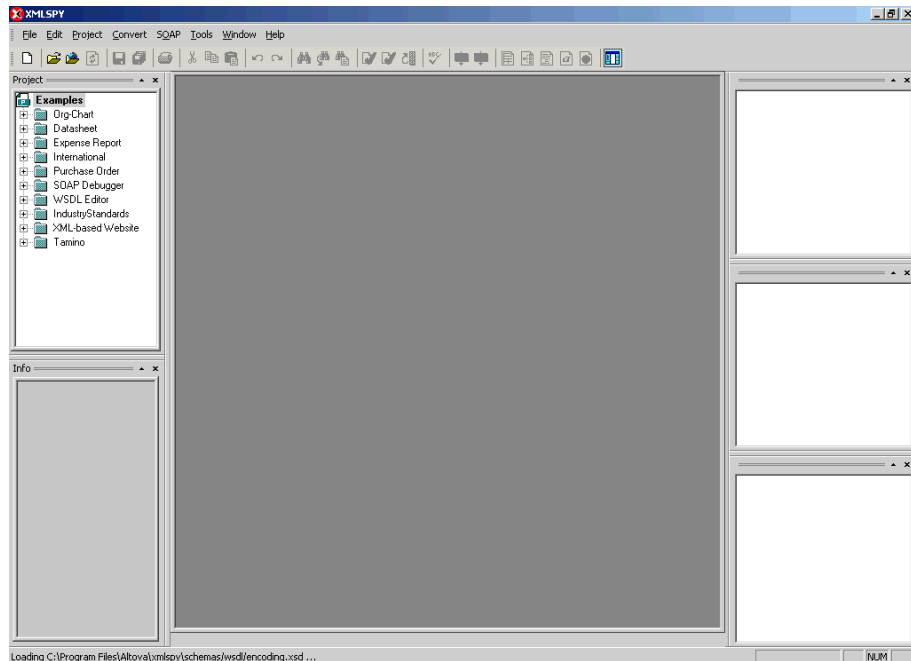
To update Web service method connection information:

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

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

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

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



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

The WSDL file location dialog box opens.

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

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

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

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

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

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



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

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

8. Locate the following section:

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

- a. For the <m:servicename> tag, replace the String placeholder with the name of the Web service.
 - b. For the <m:methodname> tag, replace the String placeholder with the name of the Web service method.
 - c. For the <m: descriptor> tag, copy and paste the contents of the <m: descriptor> tag that was returned when you retrieved Web Service method connection information.
- 9.** Modify the contents of the <m: descriptor> tag to change the existing Web Service method connection information.
- 10.** From the SOAP menu, select *Send request to server*.

Starting or Stopping a Channel Programmatically

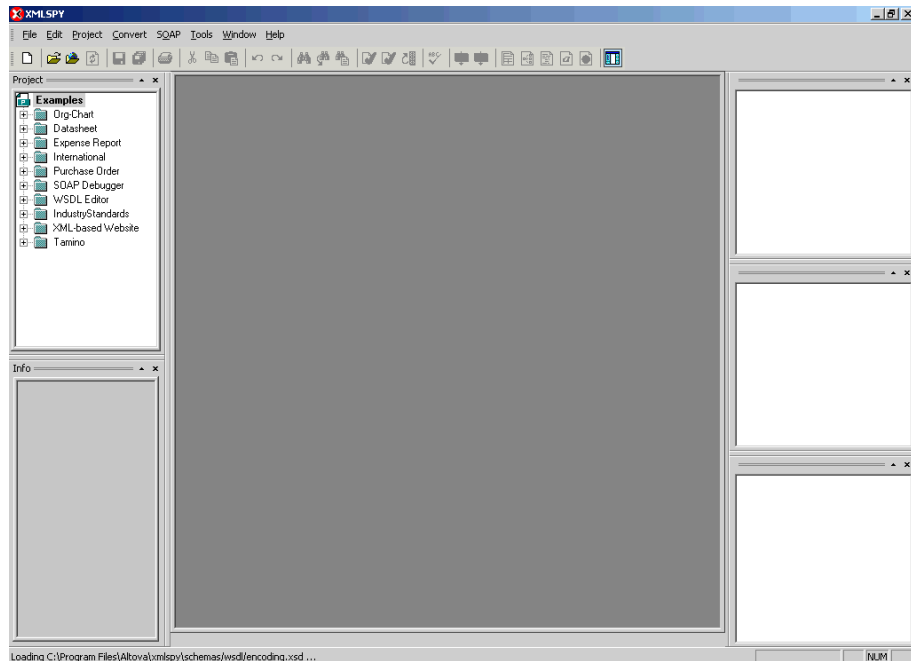
The following topic describes how to start or stop a channel programmatically.

Procedure: How to Start a Channel Programmatically

To start a channel programmatically:

1. Copy the iBSE control event URL, for example:
<http://localhost:7777/ibse/IBSEServlet/admin/iwevent.ibs?wsdl>
2. Open a third party XML editor, for example, XMLSPY.

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

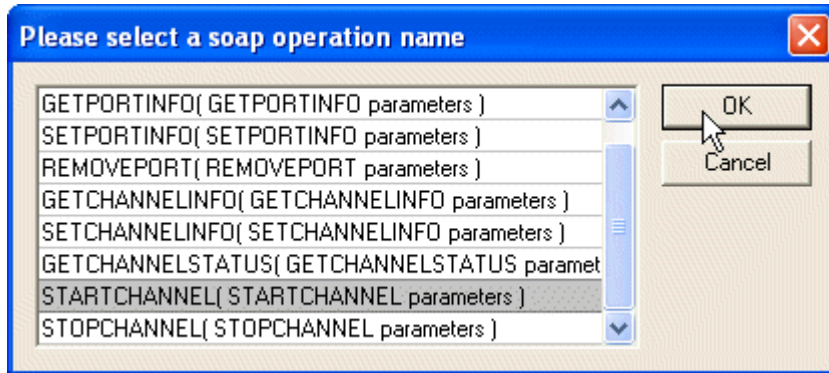


3. From the SOAP menu, select *Create new SOAP request*.

The WSDL file location dialog box opens.

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

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



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

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

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

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



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

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

8. Locate the following section:

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

9. For the `<m:channel>` tag, replace the *String* placeholder with the name of the Channel you want to start.
10. From the SOAP menu, select *Send request to server*.

Procedure: How to Stop a Channel Programmatically

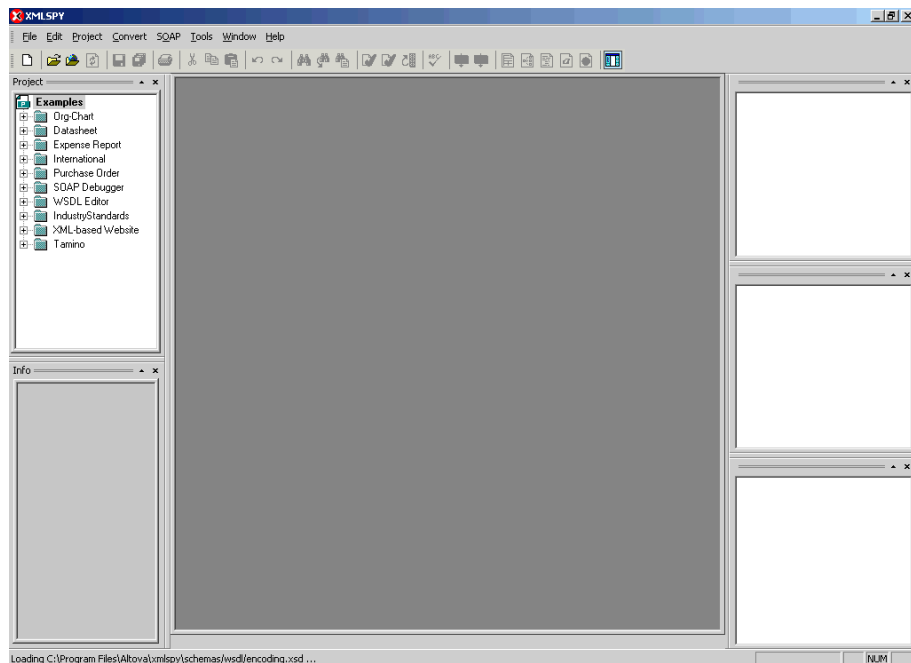
To stop a channel programmatically:

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

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

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

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

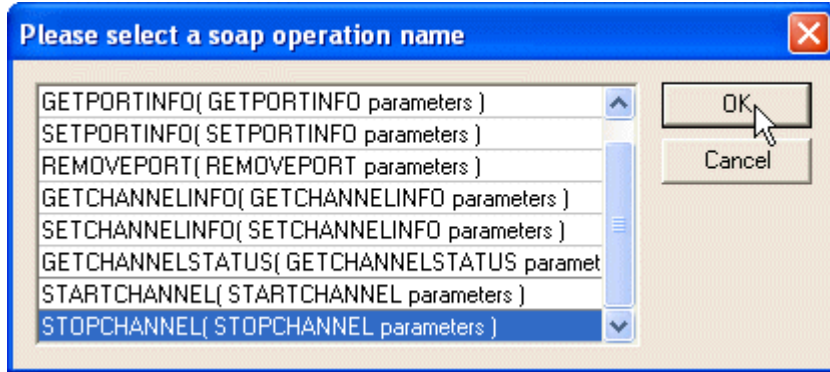


3. From the SOAP menu, select *Create new SOAP request*.

The WSDL file location dialog box opens.

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

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



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

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

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

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



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

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

8. Locate the following section:

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

9. For the `<m:channel>` tag, replace the String placeholder with the name of the Channel you want to stop.
10. From the SOAP menu, select *Send request to server*.

APPENDIX A

Using Application Explorer in BEA WebLogic Workshop to Create XML Schemas and Web Services

Topics:

- Generating Schemas and a Business Service
- Starting Application Explorer in WebLogic Workshop
- Creating and Managing a Connection
- Viewing Metadata
- Creating a Statement and Generating Schemas
- Request and Response Documents
- Multiple Executions of the Same SQL Statement or Stored Procedure
- Understanding iWay Business Services Engine
- Adding a Control for an iWay Resource in BEA WebLogic Workshop
- Adding an iWay Extensible CCI Control to a BEA WebLogic Workshop Application

This section describes how to use the Java Swing implementation of Application Explorer as deployed in BEA WebLogic Workshop. Application Explorer deployed in WebLogic Workshop is functionally similar to the Servlet Application Explorer.

This section describes how to use Application Explorer to:

- View metadata that describes your SQL statements and stored procedures.
- Create SQL statements and generate XML schemas that define request and response documents.
- Create business services (also known as Web services) for your SQL statements and stored procedures.

Generating Schemas and a Business Service

You can use Application Explorer to connect to relational databases, such as Oracle or Informix, and to non-relational databases, such as an IMS database.

When connected to non-relational databases, the adapter uses an iWay server component that enables you to take advantage of all the integration capabilities offered by iWay Software for mainframe database access. For example, when using the iWay server component, you gain relational database access to non-relational data. Stored procedures for non-relational databases can be created and stored using the server component.

Procedure How to Generate Request and Response Document Schemas or a Business Service

1. Start Application Explorer in WebLogic Workshop. For more information, see *Starting Application Explorer in WebLogic Workshop* on page A-3.
2. Open a new or existing connection to a relational or non-relational database management system, as described in *Creating and Managing a Connection* on page A-6.

After you connect to a system, you can expand the iWay Service Adapters node to view the list of adapters installed on your system. After you finish using a connection, you can close it. If you will not need the connection in the future, you can delete it.

Note: When you close Application Explorer, it automatically closes all open connections.

3. View metadata that describes your tables and stored procedures. You can use the metadata when creating request documents and when you develop logic for processing response documents. For more information, see *Viewing Metadata* on page A-11.
4. Create a prepared statement, which generates XML request and response schemas. For more information, see *Creating a Statement and Generating Schemas* on page A-15.

You can use a third-party XML tool to generate request and response instance documents from XML schemas. For information about request and response document formats, see *Request and Response Documents* on page A-25.

5. Generate a business service (also known as a Web service) for an SQL statement or stored procedure. For more information on Web services, see *Understanding iWay Business Services Engine* on page A-37.

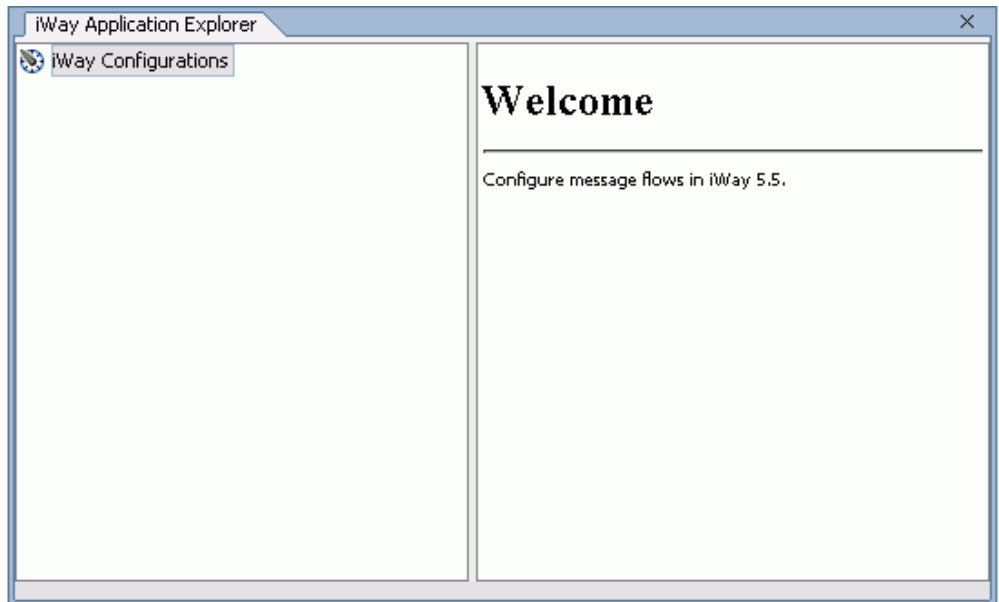
Starting Application Explorer in WebLogic Workshop

You can use Application Explorer with an iBSE or a JCA configuration. Before you can use Application Explorer, you must start BEA WebLogic Server.

Procedure How to Start Application Explorer

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

Application Explorer opens in BEA WebLogic Workshop.



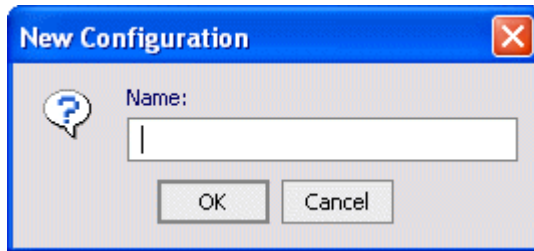
You can resize and drag-and-drop the Application Explorer window within BEA WebLogic Workshop. For example, you can drag it to the upper part of BEA WebLogic Workshop.

Procedure How to Define a New Configuration

Before you can start using Application Explorer, you must define a new configuration by performing the following steps:

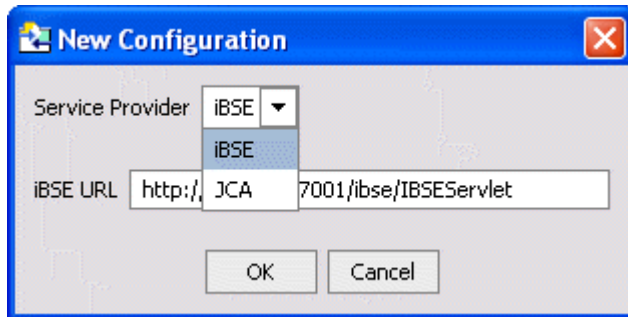
1. Right-click *Configurations* and select *New*.

The New Configuration dialog box opens:



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

The following dialog box opens:



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

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

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

where:

localhost

Is where your application server is running.

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

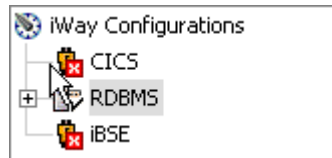
C:\Program Files\iWay55

where:

iWay55

Is the full path to your iWay installation.

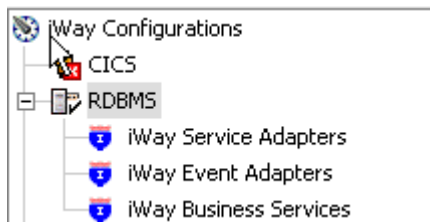
A node representing the new configuration appears under the Configurations node. The right pane provides details of the configuration you created.



Procedure How to Connect to a New Configuration

Right-click the configuration to which you want to connect (for example, RDBMS), and select *Connect*.

Nodes are displayed for iWay Service Adapters, iWay Event Adapters, and iWay Business Services (also known as Web services):



You are now ready to define new targets to RDBMS.

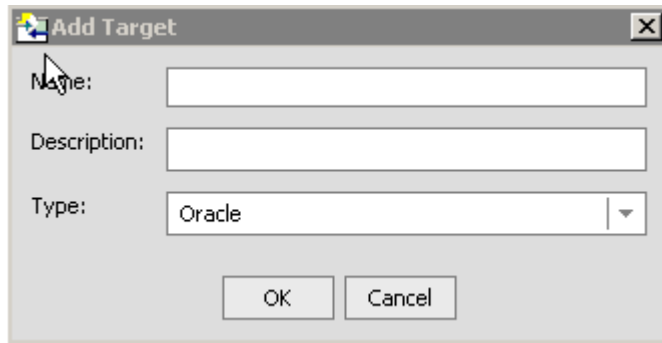
Creating and Managing a Connection

To access an adapter, you must define a target that connects to the adapter. After the defined target is created, it automatically is saved. You must establish a connection to the defined target every time you start Application Explorer or after disconnecting.

Procedure How to Define a New Target

1. Expand the *iWay Service Adapters* node.
2. Expand the *RDBMS* node.
3. Right-click the *RDBMS* node and select *Add Target*.

The Add Target dialog box opens:



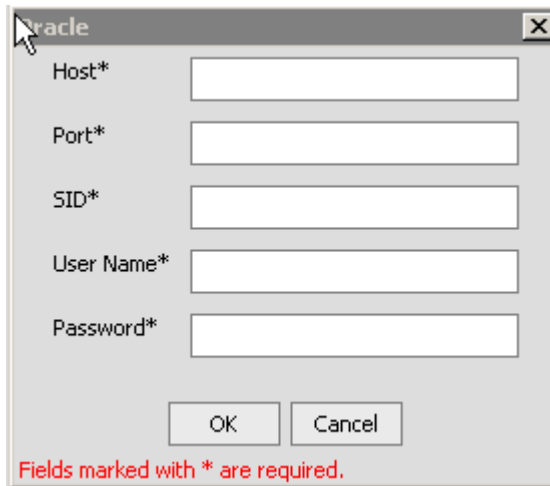
- a. Type a name for the new target (for example, *RDBMS_Connection*).
- b. Type a description (optional).
- c. From the Type drop-down list, select the type of target.

Important: Oracle is the default value. If you are connecting to a non-relational database, select *iWay Server*. This establishes a connection to the iWay server component. The server component is used to provide relational database access to non-relational databases. For more information on the server component, see Chapter 1, *Introducing the iWay XML Adapter for RDBMS*.

