

iWay

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

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 © 2003, by Information Builders, Inc. All rights reserved. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Printed in the U.S.A.

Preface

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

How This Manual Is Organized

This manual includes the following chapters:

Chapter/Appendix		Contents
1	Introducing the iWay 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 listeners to listen to a database event.
A	Using the WebLogic Workshop to Access VSAM	Describes how to use the BEA WebLogic Workshop to access a Web service created through the iWay Application Explorer..

Documentation Conventions

Delete the items that do not apply to your manual and/or add special conventions that are unique to your manual.

The following conventions apply throughout this manual:

Convention	Description
THIS TYPEFACE or <i>this typeface</i>	Denotes syntax that you must enter exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
<i>this typeface</i>	Represents a placeholder (or variable) in a text paragraph, a cross-reference, or an important term.

Convention	Description
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.
[]	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (...).
. . .	Indicates that there are (or could be) intervening or additional commands.

What You Need to Know

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

It provides information about using iWay Adapter for RDBMS tools to develop connections between a iXTE client and a RDBMS.

It is assumed that readers have a general understanding of Microsoft Windows and UNIX systems as well as:

- Some experience using Enterprise Information System (EIS) and integration products and an understanding of RDBMS and SQL products with which this software will be integrating.
- Knowledge of EIS concepts.
- General knowledge of RDBMS concepts and configuration options.
- Specific business application knowledge of the target schema.
- Knowledge of integration processes and data models for the required application area.
- General knowledge of iXTE architecture.
- General knowledge of XML concepts.

Extensive internal knowledge of the specific SQL environment is not required, but may be helpful in learning about the iWay Adapter for RDBMS.

Customer Support

Do you have questions about iWay Adapter for RDBMS?

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 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.

Information You Should Have

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

- Your six-digit site code number (xxxx.xx).
- Your iWay Software configuration:
 - The iWay Software version and release.
 - The communications protocol (for example, TCP/IP or LU6.2), including vendor and release.
- The stored procedure (preferably with line numbers) or SQL statements being used in server access.
- The database server release level.
- The database name and release level.
- The Master File and Access File.
- The exact nature of the problem:
 - Are the results or the format incorrect? Are the text or calculations missing or misplaced?
 - The error message and return code, if applicable.
 - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, your security system, communications protocol, or front-end software changed?
- Is this problem reproducible? If so, how?

- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing just the code to access the data source?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

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

Thank you, in advance, for your comments.

iWay Software Consulting and Training

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.

Contents

1. Introducing the iWay Adapter for RDBMS	1-1
Introduction	1-2
Using the Adapter With Relational Databases	1-2
iWay Adapter for RDBMS Requests	1-3
iWay Adapter for RDBMS Listener	1-3
Using the Adapter to Access Non-relational Databases	1-5
Accessing Stored Procedures for Non-relational Data Sources	1-7
Deployment Information for the iWay Adapter for RDBMS	1-7
Deployment Information Roadmap	1-8
The iWay 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 Business Services	2-2
Opening a Connection to a Database	2-3
Closing or Deleting a Connection to a Database	2-7
Viewing Metadata	2-9
Creating a Statement and Generating a Schema	2-14
Creating and Testing a Parameterized SQL Statement	2-16
Creating a Batch Statement	2-19
Generating a Schema for a Prepared Statement	2-20
Generating a Schema for a Stored Procedure for a Relational Database	2-21
Generating a Schema for a Stored Procedure for a Non-relational Database	2-22
Request and Response Documents	2-23
Regular SQL Statements	2-23
Parameterized SQL Request Schemas	2-26
Stored Procedure Schemas for an Oracle Database	2-29
The following examples are based on a schemas created for a stored procedure for an Oracle database.	2-29
Stored Procedure Schemas for a VSAM Database	2-30
Generating a Business Service for an SQL Statement or a Stored Procedure	2-32
Testing a Business Service	2-34
Generating WSDL From a Web Service	2-37
3. Listening for Database Events	3-1
Understanding iWay Event Functionality	3-2
Creating an Event Port	3-2
Creating Channels	3-5
Choosing a Listening Technique	3-8
Standard Event Processing With Row Tracking	3-9

Standard Event Processing With Row Removal	3-15
Trigger-based Event Processing	3-18
A. Using the WebLogic Workshop to Access VSAM	A-1
Using the WebLogic Workshop to Access VSAM	A-2
Running the JWSNAME Web Service from WebLogic Workshop	A-8

CHAPTER 1

Introducing the iWay Adapter for RDBMS

Topics:

- Introduction
- iWay Adapter for RDBMS Requests
- iWay Adapter for RDBMS Listener
- Using the Adapter to Access Non-relational Databases
- Deployment Information for the iWay Adapter for RDBMS

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

Introduction

Because most custom and packaged applications are built with relational databases, RDBMS systems must be taken into consideration in any enterprise integration strategy. The iWay 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.

The adapter also can be used 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 iWay Adapter for RDBMS enables integration with RDBMS systems either by:

- **Sending a Request.** The adapter sends requests, using either SQL queries or stored procedures, to the database.
- **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 iWay Adapter for RDBMS 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 (iAE), 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.

iWay Adapter for RDBMS Requests

The iWay Adapter for RDBMS 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. The adapter returns the data to the client.

The adapter can:

- Receive service requests from an external client.
- Transform the XML request document into the RDBMS data format.
The request document conforms to the XML schema generated by iWay 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 iWay Application Explorer and based on RDBMS metadata.

iWay Adapter for RDBMS Listener

The iWay Adapter for RDBMS 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 then 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 have been transferred to the workflow.

Other options may 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, allowing 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 behave and look as if they were relational databases. This enables a single, uniform approach to data access. Both the iWay Adapter for RDBMS and the separate iWay adapter, deployed with the Adapter for RDBMS, can be used for non-relational database access.

The following tables list several of the adapters that can access non-relational databases.

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

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's 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 platform such as the 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 iWay Adapter for RDBMS and the iWay adapter connect 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 and iWay adapter connections are configured similarly as to a relational database.

After you configure an adapter and create metadata using iWay Application Explorer, you can access a Web-based console, 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 a typical non-relational database access scenario, the iWay Adapter for RDBMS (hosted, for example, by the BEA WebLogic platform) connects to the iWay VSAM adapter (hosted by the server component) using JDBC standards. iWay Application Explorer is used to configure this connection. You can then create Web services for SQL and parameterized SQL using iWay Application Explorer. You also can use the iWay Application Explorer tool 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 iWay 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 has a very powerful fourth generation language that is much more robust than SQL.

The 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 iWay Adapter, an iWay stored procedures must be catalogued before iWay 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 iWay Application Explorer, the adapter creates Web services that can be used to run the stored procedures from any Web client.

Deployment Information for the iWay Adapter for RDBMS

The iWay Adapter for RDBMS works in conjunction with the following components:

- iWay Application Explorer

and

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

with either

- iWay Business Services Engine (iBSE)

or

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

iWay 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 iWay Business Services Engine or with the iWay Enterprise Connector for J2EE Connector Architecture (JCA). Both iBSE and the iWay connector for JCA are deployed to the BEA WebLogic environment with iWay Application Explorer and the adapters. The server component is deployed outside of the adapter host (for example, the 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 iWay Adapter for RDBMS and the iWay Adapter. A description of the iWay 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	<ul style="list-style-type: none">• Chapters 2, 3, and 4 of this guide• iWay Installation and Configuration for BEA WebLogic, Version 5 Release 5• iWay Application Explorer (Java Servlet Version) User's Guide, Version 5 Release 5
iWay Business Services Engine (iBSE)	<ul style="list-style-type: none">• iWay Business Services Engine User's Guide• iWay Installation and Configuration for BEA WebLogic, Version 5 Release 5
iWay Enterprise Connector for J2EE Connector Architecture (JCA)	<ul style="list-style-type: none">• iWay Connector for JCA for BEA WebLogic Server User's Guide• iWay Installation and Configuration for BEA WebLogic, Version 5 Release 5
iWay server component	<ul style="list-style-type: none">• iWay Server Installation Manual• iWay Server Administration for UNIX, Windows NT OpenVMS, OS/400, OS/390 & z/OS• iWay Data Adapter Administrator User's Guide

The iWay Business Services Engine (iBSE)

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

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

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

Web services is a distributed programming architecture that 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.

Note: Because the connector currently conforms to the JCA 1.0 specification, deployed adapters do not support event listeners. It is anticipated that the JCA 1.5 specification will enable deployment of adapters for event listening.

CHAPTER 2

Creating XML Schemas or Business Services

Topics:

- Generating Schemas and Business Services
- Opening a Connection to a Database
- Closing or Deleting a Connection to a Database
- Viewing Metadata
- Creating a Statement and Generating a Schema
- Generating a Business Service for an SQL Statement or a Stored Procedure

This section describes how to use iWay Application Explorer (IAE) to:

- **Generate XML schemas** that define request and response documents. You can use these schemas when you create request documents and when you develop logic for processing response documents.
- **View metadata** that describes your SQL statements and stored procedures. You can use the metadata when you create request documents and when you develop logic for processing response documents.
- **Create business services** (also known as Web services) for your SQL statements and stored procedures.

Generating Schemas and Business Services

You can use iWay Application Explorer to connect to relational data sources, such as Oracle or Informix and to non-relational data sources, such as an IMS data source.

When connected to non-relational data sources, 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. Although the iWay Adapter for RDBMS gives you access to data, it does not allow you to execute stored procedures. The iWay Adapter provides access to stored procedures. For more information on using the iWay Adapter to generate schemas for stored procedures, see *How to Generate a Schema for a Stored Procedure for a Non-relational Database* on page 2-22.

To generate request and response document schemas or a business service:

1. **Start iWay Application Explorer and open a new or existing connection** to a relational or non-relational database management system, as described in *Opening a Connection to a Database* on page 2-3.

After you launch iWay Application Explorer, you can expand the iWay Adapters node to view the list of adapters that represent the adapters installed on your system.

After you finish using a connection, you can close it. If you won't require the connection in the future, you can delete it. For more information, see *Closing or Deleting a Connection to a Database* on page 2-7.

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

2. **Generate XML schemas** that define request and response documents for your SQL statements and stored procedures, as described in *Creating a Statement and Generating a Schema* 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** 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 find it helpful to examine the metadata describing your SQL statements and procedures, as described in *Viewing Metadata* on page 2-9. 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.

This requires that you deployed the adapter to a server host with the servlet iWay Business Services Engine.

For configuration and deployment information, see *iWay 5.5. Installation and Configuration for BEA WebLogic Version 5 Release 5*. For more information on Web services, see *Generating a Business Service for an SQL Statement or a Stored Procedure* on page 2-32.

5. **Create events** by configuring channels and ports.

