Information Builders

iWay

iWay Technology Adapter for RDBMS User's Guide

Version 7.0.x and Higher

August 16, 2018

Contents

Preface

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

Note: This Release 7.0.x content is currently being updated to support iWay Release 8.0.x software. In the meantime, it can serve as a reference for your use of iWay Release 8. If you have any questions, please contact *Customer_Success@ibi.com*.

How This Manual Is Organized

This manual includes the following chapters:

	Chapter/Appendix	Contents
1	Introducing the iWay Technology 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.
3	Listening for Database Events	Describes how to configure a listener to listen to a database event.
A	Configuring the Adapter in an iWay Environment	Describes how to configure the adapter in the Service Manager console.
B	JDBC Drivers	Lists the supported JDBC drivers for use with the adapter.

Documentation Conventions

The following table describes the documentation conventions that are used in this manual.

Convention	Description

`THIS TYPEFACE`
or
`this typeface`
	Denotes syntax that you must enter exactly as shown.

Convention	Description
this typeface	Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option that you can click or select.
underscore	Indicates a default setting.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices. Type one of them, not the braces.
|	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis (...).
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

Visit our Technical Documentation Library at *http://documentation.informationbuilders.com*. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have any questions about this product?

Join the Focal Point community. Focal Point is our online developer center and more than a message board. It is an interactive network of more than 3,000 developers from almost every profession and industry, collaborating on solutions and sharing tips and techniques. Access Focal Point at *http://forums.informationbuilders.com/eve/forums*.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our website, *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 *http://www.informationbuilders.com* also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

Call Information Builders Customer Support Services (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your 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.

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

Help Us to Serve You Better

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

The following tables list the environment information our consultants require.

Platform	
Operating System	
OS Version	
JVM Vendor	
JVM Version	

The following table lists the deployment information our consultants require.

Adapter Deployment	For example, iWay Business Services Provider, iWay Service Manager
Container	For example, WebSphere

Version	
Enterprise Information System (EIS) - if any	
EIS Release Level	
EIS Service Pack	
EIS Platform	

The following table lists iWay-related information needed by our consultants.

iWay Adapter	
iWay Release Level	
iWay Patch	

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

Request/Question	Error/Problem Details or Information
Did the problem arise through a service or event?	
Provide usage scenarios or summarize the application that produces the problem.	
When did the problem start?	
Can you reproduce this problem consistently?	
Describe the problem.	
Describe the steps to reproduce the problem.	
Specify the error message(s).	

Request/Question	Error/Problem Details or Information
Any change in the application environment: software configuration, EIS/database configuration, application, and so forth?	
Under what circumstance does the problem *not* occur?	

The following is a list of error or problem files that might be applicable.

- ❑ Input documents (XML instance, XML schema, non-XML documents)
- ❑ Transformation files
- ❑ Error screen shots
- ❑ Error output files
- ❑ Trace files
- ❑ Service Manager package or archive to reproduce problem
- ❑ Custom functions and agents in use
- ❑ Diagnostic Zip
- ❑ Transaction log
- ❑ Archive File
- ❑ IIA

For information on tracing, see the *iWay Service Manager User's Guide*.

User Feedback

In an effort to produce effective documentation, the Technical Content Management staff welcomes your opinions regarding this document. Please use the Reader Comments form at the end of this document to communicate your feedback to us or to suggest changes that will support improvements to our documentation. You can also contact us through our website, *http://documentation.informationbuilders.com/connections.asp*.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

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

For information on course descriptions, locations, and dates, or to register for classes, visit our website (*http://education.informationbuilders.com*) or call (800) 969-INFO to speak to an Education Representative.

Chapter 1

Introducing the iWay Technology Adapter for RDBMS

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

In this chapter:

Introduction

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

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

Using the Adapter With Relational Databases

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

- **Service.** The adapter sends requests to the database, using either SQL queries or stored procedures.
- **Event.** The adapter listens for database table activity.

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

Key features of the adapter include:

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

Using the Adapter to Send a Request

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

Using the Adapter Listener

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

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

The listener can:

- Monitor data changes by repeatedly performing an SQL query.

  The SQL listener also supports customized user exits with Java™ classes to define custom operations on the row sets.
- Be configured to operate one row at a time or to operate on sets of data.

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

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

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

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

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

- Support complex configurations.

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

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

- Support user-defined exits.

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

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

Using the Adapter to Access Non-relational Databases

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

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

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

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

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

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 environment. The adapter connects to the iWay data adapters hosted in the server component using a Java-based connection.

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

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

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

In the usual non-relational database access scenario, the iWay Technology Adapter for RDBMS connects to the iWay Adapter for VSAM (hosted by the server component) using JDBC standards. iWay Explorer is used to configure this connection. You can create web services for SQL, parameterized SQL, stored procedures, table functions, and batches using iWay Explorer. You also can use iWay Explorer to create events that occur within the database, such as an insert to a VSAM file or a modification of a VSAM record.

Accessing Stored Procedures for Non-relational Data Sources

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

iWay stored procedures on the server component can be created to enable complex multiplatform joins, specialized routines, and so forth. iWay stored procedures also enable COBOL or RPG programs to be executed. To use the adapter, an iWay stored procedure must be catalogued before iWay 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 Explorer, the adapter creates web services that can be used to run the stored procedures from any web service client.

Deployment Information for Your iWay Adapter

Your iWay adapter works in conjunction with one of the following components:

- iWay Service Manager
- iWay Business Services Provider (iBSP)

When hosted in an iWay environment, the adapter is configured through iWay Service Manager and iWay Explorer. iWay Explorer is used to configure system connections, create SQL statements, batches, and web services as well as configure event capabilities. iWay Service Manager can access this configuration information through the iWay7 repository to create a robust integration solution.

iWay Service Manager

iWay Service Manager is the heart of the Universal Adapter Framework and is an open transport service bus. Service Manager uses graphical tools to create sophisticated integration services without writing custom integration code by:

- Creating metadata from target applications.
- Transforming and mapping interfaces.
- Managing stateless processes.

Its capability to manage complex adapter interactions makes it ideally suited to be the foundation of a service-oriented architecture.

iWay Explorer

iWay Explorer uses a tree metaphor to introspect a system for metadata. The explorer enables you to generate XML schemas and add web services for the associated object. In addition, you can create ports and channels to listen for events in a system. External applications that access a system through the adapter use either XML schemas or web services to pass data between the external application and the adapter.

iWay Business Services Provider (iBSP)

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

iBSP 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.

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.

Information Roadmap

The following table lists the location of deployment and user information for components of the iWay Technology Adapter for RDBMS.

Deployed Component	For more information, see
iWay Service Manager	Appendix A of this guide *iWay Service Manager User's Guide*
iWay Explorer	Chapters 2 and 3 of this guide *iWay Installation and Configuration*
iWay Business Services Provider (iBSP)	*iWay Installation and Configuration*
iWay server component	*iWay Server Installation* *iWay Server Administration for UNIX, Windows, OpenVMS, OS/400, OS/390, and z/OS* *iWay Data Adapter Administration for UNIX, Windows, OpenVMS, OS/400, OS/390, and z/OS*

Chapter 2

Creating XML Schemas or Business Services

This section describes how to use iWay Explorer to:

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

In this chapter:

Generating Schemas and Business Services

You can use iWay Explorer to connect to relational databases, such as Oracle or Informix and to non-relational databases, such as an IMS database. The majority of the interactive information that is generated for the RDBMS adapter, such as, targets, statements, and web services are stored in the iBSP repository.

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

The following highlights the tasks you will perform to generate request and response document schemas and a business service, and provides references to the related procedures found in this chapter.

- Start iWay Explorer.

  You can open a new or existing connection to a relational or non-relational database management system, as described in *Creating and Managing a Connection* on page 24.

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

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

- Generate XML schemas.

  The schemas define request and response documents for your SQL statements and stored procedures, as described in *Creating an SQL Statement and Generating Schemas* on page 54.

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

- Create request documents.

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

  You also may want to examine the metadata describing your SQL statements and procedures, as described in *Viewing Metadata* on page 35. For information about request and response document formats, see *Request and Response Documents*.

- Generate a business service (also known as a web service) for an SQL statement or stored procedure. For more information on web services, see *Understanding iWay Business Services* on page 101.

Starting iWay Explorer

The following section describes how to start iWay Explorer.

Procedure: How to Start iWay Explorer

To start iWay Explorer:

1. Ensure the server is started where iWay Explorer is running.
   a. Click the Windows *Start* menu.
   b. Select *Programs*, *iWay 7.0 Service Manager*, and then click *Start Service Manager*.
2. Enter the following URL in your browser window:

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

   where:

   hostname

   Is the machine where iWay Explorer is installed.

   port

   Is the SOAP port number for iBSP. The default SOAP port is 9000.

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

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

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

The Available Hosts drop-down list specifies to which Servlet iBSP instance you connect.

For more information on accessing different instances of a Servlet iBSP, see the *iWay Installation and Configuration* documentation.

Creating and Managing a Connection

To access an adapter, you must define a target that connects to the adapter. After the defined target is created, it automatically is saved in the iBSP repository. You must establish a connection to the defined target every time that you start iWay Explorer.

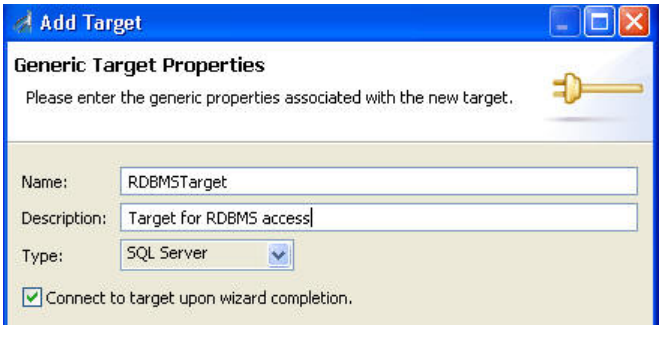
Procedure: How to Define a New Target

This procedure assumes that you are connected to Application Explorer and have expanded the Application Explorer node in the navigation tree.

To define a new target:

1. In the navigation tree, right-click *RDBMS*, and click *Add Target* from the menu.

   The Add Target / Generic Target Properties dialog box opens, as shown in the following image.



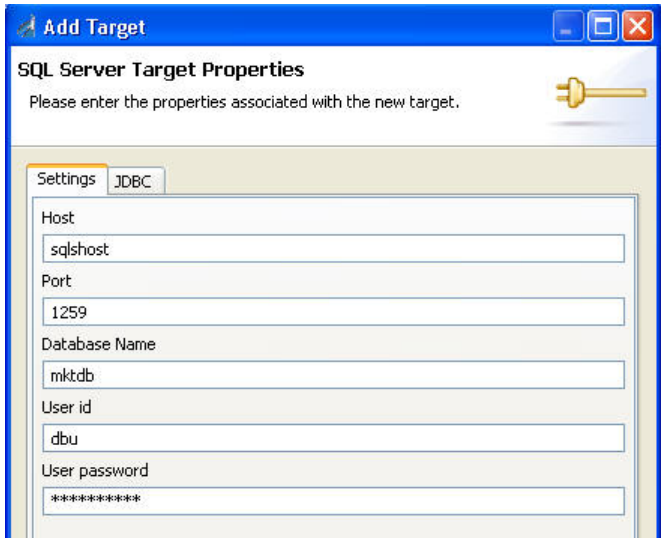
2. Supply the values for the fields on the dialog box as follows.
   a. In the Name field, type a descriptive name for the target, for example, *RDBMSTarget*.
   b. In the Description field, optionally type a brief description of the target.
   c. From the Type drop-down list, click the type of target that you will connect to. The choices are SQL Server, HSQL, EDA Server, iWay Server, WebFOCUS Server, ODBC, JDBC, and Datasource.
   d. Select the *Connect to target upon wizard completion* check box if you want Application Explorer to automatically connect to this target after you create it.