Note: The EDA Server and WebFOCUS Server target options are for connecting to Information Builders resources. The types of databases that display in the Type drop down list depends on the jar files you have installed in your lib directory.

4. Click OK.

The connection dialog box opens. The fields that appear in the connection dialog box are specific to the type of database to which you are connecting. For example, the following dialog box shows the fields for connecting to an Oracle database:



The image shows a dialog box titled "Oracle" with a close button (X) in the top right corner. Inside the dialog, there are five text input fields, each preceded by a label with an asterisk indicating it is required: "Host*", "Port*", "SID*", "User Name*", and "Password*". At the bottom of the dialog, there are two buttons: "OK" and "Cancel". Below the buttons, a red text message reads: "Fields marked with * are required."

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

5. Enter connection information that is specific to the database to which you want to connect. The following table lists and defines connection parameters:

Parameter	Definition
Host	DNS or IP name of the server where the database instance resides.
Port	Port number on which the database is listening.
Database (Data source) Name	Specific name of the database or data source to which you are connecting.
Use driver type	<p>For DB2 connections, select one of the following JDBC drivers:</p> <ul style="list-style-type: none"> • app driver. Ensure you have the DB2 client installed on the machine from which you are accessing Application Explorer. • net driver. Ensure you started the DB2 applet server. <p>For more information, see your DB2 documentation.</p>

Parameter	Definition
Server Name	For an iWay server component, the name of the service node to which you are connecting.
Service Name	For connecting to a non-relational database to access stored procedures, this parameter defaults to the DEFAULT service name of the iWay server component. Can be left blank.
Informix Server Name	Equivalent to the Informix variable name INFORMIXSERVER.
Driver	Name of the driver used to access the database to which you want to connect. For more information, see your database documentation.
URL	For a JDBC connection, the JDBC driver-specific URL used to connect to the RDBMS.
SID	For an Oracle database, the unique name of the database service selected by the database administrator or the person who installed Oracle.
User Name	Database user ID to access the database. The user ID must have database access to the interface tables that are accessed.
Password	Password associated with the specified user name.

Note: When connecting to the iWay server component for access to non-relational databases, the connection information must be the same for all databases in the server component system.

6. Click OK.

The target name appears under the node where you created the new connection. For information on connecting to the node, see *How to Connect to a Defined Target* on page A-9.

Reference Using selectMethod=Cursor

To avoid some exceptions when using the iWay XML Adapter for RDBMS for BEA WebLogic with Microsoft SQL Server 2000 Driver for JDBC, you must add `selectMethod=cursor` to the JDBC URL specification. For example,

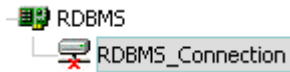
```
jdbc:microsoft:sqlserver://PMSNJC:1433;DatabaseName=dbname;selectMethod=cursor;
```

This statement determines whether Microsoft SQL Server "server cursors" are used for SQL queries. Because the adapter is not limited to a single active statement while executing a set of queries within a transaction, adding this statement to the JDBC URL allows you to specify multiple queries within a transaction. This helps to prevent errors because it addresses default settings in the adapter and in the driver.

The benefit of specifying this statement is that it enables you to have multiple concurrent statements open from a given connection, which is required for pooled connections.

Procedure How to Connect to a Defined Target

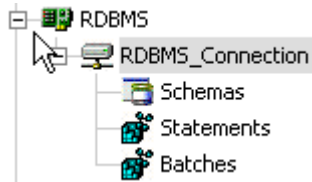
1. Expand the *iWay Service Adapters* node.
2. Expand the *RDBMS* node.
3. Click the target name (for example, *RDBMS_Connection*) under the *RDBMS* node.



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

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

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



Disconnecting From a Defined Target

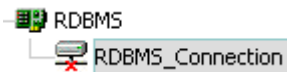
Although you can maintain multiple open connections, iWay Software recommends disconnecting from targets that are not in use.

Procedure How to Disconnect From a Defined Target

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

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

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



Editing a Defined Target

After you create a defined target using Application Explorer, you can edit any information that you provided during the creation process.

Procedure How to Edit a Defined Target

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

The Edit dialog box opens containing the connection fields.

4. Edit the information as needed and then click *OK*.

Deleting a Defined Target

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

Procedure How to Delete a Defined Target

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

The node disappears from the list of available connections.

Viewing Metadata

Viewing metadata is useful when creating request documents. You can view:

- Table metadata, as described in *How to View Table Metadata*.
- Stored procedure metadata for a relational database or non-relational database, as described in *How to View Stored Procedure Metadata* on page A-13.

Procedure How to View Table Metadata

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page A-9.
2. Expand the *Schemas* node under the desired connection.
3. Select a database.
4. Expand the *Tables* node under the desired database.
5. Scroll down and select a table to review.

Note: Although the list of tables includes all tables in the database, the user ID you specified for the connection may not have access to the table you selected. If this is the case, the creation of schemas fails.

When you select a table, a metadata summary table appears in the right pane, as shown in the following illustration.

The screenshot shows the 'Detail' tab of a metadata viewer. At the top, there are three tabs: 'Detail' (selected), 'Columns', and 'Event Schema'. Below the tabs, the 'iwaf.description' table is selected, and its 'Table Type' is 'TABLE'. The main section is titled 'Database Properties' and contains a list of database-related fields and their values:

Database Product Name	Oracle
Database Product Version	Oracle®i Enterprise Edition Release 8.1.7.3.0 - Production With the Partitioning opt
Catalog Term	
Database Transaction Isolation	2
Driver Major Version	9
Driver Minor Version	2
Driver Name	Oracle JDBC driver
Driver Version	9.2.0.3.0

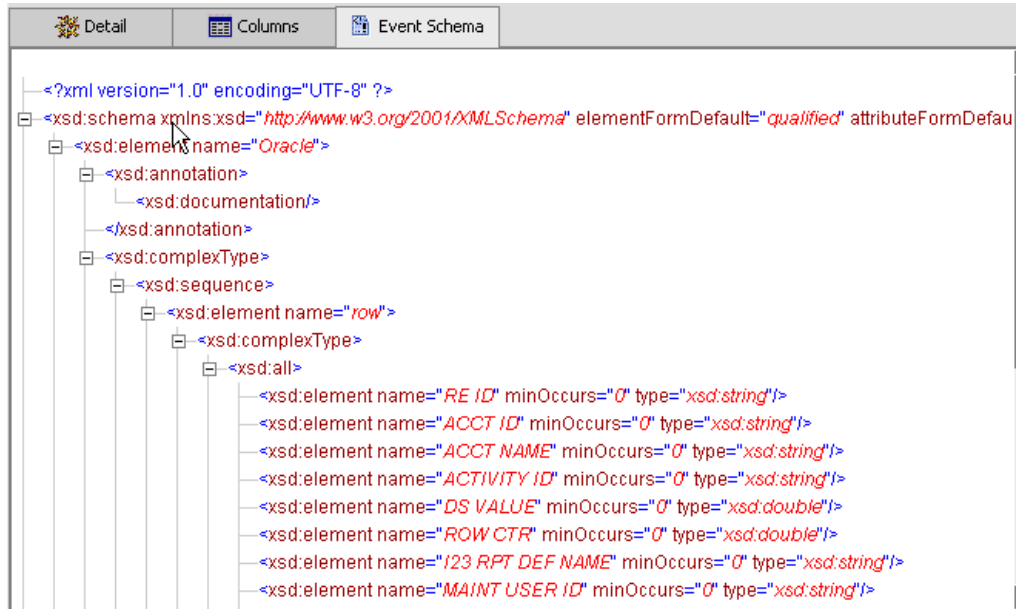
- Depending on which properties you want to review, click the *Columns* or *Event Schema* tab.

The following is an illustration of the Columns tab:

The screenshot shows the 'Columns' tab of the same metadata viewer. It displays a table with the following columns: column name, data type, type name, column size, buffer length, decimal digits, num prec r..., nullable, and remarks. The table contains 17 rows of data, with the first row highlighted in blue and the last row highlighted in green.

column name	data type	type name	column size	buffer length	decimal digits	num prec r...	nullable	remarks
RE_ID	12 VARCHAR2	12 VARCHAR2	60	0	0	10	1	
ACCT_ID	12 VARCHAR2	12 VARCHAR2	90	0	0	10	1	
ACCT_NAME	12 VARCHAR2	12 VARCHAR2	75	0	0	10	1	
ACTIVITY_ID	12 VARCHAR2	12 VARCHAR2	120	0	0	10	1	
DS_VALUE	3 NUMBER	3 NUMBER	22	0	0	10	1	
ROW_CTR	3 NUMBER	3 NUMBER	22	0	0	10	0	
I23_RPT_DE...	12 VARCHAR2	12 VARCHAR2	75	0	0	10	1	
MAINT_USE...	12 VARCHAR2	12 VARCHAR2	11	0	0	10	1	
MAINT_EFF...	12 VARCHAR2	12 VARCHAR2	1	0	0	10	1	
MAINT_TIM...	91 DATE	91 DATE	7	0	0	10	1	
ACTIVITY_N...	12 VARCHAR2	12 VARCHAR2	75	0	0	10	1	
CREATION_...	91 DATE	91 DATE	7	0	0	10	0	
CREATED_BY	3 NUMBER	3 NUMBER	15	0 0	0	10	0	
LAST_UPDA...	91 DATE	91 DATE	7	0	0	10	0	
LAST_UPDA...	3 NUMBER	3 NUMBER	15	0 0	0	10	0	
LAST_UPDA...	3 NUMBER	3 NUMBER	15	0 0	0	10	1	

The following is an illustration of the Event Schema tab:




- b. Use this information to determine the table (or tables) and fields to use when you are ready to create a schema.


Procedure How to View Stored Procedure Metadata

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page A-9.
2. Expand the *Schemas* node under the desired connection.
3. Select a database.
4. Expand the *Procedures* node under the desired database.
5. Scroll down and select the procedure to review.

Note: Although the list of procedures includes all procedures in the database, the user ID specified for the connection may not have access to the specified procedure. If this is the case, the creation of schemas fails.

When you select a procedure, a metadata summary table appears in the right pane, as shown in the following illustration:

 Detail

 Parameters

iwaf.description

Not Specified

Statement Value

{ call ABM.APPS_ARRAY_DDL(? , ? , ?) }

Schema

ABM

Database Properties

Database Product Name

Oracle

Database Product Version

Oracle8i Enterprise Edition Release 8.1.7.3.0 - Production With the Partitioning opt

Catalog Term

Database Transaction Isolation

2

Driver Major Version

9

Driver Minor Version

2

Driver Name

Oracle JDBC driver

a. Click the *Parameters* tab to view parameter information:

Parameter ...	Data Type	Length	Column Types	Remarks
LB	NUMBER	22 IN		
UB	NUMBER	22 IN		
NEWLINE_F...	VARCHAR2	0 IN		

b. Use this information to determine the procedure (or procedures) and fields to use when you are ready to create a schema.

Note: Parameters are for viewing only. Do not change parameter values.

Creating a Statement and Generating Schemas

You can create an SQL statement even when using the adapter for non-relational databases. Once you create a statement, schemas that define request and response documents are automatically generated. The metadata is stored in the iWay Repository, which can be implemented in an RDBMS (such as Oracle or Microsoft SQL Server), a file system, or a specialized XML database. You can generate the following types of statements:

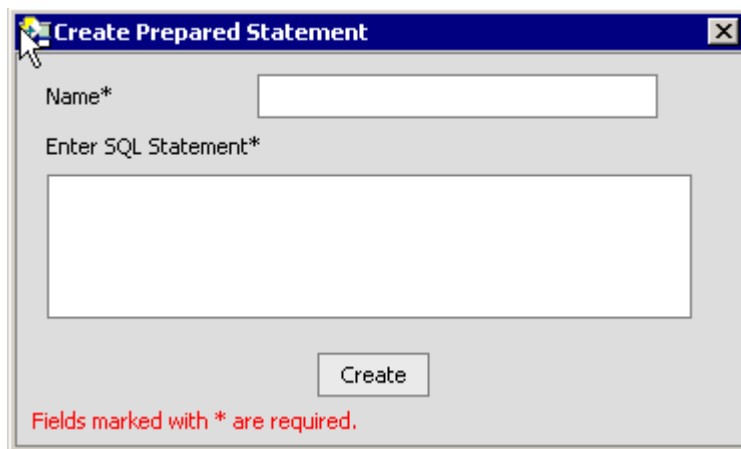
- Regular SQL Statements, as described in *How to Create a Regular SQL Statement* on page A-15.
- Parameterized SQL statements, as described in *How to Create a Parameterized SQL Statement* on page A-18.
- Batch statements, as described in *How to Create a Batch Statement* on page A-22.

Note: If you plan to create a business service, you are not required to generate a schema. For more information, see *Understanding iWay Business Services Engine* on page A-37.

Procedure How to Create a Regular SQL Statement

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page A-9, and then expand the connection node.
2. Right-click the *Statements* node and select *Create Prepared Statement*.

The Create Prepared Statement dialog box opens:



- a. Type a name for the statement.

iWay Software recommends that you specify a name that describes the service. For example, a name of CustomerIntField could represent a request against the Customer Interface table returning a Field format response document.

b. In the Enter SQL Statement field, type the SQL statement for the adapter to use.

Note: If you are not the owner of the table(s), the table name must be fully qualified.

3. Click *Create*.

The properties table for the newly created statement appears in the right pane:

The screenshot shows a software interface with three tabs: 'Detail', 'Request Schema', and 'Response Schema'. The 'Detail' tab is active. It contains the following fields:

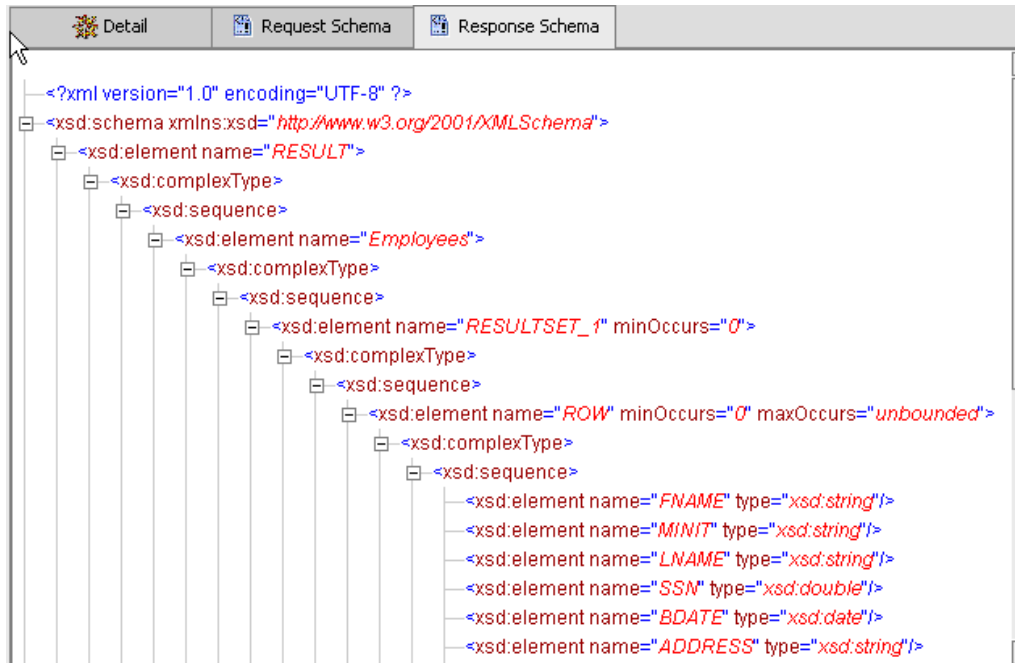
- iwaf.description:** Not Specified
- Statement:** Select * from EMPLOYEE
- Database Properties:**
 - Database Product Name:** Oracle
 - Database Product Version:** Oracle8i Enterprise Edition Release 8.1.7.3.0
 - Catalog Term:**
 - Database Transaction Isolation:** 2
 - Driver Major Version:** 9
 - Driver Minor Version:** 2
 - Driver Name:** Oracle JDBC driver
 - Driver Version:** 9.2.0.3.0