A port is a logical definition of how event data is to be directed. It includes information on where the information is to be stored and where it is to be placed after it is processed.

A channel defines the listening capability that detects events in the target EIS or database. Together, channels and ports enable you to detect and process event data from numerous EIS and database systems.

Opening a Connection to a Database

Although you create and open connections the same way, the specific connection you open depends on the type of data you wish to access. The connection you open also depends on whether you are accessing stored procedures or prepared SQL statements.

- If you are creating and generating schemas for SQL statements for a relational or non-relational database, open a connection under the *RDBMS* node.
- If you are accessing and generating schemas for stored procedures for a non-relational database, open a connection under the *iway* node.

You can connect to a database by:

- Defining a new target, as described in *How to Define a New Connection* on page 2-4.
- Connecting to an existing RDBMS target, as described in *How to Open an Existing Connection* on page 2-6.

iWay Application Explorer has three tabs at the top of the window: iWay Adapters, iWay Events, and iWay Business Services. You create and manage connections to databases in the iWay Adapters tab.

Procedure How to Define a New Connection

To define a new connection:

1. If you have not already done so, start iWay Application Explorer by entering its address in your browser, for example:

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

where:

[hostname](#)

Is the machine where your application server is installed.

[port](#)

Is the port number on which the application server is listening.

iWay Application Explorer opens. The left pane displays all the adapters supported by your version of iWay Application Explorer.

2. Click the *RDBMS* node.

Note: If you want to use stored procedures for a non-relational database, use the *iway* node.

3. In the right pane, place the cursor over *Operations* and select *Define a new target* from the pop up menu.

4. Specify the following information that is specific to the target you define:

- **Target Name.** Enter a descriptive name for the target, for example, SAPTarget.
- **Target Description.** Enter a brief description for the connection.
- **Target Type.** Select the type of target you are connecting to from the drop-down list. The default value is Oracle.

Important: If you are connecting to a non-relational database, select *EDA 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 Adapter for RDBMS*.

5. Click *Next*.

The Set connection info window appears in the right pane.

The fields that appear in the Set connection info window are specific to the type of database to which you connect.

6. 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	The DNS or IP name of the server where the database instance resides.
Port	The port number on which the database is listening.
Database (Data source) Name	The specific name of the database or data source to which you connect.
Use driver type	<p>For DB2 connections, select whether to use one of two JDBC drivers:</p> <ul style="list-style-type: none"> • app driver. Be sure you have the DB2 client installed on the machine from which you are accessing the Application Explorer. • net driver. Be sure you have started the DB2 applet server. <p>For more information about these options, see your DB2 documentation</p>
Server Name	For an iWay server component, the name of the service node to which you are connecting. For more information, see the <i>iWay Server Administration Guide</i> .
Service Name	For connecting to a non-relational database to access stored procedures, this will default to the iWay server component's DEFAULT service name. This value can be left blank
Informix Server Name	This value is equivalent to the Informix variable name INFORMIXSERVER.
Driver	The name of the driver used to access the database you wish to connect to. See your database documentation for more details.
URL	For a JDBC connection, the JDBC driver-specific URL used to connect to the RDBMS.

Parameter	Definition
SID	For an Oracle database, the unique name of the database service that was selected by the database administrator or whoever installed Oracle.
User Name	The database user ID to access the database. The user ID must have database access to the interface tables being accessed.
Password	The password associated with the specified user name.

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

7. Click *Finish*.

The connection name appears in the left pane under the node where you created the new connection. For information on connecting to the node, see *How to Open an Existing Connection* on page 2-6.

You have finished creating the new connection.

Procedure How to Open an Existing Connection

To open an existing connection:

1. If you have not already done so, start iWay Application Explorer by entering its address in the browser, for example:

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

where:

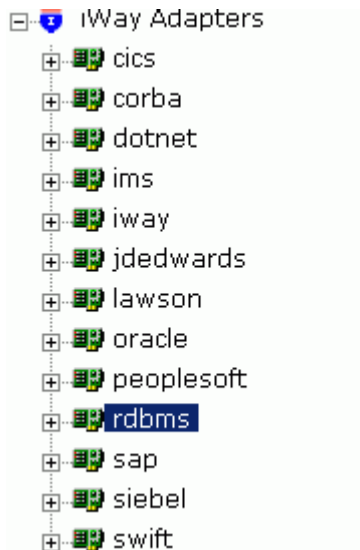
[hostname](#)

Is the machine where your application server is installed.

[port](#)

Is the port number on which the application server is listening.

iWay Application Explorer opens. The left pane displays all the adapters supported by your version of iWay Application Explorer.



2. Expand the *RDBMS* or *iway* node and select the connection you want to open.
3. In the right pane, place the cursor over *Operations* and select *Connect*.

A confirmation window opens to show connection information.

4. Supply the password, if required, edit the connection parameters that must change, and then, click *OK*.

If the parameters are correct and the RDBMS or server component is available, the node under the RDBMS node displays a plus sign. If not, an error message appears in the right pane.

Closing or Deleting a Connection to a Database

To manage connections, you can:

- **Close a connection** that is not currently in use.

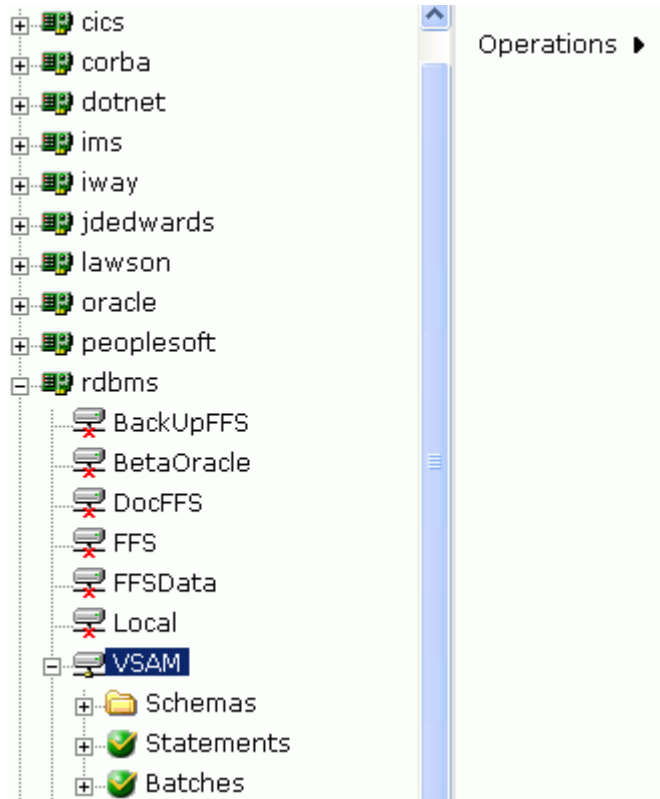
Although you can maintain multiple open connections, it is prudent to close connections that are not in use.

- **Delete a connection** that is no longer required.

You can delete a connection whether or not it is closed; if open, it automatically closes before being deleted.

Procedure How to Close a Connection

To close a connection:



1. In the left pane of iWay Application Explorer, select the connection you want to close.
In the right pane, you can select an operation to perform.



2. Place the cursor over *Operations* and select *Disconnect* from the pop-up menu.
3. If you wish to re-establish the connection, place the cursor over *Operations* and select *Connect* from the pop-up menu.

Disconnecting from the application closes the connection, but the connection remains listed in the left pane so that you can re-open it.

Note that the connection node now has a different icon, indicating that it is closed, as shown in the following figure.



Procedure How to Delete a Connection

To delete a connection from the list of existing connections:

1. If you have not already done so, connect to a database, as described in *Opening a Connection to a Database* on page 2-3.
2. Expand the *RDBMS* or *iway* node to display the list of connections.
3. Click the connection you wish to delete.
4. In the right pane, place the cursor over *Operations* and select *Delete* from the pop-up menu.

As an alternative, you can select *Disconnect* to close the connection while retaining it for future use.

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 2-9.
- Stored procedure metadata for a relational database, as described in *How to View Stored Procedure Metadata for a Relational Database* on page 2-11.
- Stored procedure metadata for a non-relational database, as described in *How to View Stored Procedure Metadata for a Non-relational Database* on page 2-12

Procedure How to View Table Metadata

To view table metadata:

1. If you have not already done so, connect to the database, as described in *Opening a Connection to a Database* on page 2-3.
2. Expand the *Tables* node under the desired connection.
3. Scroll down and select the specific table to review.

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

Viewing Metadata

When you select a specific table, a metadata summary table appears in the right pane, as shown in the following figure.

Properties for empdata

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

- a. Click the ellipsis symbol in the Columns or Database Properties row, depending on which properties you want to review.

The properties 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 creating the schema.

Procedure How to View Stored Procedure Metadata for a Relational Database

You can use the iWay Adapter for RDBMS to view metadata for a stored procedure.

Note: To view stored procedure metadata for a non-relational database, you must use the iWay Adapter. For more information, see *How to View Stored Procedure Metadata for a Non-relational Database* on page 2-12.

1. If you have not already done so, connect to the database, as described in *Opening a Connection to a Database* on page 2-3.
2. Expand the *Procedures* node under the desired connection.
3. Scroll down and select the specific 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 specific procedure, a metadata summary table appears in the right pane, as shown in the following figure.

Properties for empdata

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

- a. Click the ellipsis symbol in the Columns or Database Properties row, depending on which properties you want to review.

The properties 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 creating the schema.

Procedure How to View Stored Procedure Metadata for a Non-relational Database

To use the iWay Adapter to view metadata for iWay stored procedures:

1. If you have not already done so, connect to the database under the iWay Adapter node. For more information on connecting to a database, see *Opening a Connection to a Database* on page 2-3.
2. Expand the *Procedures* node under the desired connection.
3. Scroll down and select the specific 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 figure.

Properties for empdata

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

- a. Click the ellipsis symbol in the Columns or Database Properties row, depending on which properties you want to review.

The properties 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 creating the schema.

Creating a Statement and Generating a Schema

You can create SQL statements even when using the adapter for non-relational databases. After you create the statements, 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), 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 2-19.

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 2-20.
- Stored procedures, as described in *How to Generate a Schema for a Stored Procedure for a Relational Database* on page 2-21.
- iWay Store procedures for non-relational databases, as described in *How to Generate a Schema for a Stored Procedure for a Non-relational Database* on page 2-22

Note: To create schemas for stored procedures for a non-relational database, you must use the iWay Adapter. For more information, see *How to Generate a Schema for a Stored Procedure for a Non-relational Database* on page 2-22.

You can also remove a schema, as described in *How to Remove a Schema* on page 2-23.

If you plan to create business services, you are not required to generate a schema. For more information, see *Generating a Business Service for an SQL Statement or a Stored Procedure* on page 2-32.