      If you deselect this option, Application Explorer will not automatically connect to the target. From the navigation tree, you can connect to an individual target when you want to access RDBMS. For instructions, see *How to Connect to a Defined Target* on page 33.

3. Click *Next*.

The Add Target dialog box opens on the right pane. It includes fields that you complete to set the connection properties for the selected target type. The fields that appear on the dialog box are specific to the type of database to which you are connecting and the target type that you selected.

The following image shows the Add Target dialog box with the connection properties for the SQL Server type. The label for a required field appears in red until you have supplied a value for the field.



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

4. Enter connection information that is specific to the database to which you want to connect.

   For information on connection properties for a **SQL Server** target type, see *SQL Server Connection Properties* on page 26.

   For information on connection properties for an **HSQL** target type, see *HSQL Connection Properties* on page 27.

   For information on connection properties for an **EDA Server** target type, see *EDA Server Connection Properties* on page 28.

   For information on connection properties for an **iWay Server** target type, see *iWay Server Connection Properties* on page 28.

For information on connection properties for a **WebFOCUS Server** target type, see *WebFOCUS Server Connection Properties* on page 29.

For information on connection properties for an **ODBC** target type, see *ODBC Connection Properties* on page 30.

For information on connection properties for a **JDBC** target type, see *JDBC Connection Properties* on page 30.

For information on connection properties for a **Datasource** target type, see *Data Source Connection Properties* on page 31.

5. Supply the connection information for the target to which you want to connect, and click *Finish* when you are done.

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

   For information on connecting to the target, see *How to Connect to a Defined Target* on page 33.

Reference: SQL Server Connection Properties

The following table lists and describes the connection properties for the SQL Server target type.

Property	Description
Settings Tab	
Host	DNS or IP name of the server where the database instance resides.
Port	Port number on which the database is listening.
Database Name	Specific name of the database or data source to which you will connect.
User id	User ID that can access the database. The user ID must have database access to the interface tables that are used.
User password	Password associated with the specified user ID.
JDBC Tab	

Property	Description
URL	Allows you to specify optional settings for the JDBC URL.

Reference: HSQL Connection Properties

The following table lists and describes the connection properties for the HSQL target type.

Property	Description
Settings Tab	
Driver Type	Select from the following, depending on how you are running your HSQL database: ❑ Standalone ❑ Server ❑ Web Server ❑ In-Memory
User id	User ID that can access the database.
User password	Password associated with the specified user ID.
Standalone Tab	
Path	If you are running HSQL in Standalone mode, provide the full path to the database file.
Server/Web Server Tab	
Host	If you are running HSQL in Server or Web Server mode, provide the name of the machine on which the database is running.
Port	If you are running HSQL in Server or Web Server mode, provide the port number on which the database is listening.
Server Name	Name of the database server.

Property	Description
JDBC Tab	
URL	Allows you to specify optional settings for the JDBC URL.

Reference: EDA Server Connection Properties

The following table lists and describes the connection properties for the EDA Server target type.

Property	Description
Host	DNS or IP name of the server where the database instance resides.
Port	Port number on which the database is listening.
User id	User ID that can access the database. The user ID must have database access to the interface tables that are used.
User password	Password associated with the specified user ID.
Server Name	For an iWay server component, the name of the service node to which you are connecting.
Code Page	
Options	

Reference: iWay Server Connection Properties

The following table lists and describes the connection properties for the iWay Server target type.

Property	Description
Host	DNS or IP name of the server where the database instance resides.
Port	Port number on which the database is listening.

Property	Description
User id	User ID that can access the database. The user ID must have database access to the interface tables that are used.
User password	Password associated with the specified user ID.
Server Name	For an iWay server component, the name of the service node to which you are connecting.
Code Page	
Options	

Reference: **WebFOCUS Server Connection Properties**

The following table lists and describes the connection properties for the WebFOCUS Server target type.

Property	Description
Settings Tab	
Host	DNS or IP name of the server where the database instance resides.
Port	Port number on which the database is listening.
User id	User ID that can access the database. The user ID must have database access to the interface tables that are used.
User password	Password associated with the specified user ID.
Server Name	For an iWay server component, the name of the service node to which you are connecting.
Code Page	
Options	
JDBC Tab	

Property	Description
URL	Allows you to specify optional settings for the JDBC URL.

Reference: ODBC Connection Properties

The following table lists and describes the connection properties for the ODBC target type.

Property	Description
Data Source Name	Specific name of the database or data source to which you will connect.
User id	User ID that can access the database. The user ID must have database access to the interface tables that are used.
User password	Password associated with the specified user ID.

Reference: JDBC Connection Properties

The following table lists and describes the connection properties for the JDBC target type.

Property	Description
Driver	Name of the driver used to access the database that you want to connect to. For more information, see your database documentation.
URL	For a JDBC connection, the JDBC driver-specific URL used to connect to the RDBMS. For information on using driver options in the URL definition, for example, selectMethod=cursor for Microsoft SQL, see *Using URL Options (For JDBC Connections Only)* on page 32.
User id	User ID that can access the database. The user ID must have database access to the interface tables that are used.

Property	Description
User password	Password associated with the specified user ID.

Reference: **Data Source Connection Properties**

The following table lists and describes the connection properties for the Datasource target type.

The database types that appear in the DBMS drop-down list depend on the available JDBC drivers in the iWay7\lib directory. You must use the JDBC data source if it is available in the drop-down list for your available target types. The JDBC data source targets will appear in the drop-down list only if they are registered with the JNDI naming service. If they are not registered, you can then use the JDBC option to connect to the database.

Property	Description
Initial Context Factory	JNDI context.INITIAL_CONTEXT_FACTORY that is provided by the JNDI service provider.
Provider URL	For a JDBC connection, the JDBC driver-specific URL used to connect to the RDBMS. For information on using driver options in the URL definition, for example, selectMethod=cursor for Microsoft SQL, see *Using URL Options (For JDBC Connections Only)* on page 32. For a data source connection, the URL to use to contact the JNDI provider. The syntax of this URL depends on the JNDI provider being used. This value corresponds to the standard JNDI property, `java.naming.provider.url`
JNDI Name	JNDI name of a queue to which events are emitted.
DBMS	Oracle, SQL Server, Sybase, DB2_type2, DB2_type3, DB2_type4, Informix, EDA Server, Progress, Cache, Domino, DominoODBC, HSQL, Teradata, ODBC, or Other.

For information on connecting to the target, see *How to Connect to a Defined Target* on page 33.

Reference: Using URL Options (For JDBC Connections Only)

Each driver has specific JDBC system properties, most of which can be used in the connection string. The URL options can be added to the JDBC URL in the connection properties for a specific target. For more information on these properties, refer to the documentation on the specific driver.

- **For Microsoft SQL**

  You can use the selectMethod as a connection parameter in the URL.

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

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

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

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

- **For DB2**

  You can include a translate binary option to control how binary and varbinary data values are treated. On the OS/400 system, for example, if a field is tagged with CCSID 65535, you can set an optional translate binary parameter to true, which instructs the JDBC driver to translate the field to EBCDIC characters; for example,

  ```
  jdbc:db2://host:port/DatabaseName;translate binary=true;trace=true
  ```

  where`trace=true` turns on JDBC tracing for the DB2.

  The translate binary option forces the JDBC driver to treat binary and varbinary data values as if they were char and varchar data. The default is set to false. This setting is usually needed if the columns were created using different character coding values.

- **For Sybase**

You can ensure the use of dynamic statements by using the DYNAMIC_PREPARE option; for example,

```
jdbc:sybase:Tds:host:port/DatabaseName?&DYNAMIC_PREPARE=true
```

By default, this option is set to false.

Procedure: How to Connect to a Defined Target

1. On the iWay Explorer tab, connect to and expand the *Application Explorer* node.
2. Underneath the Application Explorer node, expand the *RDBMS* node.

   The following image shows the RDBMS node expanded, with the RDBMSTarget selected. The gray connection icon to the left of the RDBMSTarget indicates that the connection is closed.

3. Right-click the connection that you want to open (for example, *RDBMSTarget*), and click *Connect* from the menu.

   The Target Connection Dialog pane opens and displays the connection information that you supplied when you defined the target.

4. Verify the connection properties. If required, provide the password in the User password field, and then click *Finish*.

   If the properties are correct and the RDBMS component is available, the connection icon changes from gray to green to indicate that you are connected to the defined target. Otherwise, an error message appears in the right pane.

   You can expand a connected target to display the folders that are associated with that target.

Disconnecting From a Defined Target

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

Procedure: How to Disconnect From a Defined Target

1. On the iWay Explorer tab, connect to and expand the *Application Explorer* node.
2. Underneath the Application Explorer node, expand the *RDBMS* node.

The following image shows the RDBMS node expanded, with the RDBMSTarget selected. The green connection icon to the left of the RDBMSTarget indicates that the connection is open.

3. Right-click the connection that you want to close (for example, *RDBMSTarget*), and click *Disconnect from Target* from the menu.

   Disconnecting from the application closes the connection, but the connection still appears in the left pane so that you can reopen it when desired.

   The target node now has a gray connection icon, indicating that the connection is closed, as shown in the following image.

   When you want to reestablish a connection, the Connect option is available from the pop-up menu.

Editing a Defined Target

After you create a defined target using iWay Explorer, you can edit any information that you provided during the creation process. If you change any properties without editing, the changes will not be saved. You must edit the target in order for the changes to be saved.

Procedure: How to Edit a Defined Target

1. On the iWay Explorer tab, connect to and expand the *Application Explorer* node.
2. Underneath the Application Explorer node, expand the *RDBMS* node.
3. Right-click the connection that you want to edit, for example, *RDBMSTarget*, and click *Edit Target* from the menu.

   The Add Target dialog box opens and displays the connection information that you supplied when you defined the target.
4. Modify the connection information as desired and click *Finish*.

Deleting a Defined Target

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

Procedure: How to Delete a Defined Target

1. On the iWay Explorer tab, connect to and expand the *Application Explorer* node.
2. Underneath the Application Explorer node, expand the *RDBMS* node.
3. Right-click the target that you want to delete, for example, *RDBMSTarget*, and click *Delete Target* from the menu.

   A message is displayed, prompting you to confirm the deletion of the target.
4. Click *OK*.

   The target node disappears from the list of available connections.

Viewing Metadata

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

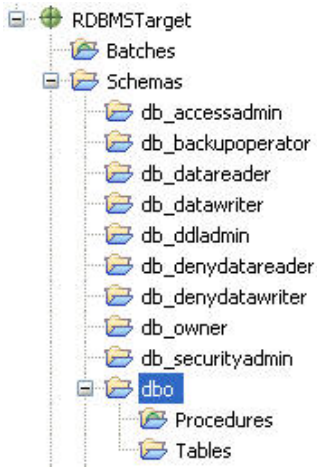
- Table metadata, as described in *How to View Table Metadata* on page 35.
- Stored procedure metadata for a relational or a non-relational database, as described in *How to View Stored Procedure Metadata* on page 37.
- Table metadata while creating an SQL statement, as described in *Creating an SQL Statement and Generating Schemas* on page 54.

Procedure: How to View Table Metadata

To view table metadata:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Underneath the desired connection, click the *Schemas* node to select it, and then expand the *Schemas* node.
3. Click a database to select it, and then expand the database. For example, click and expand the *dbo* database.

In the following image, the dbo database is selected and expanded.



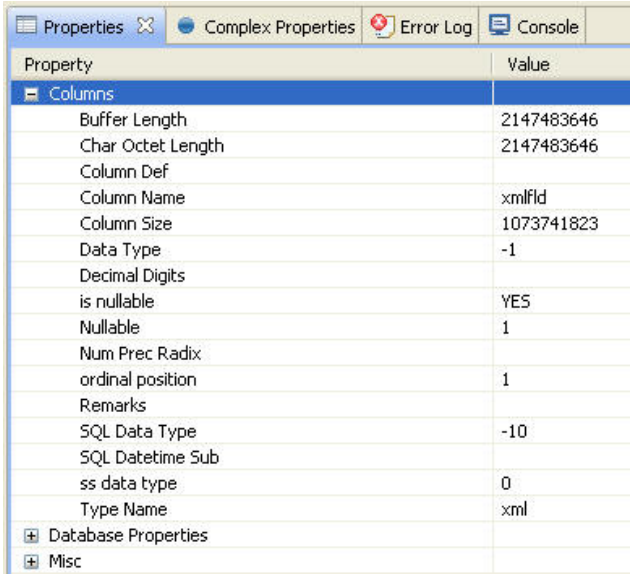
4. Select and expand the *Tables* node.
5. Scroll down and select the table whose metadata you want to view, for example, *xmltable*.