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

The screenshot shows the 'Request Schema' tab active. It displays the following XML schema information:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:m1="urn:iwaysoftware:ibse:feb2004:Request"
  <xs:element name="AdapterParams"/>
</xs:schema>
```

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



For more information about schemas, see *Schema Location* on page A-24 and *How to Export a Schema* on page A-24.

Procedure How to Test a Regular SQL Statement

1. Right-click the regular SQL statement node you want to test, and select *Test Run*.
The Test Run dialog box opens.
2. Click *Test*.
The adapter displays the results.
3. To exit the results, click *OK*.

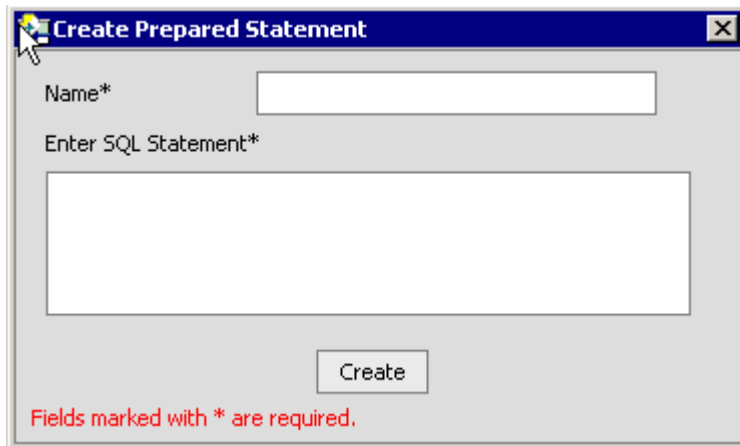
Creating and Testing a Parameterized SQL Statement

Parameterized SQL allows an SQL statement to be stored within the repository system with parameters imbedded within it. These parameters can be retrieved from XML documents at run time and executed against the SQL statements specified at design time. Application Explorer creates and maps parameters for the parameterized SQL at design time.

Procedure **How to Create a Parameterized SQL Statement**

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page A-9, and then expand the connection node.
2. Right-click the *Statements* node and select *Create Prepared Statement*.

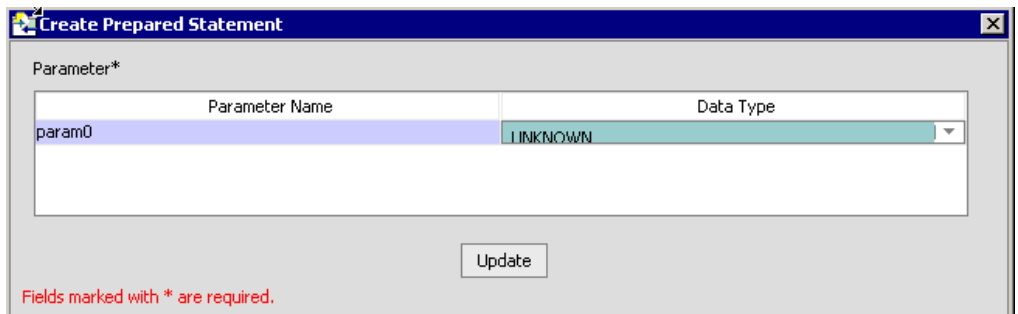
The Create Prepared Statement dialog box opens:



The dialog box titled "Create Prepared Statement" has a blue title bar with a close button. It contains two input fields: "Name*" and "Enter SQL Statement*", both marked with an asterisk to indicate they are required. Below the "Enter SQL Statement*" field is a large text area. At the bottom center is a "Create" button. At the bottom left, in red text, is the message "Fields marked with * are required."

- a. Type a name for the statement.
 - b. In the Enter SQL Statement field, type the parameterized SQL statement.
- Note:** If you are not the owner of the table(s), the table name must be fully qualified.
3. Click *Create*.

The Parameter dialog box opens:



The dialog box titled "Create Prepared Statement" (which also serves as the parameter dialog) has a blue title bar with a close button. It contains a table with two columns: "Parameter Name" and "Data Type". The first row has "param0" in the first column and "UNKNOWN" in the second column. Below the table is an "Update" button. At the bottom left, in red text, is the message "Fields marked with * are required."

Parameter Name	Data Type
param0	UNKNOWN

- a. Type a name for each parameter.
- b. From the drop down list, select a data type.

4. Click *Update*.

The properties table for the newly created statement appears in the right pane:

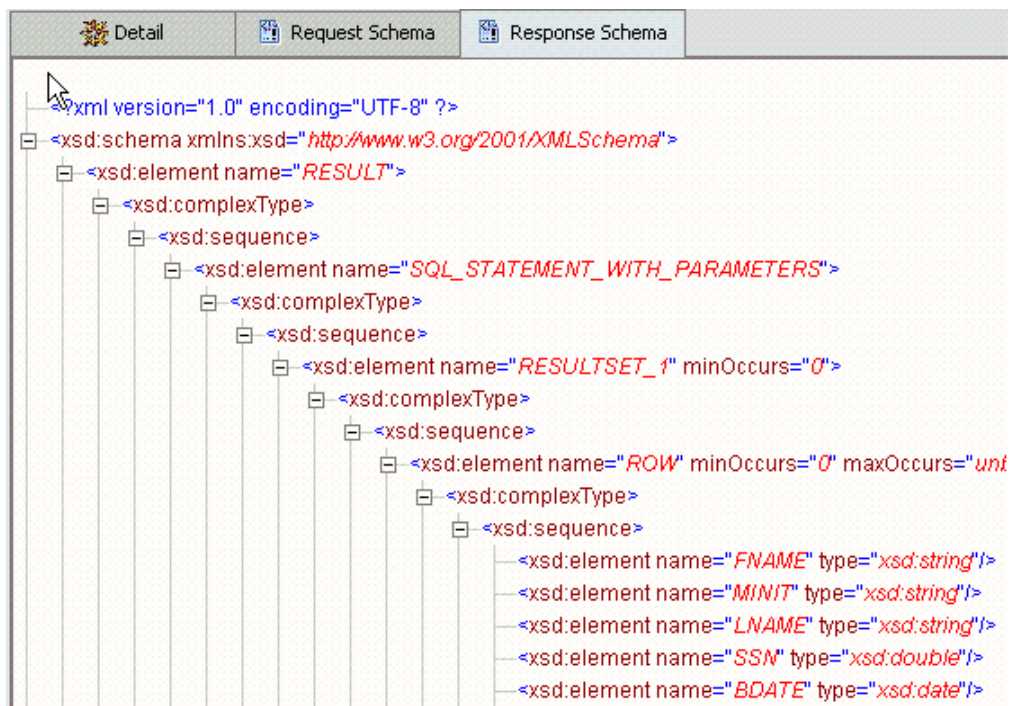
iwaf.description		Not Specified
Statement		SELECT * FROM EMPLOYEE WHERE LNAME = ?
Parameters		
Parameter Name	parameter1	
Data Type	LONGVARCHAR	
Length		
Column Types	IN	
Remarks		
Database Properties		
Database Product Name	Oracle	
Database Product Version	Oracle8i Enterprise Edition	

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

```

<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:m1="urn:iwaysoftware.ibse.feb20
  <xs:element name="AdapterParams">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="parameter1" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
  
```

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



For more information about schemas, see *Schema Location* on page A-24 and *How to Export a Schema* on page A-24.

Procedure How to Test a Parameterized SQL Statement

1. Right-click the parameterized SQL statement node you want to test, and select *Test Run*.

The Test Run dialog box opens:

The screenshot shows the 'Test Run' dialog box. It has a title bar with a 'Test Run' icon and a close button. The main area is labeled 'Parameter*' and contains a table with three columns: 'Parameter Name', 'Data Type', and 'Value'. The first row has 'parameter1' in the 'Parameter Name' column and 'LONGVARCHAR' in the 'Data Type' column. The 'Value' column is empty. Below the table is a 'Test' button. At the bottom, there is a red text label: 'Fields marked with * are required.'

Parameter Name	Data Type	Value
parameter1	LONGVARCHAR	

2. For each parameter, type a value in the Value field.

In the case of the following SQL statement:

```
Select * from employee where LNAME=?
```

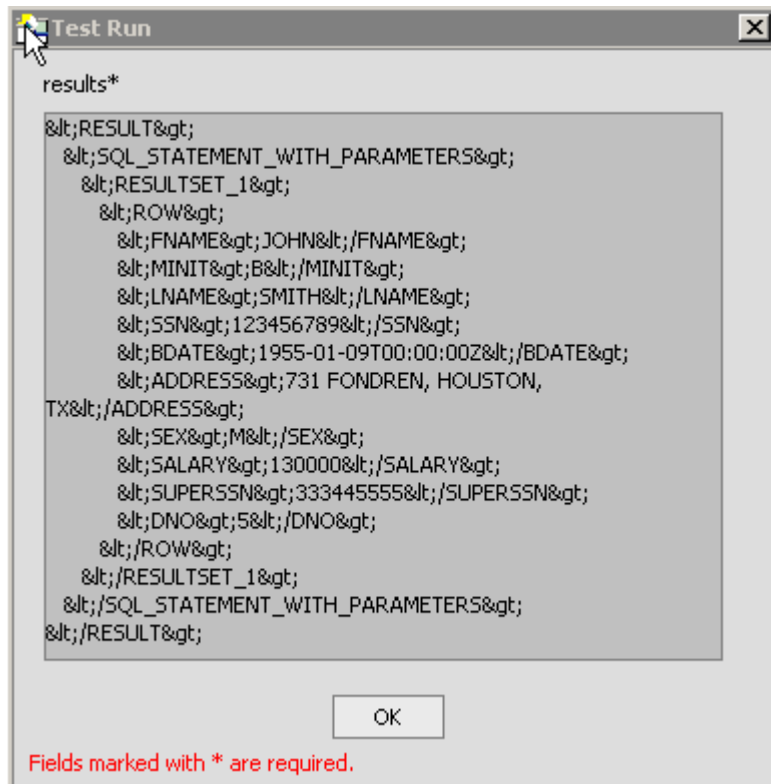
provide a sample character value, for example, SMITH.

In this example, the values correspond to values of fields found in a table.

Parameterized statements may include parameters that are input for SQL functions, for example, the Oracle SQL function TO_DATE(StringParm). In this case, the data type selected is the expected data type of the SQL function. This is why you provide the SQL type when you create the prepared parameterized SQL statement.

3. Click *Test*.

The adapter displays the results:



4. To exit the results, click *OK*.

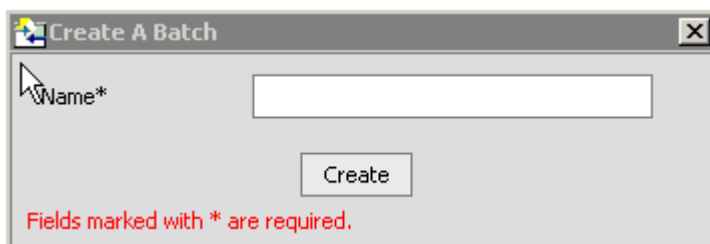
Creating a Batch Statement

Batch statements allow you to execute multiple regular SQL and/or parameterized SQL statements simultaneously.

Procedure How to Create a Batch Statement

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page A-9.
2. Right-click the *Batches* node and select *Create A Batch*.

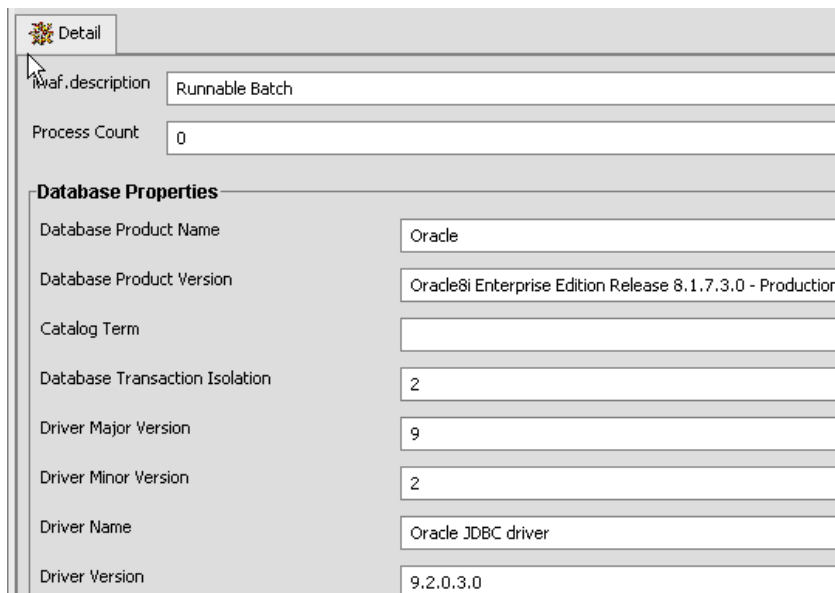
The Create A Batch dialog box opens:



The 'Create A Batch' dialog box is shown. It has a title bar with a close button. Inside, there is a label 'Name*' followed by a text input field. Below the input field is a 'Create' button. At the bottom, a red message states 'Fields marked with * are required.'

3. Type a name for the new Batch and click *Create*.

The batch properties information appears in the right pane:



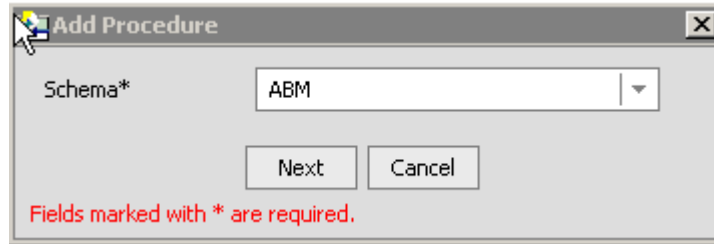
The 'Detail' tab of the batch properties form is shown. It contains the following fields:

Detail	
Batch description	Runnable Batch
Process Count	0
Database Properties	
Database Product Name	Oracle
Database Product Version	Oracle8i Enterprise Edition Release 8.1.7.3.0 - Production
Catalog Term	
Database Transaction Isolation	2
Driver Major Version	9
Driver Minor Version	2
Driver Name	Oracle JDBC driver
Driver Version	9.2.0.3.0

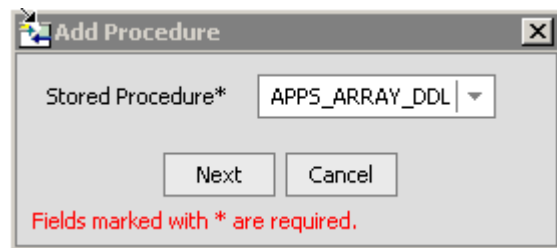
4. Right-click the batch node you created and select *Add Procedure* or *Add Statement*.

Note: If you are connected to an RDBMS node and are accessing a non-relational database, you cannot add stored procedures to a batch statement.

- a. If you selected *Add Procedure*, the Add Procedure dialog box opens:

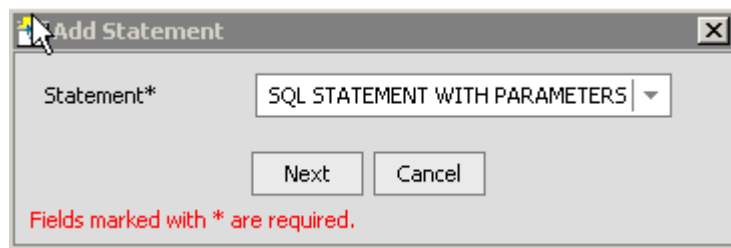


Select the schema where the stored procedure resides from the drop-down list, and click *Next*. The Stored Procedure dialog box opens:



Select the stored procedure from the drop-down list and click *Next*.

- b. If you selected *Add Statement*, the Add Statement dialog box opens:



Select the statement from the drop-down list and click *Next*.

The procedure or statement is added to the batch node.

Reference Schema Location

Application Explorer stores the schemas it creates in subdirectories under the iWay home directory of the machine where it is installed. The exact location of the schemas differs depending on whether you deploy Application Explorer with an iBSE or a JCA configuration.

- When using the adapter with an iBSE configuration, the schemas are stored under a \schemas subdirectory of the iWay home directory, for example,

`C:\Program Files\iWay55\bea\ibse\wsdl\schemas\service\RDBMS\NewTarget`

where:

`NewTarget`

Is the name of the connection to the RDBMS system as defined in Application Explorer. Under this directory, Application Explorer creates subdirectories containing schemas.

- When using the adapter with a JCA configuration, the schemas are stored under a \schemas subdirectory of the iWay home directory, for example,

`C:\Program Files\iWay55\config\base\schemas\RDBMS\NewTarget`

where:

`NewTarget`

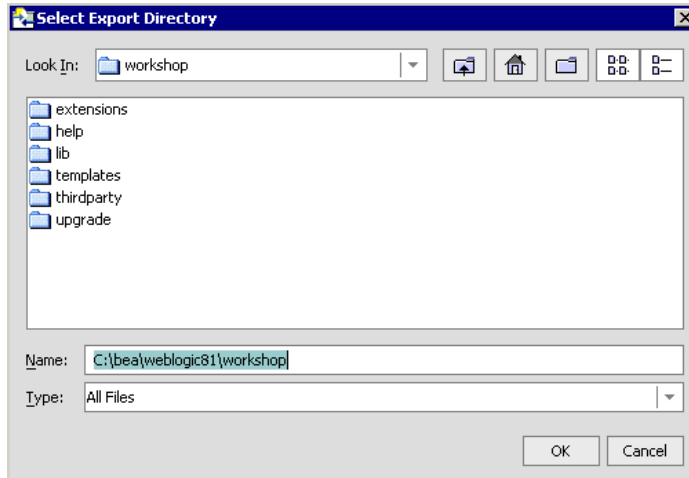
Is the name of the connection to the RDBMS system as defined in Application Explorer. Application Explorer stores the schemas in this directory.

Procedure How to Export a Schema

1. If you have not already done so, connect to the target from which you want to export a schema (for example, RDBMS_Connection).
2. Right-click the statement from which you want to export a schema, and select *Export Schema(s)*.

Note: You can also export schemas from tables, stored procedures, and batches.

3. The Select Export Directory dialog box opens:



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

Request and Response Documents

You can generate request and response document schemas using Application Explorer, as described in *Creating a Statement and Generating Schemas* on page A-15. You can generate request and response instance documents using a third party XML tool and submit those documents to the RDBMS or iWay agent.

The following topics include examples of schemas and instance documents for:

- Regular SQL Statements
- Parameterized SQL Statements
- Stored Procedure Schemas for an Oracle Database
- Stored Procedure Schemas for a VSAM Database

Regular SQL Statements

The following examples are based on a schema created for a regular SQL statement.

Example Regular SQL Request Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:22:33Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RDBMS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="stock_price_select">
          <xsd:complexType>
            <xsd:attribute name="location" type="xsd:string"
use="optional" fixed="RDBMS/Statements/stock price select"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example Regular SQL Request Instance Document

```
<?xml version="1.0" encoding="UTF-8"?>
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\stock price
select_request.xsd">
  <stock_price_select location="RDBMS/Statements/stock price select"/>
</RDBMS>
```

Example Regular SQL Response Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:22:33Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RESULT">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="stock_price_select">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="RESULTSET_1" minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="ROW" minOccurs="0" maxOccurs="unbounded">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="RIC" type="xsd:string"/>
                          <xsd:element name="PRICE" type="xsd:double"/>
                          <xsd:element name="UPDATED" type="xsd:date"/>
                          <xsd:element name="RR" type="xsd:double"/>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example Regular SQL Response Instance Document

```
<?xml version="1.0" encoding="UTF-8"?>
<RESULT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\stock price
select_response.xsd">
  <stock_price_select>
    <RESULTSET_1>
      <ROW>
        <RIC>String</RIC>
        <PRICE>3.14159265358979</PRICE>
        <UPDATED>1967-08-13</UPDATED>
        <RR>3.14159265358979</RR>
      </ROW>
      <ROW>
        <RIC>String</RIC>
        <PRICE>3.14159265358979</PRICE>
        <UPDATED>1967-08-13</UPDATED>
        <RR>3.14159265358979</RR>
      </ROW>
      <ROW>
        <RIC>String</RIC>
        <PRICE>3.14159265358979</PRICE>
        <UPDATED>1967-08-13</UPDATED>
        <RR>3.14159265358979</RR>
      </ROW>
    </RESULTSET_1>
  </stock_price_select>
</RESULT>
```


Parameterized SQL Statements

The following examples are based on a schema created for a parameterized SQL statement.

Example Parameterized SQL Request Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T21:57:19Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RDBMS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="paramempdata">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="param0">
                <xsd:complexType>
                  <xsd:simpleContent>
                    <xsd:extension base="xsd:string">
                      <xsd:attribute name="dataType"
type="xsd:string" use="required" fixed="CHAR"/>
                    </xsd:extension>
                  </xsd:simpleContent>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="location" type="xsd:string"
use="optional" fixed="RDBMS/Statements/paramempdata"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example Parameterized SQL Request Instance Document

```
<?xml version="1.0" encoding="UTF-8"?>
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\paramempdata_request.xsd">
  <paramempdata location="RDBMS/Statements/paramempdata">
    <param0 dataType="CHAR">String</param0>
  </paramempdata>
</RDBMS>
```

Example **Parameterized SQL Response Schema**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T21:57:20Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RESULT">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="paramempdata">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="RESULTSET_1" minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="ROW" minOccurs="0" maxOccurs="unbounded">
                      <xsd:complexType>
                        <xsd:sequence>
                          <xsd:element name="PIN" type="xsd:string"/>
                          <xsd:element name="LASTNAME" type="xsd:string"/>
                          <xsd:element name="FIRSTNAME" type="xsd:string"/>
                          <xsd:element name="MIDINITIAL" type="xsd:string"/>
                          <xsd:element name="DIV" type="xsd:string"/>
                          <xsd:element name="DEPT" type="xsd:string"/>
                          <xsd:element name="JOBCLASS" type="xsd:string"/>
                          <xsd:element name="TITLE" type="xsd:string"/>
                          <xsd:element name="SALARY" type="xsd:float"/>
                          <xsd:element name="HIREDATE" type="xsd:date"/>
                          <xsd:element name="AREA" type="xsd:string"/>
                        </xsd:sequence>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example **Parameterized SQL Response Instance Document**