Procedure How to Create a Regular SQL Statement

To generate schemas for request and response documents to execute an SQL statement:

1. If you have not already done so, connect to an RDBMS as described in *Opening a Connection to a Database* on page 2-3 and then, expand the connection node.
2. Click the *Statements* node.
3. In the right pane, place the cursor over *Operations* and select *Create Prepared Statement*.

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

4. Type a name for the statement.

It is good practice to specify a name that describes the service. For example, a name of CustomerIntField would represent a request against the Customer Interface table returning a Field format response document.

5. In the Enter SQL Statement field, type the SQL statement to be used by the adapter.

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

6. Click *Create*.

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

- For information on testing a regular 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 2-20.

Procedure How to Test an SQL Statement

To test an SQL statement:

1. In the right pane, place your cursor over *Operations* and select *Test Run*.
The Test Run information appears in the right pane.
2. Click *Test*.
The adapter displays the results in the right pane.

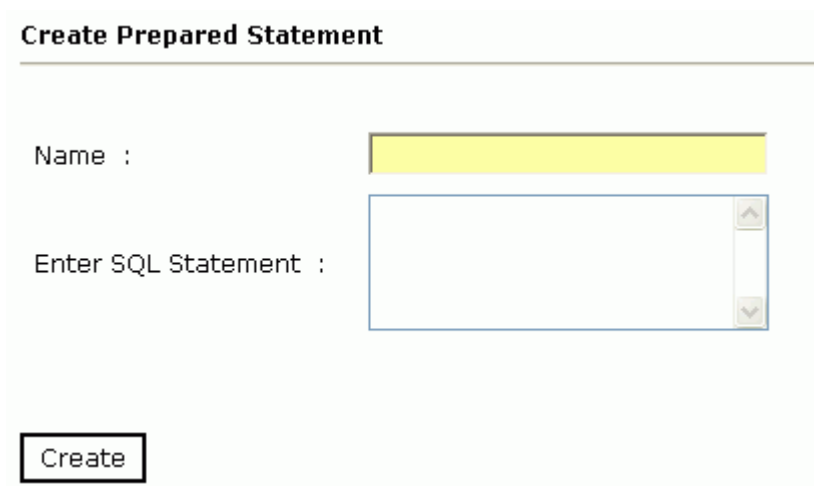
Creating and Testing a Parameterized SQL Statement

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. iWay Application Explorer is used to create and map parameters for the parameterized SQL at design time.

Procedure How to Create a Parameterized SQL Statement

To generate schemas for request and response documents to execute a parameterized SQL statement:

1. If you have not already done so, connect to a database as described in *Opening a Connection to a Database* on page 2-3 and then, select the connection node.
2. In the right pane, place the cursor over *Operations* and select *Create Prepared Statement*.



Create Prepared Statement

Name :

Enter SQL Statement :

Create

3. Type a name and the SQL statement.


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

4. Click *Create*.

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

Create Prepared Statement

params

Parameter Name	Data Type
param0	UNKNOWN 



Update

5. Type a name for each parameter and select a data type from the drop down list.
6. Click *Update*.

The properties table for the newly created statement appears in the right pane.

Operations ►

Properties for ParamSQL2

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

- For information on testing a parameterized SQL statement, see *How to Test a Parameterized SQL Statement* on page 2-17.
- For information on creating schemas for both parameterized and regular SQL statements, see *How to Generate a Schema for a Prepared Statement* on page 2-20.

Procedure How to Test a Parameterized SQL Statement

To test a parameterized SQL statement:

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

The Parameters dialog box opens for the SQL statement.

Test Run

params

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

Test

3. Type a value in the Value field for each parameter.

In the case of the following SQL statement:

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

provide a sample character value, for example, BELLA.

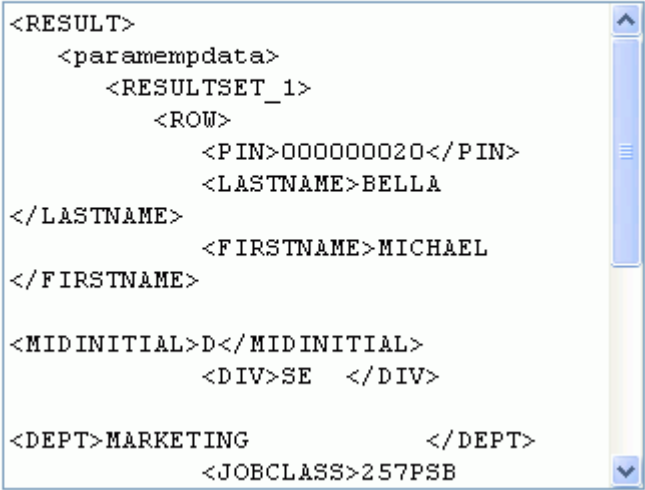
In this example, the values correspond to values of fields found in a table.

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 would be the expected data type of the SQL function. This explains why you provide the SQL type when you create the prepared parameterized SQL statement.

4. Click *Test*.

The results appear in the results window.

Test Run

results : The screenshot shows a 'Test Run' window with a 'results' section. It contains an XML document with the following structure: <RESULT> <paramempdata> <RESULTSET_1> <ROW> <PIN>000000020</PIN> <LASTNAME>BELLA </LASTNAME> <FIRSTNAME>MICHAEL </FIRSTNAME> <MIDINITIAL>D</MIDINITIAL> <DIV>SE </DIV> <DEPT>MARKETING </DEPT> <JOBCLASS>257PSB. The XML is displayed in a text area with a vertical scrollbar on the right.

```
<RESULT>
  <paramempdata>
    <RESULTSET_1>
      <ROW>
        <PIN>000000020</PIN>
        <LASTNAME>BELLA
      </LASTNAME>
        <FIRSTNAME>MICHAEL
      </FIRSTNAME>
        <MIDINITIAL>D</MIDINITIAL>
        <DIV>SE </DIV>
        <DEPT>MARKETING </DEPT>
        <JOBCLASS>257PSB
```

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

To create a batch statement:

1. If you have not already done so, connect to a database as described in *Opening a Connection to a Database* on page 2-3 and then, select the connection node.
2. Select the *Batches* node in the left pane.
3. In the right pane, place the cursor over *Operations* and select *Create A Batch*.
The Create A Batch dialog appears in the right pane.
4. Type a name for the new Batch and click *Create*.

The batch properties information appears in the right pane.

Operations ►
Properties for New Batch

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

5. Place the cursor over *Operations* and choose whether to add stored procedures or statements.

Important: If you are connected to an RDBMS node and are accessing a non-relational database, you cannot add stored procedures to a batch statement. To create batch statements that include stored procedures for non-relational databases, you must use the iWay Adapter.

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 new batch node in the left pane and then select the appropriate option from the Operations pop-up 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:

- *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 2-19

Procedure How to Generate a Schema for a Prepared Statement

To generate the schema for a prepared statement:

1. Click the *SQL statement* node.
2. Place the cursor over *Operations* in the right pane, and select *Generate Schema*.
A table that lists the available schemas appears.
3. To view a schema, click the ellipsis symbol in the Schema column.
4. To create the schema, click *OK*.

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 *Generating a Business Service for an SQL Statement or a Stored Procedure* on page 2-32.

Generating a Schema for a Stored Procedure for a Relational Database

When accessing stored procedures for non-relational databases, you use the iWay Adapter. See *How to Generate a Schema for a Stored Procedure for a Non-relational Database* on page 2-22.

Procedure How to Generate a Schema for a Stored Procedure for a Relational Database

To generate schemas for request and response documents to execute a stored procedure:

1. If you have not already done so, connect to an RDBMS as described in *Opening a Connection to a Database* on page 2-3 and then, click the desired connection node.
2. Expand the node to reveal the stored procedures and then, expand the stored procedures node.
3. Select the stored procedure.
4. In the right pane, place the cursor over *Operations* and select *Generate Schema*.

The Schemas table appears in the right pane.

Schemas

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

Help

OK

Cancel

5. To see the request schema, click the ellipsis symbol in the Request row.
6. To see the response schema, click the ellipsis symbol in the Response row.
The schemas are ready to use.
7. Click *OK*.

Generating a Schema for a Stored Procedure for a Non-relational Database

To generate schemas for iWay stored procedures created on a non-relational database, you must use the iWay Adapter. You also can remove schemas.

Procedure How to Generate a Schema for a Stored Procedure for a Non-relational Database

To generate schemas for iWay stored procedures created on a non-relational database:

1. If you have not already done so, connect to the database under the iWay Adapter node, as described in *Opening a Connection to a Database* on page 2-3, and then click the connection node you wish to use.
2. Expand the node to reveal the stored procedures and then, expand the stored procedures node.
3. Select the stored procedure.
4. In the right pane, place the cursor over *Operations* and select *Generate Schema*.

The Schemas table appears in the right pane.

Schemas

Part	Root Tag	Schema
Request	RPCIn	...
Response	RPCOut	...
Event	N/A	N/A

Help

OK

Cancel

5. Click the ellipsis symbol in the Request row to see the request schema.
6. Click the ellipsis symbol in the Response row to see the response schema.

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*.

Procedure How to Remove a Schema

To remove a schema for an SQL statement or stored procedure:

1. If you have not already done so, connect to an RDBMS, as described in *Opening a Connection to a Database* on page 2-3.
2. Select the desired SQL statement or stored procedure.
3. In the right pane, place the cursor over *Operations* and select *Remove Service Schemas*.

Request and Response Documents

You can generate request document schemas using iWay Application Explorer, as described in *Creating a Statement and Generating a Schema* 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 Procedures for an Oracle database
- Stored Procedures for a VSAM database

Regular SQL Statements

The following examples are based on a 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"?>
<!--Sample XML file generated by XMLSPY v5 rel. 4 U
(http://www.xmlspy.com)-->
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="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"?>
<!--Sample XML file generated by XMLSPY v5 rel. 4 U
(http://www.xmlspy.com)-->
<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 Request Schemas

The following examples are based on a 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"?>
<!--Sample XML file generated by XMLSPY v5 rel. 4 U
(http://www.xmlspy.com)-->
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="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"?>
<!--Sample XML file generated by XMLSPY v5 rel. 4 U
(http://www.xmlspy.com)-->
<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>
```



```

        </RESULTSET_1>
    </paramempdata>
</RESULT>

```

Stored Procedure Schemas for an Oracle Database

The following examples are based on a 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"?>
<!--Sample XML file generated by XMLSPY v5 rel. 4 U
(http://www.xmlspy.com)-->
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="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"?>
<!--Sample XML file generated by XMLSPY v5 rel. 4 U
(http://www.xmlspy.com)-->
<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

You use the iWay Adapter to create schemas for iWay stored procedures for non-relational databases.

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"?>
<!--Sample XML file generated by XMLSPY v5 rel. 4 U
(http://www.xmlspy.com)-->
<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"?>
<!--Sample XML file generated by XMLSPY v5 rel. 4 U
(http://www.xmlspy.com)-->
<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>
```

Generating a Business Service for an SQL Statement or a Stored Procedure

You can generate a business service (also known as a Web service) for an SQL statement or stored procedure. To generate a:

- **Business service** for a relational database, you must deploy the iWay Adapter for RDBMS in a business services environment using the servlet version of the iWay Business Services Engine (iBSE).
- **Web service** for a stored procedure for a non-relational data source, you must deploy the iWay Adapter in a business services environment using the servlet version of the iWay Business Services Engine (iBSE).