   Although the list of tables includes all tables in the RDBMS, the user ID that you specified for the connection may not have access to the table that you select. If that is the case, the creation of schemas will fail.
6. If it is not already selected, click the *Properties* tab in the right pane.

   Tip: If the Properties tab is not displayed, you can display it using the Window menu. From the Window menu, select *Show View* and click *Other*. Underneath General, click *Properties* and then *OK*.

   When the Properties tab is selected, summary information for the selected table is displayed in the right pane, with categories for Columns, Database Properties, and Miscellaneous (Misc). Each category contains Property and Value fields.

You can expand the category that you are interested in. In the following image, the Columns category is expanded for the selected table by default. It displays information on the column named xmlfld in the xmltable in the dbo database.



When you are ready to create a schema, use the information provided to determine the table or tables and columns to use.

Procedure: How to View Stored Procedure Metadata

To view stored procedure metadata:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Underneath the desired connection, click the *Schemas* node to select it, and then expand the *Schemas* node.
3. Click a database to select it, and then expand the database. For example, click and expand the *dbo* database.
4. Select and expand the *Procedures* node.
5. Select the procedure whose metadata you want to view, for example, *InProc*.

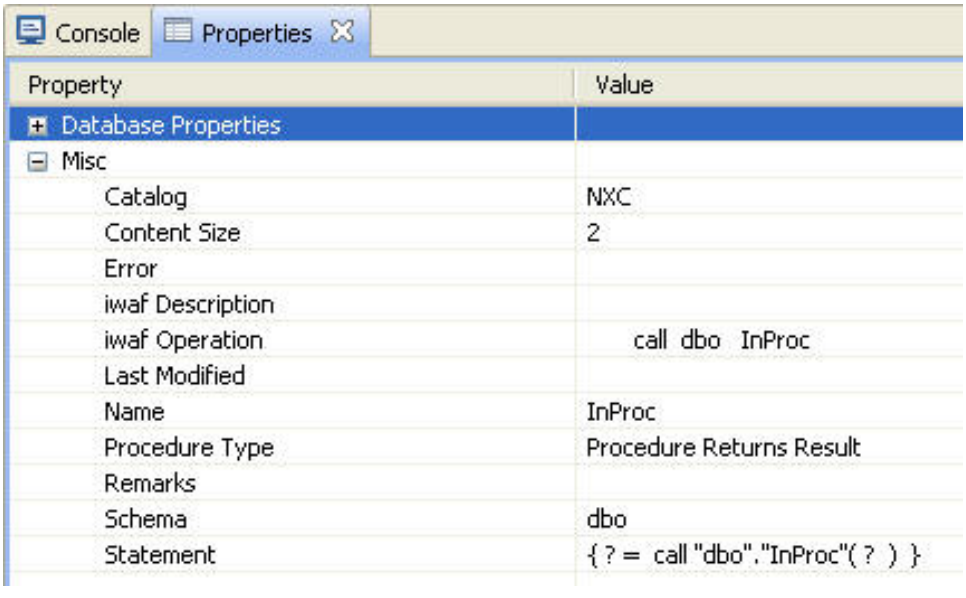
   Although the list of procedures includes all procedures in the database, the user ID that you specified for the connection may not have access to the procedure that you select. If that is the case, the creation of schemas will fail.

6. If it is not already selected, click the *Properties* tab in the right pane.

   Tip: If the Properties tab is not displayed, you can display it using the Window menu. From the Window menu, select *Show View* and click *Other*. Underneath General, click *Properties* and then *OK*.

   When the Properties tab is selected, summary information for the selected procedure is displayed in the right pane, with categories for Database Properties and Miscellaneous (Misc). Each category contains Property and Value fields.

   You can expand the category that you are interested in. In the following image, the Misc category is expanded for the selected procedure by default. It displays information on the procedure named InProc in the dbo database.



   When you are ready to create a schema, use the information provided to determine the procedure or procedures and fields to use.

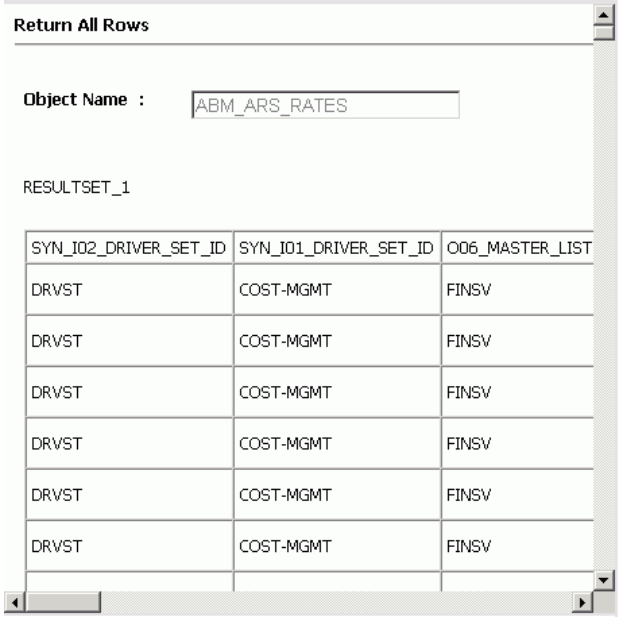
Viewing Sample Data

The iWay Technology Adapter for RDBMS provides an option that assists you in viewing sample data.

The **Return All Rows** option is available from any table node and all records for the given table. This option produces a formatted table of all the rows in the selected table.

To access this option, in the left pane of iWay Explorer right-click the table node of interest and select *Return All Rows* from the drop-down list. The Return All Rows pane opens on the right.

The following image is an example of a Return All Rows pane for a table named ABM_ARS_RATES. Use the scroll bars at the bottom and on the right of the pane to view the entire table.



About the Search Tool

The iWay Technology Adapter for RDBMS search tool allows you to find specific items in an RDBMS target. An RDBMS target is structured into three major nodes:

- Schemas
- Statements
- Batches

These nodes contain subnodes of tables, procedures, Oracle packages, statements, and batches. The search tool is available from all nodes and subnodes in the Schemas, Statements, and Batches hierarchy.

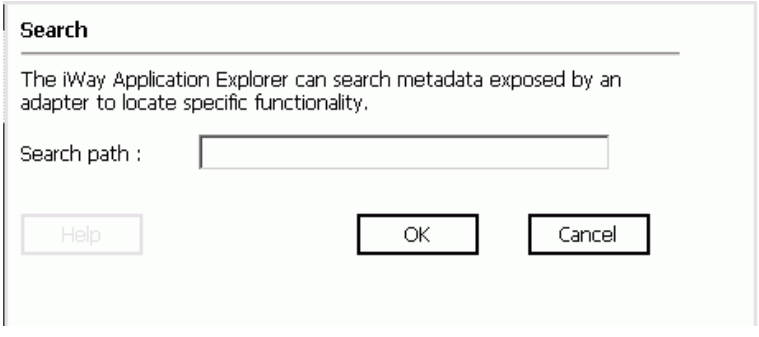
If the database contains more than 64 tables or procedures, you must use the Search Tool to locate that Table or Procedure.

Procedure: How to Use the Search Tool

To open the search tool:

1. Select a node under which you want to find one or more items. You can select either the Schemas, Statements, Batches node, or a sub node such as Tables, depending on which item you are looking for.
2. Move the cursor over *Operations* in the right pane and select *Search* from the drop-down list.

   The Search pane, shown in the following image, opens on the right.



3. In the Search path, type a specific item name or use the following search options.

   - Use %% for a recursive search.
   - Use % for a wildcard, non-case sensitive.

   The following is an example of a search entry:

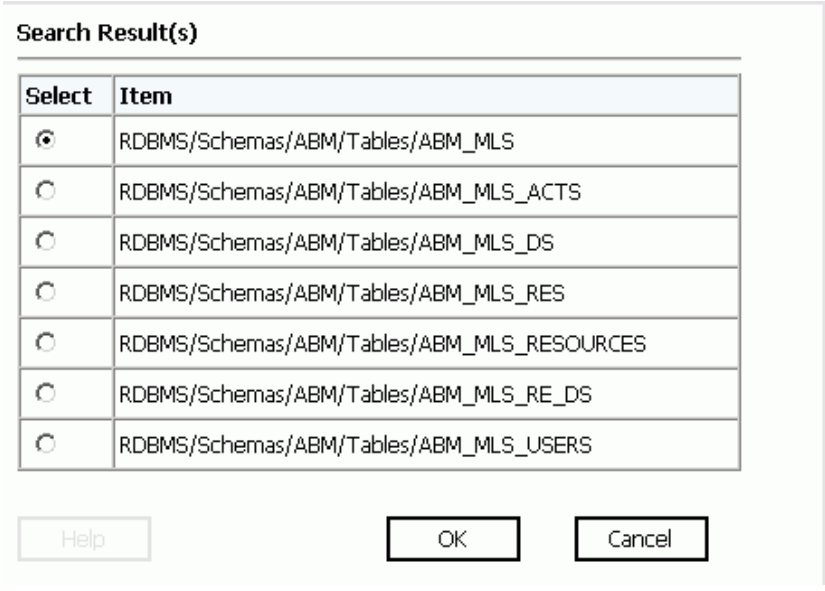
   - %%%eda% indicates to recursively search for all nodes that contain the string "eda".
   - %%eda% indicates to recursively search for all nodes that start with the string "eda".
   - %%%eda indicates to recursively search for all nodes that end with the string "eda".

4. Click *OK*.

   The Search Result(s) pane appears on the right displaying the items that match the search criteria.

   For example, under the Schemas, ABM, Tables node you can search for all tables that have MLS in the name using a search entry of %%%MLS%.

The following image shows the search results, which returns all tables with MLS in the table name.



5. In the Select column, click the item you want to view.
6. Click *OK*.

   The item you selected is highlighted in the left pane.

Understanding and Testing Stored Procedures

A stored procedure is a subroutine that is referenced by a relational database. The iWayTechnology Adapter for RDBMS provides access to stored procedures and their metadata. In Oracle databases, stored procedures and functions are listed under the stored procedure hierarchy. This section provides an overview of stored procedures and also describes how to generate XML request and response schemas for stored procedures.

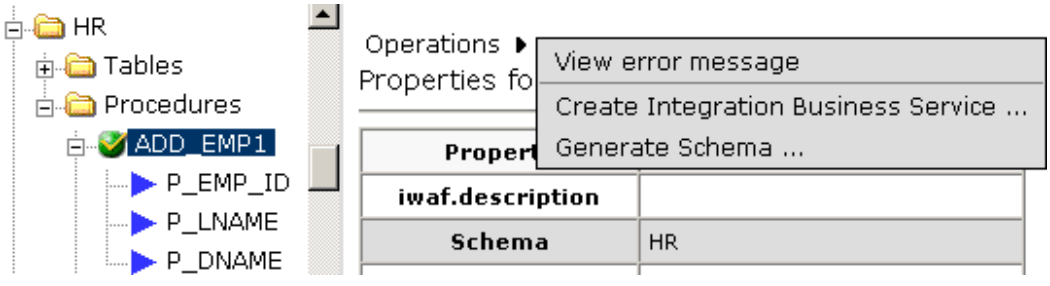
Stored Procedures With Constraints

Certain stored procedures contain constraints, meaning that if no value is supplied for an insert, a default value that meets the constraints is supplied. The adapter does not allow the creation of response schemas unless these constraints are met.