```
<?xml version="1.0" encoding="UTF-8"?>
<RESULT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\paramempdata_response.xsd">
  <paramempdata>
    <RESULTSET_1>
      <ROW>
        <PIN>String</PIN>
        <LASTNAME>String</LASTNAME>
        <FIRSTNAME>String</FIRSTNAME>
        <MIDINITIAL>String</MIDINITIAL>
        <DIV>String</DIV>
        <DEPT>String</DEPT>
        <JOBCLASS>String</JOBCLASS>
        <TITLE>String</TITLE>
        <SALARY>3.14159</SALARY>
        <HIREDATE>1967-08-13</HIREDATE>
        <AREA>String</AREA>
      </ROW>
      <ROW>
        <PIN>String</PIN>
        <LASTNAME>String</LASTNAME>
        <FIRSTNAME>String</FIRSTNAME>
        <MIDINITIAL>String</MIDINITIAL>
        <DIV>String</DIV>
        <DEPT>String</DEPT>
        <JOBCLASS>String</JOBCLASS>
        <TITLE>String</TITLE>
        <SALARY>3.14159</SALARY>
        <HIREDATE>1967-08-13</HIREDATE>
        <AREA>String</AREA>
      </ROW>
      <ROW>
        <PIN>String</PIN>
        <LASTNAME>String</LASTNAME>
        <FIRSTNAME>String</FIRSTNAME>
        <MIDINITIAL>String</MIDINITIAL>
        <DIV>String</DIV>
        <DEPT>String</DEPT>
        <JOBCLASS>String</JOBCLASS>
        <TITLE>String</TITLE>
        <SALARY>3.14159</SALARY>
        <HIREDATE>1967-08-13</HIREDATE>
        <AREA>String</AREA>
      </ROW>
    </RESULTSET_1>
  </paramempdata>
```

</RESULT>

Stored Procedure Schemas for an Oracle Database

The following examples are based on a schema created for a stored procedure for an Oracle database.

Example Stored Procedure Request Schema for an Oracle Database

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:12:21Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RDBMS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="PROCIN">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Y" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="location" type="xsd:string"
use="optional" fixed="RDBMS/Schemas/EDARPK/Procedures/PROCIN"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example Stored Procedure Request Instance Document for an Oracle Database

```
<?xml version="1.0" encoding="UTF-8"?>
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\PROCIN_request.xsd">
  <PROCIN location="RDBMS/Schemas/EDARPK/Procedures/PROCIN">
    <Y>String</Y>
  </PROCIN>
</RDBMS>
```

Example **Stored Procedure Response Schema for an Oracle Database**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:18:44Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RESULT">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="PROCIN">
          <xsd:complexType>
            <xsd:sequence/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example **Stored Procedure Response Instance Document for an Oracle Database**

```
<?xml version="1.0" encoding="UTF-8"?>
<RESULT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\PROCIN_response.xsd">
  <PROCIN/>
</RESULT>
```

Stored Procedure Schemas for a VSAM Database

Use the iWay Adapter to create schemas for an iWay stored procedure for a VSAM database.

Example **Stored Procedure Request Schema for a VSAM Database**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:05:56Z -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="RPCIn">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="1" type="xs:string"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="optional"
default="RPCVSM"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example Stored Procedure Request Instance Document for a VSAM Database

```
<?xml version="1.0" encoding="UTF-8"?>
<RPCIn xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\RPCVSM_request.xsd"
name="RPCVSM">
  <l>String</l>
</RPCIn>
```

Example Stored Procedure Response Schema for a VSAM Database

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2004-01-13T22:05:56Z -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="RPCOut">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Row" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="COMP_NAME" type="xs:string"/>
              <xs:element name="EMP_ID" type="xs:string"/>
              <xs:element name="EMPID" type="xs:string"/>
              <xs:element name="FIRST_NAME" type="xs:string"/>
              <xs:element name="LAST_NAME" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="status" type="xs:string" use="required"/>
      <xs:attribute name="reason" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example **Stored Procedure Response Instance Document for a VSAM Database**

```
<?xml version="1.0" encoding="UTF-8"?>
<RPCOut xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\RPCVSM_response.xsd"
status="String" reason="String">
  <Row>
    <COMP_NAME>String</COMP_NAME>
    <EMP_ID>String</EMP_ID>
    <EMPID>String</EMPID>
    <FIRST_NAME>String</FIRST_NAME>
    <LAST_NAME>String</LAST_NAME>
  </Row>
  <Row>
    <COMP_NAME>String</COMP_NAME>
    <EMP_ID>String</EMP_ID>
    <EMPID>String</EMPID>
    <FIRST_NAME>String</FIRST_NAME>
    <LAST_NAME>String</LAST_NAME>
  </Row>
  <Row>
    <COMP_NAME>String</COMP_NAME>
    <EMP_ID>String</EMP_ID>
    <EMPID>String</EMPID>
    <FIRST_NAME>String</FIRST_NAME>
    <LAST_NAME>String</LAST_NAME>
  </Row>
</RPCOut>
```

Multiple Executions of the Same SQL Statement or Stored Procedure

The iWay XML Adapter for RDBMS for BEA WebLogic allows you to execute a prepared SQL statement or stored procedure multiple times with different input parameters each time. The major benefits of this feature are as follows:

- **Connection Resource Utilization.** One connection or thread is established to the backend database. This minimizes resource consumption. Note that the number of cursors for select statements created is based on the number of parameter sets in the submitted request. If there are 3 sets of parameters for a select statement, there will be 3 cursors created and closed sequentially.
- **Logical Unit of Work (LUW).** A logical unit of work (LUW) is established that will roll back all of the updates or inserts that were issued by the SQL statement of prior executions. A transaction starts at the first set of parameters. The sets of parameters are executed in order from top to bottom of the message sequentially. For example, if an insert statement is submitted along with 3 sets of parameters, and the third set of parameters fails to insert, the previous 2 inserts will be rolled back.

- **Integration of Data From an Outside Source.** For integration scenarios where external data is used to "feed" SQL or stored procedures, this feature allows external data to be mapped to one XML execution block for the adapter to execute.

Note: Multiple processes in one message is not supported. Only one SQL statement or stored procedure can be executed in a single XML execution request block.

Example **Stored Procedure Request Instance Document for an SQL Server Database**

The following is a sample XML request document to execute a Microsoft SQL Server stored procedure called CustOrderHist. Note that the PARAMS tag surrounds the parameter(s) to be used as input. The adapter executes the stored procedure twice, with different input values for each execution.

```
<RDBMS>
  <ITERATE location="RDBMS/Schemas/dbo/Procedures/CustOrderHist">
    <PARAMS>
      <parm>ALFKI</parm>
    </PARAMS>
    <PARAMS>
      <parm>NNN*</parm>
    </PARAMS>
  </ITERATE>
</RDBMS>
```

Example **Stored Procedure Response Instance Document for an SQL Server Database**

The following is a sample XML response document from the stored procedure CustOrderHist. Note the multiple result sets that correspond to the two executions of the stored procedure. The second and last result set contains 0 records found for the input value "NNN*".

```
<?xml version="1.0" encoding="UTF-8" ?>
- <RESULTS>
- <RESULT>
- <CustOrderHist>
- <RESULTSET_1>
- <ROW>
  <ProductName>Aniseed Syrup</ProductName>
  <Total>6</Total>
</ROW>
- <ROW>
  <ProductName>Chartreuse verte</ProductName>
  <Total>21</Total>
</ROW>
```



```
- <ROW>
  <ProductName>Escargots de Bourgogne</ProductName>
  <Total>40</Total>
</ROW>
- <ROW>
  <ProductName>Flotemysost</ProductName>
  <Total>20</Total>
</ROW>
- <ROW>
  <ProductName>Grandma's Boysenberry Spread</ProductName>
  <Total>16</Total>
</ROW>
- <ROW>
  <ProductName>Lakkalikööri</ProductName>
  <Total>15</Total>
</ROW>
- <ROW>
  <ProductName>Original Frankfurter grüne Soße</ProductName>
  <Total>2</Total>
</ROW>
- <ROW>
  <ProductName>Raclette Courdavault</ProductName>
  <Total>15</Total>
</ROW>
</RESULTSET_1>
<RETURN_VALUE>0</RETURN_VALUE>
</CustOrderHist>
</RESULT>
- <RESULT>
- <CustOrderHist>
  <RESULTSET_1 />
  <RETURN_VALUE>0</RETURN_VALUE>
</CustOrderHist>
</RESULT>
</RESULTS>
```

Understanding iWay Business Services Engine

Application Explorer provides Web developers with a simple, consistent mechanism for extending the capabilities of the adapter. The iWay Business Services Engine exposes functionality as Web services. It serves as a gateway to heterogeneous back-end applications and databases.

A Web service is a self-contained, modularized function that can be published and accessed across a network using open standards. It is the implementation of an interface by a component and is an executable entity. For the caller or sender, a Web service can be considered as a “black box” that may require input and delivers a result. A Web service integrates within an enterprise as well as across enterprises on any communication technology stack, whether asynchronous or synchronous, in any format.

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

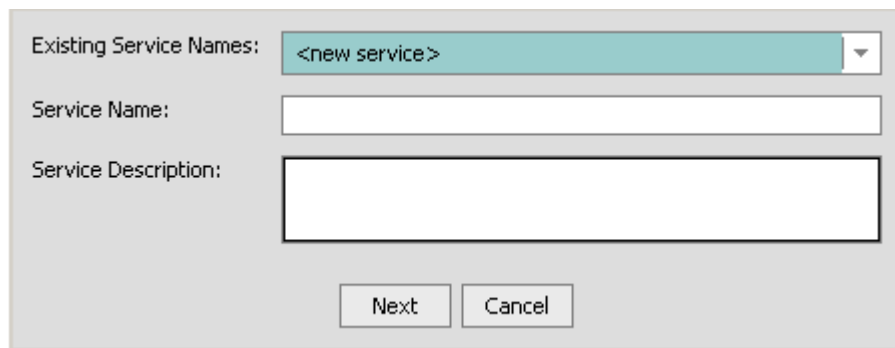
Creating a Business Service

You can create a business service for an SQL statement or a stored procedure.

Procedure How to Generate a Business Service

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page A-9.
2. Expand the node to display the statements or procedures.
3. Right-click the SQL statement or stored procedure for which you want to create a business service, and select *Create iWay Business Service*.

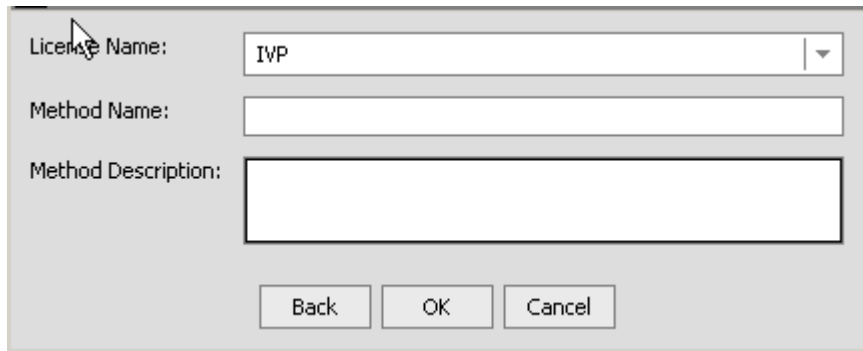
The Create iWay Business Service dialog box opens:

The image shows a 'Create iWay Business Service' dialog box. It has a light gray background. At the top, there is a label 'Existing Service Names:' followed by a dropdown menu showing '<new service>'. Below this is a label 'Service Name:' followed by a text input field. Underneath that is a label 'Service Description:' followed by a larger text area. At the bottom of the dialog, there are two buttons: 'Next' and 'Cancel'.

4. Choose whether to create a new service or use an existing service.

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

The license and method dialog box opens:

A screenshot of a 'License and Method' dialog box. It has a light gray background. At the top, there is a label 'License Name:' followed by a dropdown menu showing 'IVP'. Below that is a label 'Method Name:' followed by an empty text input field. Underneath is a label 'Method Description:' followed by a larger empty text area. At the bottom, there are three buttons: 'Back', 'OK', and 'Cancel'.

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

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

Testing a Business Service

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

Procedure How to Test a Business Service

1. Expand the *iWay Business Services* node.
2. Expand the *Services* node.
3. Select the name of the business service you want to test.

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

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

The test option appears in the right pane.

- If you are testing a Web service that requires XML input, an input field appears, as shown by the following illustration:

The screenshot shows a web interface with a vertical sidebar on the left containing 'Add' and 'Test' links. The main area is titled 'Test' and contains the text: 'To test the operation using the [SOAP protocol](#), click the 'Invoke' button.' Below this is a large text area labeled 'input xml:'. At the bottom of the main area, there is a small text input field, a 'Browse...' button, and an 'Invoke' button.

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

- If you are testing a Web service for a parameterized SQL statement, an input area appears, as shown by the following illustration:

The screenshot shows a web interface with a vertical sidebar on the left containing 'Service' and 'Test' links. The main area is titled 'Test' and contains the text: 'To test the operation using the [SOAP protocol](#), click the 'Invoke' button.' Below this is a table with two columns: 'Parameter' and 'Value'. The first row has 'parameter1:' in the 'Parameter' column and an empty text input field in the 'Value' column. At the bottom right of the main area, there is an 'Invoke' button.

5. Provide the appropriate input.
6. Click *Invoke*.

Application Explorer displays the results in the right pane:

The image shows a screenshot of the Application Explorer's right pane, which displays an XML response. The XML is a SOAP message. The root element is <?xml version="1.0" encoding="UTF-8" ?>. It contains a <SOAP-ENV:Envelope> element with namespaces for xsd, SOAP-ENV, and xsi. Inside the envelope is a <SOAP-ENV:Body> element containing a <SPmethResponse> element. The SPmethResponse element has a namespace of urn:iwaysoftware:ibse:jul2003:SPmeth:response and a cid attribute with the value "16B82D8AAFC70DF1B04C2C80028A05C8". It contains a <RPCOut> element with a reason attribute and a status attribute set to "success". The RPCOut element contains a <Row> element, which in turn contains a <DBLNAME>SYSDATABASE</DBLNAME> element. This is followed by a series of elements: <EDAACCESS>R</EDAACCESS>, <ISOLAT>C</ISOLAT>, <IDENTIFY>N</IDENTIFY>, <ENGINE>EDA</ENGINE>, and <DBDESCR>EDA Database Information Tables</DBDESCR>. The <Row> element is closed, and then another <Row> element is shown, which contains <DBLNAME>SYSEXTENDED</DBLNAME>, <EDAACCESS>R</EDAACCESS>, <ISOLAT>C</ISOLAT>, and <IDENTIFY>N</IDENTIFY>. The XML is displayed in a text editor with a scrollbar on the right and a status bar at the bottom.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance">
- <SOAP-ENV:Body>
  - <SPmethResponse
    xmlns="urn:iwaysoftware:ibse:jul2003:SPmeth:respo
    cid="16B82D8AAFC70DF1B04C2C80028A05C8">
  - <RPCOut reason="" status="success">
    - <Row>

      <DBLNAME>SYSDATABASE</DBLNAME>
      <EDAACCESS>R</EDAACCESS>
      <ISOLAT>C</ISOLAT>
      <IDENTIFY>N</IDENTIFY>
      <ENGINE>EDA</ENGINE>
      <DBDESCR>EDA Database
      Information Tables</DBDESCR>
    </Row>
  - <Row>

      <DBLNAME>SYSEXTENDED</DBLNAME>
      <EDAACCESS>R</EDAACCESS>
      <ISOLAT>C</ISOLAT>
      <IDENTIFY>N</IDENTIFY>
```

Generating WSDL From a Web Service

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

Procedure How to Generate WSDL From a Web Service

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

The Save dialog box opens.

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

Note: The file extension must be .wsdl.

5. Click *Save*.

Example Viewing WSDL Generated from a Web Service

The following is an example of a WSDL file for a Web service called MPS generated from a parameterized SQL statement against a VSAM database.

```
<definitions xmlns:tns="urn:schemas-iwaysoftware-com:iwse"
targetNamespace="urn:schemas-iwaysoftware-com:iwse"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:m1="urn:iwaysoftware:ibse:jul2003:VSAM:response"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:m1="urn:iwaysoftware:ibse:jul2003:VSAM"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"><types><xs:schema
targetNamespace="urn:schemas-iwaysoftware-com:iwse"
elementFormDefault="qualified"><xs:element
name="ibsinfo"><xs:complexType><xs:sequence><xs:element type="xs:string"
name="service"/><xs:element type="xs:string" name="method"/><xs:element
type="xs:string" name="license"/><xs:element type="xs:string"
minOccurs="0" name="disposition"/><xs:element type="xs:string"
minOccurs="0" name="Username"/><xs:element type="xs:string" minOccurs="0"
name="Password"/><xs:element type="xs:string" minOccurs="0"
name="language"/></xs:sequence></xs:complexType></xs:element>
</xs:schema><xs:schema
targetNamespace="urn:schemas-iwaysoftware-com:iwse"
elementFormDefault="qualified"><xs:element
name="adapterexception"><xs:complexType><xs:sequence><xs:element
type="xs:string"
name="error"/></xs:sequence></xs:complexType></xs:element>
</xs:schema><xs:schema
targetNamespace="urn:iwaysoftware:ibse:jul2003:VSAM"
xmlns:m1="urn:iwaysoftware:ibse:jul2003:VSAM"
elementFormDefault="qualified"><xs:element
name="VSAM"><xs:complexType><xs:sequence><xs:element type="xs:string"
name="emp_id"/></xs:sequence></xs:complexType></xs:element>
</xs:schema><xs:schema
targetNamespace="urn:iwaysoftware:ibse:jul2003:VSAM:response"
xmlns:m1="urn:iwaysoftware:ibse:jul2003:VSAM:response"
elementFormDefault="qualified"><xs:element
name="VSAMResponse"><xs:complexType><xs:sequence><xs:element
name="RESULT"><xs:complexType><xs:sequence><xs:element
name="MPSVSAM"><xs:complexType><xs:sequence><xs:element minOccurs="0"
name="RESULTSET_1"><xs:complexType><xs:sequence><xs:element minOccurs="0"
name="ROW" maxOccurs="unbounded"><xs:complexType><xs:sequence><xs:element
type="xs:string" name="EMP_ID"/><xs:element type="xs:string"
name="FIRST_NAME"/><xs:element type="xs:string"
name="LAST_NAME"/><xs:element type="xs:string" name="DEPT"/><xs:element
type="xs:string"
```

```
<name="COMP_NAME"/></xs:sequence></xs:complexType></xs:element></xs:sequence></xs:complexType></xs:element></xs:sequence></xs:complexType></xs:element>
</xs:sequence><xs:attribute type="xs:string" use="required"
name="cid"/></xs:complexType></xs:element></xs:schema>    </types><message
name="VSAMIn"><part element="m1:VSAM" name="parameters"/>
</message><message name="VSAMOut"><part element="m11:VSAMResponse"
name="parameters"/>    </message><message name="MPSHeader"><part
element="tns:ibsinfo" name="header"/>    </message><message
name="AdapterException"><part element="tns:adapterexception"
name="fault"/>
    </message><portType name="MPSSoap"><operation
name="VSAM"><documentation/><input message="tns:VSAMIn"/><output
message="tns:VSAMOut"/><fault message="tns:AdapterException"
name="AdapterExceptionFault"/></operation>
    </portType><binding type="tns:MPSSoap" name="MPSSoap"><soap:binding
style="document"
transport="http://schemas.xmlsoap.org/soap/http"/><operation
name="VSAM"><soap:operation style="document"
soapAction="MPS.VSAMRequest@production@@"/><input><soap:body
use="literal"/><soap:header part="header" message="tns:MPSHeader"
use="literal"/>
        </input><output><soap:body use="literal"/>
        </output><fault name="AdapterExceptionFault"><soap:fault
use="literal" name="AdapterExceptionFault"/></fault></operation>
    </binding><service name="MPS"><documentation>MPS</documentation><port
binding="tns:MPSSoap" name="MPSSoap1"><soap:address
location="http://iwayntk1.7001/ibse/IBSEServlet/XDSOAPRouter"/></port></s
ervice></definitions>
```

For more information on using WSDL in BEA WebLogic Workshop, including an example, see Appendix C, *Using WebLogic Workshop to Access VSAM*.