Ensure you properly configure the servlet iBSE. For more information on installing and deploying iWay components, see the *iWay Installation and Configuration for BEA WebLogic, Version 5 Release 5* manual.

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, Version 5 Release 5* manual and the *iWay Connector for JCA for BEA WebLogic Server User's Guide, Version 5 Release 5*.

Procedure How to Generate a Business Service for an SQL Statement or Stored Procedure

To generate a business service for an SQL statement or a stored procedure for a relational or non-relational database:

1. If you have not already done so, connect to a database, as described in *Opening a Connection to a Database* on page 2-3.
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.

Note: To create a Web service for a stored procedure, you must access the stored procedure using the iWay Adapter.

4. In the right pane, place the cursor over *Operations* and select *Create iWay 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, and you must select the business service to which you want to add the new service.

If you select *Create a new service*, the following screen appears.

Create Web Service for Param Car

Service Name:

Description:

License:

- production
- test

6. Provide the following information:

Service Name. Type a name to identify the Web service under the Service node in the left pane of the iWay Business Services tab.

Description. Type a brief description of the Web service.

License. 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.

7. Click *Next*.

Provide the following information:

Method Name. Type a name to specify the name of the SQL statement or stored procedure to be added to the business service.

Description. Type a brief description of the method.

8. Click *Finish*.

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

Testing a Business Service

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

Procedure How to Test a Business Service

To test a business service:

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

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

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

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

The test option appears in the right pane.

If you are testing a Web service that requires XML input, an input field appears as illustrated by the following figure.

Test

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



The screenshot shows a dialog box titled "Test". Inside, there is a section labeled "input xml:" with a large, empty text area for entering XML. Below the text area, there are four buttons: "Browse...", "Upload", "More", and "Invoke". The "Invoke" button is highlighted, indicating it is the next step in the process.

If you are testing a Web service for a parameterized SQL statement, an input area appears as illustrated by the following figure.

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"/>

Invoke

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

Application Explorer displays the results in the results window in the right pane.



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

Procedure How to Generate WSDL From a Web Service

To generate WSDL from a Web service:

1. If you are not already in the iWay Business Services tab, click the tab to access it.

Generating a Business Service for an SQL Statement or a Stored Procedure

2. In the left pane, expand the list of services to display the service for which you want to generate a WSDL.
3. Click the service.
The link for the service appears in the right pane.
4. Right-click the link and choose *Save Target As* from the pop-up menu.
5. Choose a location for the file and add a .wsdl file extension.
6. Click *Save*.

For example, saving a Web service called PMS for a VSAM database creates a file named PMS.wsdl.

Note: The file extension must be .wsdl.

The following is an example of a WSDL file for a Web service called PMS that was created 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="PMSVSAM"><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="DEPT"/><xs:element
type="xs:string"
name="COMP_NAME"/></xs:sequence></xs:complexType></xs:element></xs:sequen
```