The adapter allows you to input default parameters that meet the criteria of the stored procedure in order to generate the response schema. The values which you input must adhere to the rules of the stored procedure. Once the proper values are entered, you can right-click the stored procedure name again and select the Test option from the context menu.

Procedure: How to View an Error Message and Set Parameter Values

The following image shows a stored procedure selected in the left pane and the Operations menu in the right pane.

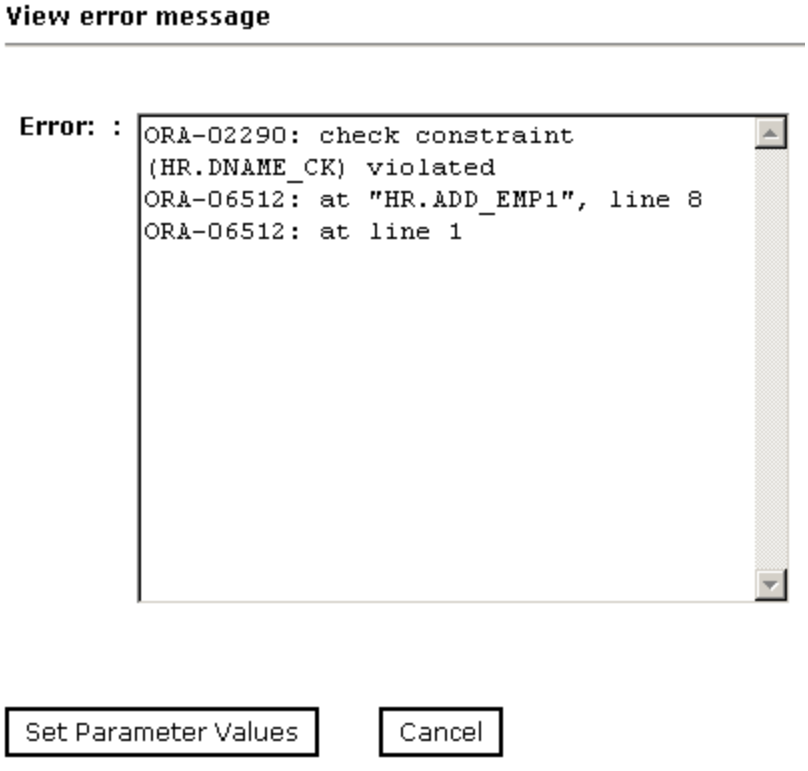


To set the parameter values:

1. Select the stored procedure name.
2. In the right pane, move the pointer over *Operations*, and select *View error message*.

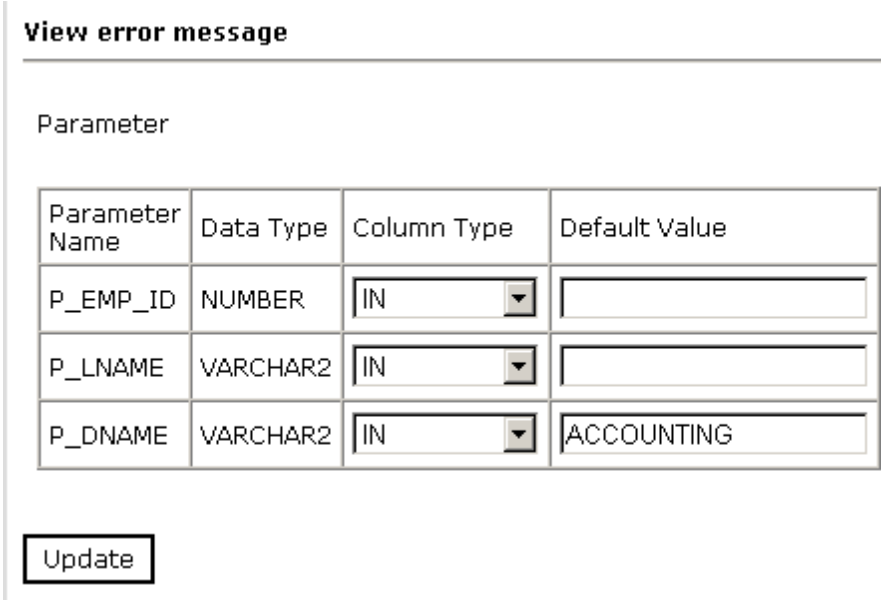
   When you click *View error message*, a constraint error or primary key violation error is generated. This means that you must first supply a value in order for a response schema to be returned.

Note: If the parameters are not set, then you will not be able to generate a web service because the response schema is not available.



3. Click *Set Parameter Values*.

The View error message pane opens on the right and contains columns for Parameter Name, Data Type, Column Type, and Default Value.

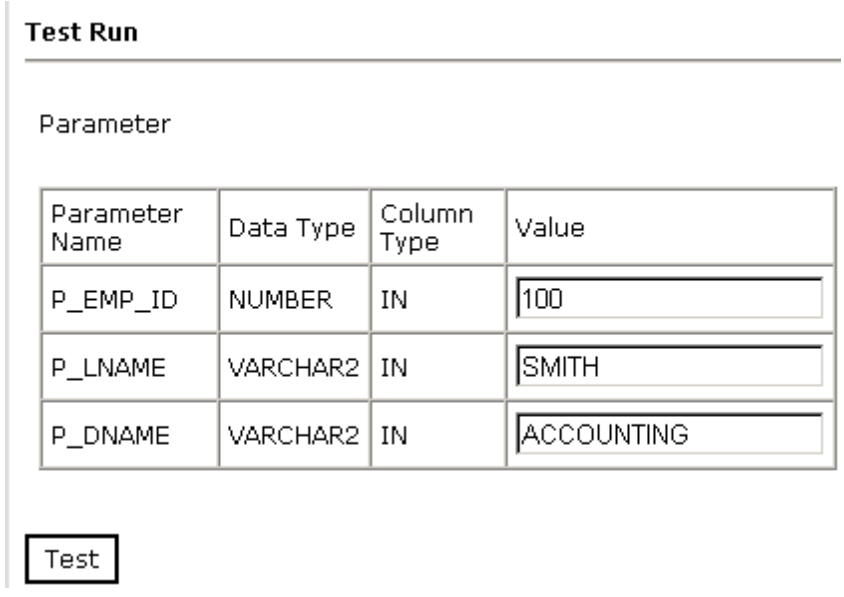


4. In the Default Value fields, type all of your default constraints.

   In this example, the P_DNAME field is the one with the constraint, so it requires a default value, for example, ACCOUNTING.

5. After you enter the value(s), click *Update* to return to the main properties window.
6. In the right pane, move the pointer over Operations and select *Test Run*.

The Test Run pane opens on the right, where you can test the stored procedure. ACCOUNTING is automatically populated as the default value for P_DNAME.



Test Run

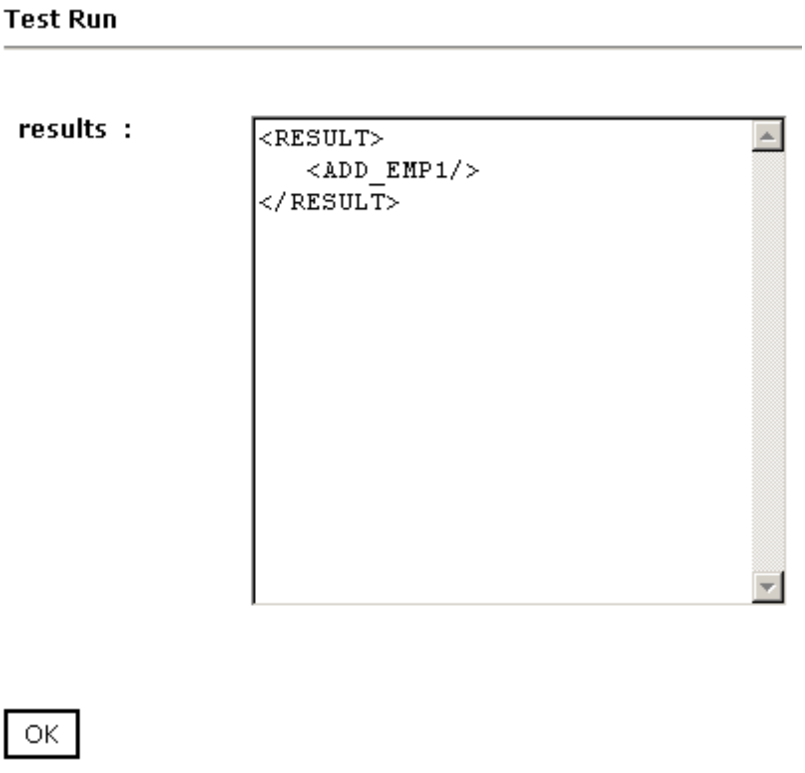
Parameter

Parameter Name	Data Type	Column Type	Value
P_EMP_ID	NUMBER	IN	100
P_LNAME	VARCHAR2	IN	SMITH
P_DNAME	VARCHAR2	IN	ACCOUNTING

Test

7. Enter additional parameter values (for example, 100 and SMITH) and click *Test*.

The results appear in the Test Run results window as shown in the following image.



8. To return to the main properties window where you can generate schemas and create web services for the stored procedure, click *OK*.

Generating a Schema for a Stored Procedure

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

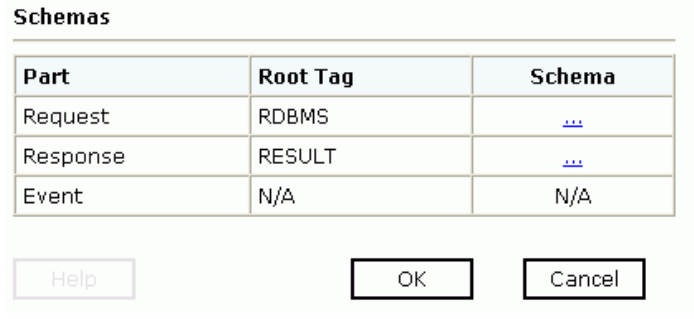
Procedure: How to Generate a Schema for a Stored Procedure

To generate a schema for a stored procedure:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Expand the *Schemas* node under the desired connection.
3. Select the database containing the stored procedure for which you want to generate a schema.

4. Expand the *Procedures* node.
5. Select the stored procedure.
6. In the right pane, move the pointer over *Operations* and select *Generate Schema*.

   The Schemas table appears in the right pane containing three columns named Part, Root Tag, and Schema and three rows named Request, Response, and Event as shown in the following image.



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

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

7. Click *OK*.

Testing a Stored Procedure

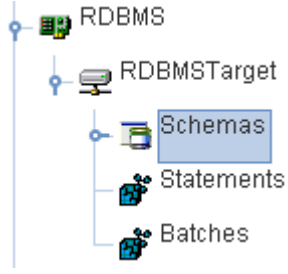
This section describes how to test a stored procedure using iWay Explorer.

Procedure: How to Test a Stored Procedure

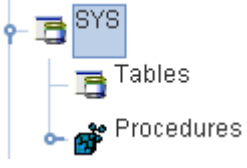
Perform the following steps to test a stored procedure:

1. Connect to an available RDBMS target that is configured for an Oracle connection.

For more information, see *How to Define a New Target* on page 24.