Identity Propagation

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

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

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

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

Adding a Control for an iWay Resource in BEA WebLogic Workshop

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

Adding a Web Service Control to a BEA WebLogic Workshop Application

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

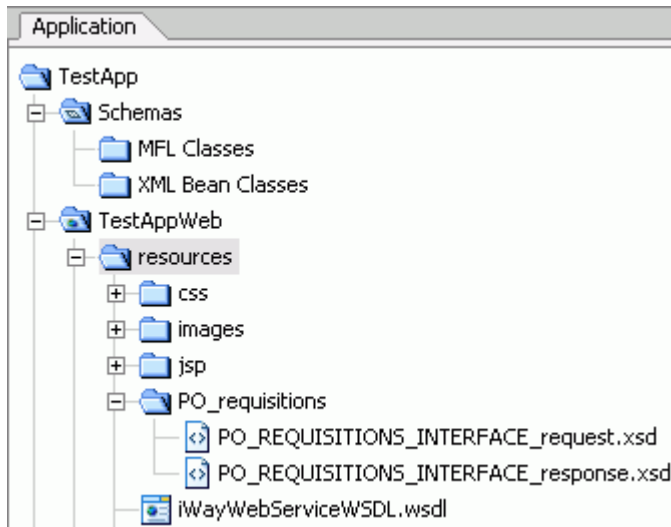
For more information on exporting a WSDL file, see *How to Generate WSDL From a Web Service* on page A-42.

Procedure How to Add a Web Service Control

To add a Web service control:

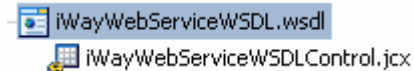
1. After exporting the WSDL file from Application Explorer, locate the file in the Application tab of your BEA WebLogic Workshop application.

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



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

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



Adding an iWay Extensible CCI Control to a BEA WebLogic Workshop Application

An iWay control enables access to resources provided by Application Explorer when it is used in conjunction with a JCA deployment. You must add an iWay control before using it in a BEA WebLogic Workshop application workflow.

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

Overview

The extensible iWay CCI control provides:

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

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

```
public ResponseDataType MethodName(RequestDataType VariableName) throws  
Exception;
```

where:

ResponseDataType

Is the XML Bean Class value that is generated from the response schema.

MethodName

Is the method name used by the extensible CCI control.

RequestDataType

Is the XML Bean Class value that is generated from the request schema.

VariableName

Is the request variable that stores the request document, which is used as input by the extensible CCI control.

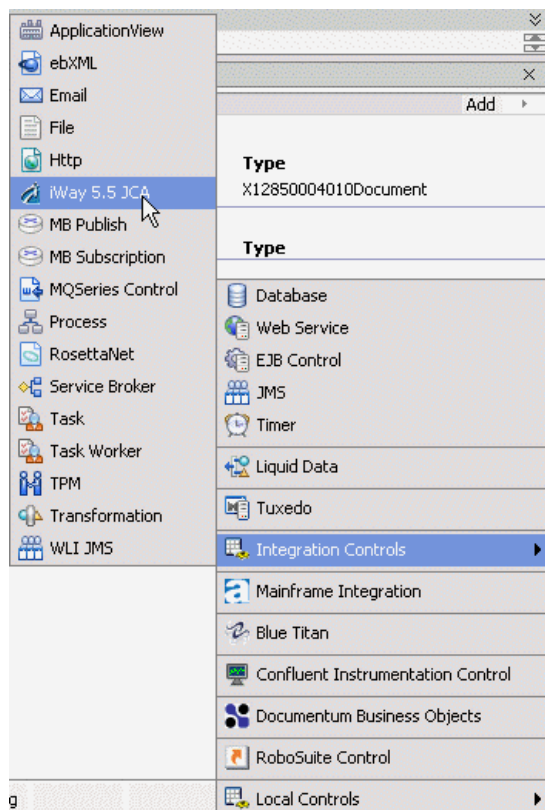
In addition, the extensible CCI control now generates a JCX file to which you can add your own methods. For more information, see *Defining a Control Using the Extensible CCI Control* on page A-47.

You can also use dynamic class casting to specify schema-based input or output XmlObjects to be casted into a pure XmlObject as a service method, which is expected by the CCI control. For more information, see *Using Dynamic Class Casting* on page A-53.

Example Defining a Control Using the Extensible CCI Control

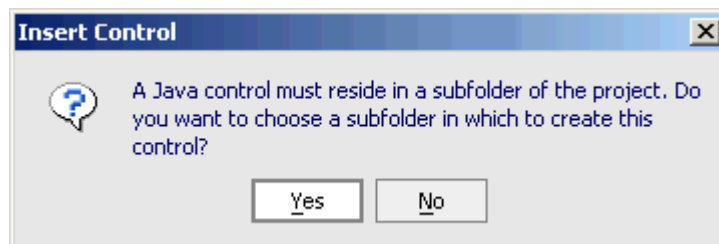
The following sample JCX demonstrates how to define a control for RDBMS using the extensible CCI control in BEA WebLogic Workshop.

1. Start BEA WebLogic Workshop and create a new project.



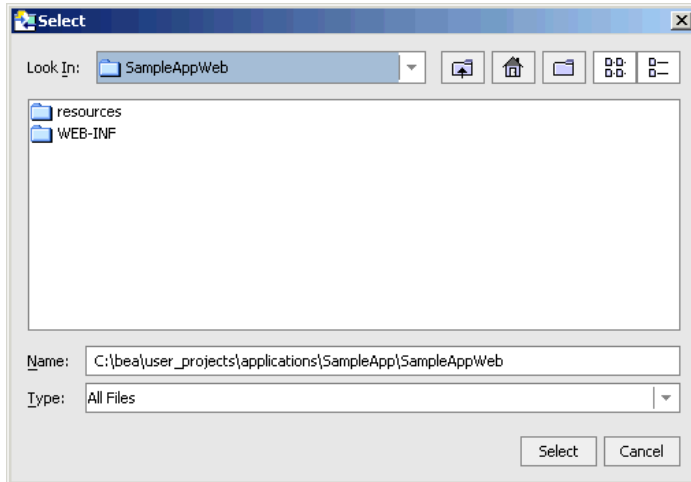
2. Click *Add* from the Controls section in the Data Palette tab, select *Integration Controls*, and click *iWay 5.5 JCA*.

The Insert Control message box opens.



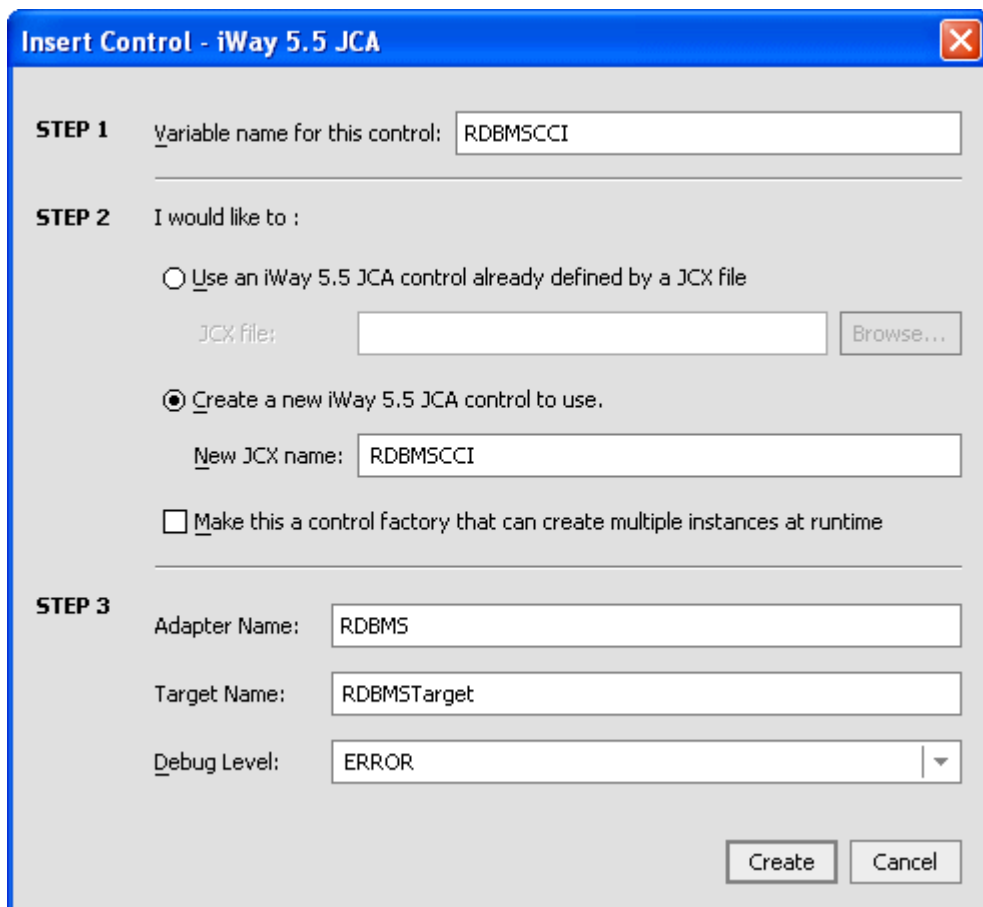
3. Click *Yes*.

The Select dialog box opens.



4. Choose a subfolder for the CCI control and click *Select*.

The Insert Control - iWay 5.5 JCA dialog box opens.



The dialog box is titled "Insert Control - iWay 5.5 JCA" and contains three steps for configuring a new JCA control.

STEP 1 Variable name for this control:

STEP 2 I would like to :

☐ Use an iWay 5.5 JCA control already defined by a JCX file

JCX file:

☒ Create a new iWay 5.5 JCA control to use.

New JCX name:

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

STEP 3

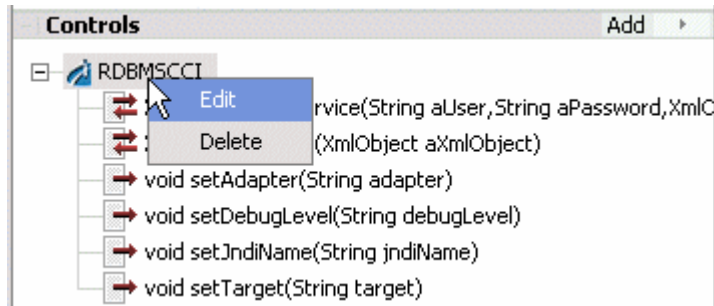
Adapter Name:

Target Name:

Debug Level:

- a. Provide a variable name for the control.
 - b. Click *Create a new iWay 5.5 JCA control to use* and provide a new JCX name.
 - c. Enter the adapter name, target name, and select a debug level from the drop-down list.
5. Click *Create*.

A new JCX file is created.

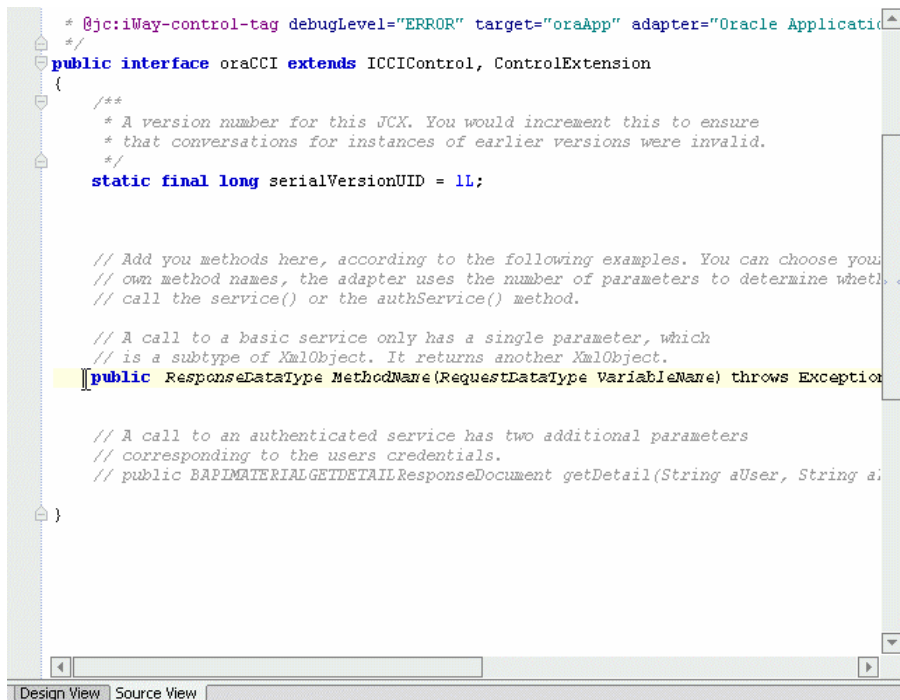


6. Right-click the control, for example, RDBMSCCI, and select *Edit*.

The Design View for the control opens.

7. Click the *Source View* tab.

The Source View for the control opens.

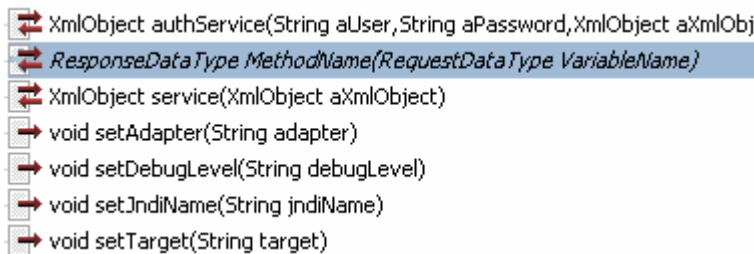


Perform the following steps:

- a. Uncomment the public class definition.

- b. Change the existing response data type to match your response data type that is generated from your RDBMS response schema.
- c. Change the existing method name to match your method.
- d. Change the existing request data type to match your request data type that is generated from your RDBMS request schema.

The following control is now available in BEA WebLogic Workshop and can be added to a workflow:



Note: You can view available data types under the *XML Bean Classes* folder in the *Application* tab, which are added once you import your XML request or response schemas from Application Explorer.

These data types are case sensitive and must be entered exactly as shown.

Using the Extensible CCI Control

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

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

For example, instead of the generic `XmlObject`:

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

you call:

```
public ResponseDataType MethodName(RequestDataType VariableName) throws  
Exception;
```

where:

```
ResponseDataType
```

Is the XML Bean Class value that is generated from the response schema.

MethodName

Is the method name used by the extensible CCI control.

RequestDataType

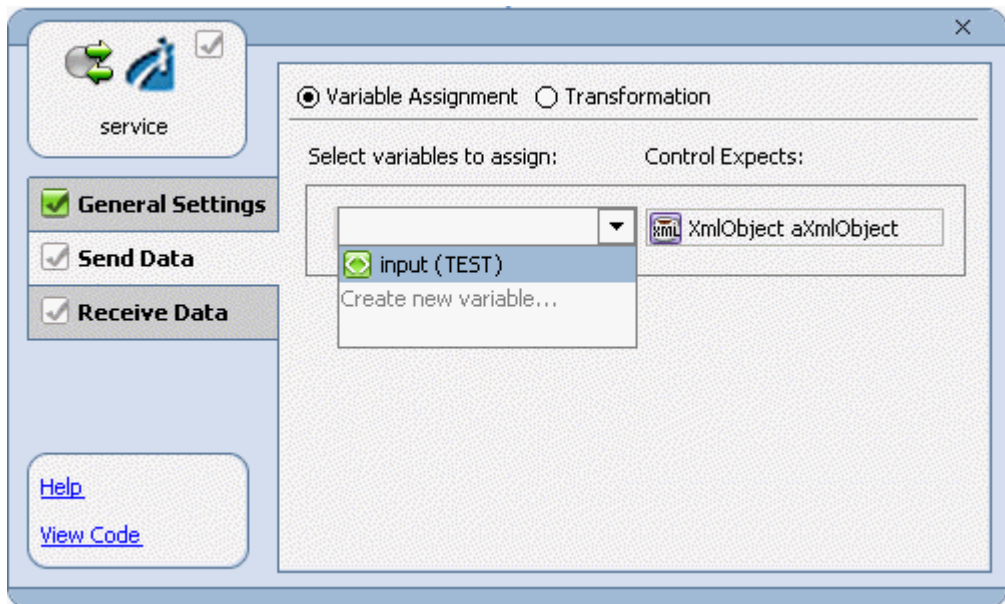
Is the XML Bean Class value that is generated from the request schema.

VariableName

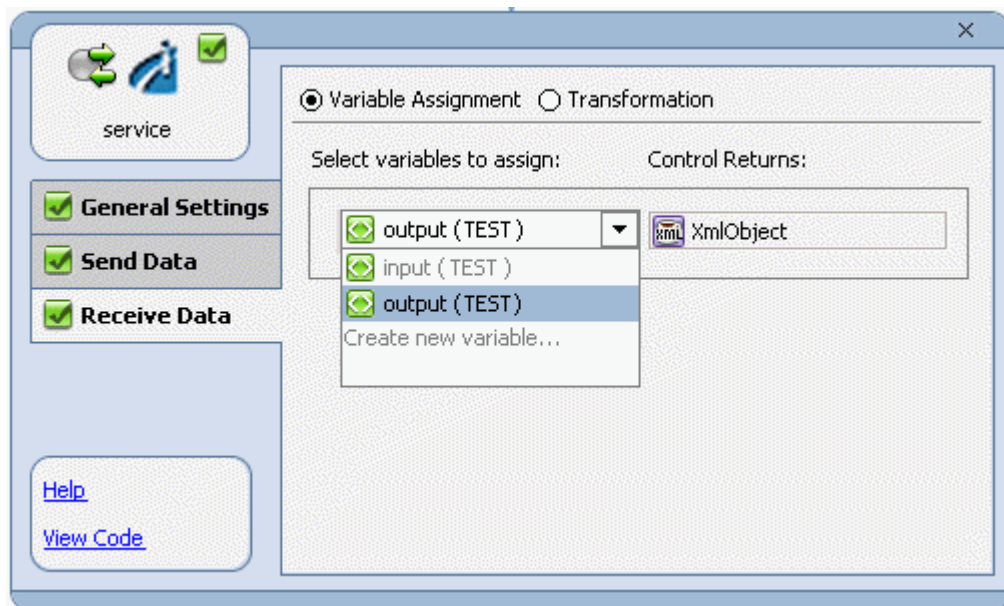
Is the request variable that stores the request document, which is used as input by the extensible CCI control.

Example Using Dynamic Class Casting

The following example uses dynamic class casting to specify a schema-based input XmlObject to be casted into a pure XmlObject as a service method, which is expected by the CCI control.



The following example uses dynamic class casting where the CCI control returns a pure XmlObject, which is casted dynamically into a schema-based output XmlObject.