```
ce></xs:complexType></xs:element></xs:sequence></xs:complexType></xs:elem  
ent></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="m1:VSAMResponse"  
name="parameters"/> </message><message name="PMSHeader"><part  
element="tns:ibsinfo" name="header"/> </message><message  
name="AdapterException"><part element="tns:adapterexception"  
name="fault"/>  
    </message><portType name="PMSSoap"><operation  
name="VSAM"><documentation/><input message="tns:VSAMIn"/><output  
message="tns:VSAMOut"/><fault message="tns:AdapterException"  
name="AdapterExceptionFault"/></operation>  
    </portType><binding type="tns:PMSSoap" name="PMSSoap"><soap:binding  
style="document"  
transport="http://schemas.xmlsoap.org/soap/http"/><operation  
name="VSAM"><soap:operation style="document"  
soapAction="PMS.VSAMRequest@production@@"/><input><soap:body  
use="literal"/><soap:header part="header" message="tns:PMSHeader"  
use="literal"/>  
    </input><output><soap:body use="literal"/>  
    </output><fault name="AdapterExceptionFault"><soap:fault  
use="literal" name="AdapterExceptionFault"/></fault></operation>  
    </binding><service name="PMS"><documentation>PMS</documentation><port  
binding="tns:PMSSoap" name="PMSSoap1"><soap:address  
location="http://iwayntk1:7001/ibse/IBSEServlet/XDSOAPRouter"/></port></s  
ervice></definitions>
```

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

CHAPTER 3

Listening for Database Events

Topics:

- Understanding iWay Event Functionality
- Standard Event Processing With Row Tracking
- Standard Event Processing With Row Removal
- Trigger-based Event Processing

This section describes how to use the iWay Adapter for RDBMS or iWay, 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 do something 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 the Servlet iWay Application Explorer. To create an iWay Event, you must:

1. Create a **Port**.

A port associates a particular business object exposed by the iWay Adapter with a particular disposition. A disposition is a URL that defines the protocol and location of the event data. The port defines the endpoint of the event consumption. For example, you can use a JMS protocol to route the result of polling a VSAM database to a JMS queue hosted by the BEA WebLogic Server. For more information, see *Creating an Event Port* on page 3-2.

2. Create a **Channel**.

A channel represents configured connections to particular instances of back-end systems. A channel binds one or more event ports to a particular listener managed by the iWay Adapter. For more information, see *Creating Channels* on page 3-5.

Creating an Event Port

The following procedure describes how to create an arbitrary or specific event port using the iWay Servlet iWay Application Explorer. In addition, the second and third procedures provide information on how to modify and delete each type of event port.

Procedure How to Create an Event Port

To create a specific event port using the iWay Servlet iWay Application Explorer:

1. Click the *iWay Events* tab.

The iWay Adapters window opens.

2. In the right pane, click *Operations* and select *Add a new port*.

The Create Event Port window opens in the right pane.

- a. Type a name for the event port and provide a brief description.
- b. Select a disposition protocol (for example, File) from the drop-down list.
- c. In the Disposition URL field, provide a destination where the event data will be written, for example,

`file://c:\temp\x.txt`

3. If you want to configure a JMS queue as a disposition, configure as follows:

- In the Disposition Protocol drop-down list, select *JMSQ*.
- In the Disposition field, enter a JMS destination, using the following format:

`jmsq:myQueueName@javax.jms.QueueConnectionFactory;jndiurl=t3://localhost:7001;jndifactory=weblogic.jndi.WLInitialContextFactory`

The following table lists the parameters and a description of each.

Parameter	Description
queue	JNDI name of a queue to which events are emitted.
Connection Factory	A resource which contains information about the JMS Server. The WebLogic connection factory is: <code>javax.jms.QueueConnectionFactory</code>
jndiurl	The URL of the Weblogic Server, <code>t3://host:port</code> Host is the machine name where the Weblogic Server is installed. Port 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> .

4. Click *OK*.



Port Name	Description	Disposition	Target
New	new	file://D:\in\A.txt	rdbms

5. Expand the iWay Event Adapters node to display the port.

Creating an Event Port

In the right pane, a table appears that summarizes all the information associated with the event port you created .

The event port also is listed under the ports node in the left pane.

You are now ready to associate the event port with a channel.

Procedure How to Edit an Event Port

To edit an existing event port:

1. In the left pane, select the event port you want to edit.
2. In the right pane, click *Operations* and select *Edit*.

The Edit Port window opens in the right pane.

Edit Port

Choose parameters of the port that you wish to edit.

Port Name:

Description:

Disposition Protocol:

Disposition:

3. In the event port configuration window, make any required changes and click *OK*.

Procedure How to Delete an Event Port

To delete an existing event port:

1. Select the event port you want to delete.
2. In the right pane, click *Operations* and select *Delete*.

A confirmation dialog box opens.

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

The event port is removed from the list in the left pane.

Creating Channels

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

Procedure How to Create a Channel

To create a channel using the iWay Servlet iWay Application Explorer:

1. Click the *iWay Events* tab.

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

2. Expand the *iWay Adapter* node.

The ports and channels nodes appear in the left pane.

3. Click the *channels* node.

4. In the right pane, click *Operations* and select *Add a new channel*.

The Add a new channel window opens in the right pane.

- a. Type a name for the channel, for example, New_CHANNEL.
 - b. Type a brief description.
 - c. Select a channel type from the drop-down list.
5. Click *Next*.

The Edit channels window opens in the right pane.

Edit NewChannel

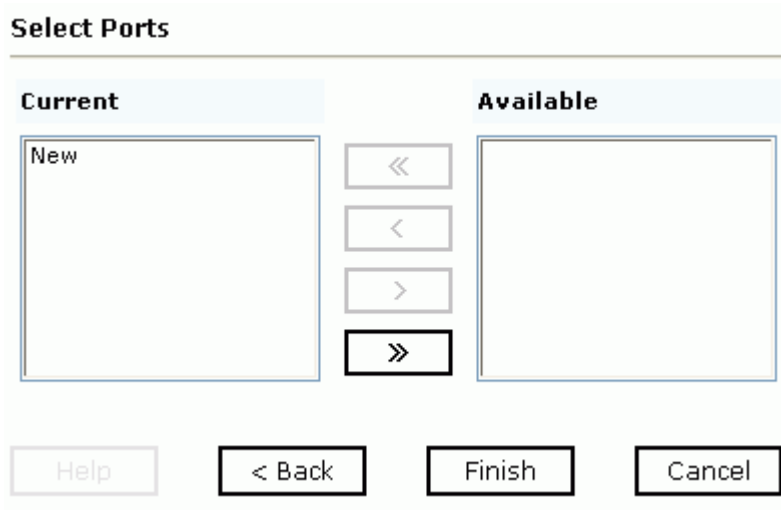
<u>Oracle</u> <u>Listener</u>	<u>SQL Server</u> <u>Listener</u>	EDA Server Listener
Host: <input type="text" value="localhost"/>		
Port: <input type="text" value="3446"/>		
Database Name: <input type="text" value="car"/>		
User: <input type="text"/>		
Password: <input type="password"/>		
Polling Interval: <input type="text" value="20"/>		
SQL Query: <input type="text" value="select * from car"/>		
Post Query: <input type="text" value="none"/>		
<div><input type="button" value="Help"/> <input type="button" value=" < Back"/> <input type="button" value=" Next >"/> <input type="button" value="Cancel"/></div>		

6. Select either Oracle, SQL, or EDA Server listener and enter the system information that is specific to your EIS.

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

7. Click *Next*.

The Select Ports window opens in the right pane.



8. Select an event port from the list of current ports and click the single right (>) arrow button to transfer it to the list of available ports.

To associate all the event ports click the double right (>>) arrow button.

9. Click *Finish*.

The summary window opens in the right pane.



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

The channel also is listed 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.

10. In the right pane, click *Operations* and select *Start the channel*.

The channel you created is now active..



The X that was over the icon in the left pane disappears.

11. If you want to stop the channel at any time, click *Operations* in the right pane and select *Stop the channel*.

Procedure How to Edit a Channel

To edit an existing channel:

1. In the left pane, select the channel you want to edit.
2. In the right pane, click *Operations* and select *Edit*.

The Edit channels window opens in the right pane.

3. Make any necessary changes to the channel configuration windows and click *Finish*.

Procedure How to Delete a Channel

Perform the following steps to delete an existing channel.

1. Select the channel you want to delete in the left pane.
2. Click *Operations* in the right pane and select *Delete*.

The following confirmation dialog box opens.

3. Click *OK* to delete the channel you selected.

The channel is removed from the list in the left pane.

Choosing a Listening Technique

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

An elaborate polling technology enables the specification of SQL SELECT statements to be executed on a periodic basis.

You can use the following techniques to listen to an RDBMS event:

- **Standard event processing with row tracking.** The listener polls a table, sends each newly inserted row to a preset destination, and uses a control table to track which row was most recently read. The control table prevents the most recently read row from being reread 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 3-9.

- **Standard event processing with row removal.** The listener polls a table, sends each newly inserted row to the Reply_to destination, and then deletes the new row from the table to prevent it from being reread during the next listening cycle.

You apply this technique when the source table is used to pass data to the adapter, and the table's 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 3-15.

- **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 the destination. The listener deletes new rows from the control table to prevent them from being reread 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 3-18.

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 RDBMS Table Listener read from the source table.

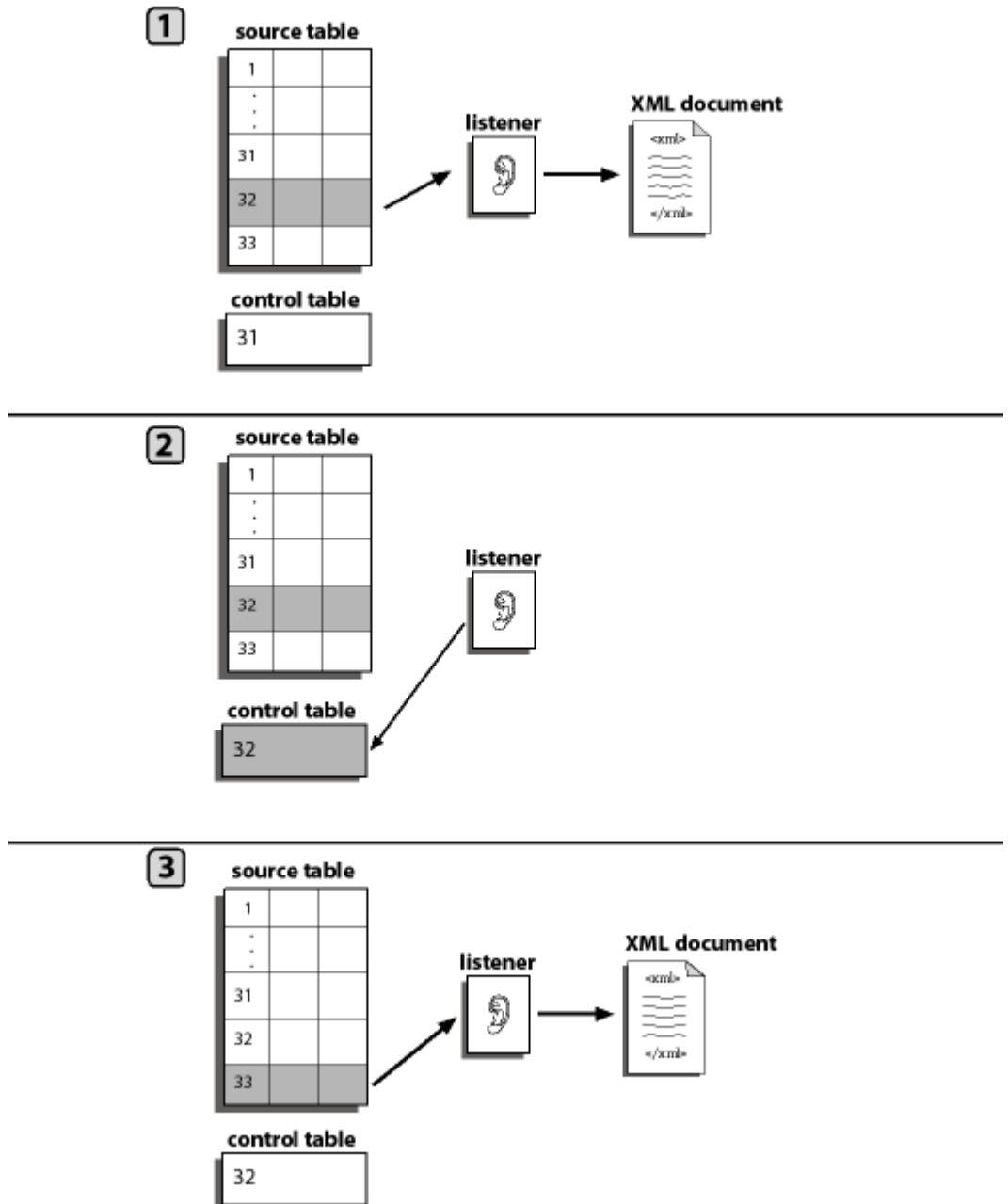
The single column of the control table corresponds to a column (or to a group of columns) in the source table that is unique, sortable, and indicates how recently the row was added to the source table relative to the other rows. For example, the first row added to the source table has the lowest value, 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's properties, configure the listener's SQL Post-query property to automatically update the control table's event key.

Standard Event Processing With Row Tracking

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 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 just read from the source table.

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's event key. The listener copies the row to an XML document and sends to the Reply_to destination.
2. The listener then 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*.
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 instructions about how to configure a listener, see *The SQL Post-query Parameter's Operators* on page 3-13.

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 SQL Post-query Parameter's Operators* on page 3-13.

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.

After creating a simple table in Oracle, the listener must be configured. .

Reference The SQL Post-query Parameter's Operators

When you configure an RDBMS 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, we will listen to the trans_event table using the row tracking technique, and use last_trans as the control table that will contain the last value of the primary key read from trans_event.

Last_trans is to contain a single value in a single row, and must be set up prior to configuring the RDBMS 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 be seeded with 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.

Using a SQL query/data manipulation tool supplied by the RDBMS vendor insert a record into the trans_event table:

- EVENT_ID=1
- LAST_NAME='Kaplan'
- TRANS_ID='03'

When setting up the listener, a Reply_to destination path is configured. A response document with the records data is deposited into the directory after the insert has been made.

The following is an example of a Reply-to/response document for the listener:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <RDBMSSLListener table="TRANS_EVENT">
    <row>
      <EVENT_ID type="2">1</EVENT_ID>
      <LAST_NAME type="12">Kaplan</LAST_NAME>
      <TRANS_ID type="1">03</TRANS_ID>
    </row>
  </RDBMSSLListener>
```

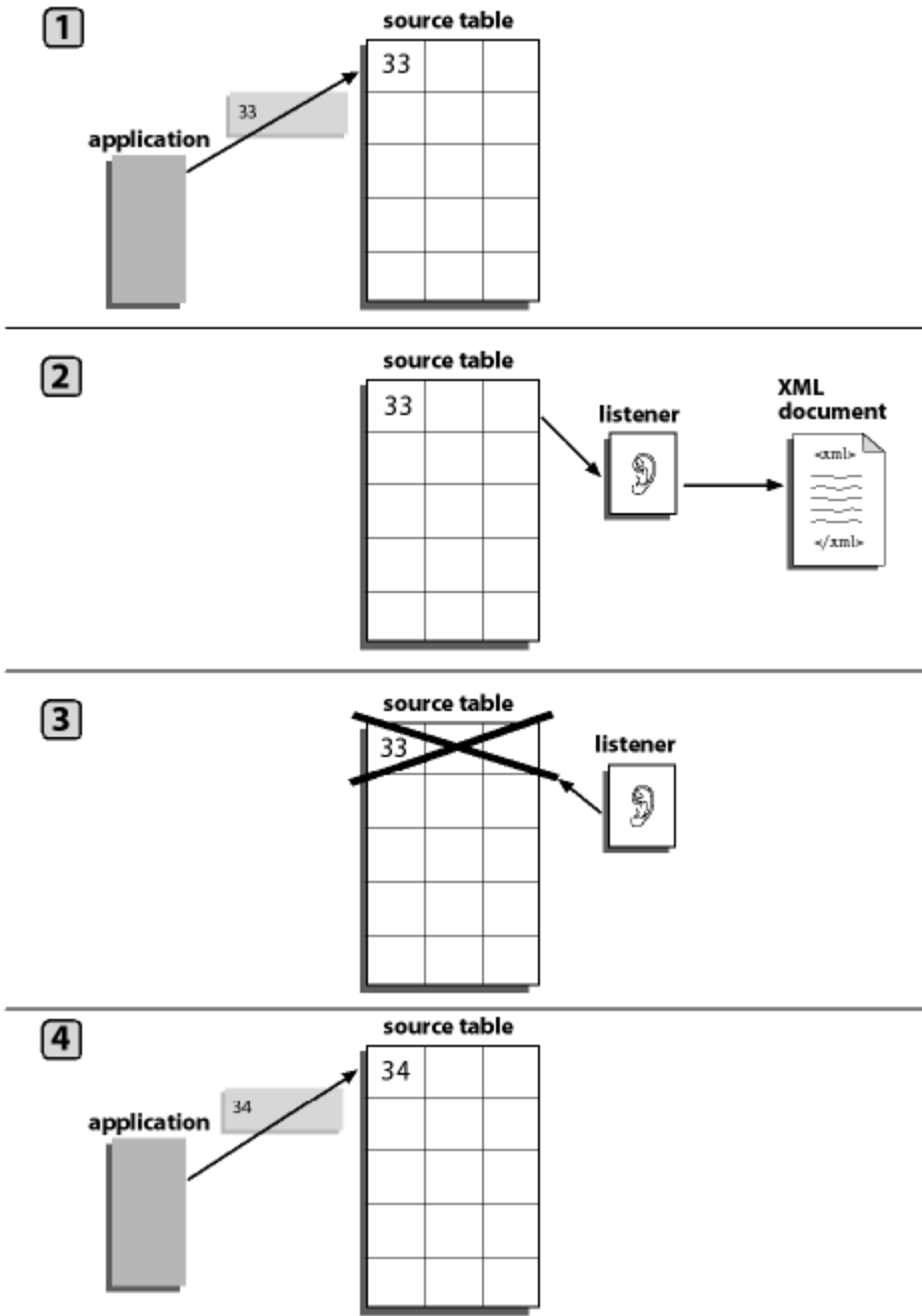
Configure the listener by specifying these properties:

- **Driver.** The JDBC Driver. For example,
`oracle.jdbc.driver.OracleDriver`
- **URL.** The URL of the RDBMS. For example,
`jdbc:oracle:thin:@Oracle11i.ubi.com:1521:test`
- **User Name.** The user name that is registered with the back-end RDBMS.
- **Password.** The password of the user name.
- **Table.** The table name containing the value of the highest primary key processed. For example: `Last_Trans`
- **Maximum Rows.** 1
- **SQL Query.** `SELECT * FROM TRANS_EVENT WHERE EVENT_ID>(select EVENT_ID from LAST_TRANS.EVENT_ID)`
- **SQL Post Query.** `UPDATE LAST_TRANS SET ?EVENT_ID`
- **Delete Keys.** (empty)
- **Generated XML format.** (field)
- **Polling Interval.** Interval in seconds.
- **Character Set Encoding.** UTF-8

Standard Event Processing With Row Removal

The standard event processing with row removal technique assumes that the source table is being used as a conduit to pass the data to the adapter, and that the table's rows do not need to persist. The RDBMS Table Listener periodically queries the source table: when it finds a row, it reads it and returns it to the Reply_to destination via an XML document. To ensure that the row is not read again when the listener next queries the table, the listener then deletes the row from the table.

Consider the following figure:



In this 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. It sends the document to the Reply_to destination.
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.

To implement this event processing technique, see *How to Implement Standard Event Processing With Row Removal* on page 3-17.

Procedure How to Implement Standard Event Processing With Row Removal

To implement the standard event processing with row removal technique, configure an RDBMS Table Listener. In addition to the required listener properties, for standard event processing with row removal you must also provide values for the following optional properties:

- **SQL Query**, the SQL SELECT statement that identifies the source table to which the adapter will listen, and with which it will query the table.
- **Delete Keys**, the list of keys that identify the rows that the adapter will automatically delete from the table.

For detailed instructions about configuring a listener, see *The SQL Post-query Parameter's Operators* on page 3-13.

Example Listening to stock_prices Using the Row Removal Technique

In this example, we will 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 Reply_to property associated with this RDBMS Table Listener. After generating the document the record is deleted from the table.

Configure the listener by specifying these properties:

- **Driver.** The JDBC Driver. For example,

```
oracle.jdbc.driver.OracleDriver
```

- **URL.** The URL of the RDBMS. For example,

```
jdbc:oracle:thin:@Oracle11i.ubi.com:1521:test
```
- **User Name.** The user name that is registered with the back-end RDBMS.
- **Password.** The Oracle Applications user ID authorized to access the Oracle Applications system.
- **Table.** stock_prices
- **Maximum Rows.** 1
- **SQL Query.** (select * from stock_prices)
- **SQL Post Query.** (empty)
- **Delete Keys.** (ric)
- **Polling Interval.** Interval in seconds.

For a description of these properties, see *The SQL Post-query Parameter's Operators* on page 3-13.

Trigger-based Event Processing

Trigger-based event processing is a technique for listening to multiple joined relational tables. It is also helpful for detecting when a row has been deleted or updated.

The trigger-based technique provides several benefits:

- **It improves performance** when listening for events in a group of large joined tables. When processing joined tables, the RDBMS creates a Cartesian product working table; when the joined tables are large, the interim working table will be *very* large. The standard technique of processing RDBMS 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 RDBMS Table Listener to query a single small control table, and by writing to the control table only when an event actually occurs.
- **It increases the number of event types that the adapter recognizes.** Using the trigger-based technique, you can tell when a row has been updated, deleted, or inserted. Using the standard technique, you can tell only when a row has been inserted.

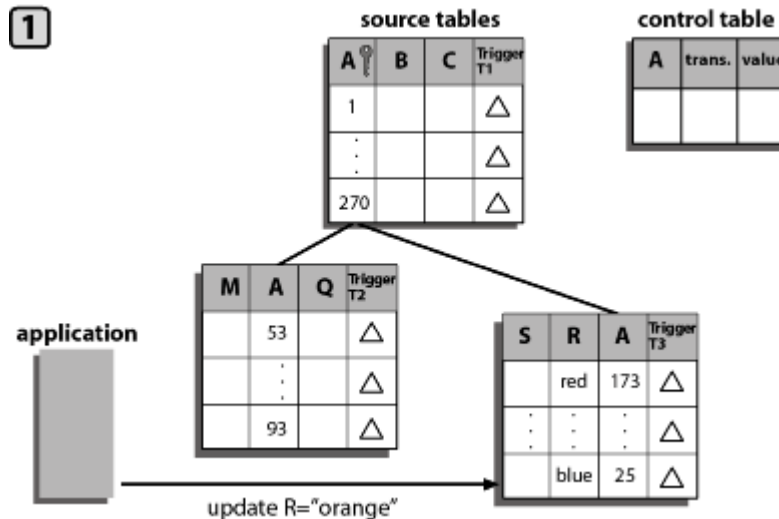
To use the trigger-based technique, you assign a trigger to each table that you want to monitor. When a value changes, it fires the corresponding trigger, which writes data to a control table. The iWay Adapter for RDBMS listens to this control table by running a query against it. When it finds a row in the control table, it reads it and returns it to the Reply_to destination via an XML document. To ensure that the row is not read again when the listener next queries the table, the listener then 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 which is represented in the database by a hierarchical set of tables.

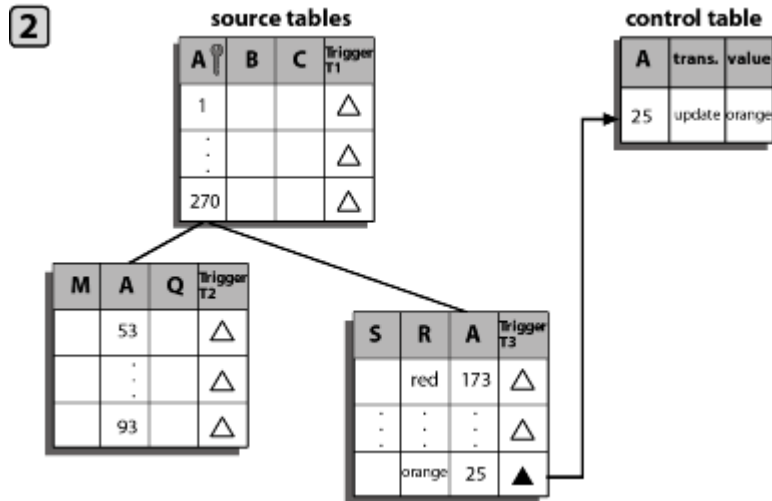
You manage the triggers using a native RDBMS tool (such as SQL*Plus for Oracle tables), and configure the listener using the iWay Web Console.

Consider the following figures:

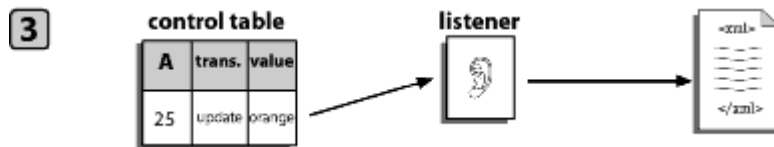
1. Your application updates a row in a group of related source tables.



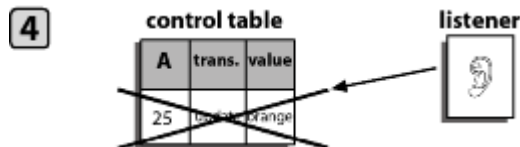
- The update causes a row trigger to fire in the changed table. The trigger inserts a row into the control table. The new control table row includes the key value (25), the type of transaction (update), and the new cell value (orange).



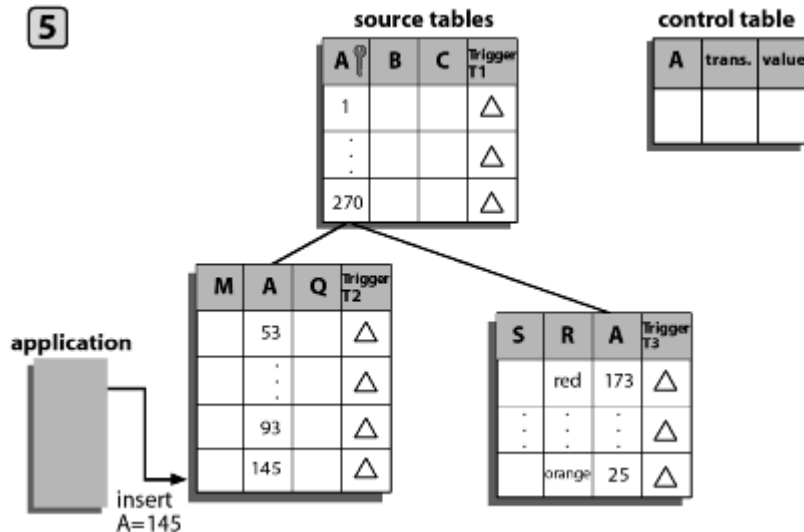
- The listener queries the control table and copies the new row to an XML document. It sends the document to the Reply_to destination.



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



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



For an summary of how to implement this technique, see *How to Implement Trigger-based Event Processing* on page 3-21.

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 has changed, regardless of which of the entity's tables have changed.
 You can store a variety of information in the control table, from the key of the entity that was inserted, updated, or deleted, to the name of the table and field that was updated.
 The control table's design is a function of your application's business logic. 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 which kinds of events to monitor (insertions, deletions, and/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. Which triggers you assign, and to which tables you assign them, will be determined by what kind of change you want to monitor. The triggers will implement much of the event-processing logic. For a sample trigger, see *Trigger on WIP_ENTITY_NAME Column in an Oracle Table* on page 3-22.

For example, consider a bill of material 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, where 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).

As another example, 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 would be logged.