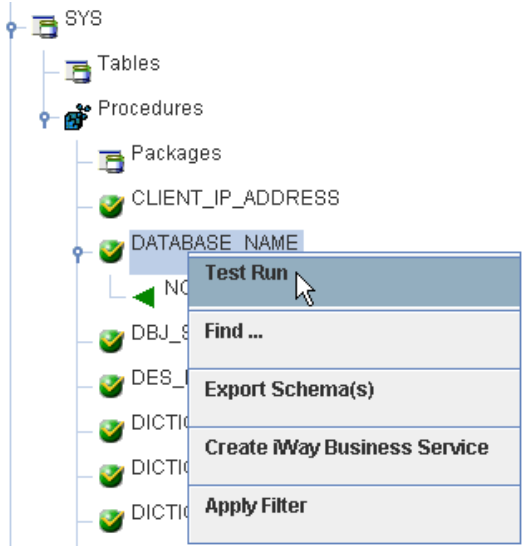


2. In the left pane, expand the *Schemas* node.



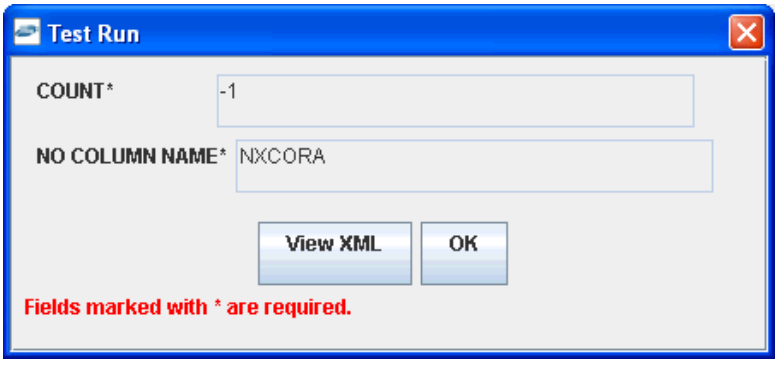
3. Expand an available group node that contains the stored procedure you want to test, for example, SYS.

4. Expand the *Procedures* node.

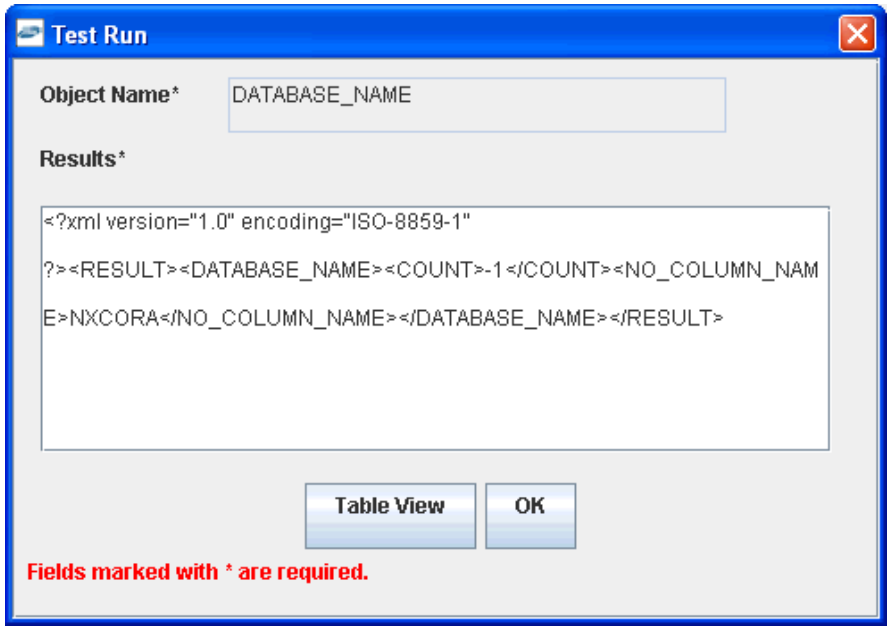


5. Right-click a stored procedure, for example, DATABASE_NAME, and select *Test Run* from the context menu.

   The Test Run dialog box opens.



To view an XML representation, click *View XML*.



To return to the default view, click *Table View*.

6. Click *OK* when you are finished.

Understanding and Testing Stored Procedures and Functions Contained in an Oracle Package

An Oracle database allows a developer to create stored procedures and functions that are contained within packages. An Oracle package is used to store related items in a single unit. The iWay Technology Adapter for RDBMS provides access to stored procedures within a package. Each stored procedure is executed separately and in the same manner that a regular stored procedure is executed.

This section describes how to test a stored procedure in an Oracle package and how to generate a schema for the stored procedure.

Procedure: How to Test a Stored Procedure and Function in an Oracle Package

You can test a stored procedure and function in an Oracle package using the same technique as for a regular stored procedure. Perform the following steps to test a stored procedure and function in an Oracle package:

1. In the left pane, expand the *Schemas*, *Stored Procedures*, and *Packages* nodes.
2. Right-click the stored procedure or function that you want to test.
3. Click *Test Run*.

   Note: Constraints are addressed in the same manner as for a regular stored procedure.

Procedure: How to Generate a Schema for a Stored Procedure Contained in an Oracle Package

To generate a schema for a stored procedure contained in an Oracle package:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Expand the *Schemas* node under the desired connection.
3. Select the database containing the stored procedure for which you want to generate a schema.
4. Expand the *Procedures* node.
5. Expand the *Packages* node.
6. Select the stored procedure.
7. In the right pane, move the pointer over *Operations* and select *Generate Schema*.

Examples

This section provides sample request and response documents.

Example: Stored Procedure Request Schema for an Oracle Database

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

Example: Stored Procedure Request Instance Document for an Oracle Database

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

Example: Stored Procedure Response Schema for an Oracle Database

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

Example: **Stored Procedure Response Instance Document for an Oracle Database**

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

Creating an SQL Statement and Generating Schemas

You can create an SQL statement even when using the adapter for non-relational databases. iWay Explorer provides the convenience to browse and view database tables and its metadata while you create a statement. You can also edit existing SQL statements and parameters. The edit feature also allows you to view tables while you edit.

After you create the statement, you can generate schemas that define request and response documents. The metadata is stored in the iWay Repository, which can be implemented in an RDBMS (such as Oracle or Microsoft SQL Server), a file system, or a specialized XML database.

You can generate the following types of statements:

- Regular SQL Statements, as described in *How to Create a Regular SQL Statement* on page 55.
- Parameterized SQL statements, as described in *How to Create a Parameterized SQL Statement* on page 59.

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 75.
- Stored procedures for relational databases, and iWay stored procedures for non-relational databases, as described in *Generating a Schema for a Stored Procedure* on page 46.
- Oracle stored procedures and functions that are contained in a package.
- Table functions
- Batches (full Edit Batch schemas or AnyBatch schemas)

Creating and Testing a Regular SQL Statement

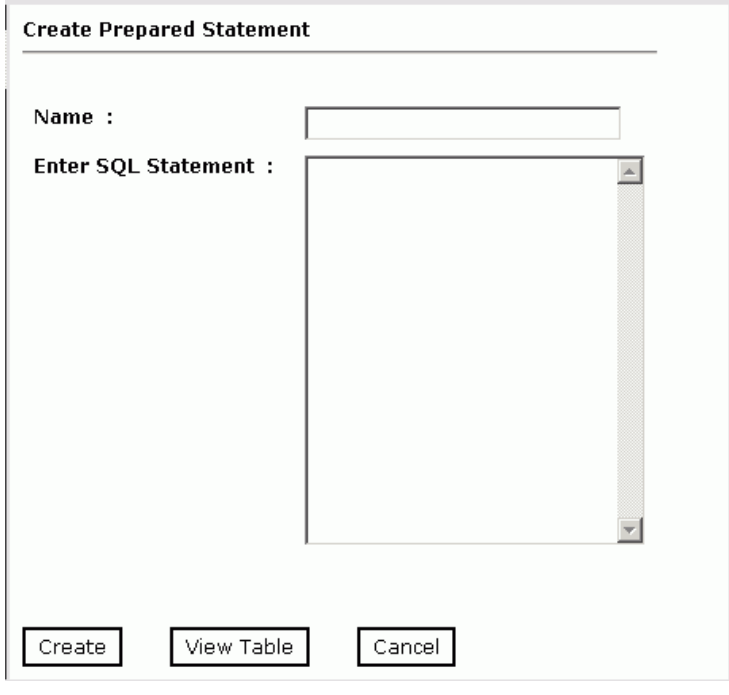
This section explains how to create and test a regular SQL statement. In addition, it describes how to edit an existing SQL statement.

Procedure: How to Create a Regular SQL Statement

To create an SQL statement:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Click the *Statements* node.
3. In the right pane, move the pointer over *Operations* and select *Create Prepared Statement*.

   The Create Prepared Statement options open in the right pane, as shown in the following image.



4. In the Name field, type a name for the statement.

   iWay Software recommends that you specify a name that describes the service. For example, a name of CustomerIntField could represent a request against the Customer Interface table returning a Field format response document.
5. In the Enter SQL Statement field, type the SQL statement for the adapter to use.

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

To view table metadata while you edit:

a. Click *View Table*.

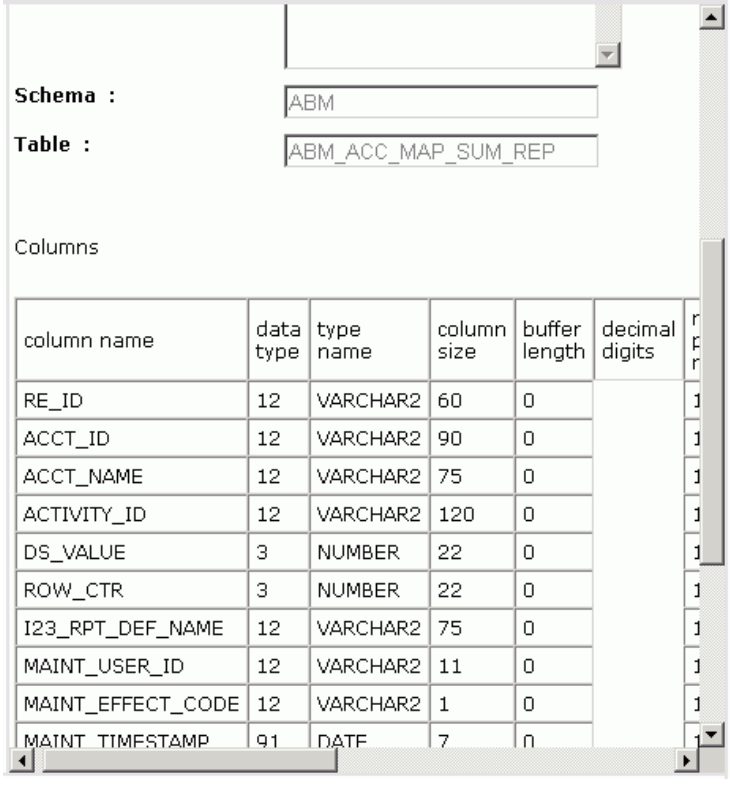
   The *Schema* drop-down list is now available at the bottom of the pane.

b. Select a schema from the drop-down list and click *Get Tables*.

   The *Table* drop-down list is now available under the *Schema* field.

c. Select the table you want to view from the drop-down list and click *Get Table*.

   The table metadata appears at the bottom of the Create Prepared Statement pane. Use the horizontal and vertical scroll bars to view the entire table. An example of a table displayed in the Create Prepared Statement pane is shown in the following image.



   You can browse additional tables if needed using *Get Tables* and *View Another Table* found at the bottom of this pane.

6. When the SQL statement is complete, click *Create*.

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

- For information on testing a regular SQL statement, see *How to Test an SQL Statement* on page 57.
- For information on creating schemas for both parameterized and regular SQL statements, see *How to Generate a Schema for a Prepared Statement* on page 75.