APPENDIX B

Using Application Explorer in BEA WebLogic Workshop for Event Handling

Topics:

- Starting Application Explorer in WebLogic Workshop
- Understanding iWay Event Functionality
- Creating, Editing, and Deleting an Event Port
- Creating, Editing, and Deleting an Event Channel
- Deploying iWay Components in a Clustered BEA WebLogic Environment
- For More Information

This section describes how to use the Java Swing implementation of Application Explorer as deployed in BEA WebLogic Workshop. Application Explorer deployed in WebLogic Workshop is functionally similar to the Servlet Application Explorer.

This section describes how to use the iWay XML Adapter for RDBMS for BEA WebLogic to listen for events in a relational table. Several listening techniques are available, enabling you to choose the technique that best suits your requirements.

Starting Application Explorer in WebLogic Workshop

You can use Application Explorer with an iBSE or a JCA configuration. Before you can use Application Explorer, you must start BEA WebLogic Server.

Procedure How to Start Application Explorer

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

Application Explorer opens in BEA WebLogic Workshop.



You can resize and drag-and-drop the Application Explorer window within BEA WebLogic Workshop. For example, you can drag it to the upper part of BEA WebLogic Workshop.

Understanding iWay Event Functionality

This section describes how to use the iWay XML Adapter for RDBMS for BEA WebLogic to listen for events in a relational table. Several listening techniques are available, enabling you to choose the technique that best suits your requirements.

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

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

- **Port**

A port associates a particular business object exposed by an adapter with a particular disposition. A disposition defines the protocol and resulting location of the event data. The port defines the end point of the event consumption. For more information, see *Creating, Editing, and Deleting an Event Port*.

- **Channel**

A channel represents configured connections to particular instances of back-end or other types of systems. A channel binds one or more event ports to a particular listener managed by an adapter. For more information, see *Creating, Editing, and Deleting an Event Channel* on page B-18.

Creating, Editing, and Deleting an Event Port

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

Creating an Event Port From the iWay Event Adapters Tab

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

- File
- iBSE
- MSMQ
- JMSQ
- SOAP

- HTTP
- MQ Series

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

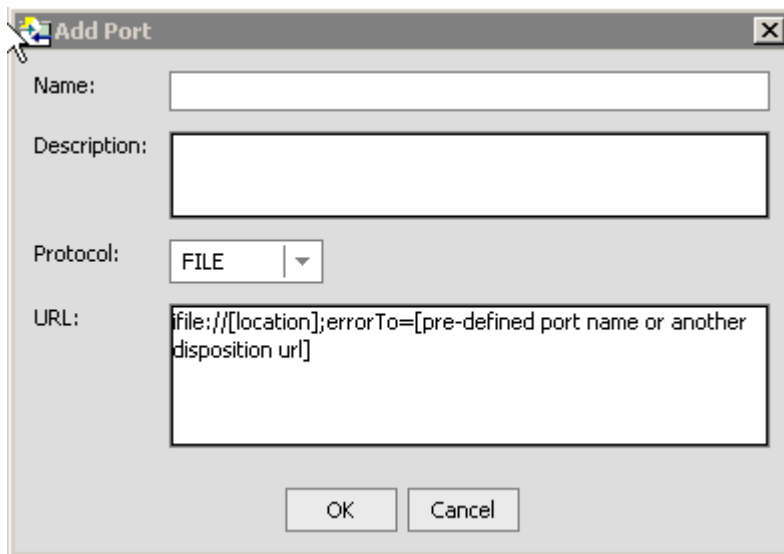
The following dispositions are available when using Application Explorer in conjunction with a JCA connector deployment.

- File
- JMSQ
- HTTP
- MQ Series

Procedure How to Create an Event Port for File

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

The Add Port dialog box opens:

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

- a. Type a name for the event port and provide a brief description.
- b. From the Protocol drop-down list, select *File*.
- c. In the URL field, specify a destination file to which the event data is written.

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

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

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

The following table lists and defines the parameters for the File disposition:

Parameter	Description
location	Destination and file name of the document where event data is written. For example, D:\in\x.txt
errorTo	Location to which error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

4. Click *OK*.

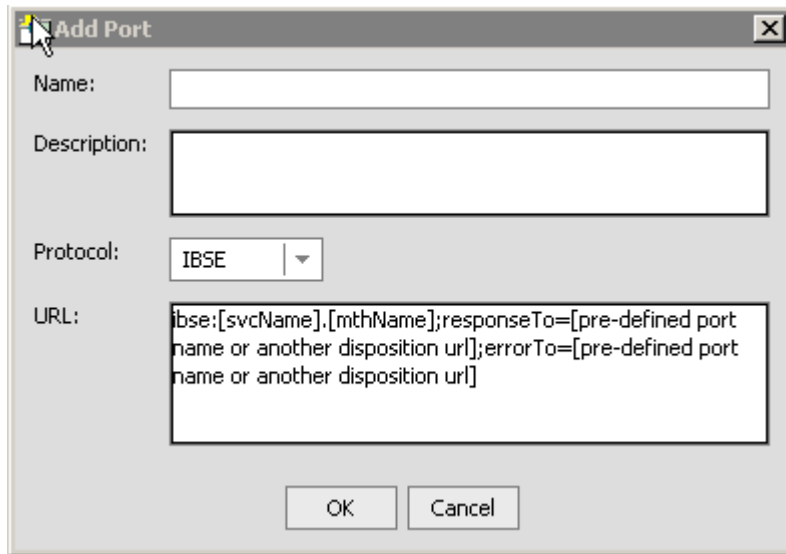
The port appears under the ports node in the left pane. To review the port settings, select the port name. A table summarizing the port settings appears in the right pane.

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting an Event Channel* on page B-18.

Procedure How to Create an Event Port for iBSE

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

The Add Port dialog box opens:

The image shows a dialog box titled "Add Port" with a close button (X) in the top right corner. It contains four input fields: "Name:" (a single-line text box), "Description:" (a multi-line text box), "Protocol:" (a dropdown menu with "IBSE" selected), and "URL:" (a multi-line text box). The URL field contains the text: `ibse:[svcName].[mthName];responseTo=[pre-defined port name or another disposition url];errorTo=[pre-defined port name or another disposition url]`. At the bottom are "OK" and "Cancel" buttons.

Add Port

Name:

Description:

Protocol:

URL:

- a. Type a name for the event port and provide a brief description.
- b. From the Protocol drop-down list, select *ibSE*.
- c. In the URL field, enter an iBSE destination in the following format:
`ibse:[svcName].[mthName];responseTo=[pre-defined port name or another disposition url];errorTo=[pre-defined port name or another disposition url]`

The following table lists and defines the parameters for the iBSE disposition:

Parameter	Description
svcName	Name of the service created with iBSE.
mthName	Name of the method created for the Web service.
responseTo	Location to which responses are posted. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.
errorTo	Location to which error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

4. Click *OK*.

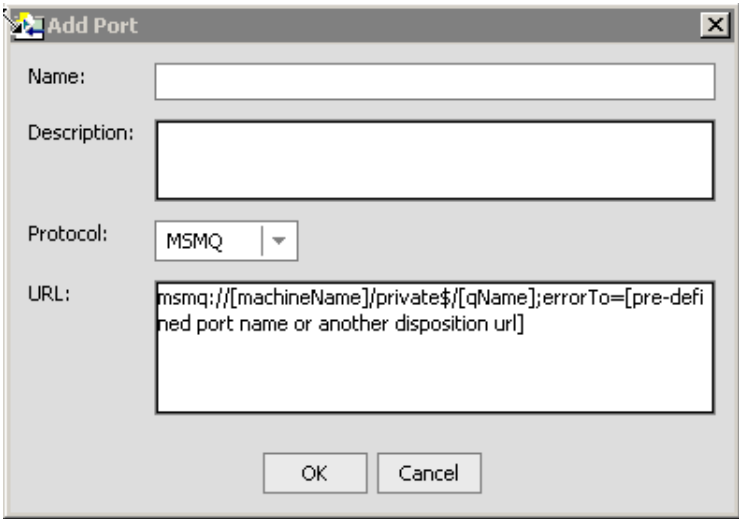
The port appears under the ports node in the left pane. To review the port settings, select the port name. A table summarizing the port settings appears in the right pane.

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting an Event Channel* on page B-18.

Procedure How to Create an Event Port for MSMQ

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

The Add Port dialog box opens:



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

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

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

The following table lists and defines the parameters for the MSMQ disposition:

Parameter	Description
machineName	Machine name where the Microsoft Queuing system is running.
qName	Name of the private queue where messages are placed.
errorTo	Location to which error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

- 4. Click *OK*.

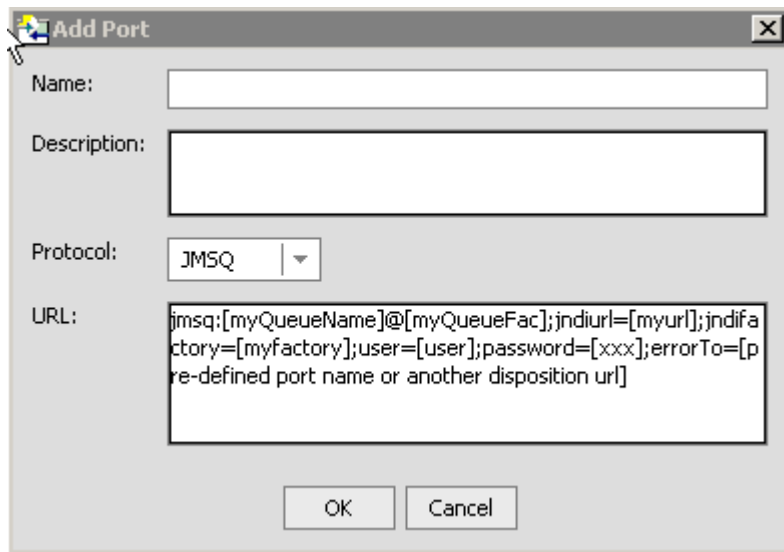
The port appears under the ports node in the left pane. To review the port settings, select the port name. A table summarizing the port settings appears in the right pane.

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting an Event Channel* on page B-18.

Procedure How to Create an Event Port for JMSQ

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

The Add Port dialog box opens:

The image shows a dialog box titled "Add Port" with a standard Windows-style title bar (minimize, maximize, close buttons). The dialog contains four labeled fields: "Name:" with a single-line text input; "Description:" with a multi-line text input; "Protocol:" with a drop-down menu currently showing "JMSQ"; and "URL:" with a multi-line text input containing a JMSQ URL template: `jmsq:[myQueueName]@[myQueueFac];jndiurl=[myurl];jndifa
ctory=[myfactory];user=[user];password=[xxx];errorTo=[p
re-defined port name or another disposition url]`. At the bottom of the dialog are two buttons: "OK" and "Cancel".

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

- c. In the URL field, enter a JMS destination.

When pointing Application Explorer to an **iBSE** deployment, use the following format:

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

When pointing Application Explorer to a **JCA** deployment, use the following format:

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

The following table lists and defines the parameters for the JMSQ disposition:

Parameter	Description
myQueueName or jmsqueue	JNDI name of a queue to which events are emitted.
myQueueFac or jmsfactory	Resource that contains information about the JMS Server. The WebLogic connection factory is: <code>javax.jms.QueueConnectionFactory</code>
jndiurl	URL to use to contact the JNDI provider. The syntax of this URL depends on the JNDI provider being used. This value corresponds to the standard JNDI property, <code>java.naming.provider.url.</code> The URL of the WebLogic Server is <code>t3://host:port</code> where: <code>host</code> Is the machine name where WebLogic Server is installed. <code>port</code> Is the port on which WebLogic Server is listening. The default port, if not changed at installation, is 7001.
jndifactory	Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider. For WebLogic Server, the WebLogic factory is: <code>weblogic.jndi.WLInitialContextFactory.</code>
user	Valid user name required to access a JMS server.

Parameter	Description
password	Valid password required to access a JMS server.
errorTo	Location to which error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

4. Click *OK*.

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

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting an Event Channel* on page B-18.

Procedure How to Create an Event Port for SOAP

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

The Add Port dialog box opens:

Add Port

Name:

Description:

Protocol: SOAP ▾

URL:

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

- c. In the URL field, enter a SOAP destination in the following format:

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

The following table lists and defines the parameters for the SOAP disposition:

Parameter	Description
wsdl-url	<p>The URL to the WSDL file that is required to create the SOAP message. For example:</p> <pre>http://localhost:7001/ibse/IBSEServlet/test/ webservice.ibs?wsdl</pre> <p>where:</p> <pre>webservice</pre> <p>Is the name of the Web service you created using Application Explorer.</p> <p>This value can be found by navigating to the Integration Business Services tab and opening the <i>Service Description</i> link in a new window. The WSDL URL appears in the Address field.</p> <p>You can also open the WSDL file in a third party XML editor (for example, XMLSPY) and view the SOAP request settings to find this value.</p>
soapaction	The method that will be called by the SOAP disposition. This value can be found in the WSDL file.
method	Web service method you are using. This value can be found in the WSDL file.
namespace	XML namespace you are using. This value can be found in the WSDL file.
responseTo	<p>Location to which responses are posted. Optional.</p> <p>A predefined port name or another disposition URL. The URL must be complete, including the protocol.</p>
errorTo	<p>Location to which error logs are sent. Optional.</p> <p>A predefined port name or another disposition URL. The URL must be complete, including the protocol.</p>

4. Click OK.

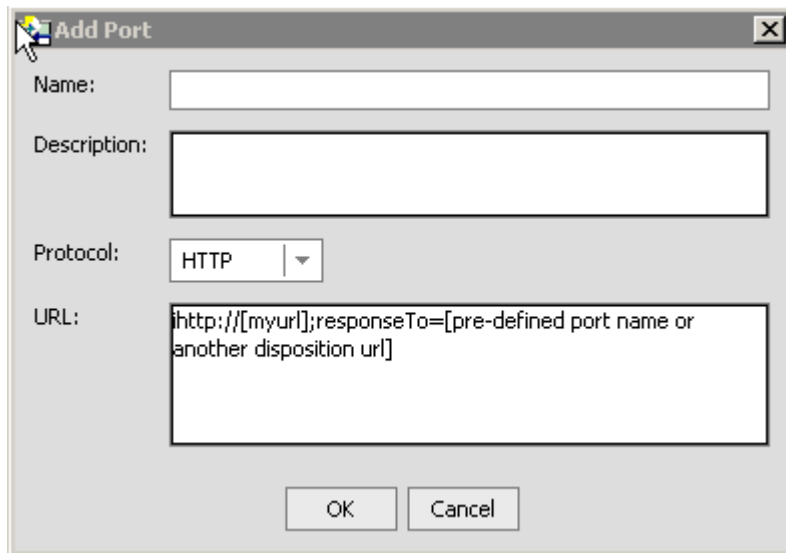
The port appears under the ports node in the left pane. To review the port settings, select the port name. A table summarizing the port settings appears in the right pane.

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting an Event Channel* on page B-18.

Procedure How to Create an Event Port for HTTP

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

The Add Port dialog box opens:

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

- a. Type a name for the event port and provide a brief description.
- b. From the Protocol drop-down list, select *HTTP*.
- c. In the URL field, enter an HTTP destination.

When pointing Application Explorer to an **iBSE** deployment, use the following format:

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

When pointing Application Explorer to a **JCA** deployment, use the following format:

<http://host:port/uri>

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

Parameter	Description
myurl	URL target for the post operation, for example, http://myhost:1234/docroot
responseTo	Location to which responses are posted. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

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

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

4. Click OK.

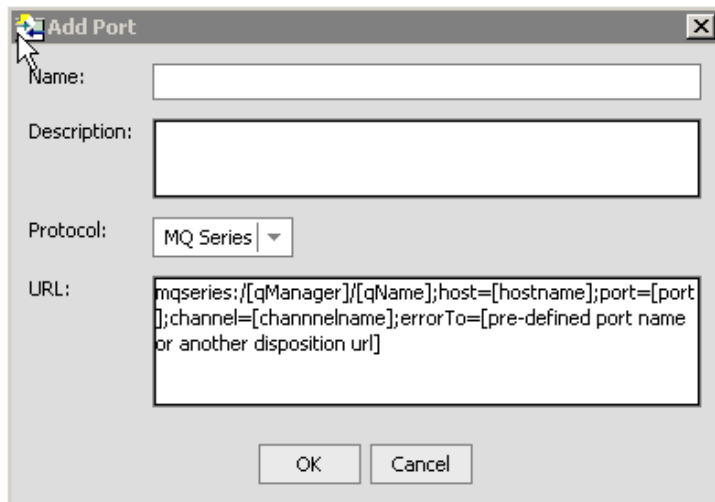
The port appears under the ports node in the left pane. To review the port settings, select the port name. A table summarizing the port settings appears in the right pane.

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting an Event Channel* on page B-18.

Procedure How to Create an Event Port for MQ Series

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

The Add Port dialog box opens:

The image shows a dialog box titled "Add Port" with a close button in the top right corner. It contains four fields: "Name:" with a text input box, "Description:" with a larger text area, "Protocol:" with a dropdown menu showing "MQ Series", and "URL:" with a text area containing a template URL: `mqseries:/[qManager]/[qName];host=[hostname];port=[port];channel=[channelname];errorTo=[pre-defined port name or another disposition url]`. At the bottom are "OK" and "Cancel" buttons.

- a. Type a name for the event port and provide a brief description.
- b. From the Protocol drop-down list, select *MQ Series*.
- c. In the URL field, enter an MQ Series destination.

When pointing Application Explorer to an **iBSE** deployment, use the following format:

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

When pointing Application Explorer to a **JCA** deployment, use the following format:

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

The following table lists and defines the parameters for the MQ Series disposition:

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

4. Click **OK**.

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

You are ready to associate the event port with a channel. For more information, see *Creating, Editing, and Deleting an Event Channel* on page B-18.

Editing and Deleting an Event Port

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

Procedure How to Edit an Event Port

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

Procedure How to Delete an Event Port

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

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

Creating, Editing, and Deleting an Event Channel

The following topics describe how to create, edit, and delete a channel for your iWay Event. All defined event ports must be associated with a channel.

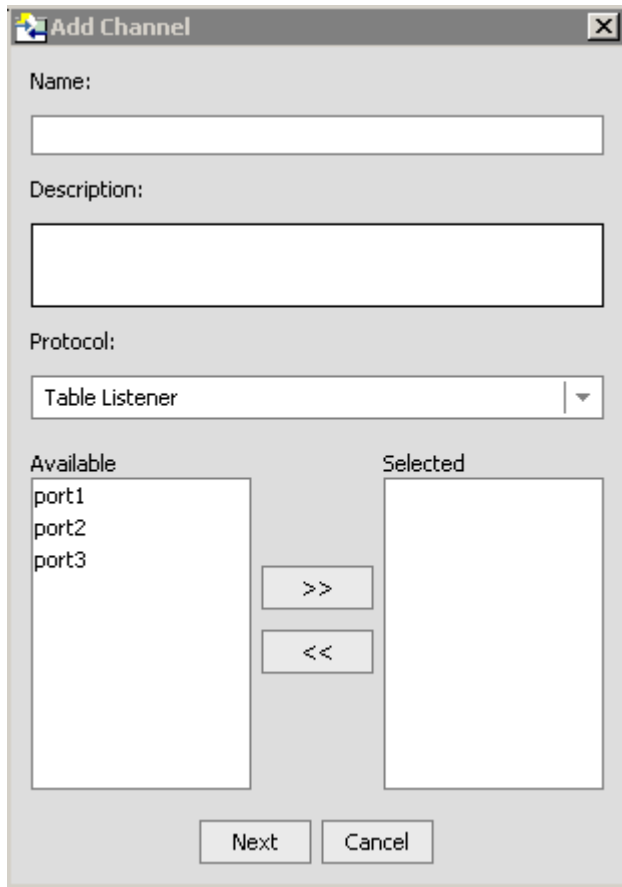
Creating a Channel

The following procedure describes how to create a channel.