3. Configure the listener in the iWay Web Console. In addition to the required listener properties, for trigger-based event processing you must also provide values for the following optional properties:
 - SQL Query, the SQL SELECT statement that identifies the control table to which the adapter will listen, and with which it will query the table to determine changes in the source tables.
 - Delete Keys, the list of keys that identify the rows that the adapter will automatically delete from the control table.

For detailed instructions about configuring a listener, see *The SQL Post-query Parameter's Operators* on page 3-13.

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, it 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;
```

APPENDIX A

Using the WebLogic Workshop to Access VSAM

Topics:

- Using the WebLogic Workshop to Access VSAM
- Running the JWSNAME Web Service from WebLogic Workshop

This topic describes how to produce and access a Web services generated from a VSAM database.

Using the WebLogic Workshop to Access VSAM

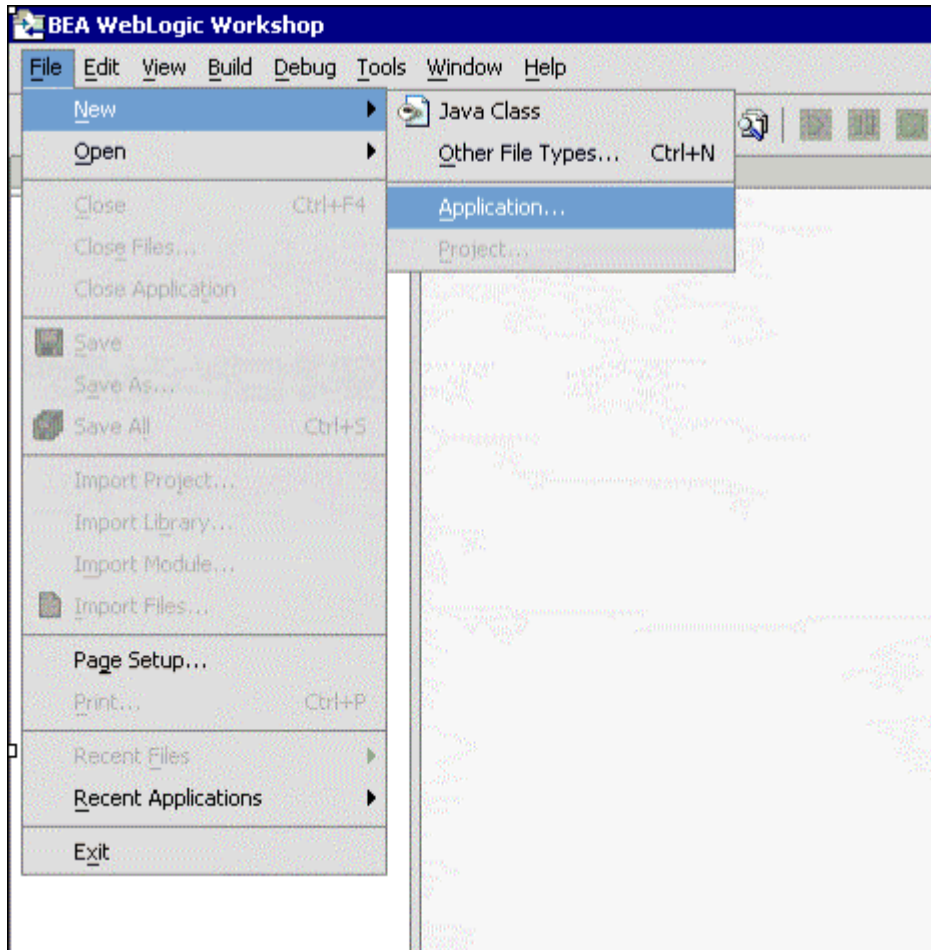
WebLogic Workshop provides a framework for building Web services. The Web services that you build with WebLogic Workshop are enterprise-class services, and WebLogic Workshop provides simple controls for connecting to your enterprise resources. At the same time, WebLogic Workshop simplifies the process 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 have already created and tested a Web service using the iWay 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*.

1. To start WebLogic Workshop, from the Start menu, choose *Programs, WebLogic Platform 8.1, WebLogic Workshop*, and then *WebLogic Workshop*.
2. Create a new application:

- a. From the File menu, select *New*, and then *Application*.

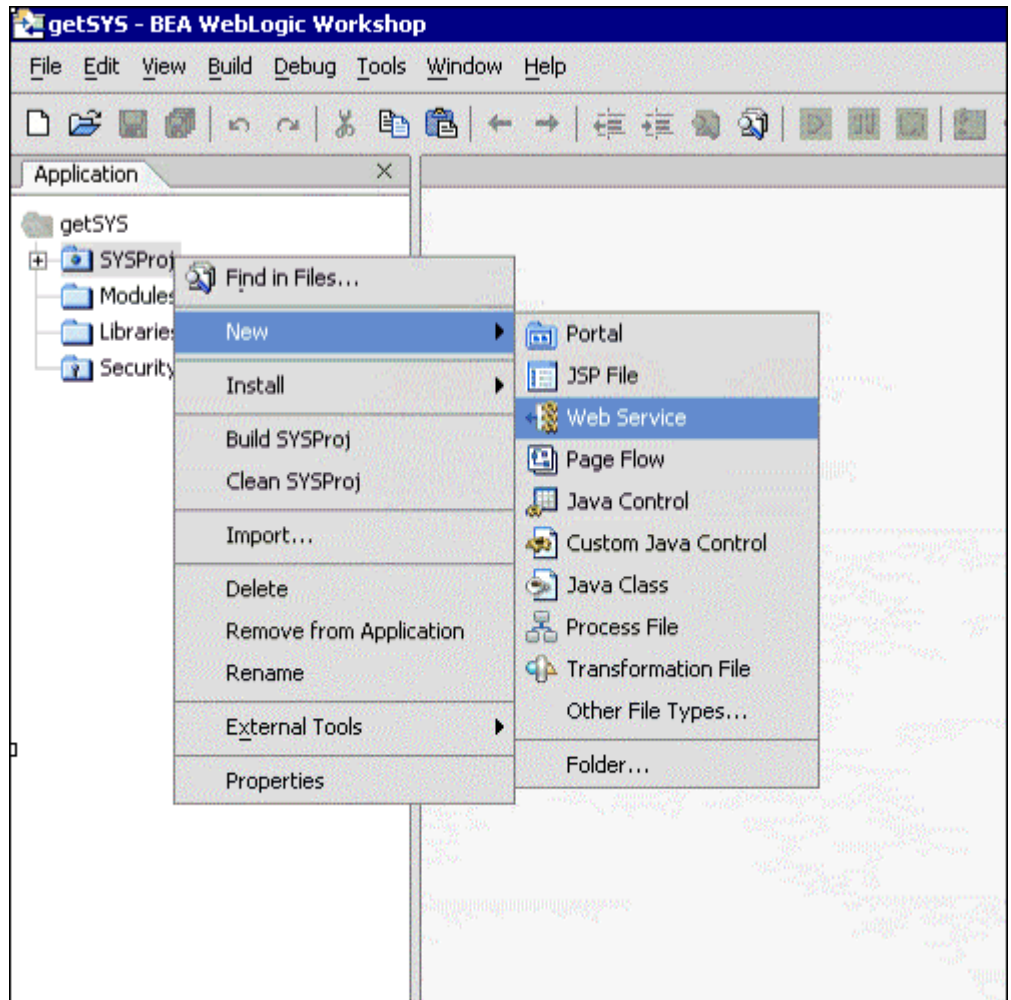


- b. In the upper-left pane, select all, and then select *Empty Application*.
- c. In the directory field, type *C:\IWAYSrv*.
3. Click *Create*.
4. In the Application tab, right-click the IWAYSrv folder, and select *New Project*.
5. In the upper-left pane, select all, and then select *Web Project*. In the name field, type *BAPIproj*.
6. 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. But, 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.

7. In the Application tab, right-click the BAPIProj folder and select *New*, and then *Web Service*.

The New Web Service dialog box opens, as shown in the following graphic.



8. In the upper-left pane, select all, and then select *Web Service* in the right pane. In the name field, type *JWSNAME.jws*.

9. Click *Create*.

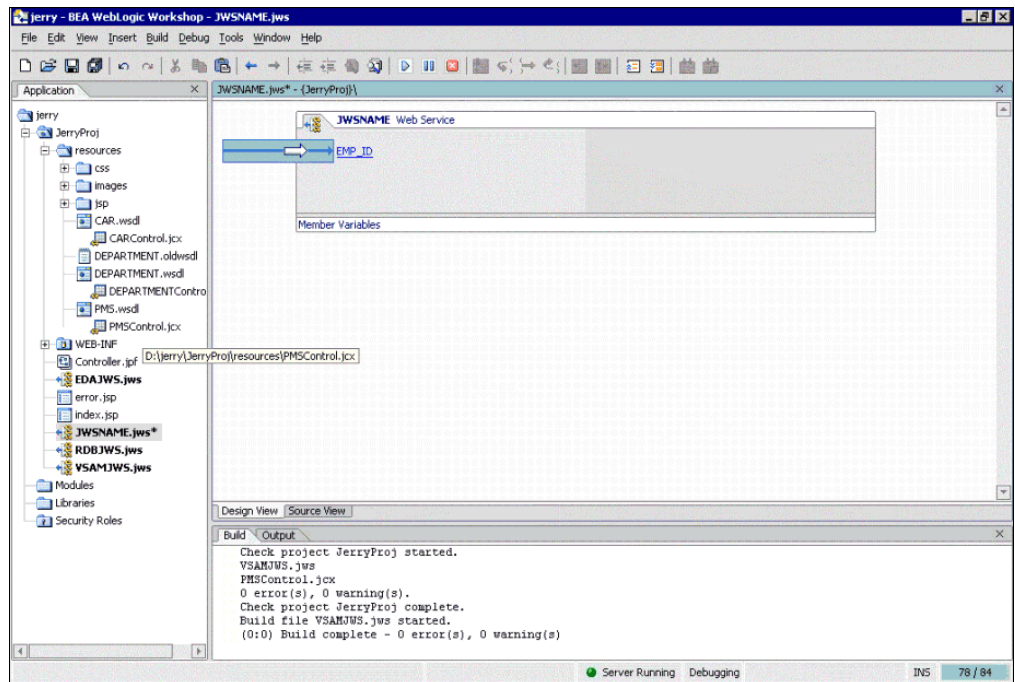
The design view window opens.

Web services expose their functionality through methods, which clients invoke when they want to request something from the Web service. In this case, clients will invoke a method to call the PMS Control, which will be exposed later in this procedure.

10. If it is not selected already, click the *Design View* tab.

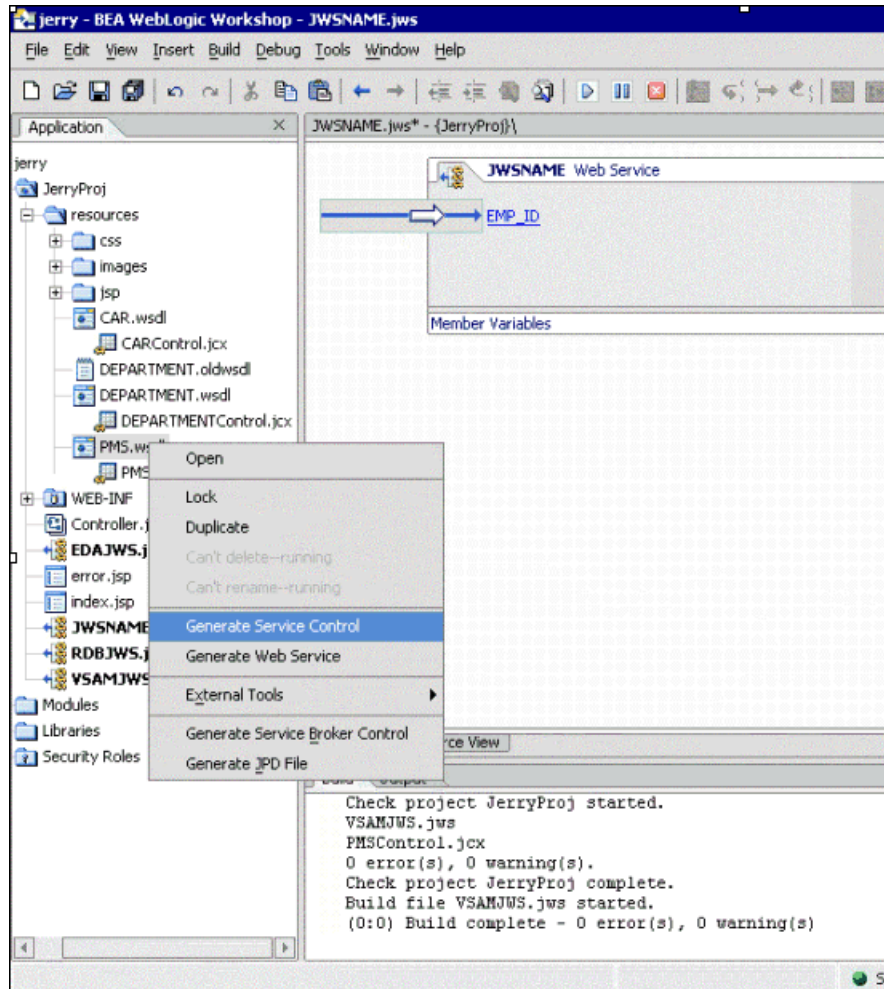
11. From the Insert menu, select *Method*.

12. In the space provided, replace method1 with *EMP_ID*, and press *Enter*.

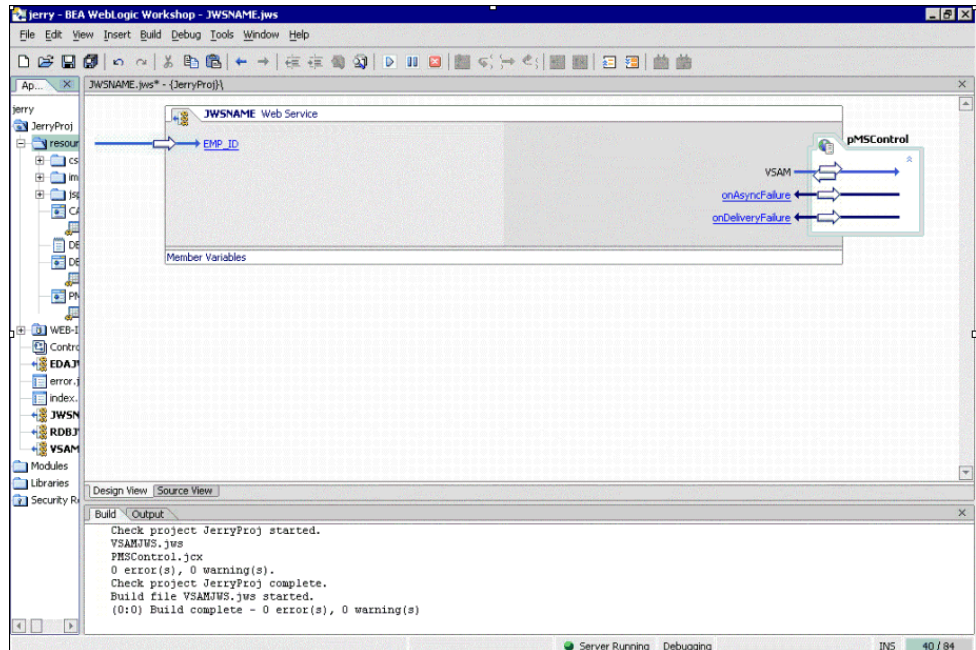


13. Right-click the Resources sub-folder project and select *Import*. Import the PMS.WSDL saved in Creating WSDL to Access the Web Service. For more information on creating a WSDL file, see Chapter 2, *Creating XML Schemas or Business Services*.

14. Right-click the WSDL file to generate a Java Control file.



15. Drag the PMS.jcx file onto the JWSNAME Web service as follows:



16. Click the Source View tab to modify the source code and call the iWay PMS Web service.

- a. 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;
}
```