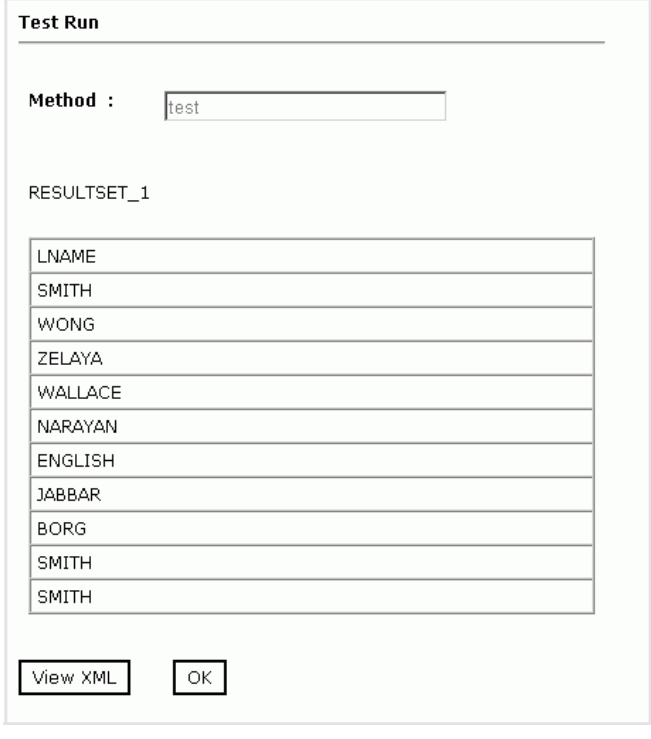
Procedure: How to Test an SQL Statement

To test an SQL statement:

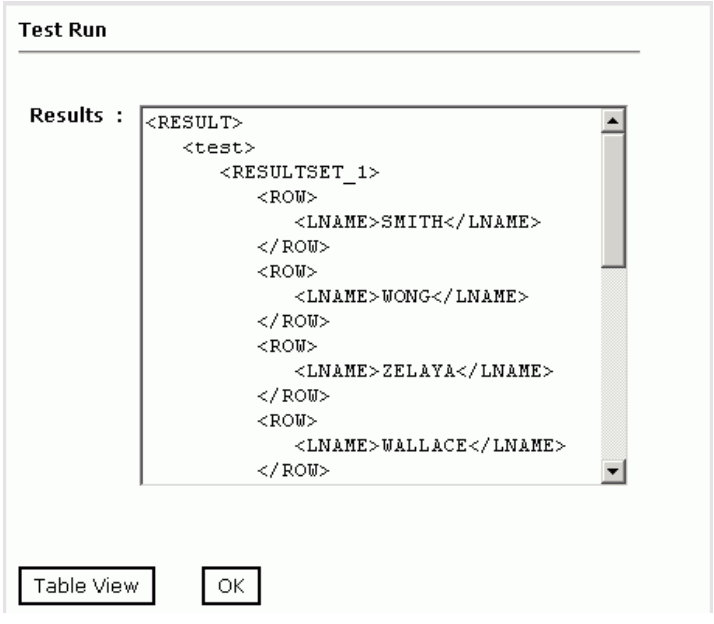
1. Select the SQL statement node you want to test.
2. In the right pane, move the pointer over *Operations* and select *Test Run*.

   The Test Run pane opens on the right.
3. Click *Test*.

   The results appear in the Test Run pane in a table format. The following image is an example of an SQL statement test results.



To see the results in XML, click *View XML*. The following image is an example of test results in XML format.



Click *Table View* to return to the table format display.

4. To exit the results window, click *OK*.

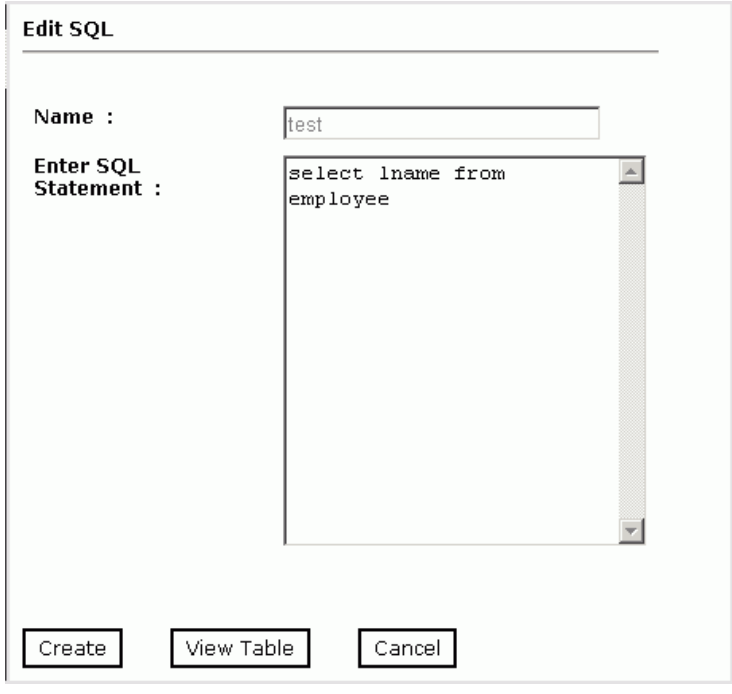
Procedure: How to Edit an SQL Statement

You can follow this procedure to edit the SQL for a parameterized SQL statement as well.

To edit an SQL statement:

1. Expand the Statements node in the left pane.
2. Select the regular SQL statement you want to edit.
3. Move the mouse pointer over *Operations* and select *Edit SQL*.

The Edit SQL pane opens on the right, as shown in the following image.



You can view table metadata by clicking *View Table*. For more details see *How to Create a Regular SQL Statement* on page 55.

4. When your edits are complete, click *Create*.

Creating and Testing a Parameterized SQL Statement

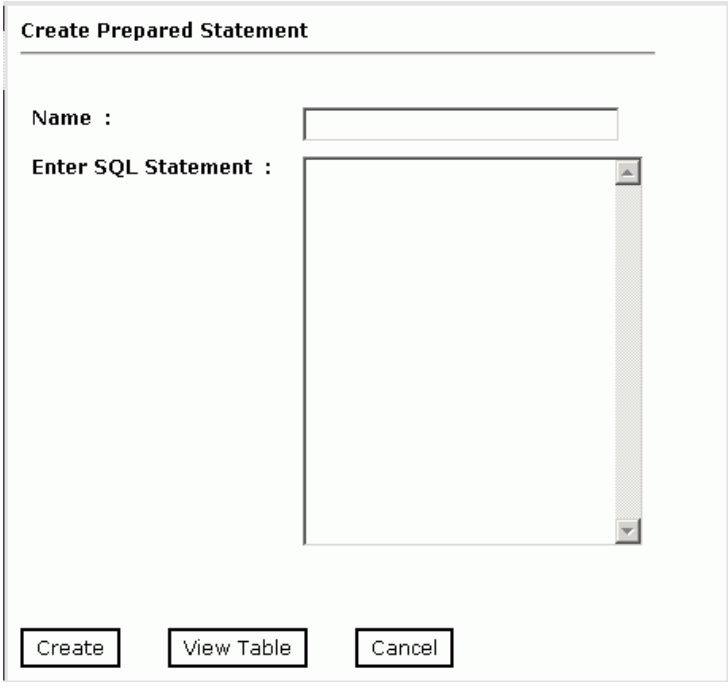
Parameterized SQL allows an SQL statement to be stored within the repository system with parameters imbedded within it. These parameters can be retrieved from XML documents at run time and executed against the SQL statements specified at design time. iWay Explorer creates and maps parameters for the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

Procedure: How to Create a Parameterized SQL Statement

To create a parameterized SQL statement:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Click the *Statements* node.
3. In the right pane, move the pointer over *Operations* and select *Create Prepared Statement*.

The Create Prepared Statement pane opens on the right, as shown in the following image.



4. In the Name field, type a name for the statement.
5. In the Enter SQL Statement field, type the parameterized SQL statement.

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

   To view table metadata while you edit:

   a. Click *View Table*.

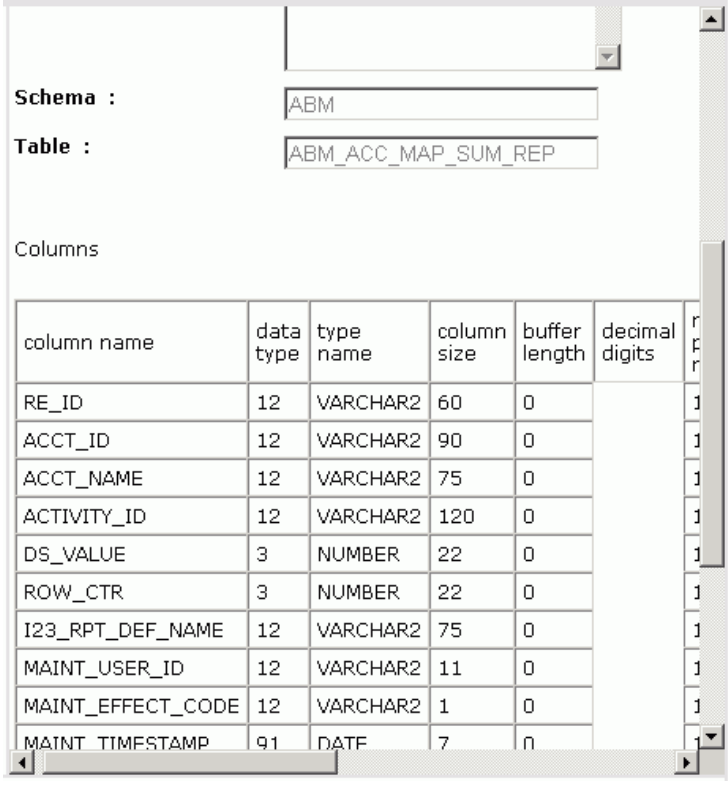
      The *Schema* drop-down list is now available at the bottom of the pane.

   b. Select a schema from the drop-down list and click *Get Tables*.

      The *Table* drop-down list is now available under the *Schema* field.

   c. Select the table you want to view from the drop-down list and click *Get Table*.

The table metadata appears at the bottom of the Create Prepared Statement pane. Use the horizontal and vertical scroll bars to view the entire table. An example of a table displayed in the Create Prepared Statement pane is shown in the following image.



Schema : ABM

Table : ABM_ACC_MAP_SUM_REP

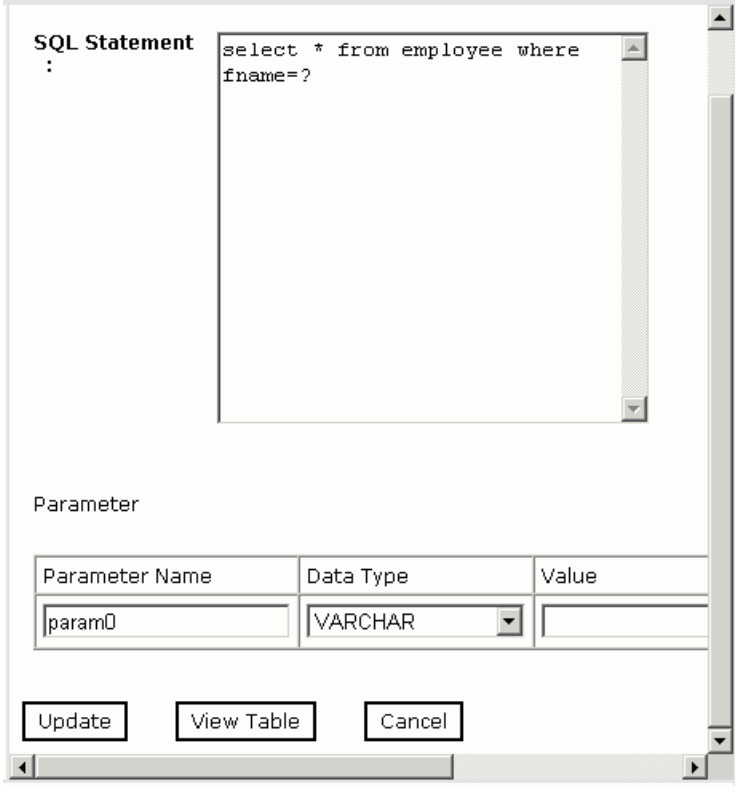
Columns

column name	data type	type name	column size	buffer length	decimal digits	r p r
RE_ID	12	VARCHAR2	60	0		1
ACCT_ID	12	VARCHAR2	90	0		1
ACCT_NAME	12	VARCHAR2	75	0		1
ACTIVITY_ID	12	VARCHAR2	120	0		1
DS_VALUE	3	NUMBER	22	0		1
ROW_CTR	3	NUMBER	22	0		1
I23_RPT_DEF_NAME	12	VARCHAR2	75	0		1
MAINT_USER_ID	12	VARCHAR2	11	0		1
MAINT_EFFECT_CODE	12	VARCHAR2	1	0		1
MAINT_TIMESTAMP	91	DATE	7	0		1

You can browse additional tables if needed using *Get Tables* and *View Another Table* found at the bottom of this pane.