Procedure How to Create a Channel

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

The Add Channel dialog box opens:

The image shows a Windows-style dialog box titled "Add Channel". It has a standard title bar with a close button (X). The dialog contains several fields: a "Name:" label followed by a text input box; a "Description:" label followed by a larger text input box; and a "Protocol:" label followed by a drop-down menu currently showing "Table Listener". Below these are two list boxes: "Available" on the left containing "port1", "port2", and "port3"; and "Selected" on the right, which is currently empty. Between the two list boxes are two buttons: ">>" (double right arrow) and "<<" (double left arrow). At the bottom of the dialog are two buttons: "Next" and "Cancel".

4. Specify information for the channel you are creating.
 - a. Type a name for the channel (for example, Channel1) and provide a brief description.
 - b. From the *Protocol* drop-down list, select a protocol (for example, Table Listener).
 - c. Select an event port from the list of available ports. To select more than one, hold down the *Ctrl* key and click the ports.
 - d. Click the double right arrow (>>) to transfer the port(s) to the list of selected ports.
5. Click *Next*.

The Table Listener dialog box opens:

Table Listener

JDBC-ODBC Bridge Listener

EDA Server Listener

SQL Server Listener

Oracle Listener

Host*

Port*

SID*

User*

Password*

Polling Interval*

SQL Query*

Post Query*

OK Cancel

Fields marked with * are required.

6. Select either a JDBC-ODBC Bridge, EDA Server, SQL Server, or Oracle Listener.

Note: If you are configuring listening capabilities for a non-relational database, select EDA Server Listener.

The following table lists and describes the parameters for all of the listeners.

Parameter	Description
Host	Name or URL of the machine where the database is installed.
Port	Port on which the Host database is listening.
Database Name (For SQL Server and EDA Server Listener)	Database name of the database where the table specified in the SQL statement is located. Note: When you access a non-relational database, and the server component is an SSCTL server component, the database name must be the service name and you must specify it. If the server component is installed on USS, you can leave the database field blank. For more information about the server component, see Chapter 1, <i>Introducing the iWay XML Adapter for RDBMS</i> .
SID (For Oracle Listener)	For an Oracle Listener, the SID is a unique name for the database service, chosen by the database administrator or the person who installed Oracle E-Business Suite.
Data Source (For JDBC-ODBC Bridge Listener)	For JDBC-ODBC Bridge Listener, this is the name of the data source configured under the ODBC Driver Manager. For more information, see your ODBC Driver Manager documentation.
User	Database user ID to access the table.
Password	Database password associated with the user ID.
Polling Interval	Interval, in milliseconds, at which to check for new input.

Parameter	Description
SQL Query	<p>SQL SELECT statement that the listener issues to poll the table.</p> <p>If the SQL statement includes a date column or long text column, you must provide a value for the SQL Post-query parameter. The value you provide must not contain a date column or a long text column. This applies whether you provide an SQL statement here or rely upon the default.</p> <p>For example, the following SELECT statement retrieves all unprocessed records from the DISCRETE_JOBS table:</p> <pre>SELECT * FROM WIP_DISCRETE_JOBS D WHERE DJ.WIP_ENTITY_ID > (SELECT WIP_ENTITY_ID FROM WIP.TEMP_NEW_WORK_ORDER_ENTITY_ID)</pre> <p>Important: When a SQL Query joins two or more tables, a SQL Post Query must be used. Also, do not use a semicolon at the end of a SQL statement for a SQL Query or a SQL Post Query.</p>

Parameter	Description
Post Query	<p>A SQL statement that is executed after each new record has been read from the table. This is case sensitive: the case used to specify the column names must match the case used in the SELECT statement that polled the table. If the SQL Query property was omitted so that a default SELECT statement polled the table, the case used to specify the column names must match the case used to define the columns in the DBMS's native schema.</p> <p>If you do not specify a value for SQL Post-query, each record read from the table will be deleted after it has been read. How this happens depends on whether you specify the Delete Keys property. If you:</p> <ul style="list-style-type: none"> • Specify the Delete Keys property, by default the adapter issues a DELETE statement with a WHERE clause containing every key column specified for the Delete Keys property. At run-time this will be faster than if you had not specified the Delete Keys property if there is an index on the key, or if there are fewer key columns than there are columns in the SELECT statement that polled the table. • Do not specify the Delete Keys property, by default the adapter issues a DELETE statement with a WHERE clause that specifies every column from the SELECT statement that polled the table. <p>You can choose to retain the table's data once it has been read by specifying a value for this parameter, as shown in the examples that follow.</p> <p>Note that the SQL Post-query and Delete Keys parameters are mutually exclusive, as Delete Keys applies to the default DELETE statement, and SQL Post-query overrides the default DELETE statement. You can provide a value for one or the other, but not for both.</p> <p>There are two field operators, ? and ^, that you can use in a post-query SQL statement; for more information, see <i>The Post-query Parameter Operators</i> on page 3-30.</p> <p>Important: When a SQL Query joins two or more tables, a SQL Post Query must be used. Also, do not use a semicolon at the end of a SQL statement for a SQL Query or a SQL Post Query.</p>

Parameter	Description
Delete Keys	<p>Comma-separated list of key columns to be used in the default DELETE statement. DELETE operates on keys, so specify the table's key columns.</p> <p>This is case sensitive: the case used to specify the column names must match the case used in the SELECT statement that polled the table. If the SQL Query property was omitted so that a default SELECT statement polled the table, the case used to specify the column names must match the case used to define the columns in the DBMS's native schema.</p> <p>Note that the Delete Keys and SQL Post Query parameters are mutually exclusive, as Delete Keys applies to the default DELETE statement, and SQL Post Query overrides the default DELETE statement. You can provide a value for one or the other, but not for both. For more information, see the description of the SQL Post-query parameter in this table.</p>

7. Enter the system information that is specific to the database on which you are listening based on the descriptions in the previous table.
8. Click OK.

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

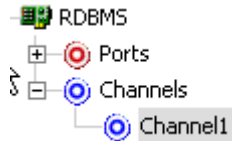


An X over the icon indicates that the channel is currently disconnected. You must start the channel to activate your event configuration.

To review the settings for the channel, select the channel. The right pane contains tabs that summarize the channel settings.

Procedure How to Start and Stop a Channel

1. Expand the *iWay Event Adapters* node.
2. Expand the *RDBMS* node.
3. To start a channel, right-click the channel node and select *Start*.
The channel becomes active and the X over the icon disappears.



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

Editing and Deleting a Channel

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

Procedure How to Edit a Channel

1. Expand the *iWay Event Adapters* node.
2. Expand the *RDBMS* node.
3. Right-click the channel you want to edit and select *Edit*.
The Edit Channel dialog box appears.
4. Make the necessary changes to the channel configuration.
5. Click *OK*.

Procedure How to Delete a Channel

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

Deploying iWay Components in a Clustered BEA WebLogic Environment

Events can be configured in a clustered BEA WebLogic environment. You can deploy iBSE or JCA to this environment. This topic uses iBSE as an example, but you can follow the same procedures when deploying JCA. The only difference is that you need to deploy the JCA connector .RAR file to the clustered environment.

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

Clustering provides the following benefits:

- Load balancing
- High availability

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

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

Procedure How to Deploy iWay Components in a Clustered Environment

To deploy iWay components in a clustered environment:

1. Using the BEA Configuration Wizard:
 - a. Configure an administrative server to manage the managed servers.
 - b. Add and configure as many managed servers as required.
 - c. Add and configure an HTTP router. This does not have to be a part of WebLogic and can be an outside component.
 - d. If you configure the HTTP router within WebLogic, start it by entering the following command:

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

where:

```
HTTPROUTER
```

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

```
http://localhost:7001
```

Is the location of the admin console.

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

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

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

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

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

Deploy a Web Application Module

Select the archive for this Web application module

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

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

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

<input type="checkbox"/>	ibse
<input checked="" type="checkbox"/>	iwaee
<input type="checkbox"/>	iwjcaivp

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

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




Deploy a Web Application Module

Select the archive for this Web application module

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

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

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

<input type="radio"/>	 ibse
<input checked="" type="radio"/>	 iwaee
<input type="radio"/>	 iwaycaivp

Target Module

The following window opens.

Select targets for this Web application module

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

Independent Servers

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

Clusters

<input checked="" type="checkbox"/>	MYCluster
<input checked="" type="radio"/>	All servers in the cluster
<input type="radio"/>	Part of the cluster
<input type="checkbox"/>	MS1
<input type="checkbox"/>	MS2

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

The following window opens.

Source Accessibility

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

How should the source files be made accessible?

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

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

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

C:\iWay55\bea\ibse

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

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

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

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

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

where:

[hostname](#)

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

[port](#)

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

8. Click *Deploy*.

Procedure Configuring Ports and Channels in a Clustered Environment

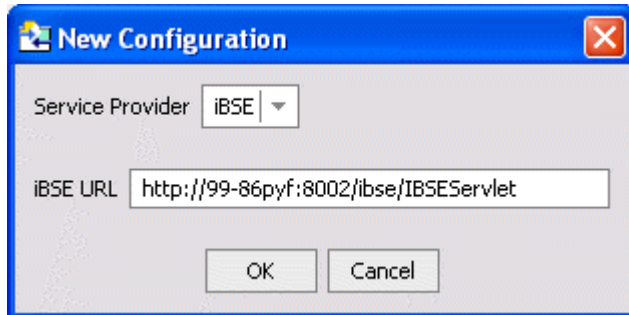
You can use Swing Application Explorer deployed in BEA WebLogic WorkShop or Servlet Application Explorer to configure ports and channels in a clustered environment.

Note: Before using Servlet Application Explorer in a clustered environment, you must edit the web.xml file and specify the correct URL to your iBSE deployment. The default location on Windows is:

`C:\Program Files\iWay55\bea\iwae\WEB-INF\web.xml`

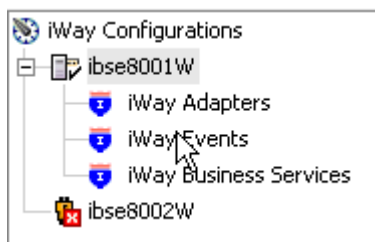
For more information on configuring the web.xml file for Servlet Application Explorer, see the *iWay Installation and Configuration for BEA WebLogic* documentation.

1. Open Swing Application Explorer in BEA WebLogic Workshop.
2. Create a new connection to the iBSE instance. For information on creating a new configuration, see *How to Define a New Configuration* on page A-4.



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

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

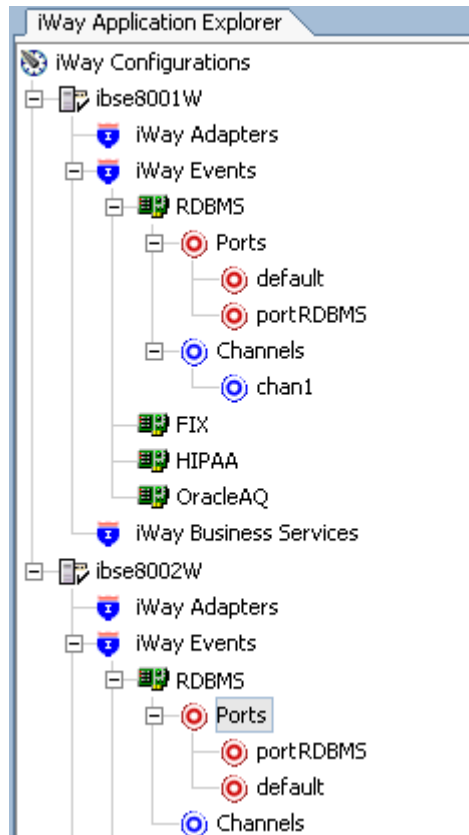


4. Add a new port for the RDBMS adapter. For more information, see *Creating an Event Port From the iWay Event Adapters Tab* on page B-4
5. Create a channel and add the port you created. For more information, see *Creating, Editing, and Deleting an Event Channel* on page B-18.

6. Click *Next* and enter the application server parameters.
7. Start the channel.
8. Create a new configuration and connect to the second iBSE instance.

The connection to iBSE must be configured to each instance of the managed server.

The following graphic shows two configurations.



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

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

The following operations must be performed on each server:

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

For More Information

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

- *Choosing a Listening Technique*
- *Standard Event Processing With Row Tracking*
- *Standard Event Processing With Row Removal*
- *Trigger-based Event Processing*

For More Information

APPENDIX C

Using WebLogic Workshop to Access VSAM

Topics:

- Using WebLogic Workshop to Access VSAM
- Running the JWSNAME Web Service from WebLogic Workshop

This section describes how to produce and access a Web service generated from a VSAM database.

Using WebLogic Workshop to Access VSAM

WebLogic Workshop provides a framework for building Web services. The Web services that you build with WebLogic Workshop are enterprise-class services, and WebLogic Workshop provides simple controls for connecting to your enterprise resources.

WebLogic Workshop simplifies the process of creating Web services by insulating developers from the low-level implementation details that have traditionally made Web service development the domain of sophisticated J2EE developers. With WebLogic Workshop, you can build powerful Web services whether you are an application developer or a J2EE expert.

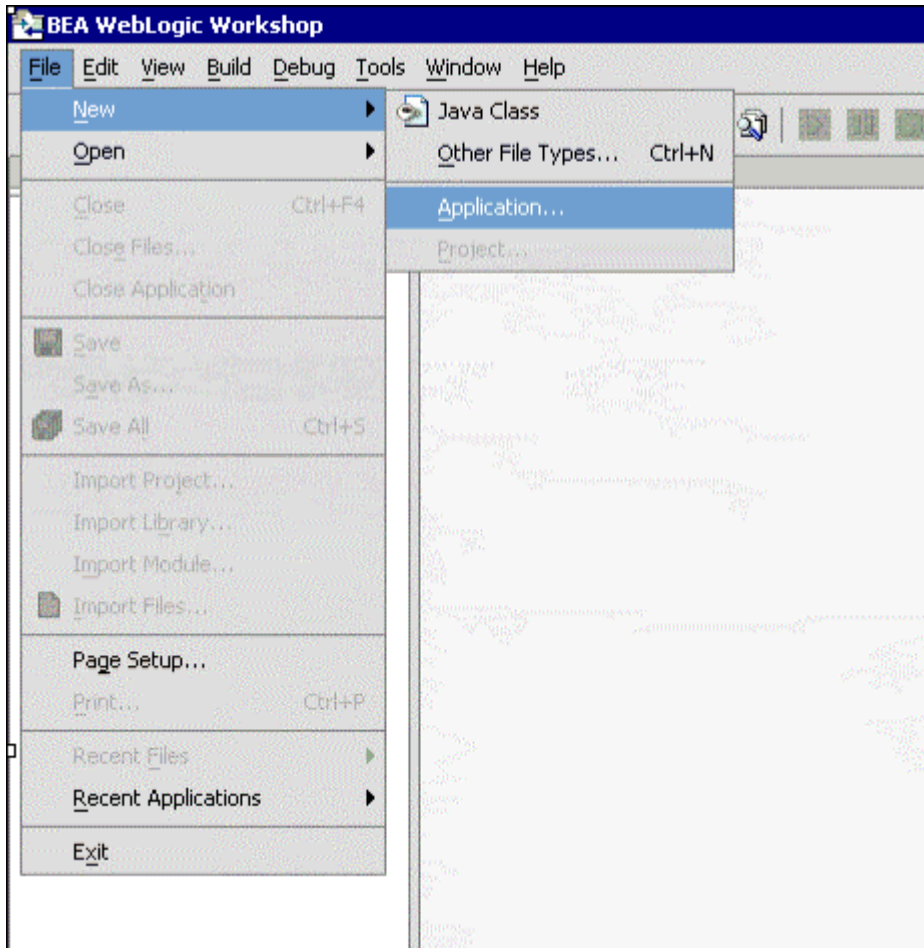
Procedure How to Use WebLogic Workshop to Access VSAM

This procedure assumes you already created and tested a Web service using Application Explorer. It also assumes you have created the WSDL used to access the service. For more information on creating Web services and the accompanying WSDL, see Chapter 2, *Creating XML Schemas or Business Services*.

To access VSAM using WebLogic Workshop:

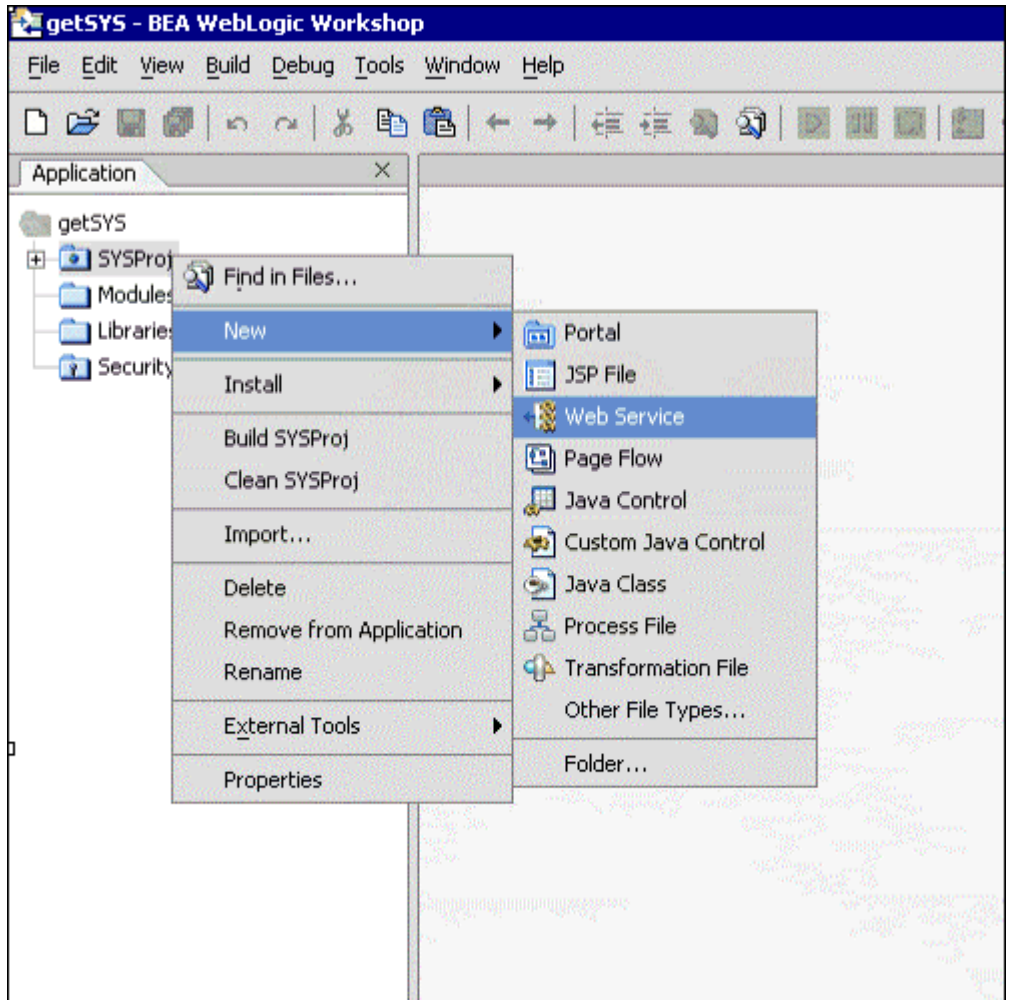
1. From the Start menu, choose *Programs, WebLogic Platform 8.1*, and then *WebLogic Workshop 8.1*.