17. Use the Control + S key sequence to save your current work.

The resulting Java code should look 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;
    }
}
```

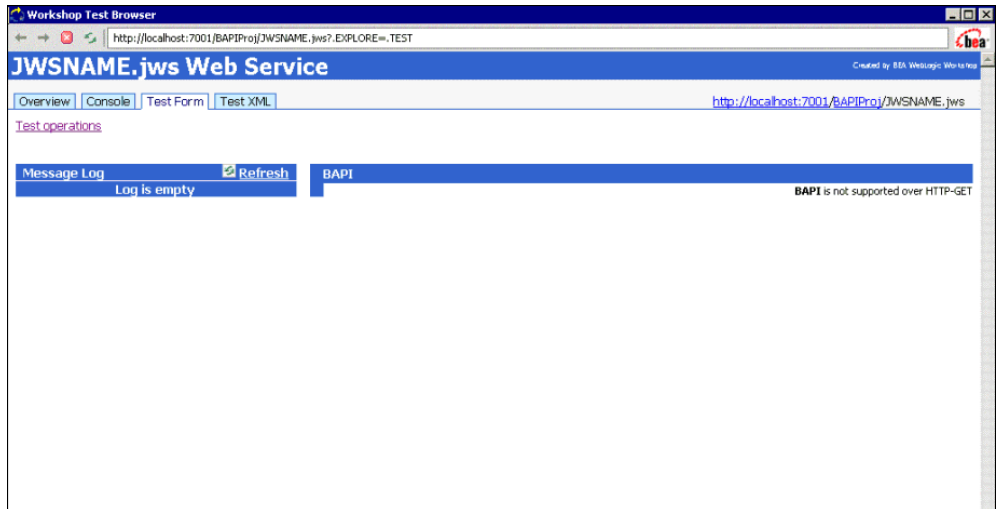
Running the JWSNAME Web Service from WebLogic Workshop

When you have created a new Web service tutorial application, you must ensure that WebLogic Server is running while you build your Web service. You can confirm whether WebLogic Server is running by looking at the status bar at the bottom of WebLogic Workshop. If WebLogic Server is running, a green ball is displayed. If WebLogic Server is not running, a red ball is displayed. If you see the red ball in the status bar, then start WebLogic Server, as described in the following procedure.

Procedure How to Start WebLogic Server

1. From the Tools menu, select *WebLogic Server*, and then *Start WebLogic Server*.
2. To deploy the application to WebLogic, select *Tools*, and then *Deploy Application*.
3. Click the *Start* button on the toolbar to start the application.

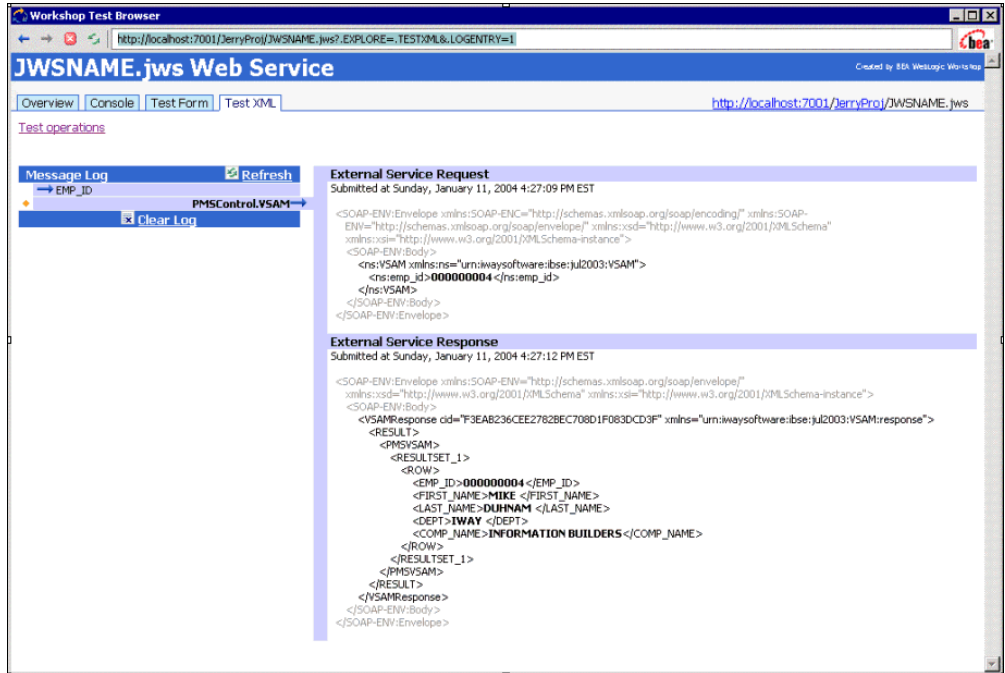
The following test window opens.



4. Click the Test XML tab to enter and test the XML stream to be passed to the Web service.
5. Replace the string XML input as follows:


```
<EMP_ID xmlns="http://www.openuri.org/">
  <!--Optional:-->
  <empid>000000004</empid>
</EMP_ID>
```
6. Click the *EMP_ID* button to submit the request.

Once the SOAP request is sent to the VSAM adapter, the following response is returned:



Workshop Test Browser

http://localhost:7001/JerryProj/JWSNAME.jws?EXPLORE=.TESTXML&LOGENTRY=1

JWSNAME.jws Web Service

Created by EDA WebLogic Workshop

Overview Console Test Form Test XML

Test operations

Message Log Refresh

EMP_ID PMSControl.VSAM Clear Log

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>
```

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