6. When the parameterized statement is complete, click *Create*.

The Parameter Name, Data Type, and Value selection information appears at the bottom of the pane, as shown in the following image.



a. In the Parameter Name column, type a name for each parameter.

b. In the Data Type column, select a data type for each parameter from the drop-down list, which matches the datatype in the database.

c. In the Value field, type a value for the parameter.

7. Click *Update*.

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

Property	Value
iwaf description	
iwaf operation	select from edarpk department where DNAME
Statement	select * from edarpk.department where DNAME=?
Last Modified	2006-03-20 15:50:42.296
Parameters	...
Database Properties	...

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

If your statement is not accurate, an error message appears at the bottom of the pane.

The date on which the statement is created is saved along with other data about the statement. In addition, the date for each parameter is saved. The date information can help debug problems. For example, if a batch statement that references a prepared statement has a modified date earlier than the date listed for the prepared statement, the batch might behave differently than expected. Also, when the design-time and run-time repositories are the same, a deployed service with a date earlier than the modified date shown for the service in design-time might mean that the service behaves differently than intended. In this case, the service should be redeployed.

For information on testing a parameterized SQL statement, see *How to Test a Parameterized SQL Statement* on page 63.

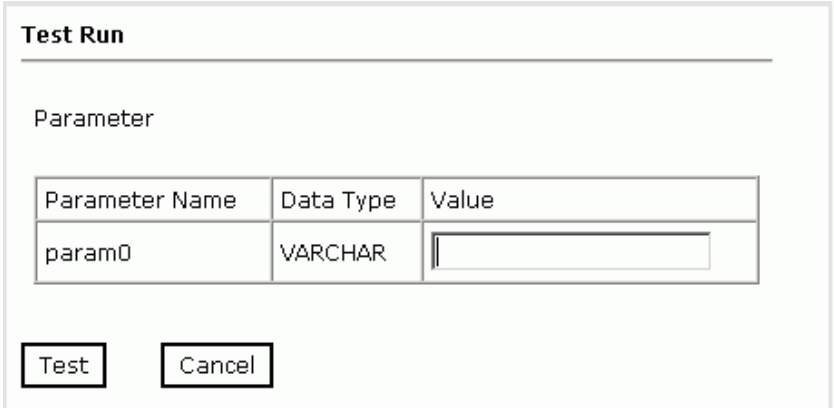
For information on creating schemas for both parameterized and regular SQL statements, see *How to Generate a Schema for a Prepared Statement* on page 75.

Procedure: How to Test a Parameterized SQL Statement

To test a parameterized SQL statement:

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

The Test Run pane opens on the right for the SQL statement. This pane contains the parameter name, data type, and an input box where you can type the parameter value, as shown in the following image.



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

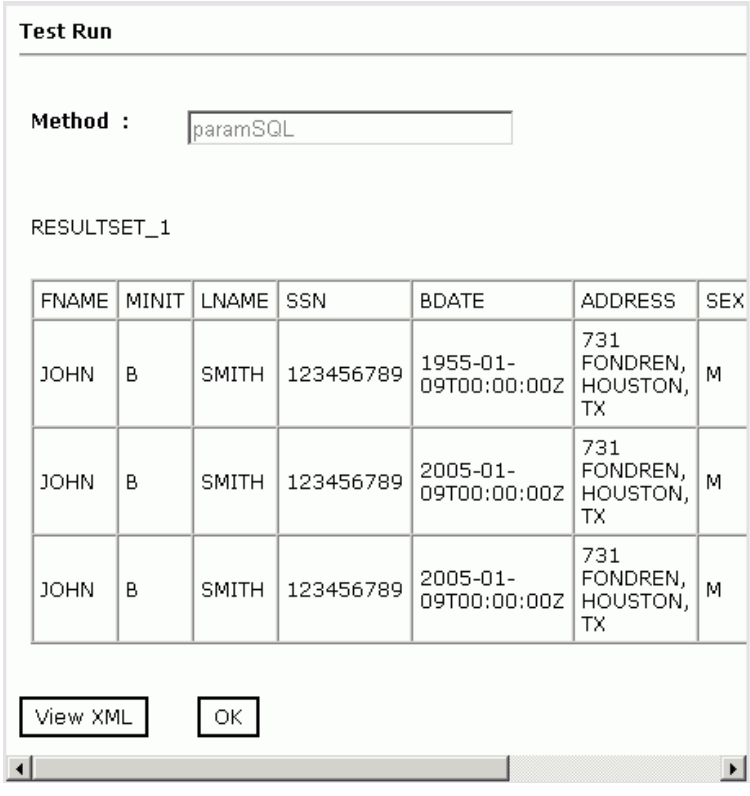
   For example, provide a sample character value, for example, SMITH, for the following SQL statement:

   ```
   select * from employee where lname=?
   ```

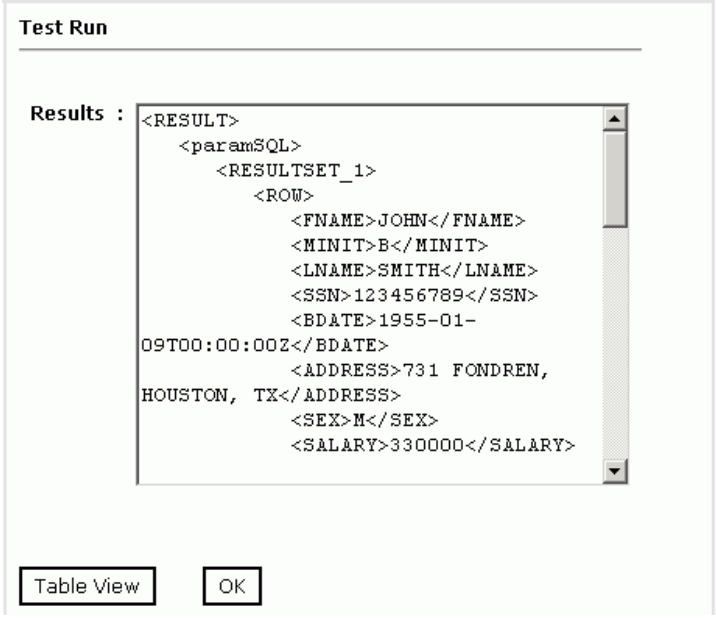
   In this example, the values correspond to values of fields found in a table. Parameterized statements may include parameters that are input for SQL functions, for example, the Oracle SQL function TO_DATE(StringParm). In this case, the data type selected is the expected data type of the SQL function.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

4. Click *Test*.

The results appear in the Test Run results window in a table format. An example of test results is shown in the following image.



To see the results in XML, click *View XML*. The following image is an example of test results in XML format.



Click *Table View* to return to the table format display.

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

Procedure: How to Edit a Parameterized Statement

You can edit the parameter name, data type, and value through the Edit Parameters pane.

To edit a Parameterized SQL statement:

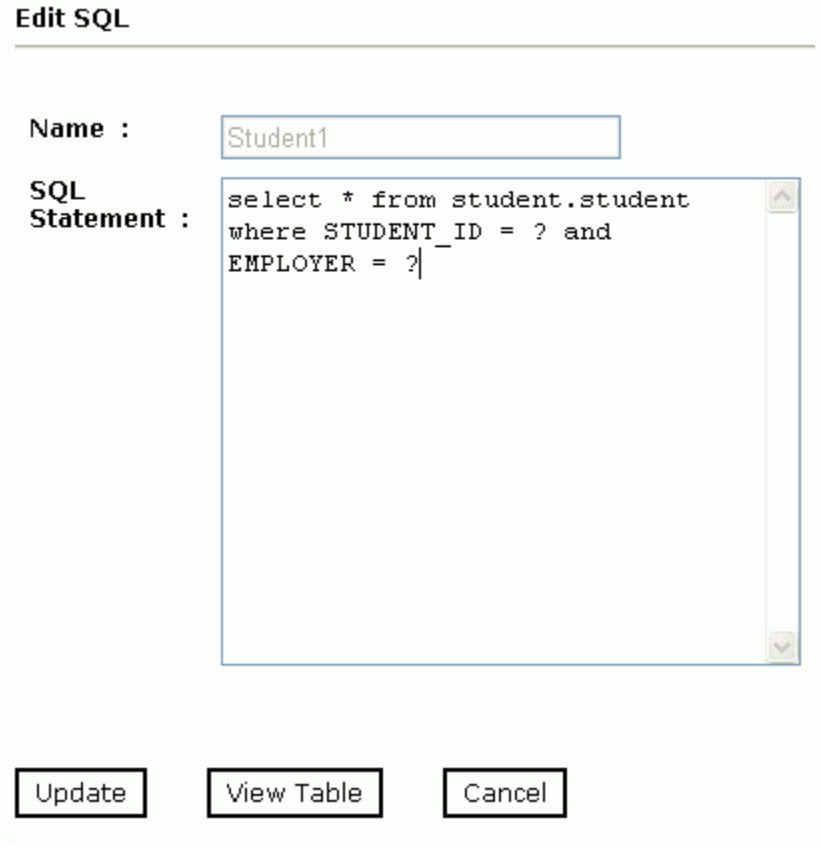
1. Expand the Statements node in the left pane.
2. Select the Parameterized SQL statement you want to edit.
3. In the right pane, move the mouse pointer over *Operations* and select *Edit Parameters*.
4. Type the changes to parameter name and value, and select the data type as necessary.

   To view table metadata while you edit, click *View Table*. For more details see *How to Create a Parameterized SQL Statement* on page 59.
5. When your edits are complete, click *Update*.

   If your statement is not accurate, an error message appears at the bottom of the pane.

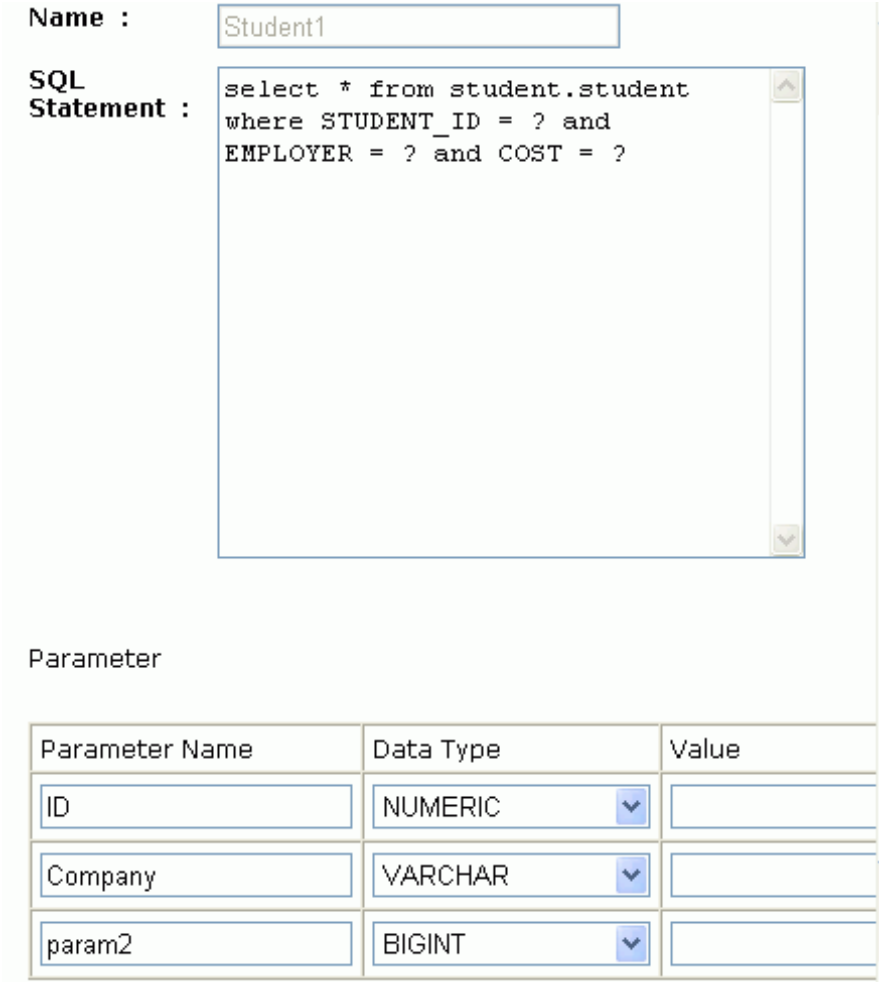
6. If you need to add or delete a parameter, select the parameterized statement, move the mouse pointer over *Operations*, and select *Edit SQL*.