BEA WebLogic Workshop opens.



2. Create a new application.
 - a. From the File menu, select *New* and then, *Application*.
 - b. In the upper-left pane, select all and then, select *Empty Application*.
 - c. In the directory field, type *C:\WAYSRV*.
 - d. Click *Create*.
3. In the Application tab, right-click the *IWAYSrv* folder and select *New Project*.
4. In the upper left pane, select all and then, select *Web Project*.
5. In the name field, type *SYSProj* and click *Create*.

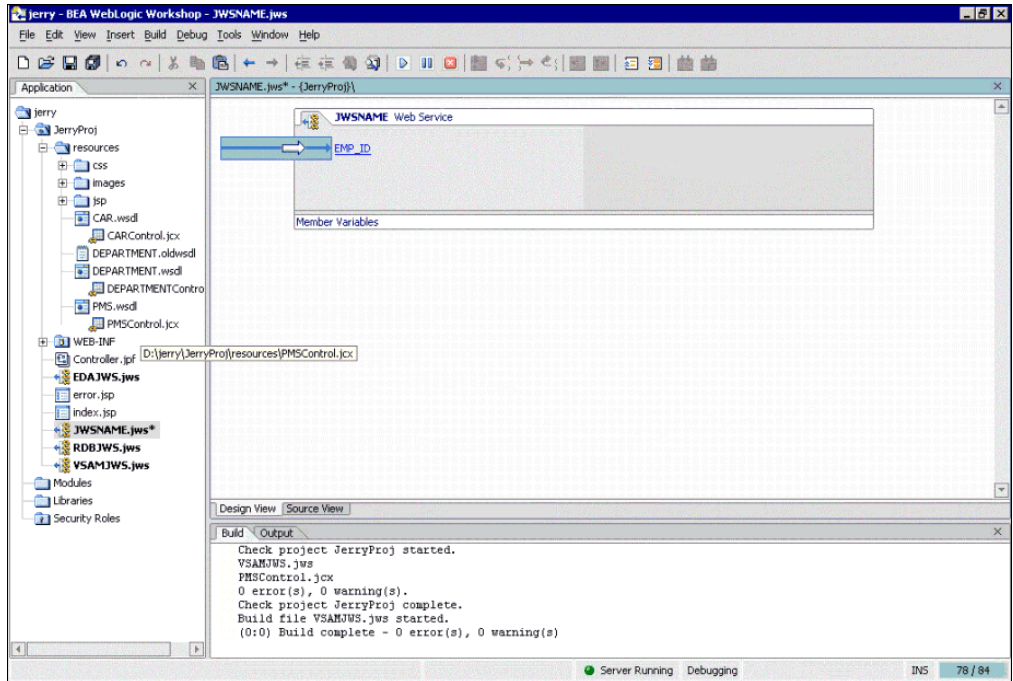
The code for a Web service is contained within a JWS (Java Web Service) file. A JWS file is a JAVA file in that it contains code for a Java class. However, because a file with a JWS extension contains the implementation code intended specifically for a Web service class, the extension gives it special meaning in the context of the WebLogic Server.



6. On the Application tab, right-click the *SYSProj* folder.
 - a. Select *New*.
 - b. Select *Web Service*.
 - c. In the upper left pane, select all.
 - d. In the right pane, select *Web Service*.

- e. In the name field, type *JWSNAME.jws*.
7. Click *Create*.

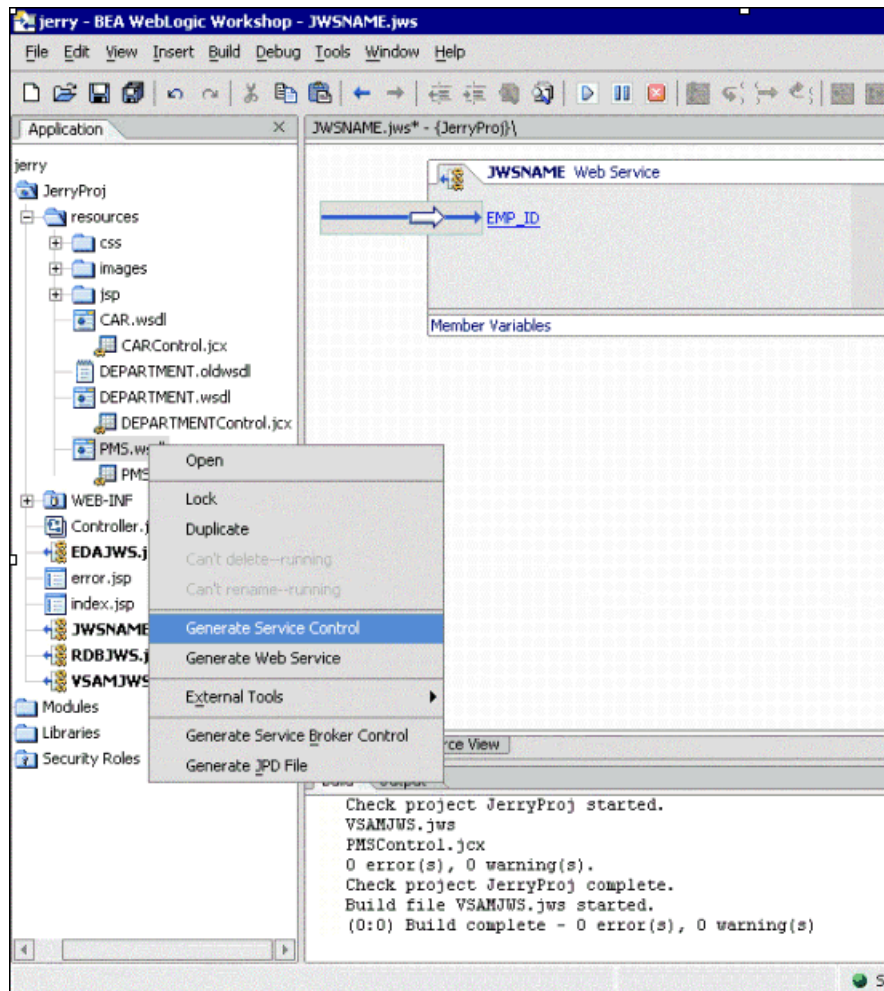
The design view window opens.



Web services expose their functionality through methods that clients invoke when they want to request something from the Web service. In this case, clients invoke a method to call the PMS Control that is exposed later in this procedure.

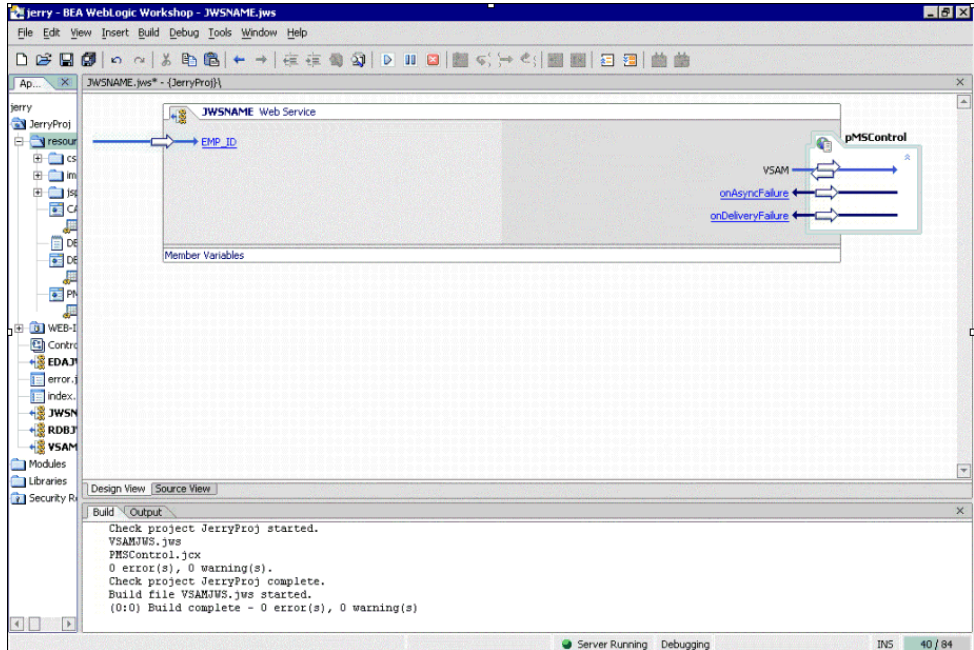
8. If it is not already selected, click the *Design View* tab.
 - a. From the Insert menu, select *Method*.
 - b. In the space provided, replace method1 with *EMP_ID*.
9. Press *Enter*.
10. Right-click the *resources* subfolder.
 - a. Select *Import*.
 - b. Import *PMS.WSDL* (saved in *How to Generate WSDL From a Web Service* in Chapter 2, *Creating XML Schemas or Business Services*).

For more information on creating a WSDL file, see Chapter 2, *Creating XML Schemas or Business Services*.



11. To generate a Java Control file, right-click the *PMS.wsdl* file and select *Generate Service Control*.

12. Drag the *PMS.jcx* file onto the JWSNAME Web service as follows:



- a. To modify the source code and call the iWay PMS Web service, click the *Source View* tab.
- b. Modify the common section of the code to:


```

public String EMP_ID(String empid) throws Exception
{
    return PMSControl.VSAM(empid).RESULT.PMSVSAM.RESULTSET_1[0].EMP_ID;
}

```
- c. To save your current work, press *Control + S*.

The resulting Java code looks similar to the following:

```
public class JWSNAME implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private resources.PMSControl PMSControl;

    static final long serialVersionUID = 1L;

    /**
     * @common:operation
     */
    public String EMP_ID(String empid) throws Exception
    {
        return PMSControl.VSAM(empid).RESULT.PMSVSAM.RESULTSET_1[0].EMP_ID;
    }
}
```

Running the JWSNAME Web Service from WebLogic Workshop

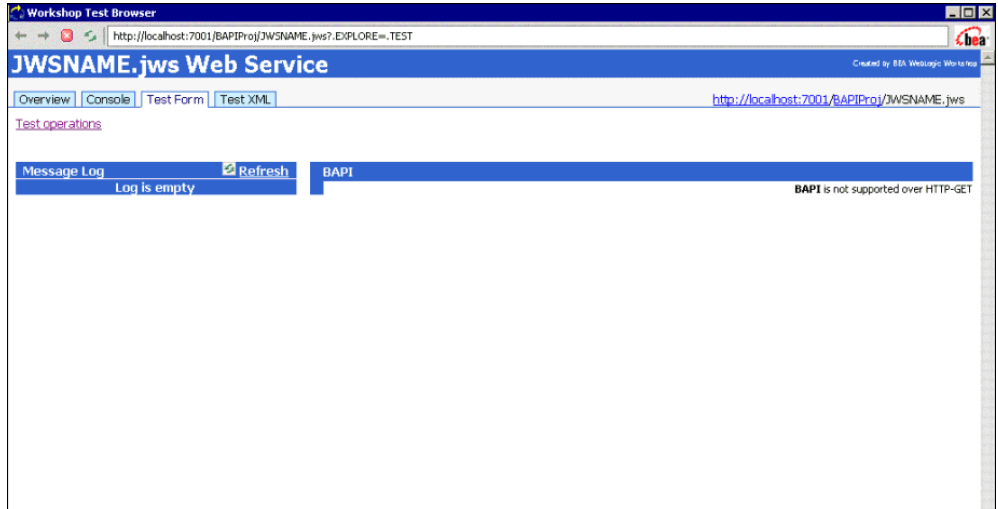
When you create a new Web service tutorial application, you must ensure that BEA WebLogic Server is running while you build your Web service. You can confirm whether BEA WebLogic Server is running by checking the status bar in WebLogic Workshop.

If WebLogic Server is running, a green ball appears in the status bar. If WebLogic Server is not running, a red ball appears. If you see the red ball, start BEA WebLogic Server, as described in the following procedure.

Procedure How to Start BEA WebLogic Server

1. From the Tools menu, select *WebLogic Server* and then, *Start WebLogic Server*.
2. To deploy the application to BEA WebLogic, select *Tools* and then, *Deploy Application*.
3. To start the application, click the *Start* button on the toolbar.

The following test window opens.



- a. To enter and test the XML stream to be passed to the Web service, click the *Test XML* tab.

- b. Replace the string XML input with the following:

```
<EMP_ID xmlns="http://www.openuri.org/">
  <!--Optional:-->
  <empid>000000004</empid>
</EMP_ID>
```

4. To submit the request, click the *EMP_ID* button.

After the SOAP request is sent to the VSAM adapter, the following response is returned:

The screenshot shows the Workshop Test Browser interface for the JWSNAME.jws Web Service. The browser window title is "Workshop Test Browser" and the address bar shows "http://localhost:7001/JerryProj/JWSNAME.jws?_EXPLORE=&_TESTXML&_LOGENTRY=1". The main content area is titled "JWSNAME.jws Web Service" and includes a "Test operations" section with a "Message Log" and a "Refresh" button. The "Message Log" shows a message from "EMP_ID" to "PMSControl.VSAM" with a "Clear Log" button. The "External Service Request" section shows a SOAP request submitted at Sunday, January 11, 2004 4:27:09 PM EST. The "External Service Response" section shows a SOAP response submitted at Sunday, January 11, 2004 4:27:12 PM EST.

External Service Request
Submitted at Sunday, January 11, 2004 4:27:09 PM EST

```
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns: xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <ns:VSAM xmlns:ns="urn:iwaysoftware:ibse:jul2003:VSAM">
      <ns:emp_id>000000004</ns:emp_id>
    </ns:VSAM>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

External Service Response
Submitted at Sunday, January 11, 2004 4:27:12 PM EST

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <VSAMResponse id="F3EAB236CEE2782BEC708D1F083DCD9F" xmlns="urn:iwaysoftware:ibse:jul2003:VSAM:response">
      <RESULT>
        <PMSVSAM>
          <RESULTSET_1>
            <ROW>
              <EMP_ID>000000004</EMP_ID>
              <FIRST_NAME>MIKE</FIRST_NAME>
              <LAST_NAME>DUHNAME</LAST_NAME>
              <DEPT>IWAY</DEPT>
              <COMP_NAME>INFORMATION BUILDERS</COMP_NAME>
            </ROW>
          </RESULTSET_1>
        </PMSVSAM>
      </RESULT>
    </VSAMResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

APPENDIX D

JDBC Drivers

Topic:

- Copying and Collecting a JDBC File

This section lists the supported JDBC drivers for use with the iWay XML Adapter for RDBMS for BEA WebLogic.

Copying and Collecting a JDBC File

The iWay XML Adapter for RDBMS for BEA WebLogic requires JDBC drivers when connecting to certain databases.

To enable these adapters:

1. Use the following table to determine the JDBC driver files required for your database.
2. Copy the required JDBC files into the *iWay 5.5 lib* directory.

The default location for this directory on Windows is:

`C:\Program Files\iWay55\lib`

On other platforms, use the corresponding location.

iWay Adapter for...	Required Libraries or Drivers
CICS	None
DB2 Net Driver DB2 App Driver	<p>JDBC driver for DB2 (db2java.zip).</p> <p>Installed as part of the DB2 server. The default location on Windows is one of the following:</p> <p><code>C:\SQLLIB\java\db2java.zip</code></p> <p><code>C:\Program Files\SQLLIB\java\db2java.zip</code></p> <p>Because the iWay XML Adapter for RDBMS for BEA WebLogic uses JDBC 2.0 features, you must run the usejdbc2.bat file supplied with DB2. Obtain the usejdbc2.bat from your DB2 server or download the current version from the DB2 JDBC driver download Web page.</p> <p>This process builds the proper JDBC 2.0-compliant version of the db2java.zip file.</p> <p>Note: To run the usejdbc2 procedure, you must stop the DB2 instance.</p>
IMS	None

iWay Adapter for...	Required Libraries or Drivers
Informix	<p>JDBC driver for Informix (ifxjdbc.jar).</p> <p>Download the driver from the Informix Web site.</p> <p>For Informix 7:</p> <p>ifxjdbc.jar</p> <p>For Informix 9:</p> <p>ifxjdbc.zip</p>
Oracle E-Business Suite	<p>Oracle JDBC drivers (thin type4 or OCI type2) and/or Oracle Client NET8 or NET9.</p> <p>All calls to Oracle E-Business Suite occur through these drivers. If you do not have the appropriate JDBC driver, Oracle Technology Network (OTN) provides a download site at:</p> <p>http://otn.oracle.com/software/tech/java/sqlj_jdbc/content.html</p> <p>Note: To download the drivers, you require a logon ID.</p> <p>If you are using OCI drivers, you must install and configure Oracle Client on the machine with the iWay Adapter for Oracle E-Business Suite.</p> <p>To use iWay Concurrent Program request functionality, you must install and configure Oracle Client on the Oracle database that supports Oracle E-Business Suite.</p>

iWay Adapter for...	Required Libraries or Drivers
Oracle	<p>Oracle JDBC driver (classes12.zip).</p> <p>You can download this driver from the Oracle Web site: http://otn.oracle.com/software/tech/java/sqlj_jdbc/content.html</p> <p>Note: To download the drivers, you require a logon ID.</p> <p>For more information on Oracle JDBC issues, see the Oracle JDBC FAQ: http://otn.oracle.com/tech/java/sqlj_jdbc/htdocs/jdbc_faq.htm</p>
SQL Server 2000	<p>SQL Server 2000 JDBC driver.</p> <p>The JDBC driver includes the following three files: msbase.jar mssqlserver.jar msutil.jar</p> <p>You can download the driver free of charge from: http://microsoft.com</p> <p>To download and run an installation program to install the driver, search the site for <i>SQL Server 2000 JDBC driver</i>. The installation program installs the three driver files in: drive:\Program Files\Microsoft SQL Server 2000 Driver for JDBC\lib</p> <p>Note: Running the driver installation program is not required. If these files are on another machine, simply copy them.</p>
Sybase	<p>JDBC driver for Sybase servers (jConnect for JDBC).</p> <p>You can download the driver from the Sybase downloads Web site: http://www.sybase.com/downloads</p> <p>Select <i>jConnect for JDBC</i> and review information on the jConnect for JDBC Web site.</p> <p>To extract the JDBC driver, obtain the <i>jConnect.zip</i> that corresponds to your version of Sybase and follow the steps described on the jConnect download page.</p>

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

Information Builders, Two Penn Plaza, New York, NY 10121-2898

(212) 736-4433

iWay XML Adapter for RDBMS for BEA WebLogic User's Guide

DN3501320.0205

Version 5 Release 5