   For example, the following image shows a parameterized statement before it is edited:



7. Edit the SQL to add the new parameter, for example, COST, and click *Update*.

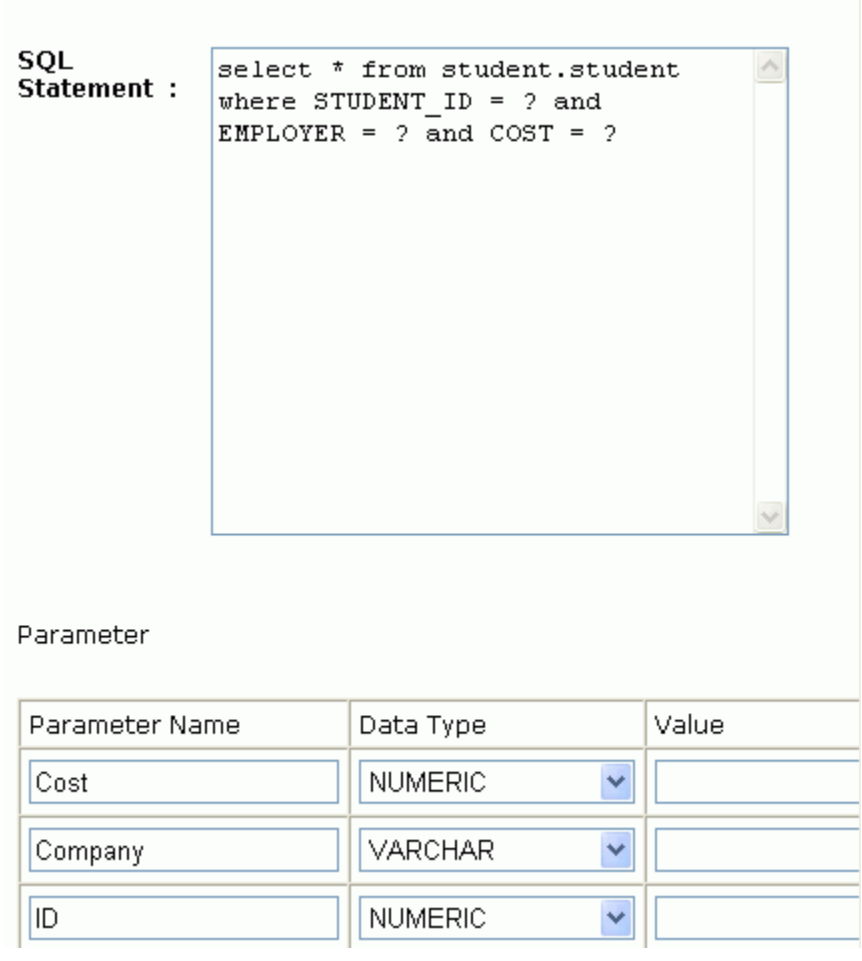
When a new parameter is added to the SQL statement, iWay Explorer adds the new, unnamed parameter automatically to the parameter list. The following image shows the edited SQL statement and the parameters listed below the SQL Statement box:



You can edit the parameter name and data type.

8. Edit the parameter as needed and click *Finish* or click *Update* if you are not done editing the statement.

The following image shows the edited parameter in the parameter table.



You can change the position of the parameters in the parameter table to match the order of parameters in the SQL statement by specifying the position in the Position/Delete column. The following image shows the reordering of the parameter rows in the Parameter table based on the position selected. As shown in the image, the third row will become the top row, followed by the two other rows.

9. Click *Change* to view the changes in the parameter table without committing them, or click *Finish* to commit the changes.
10. To delete a parameter, edit the SQL to remove the parameter and click *Change*.

    The delete setting appears by default in the Position/Delete column of one of the parameter rows. If it is not the parameter you want to delete, you can change the setting for that row.
11. Select *delete* from the Position/Delete column for the parameter you are deleting.

    The following image shows delete selected in the second row, which contains the Company parameter:

You can also change the position of a parameter relative to other parameters in the list by selecting a position number in the parameter row whose position you want to change. This is useful if you delete the middle parameter from a SQL statement and want to reposition the other parameters in the parameter table to match the sequence of parameters in the SQL statement. The following image shows delete selected in the middle row and a position of 1 selected in the third row of the parameter table. This will move the position of the this parameter from the third row to the second row:

12. Click *Change* to perform the modification without committing the changes, or click *Finish* to complete the edits and commit the changes.
13. If you want to clear the edits and return the parameter list to its original form, click *Undo Edit*.
14. If you want to remove all parameters from the list and start fresh, click *Clear All*.
15. If you are not satisfied with the edits, click *Undo Edit* to clear the edits you are making and return the parameter list to its original form or click *Clear All* to return the list to a list of unnamed parameters.
16. Click *Finish* when you are done with all the edits and want to commit the changes.

Using Date and Time Formatting

Because dates can be formatted in many different ways, some applications might have to take extra time transforming a date and time string to the correct XML format. By default, the XML date format is used for a request/input document. The iWay Technology Adapter for RDBMS allows you to specify the date and time format when you design a service so that it does not have to be done in the flow of the message. Editing a date format can only be performed for parameterized SQL statements, which include a DATE or TIMESTAMP datatype. You can configure and test the date formatter the adapter uses at runtime to parse the request value.

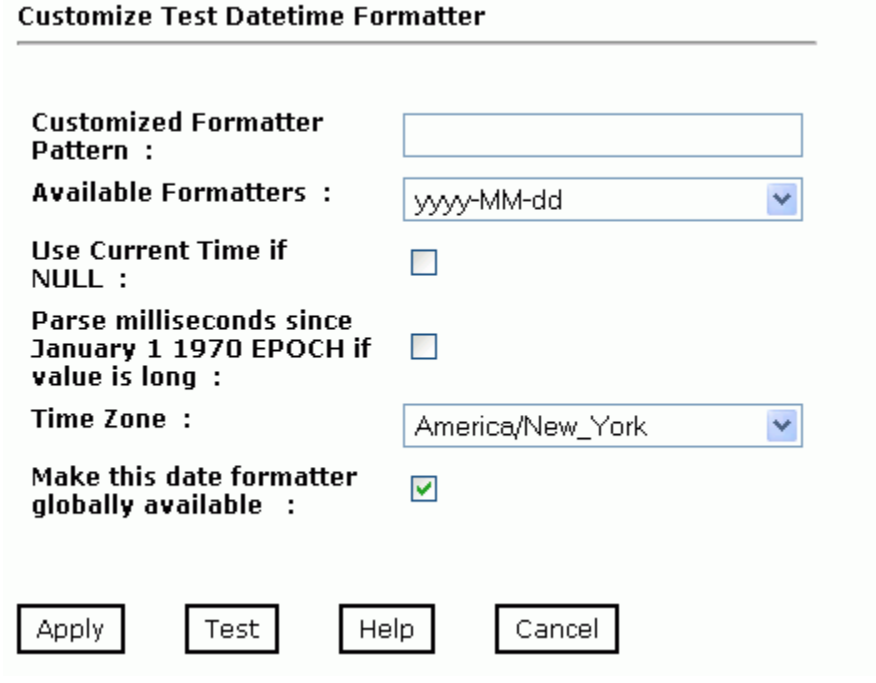
Note: The date format you specify structures the date format for the request document. The date format returned in the response will conform to the date format defined in the database.

Procedure: How to Use Date and Time Formatting Options

To specify the date and time format in a request:

1. Select the date parameter in the parameterized SQL statement for which you want to format the date and time.
2. In the right pane, move the mouse pointer over *Operations* and select *Customize Test Datetime Formatter*.

   The Customize Datetime Formatter pane appears on the right, as shown in the following image:

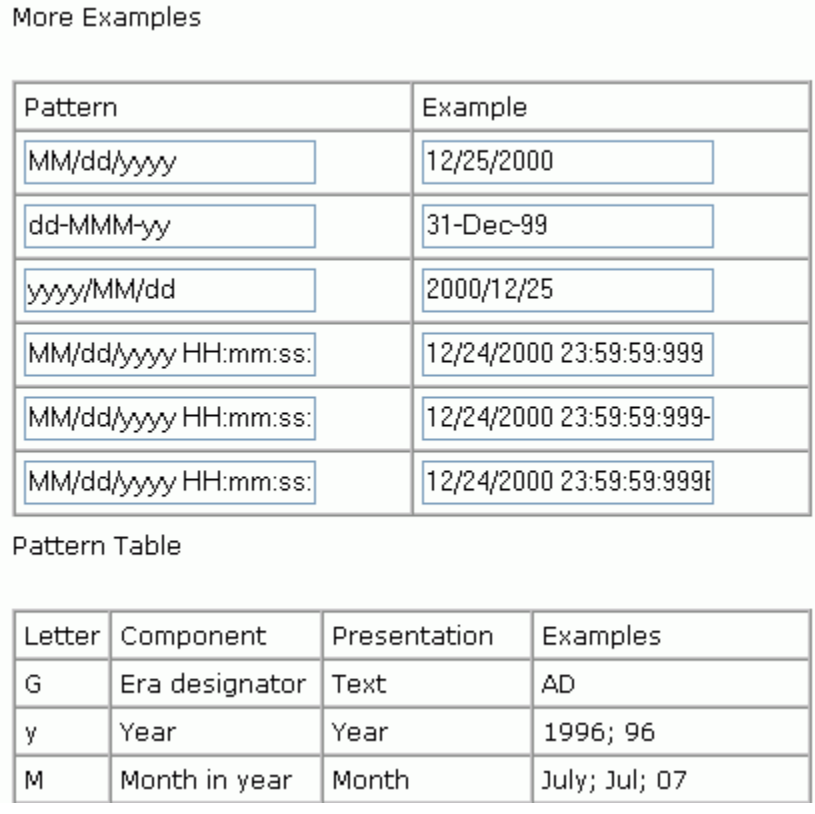


3. Provide the information as follows:

   a. In the Customized Formatter Pattern box, provide the date and time pattern, or select a Formatter from the Available Formatters drop-down box.

   b. Click *Help* if you want to see two tables that assist you in configuring the custom formatter, the Pattern Table and an examples table.

The Examples and Pattern tables appear. The following image shows the Examples Table and the Pattern Table. The Examples Table lists date examples. The Pattern Table specifies the letter, component of the date, the date presentation, and examples.



More Examples

Pattern	Example
MM/dd/yyyy	12/25/2000
dd-MMM-yy	31-Dec-99
yyyy/MM/dd	2000/12/25
MM/dd/yyyy HH:mm:ss:	12/24/2000 23:59:59:999
MM/dd/yyyy HH:mm:ss:	12/24/2000 23:59:59:999-
MM/dd/yyyy HH:mm:ss:	12/24/2000 23:59:59:999[

Pattern Table

Letter	Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07

c. Select *Use Current Time if NULL* to use the current time as the default value if the request value is null.

d. Select *Parse milliseconds since January 1, 1970 (EPOCH)* if you want to have the adapter check if the value returned is a long value representing the number of milliseconds since January 1, 1970 (EPOCH).

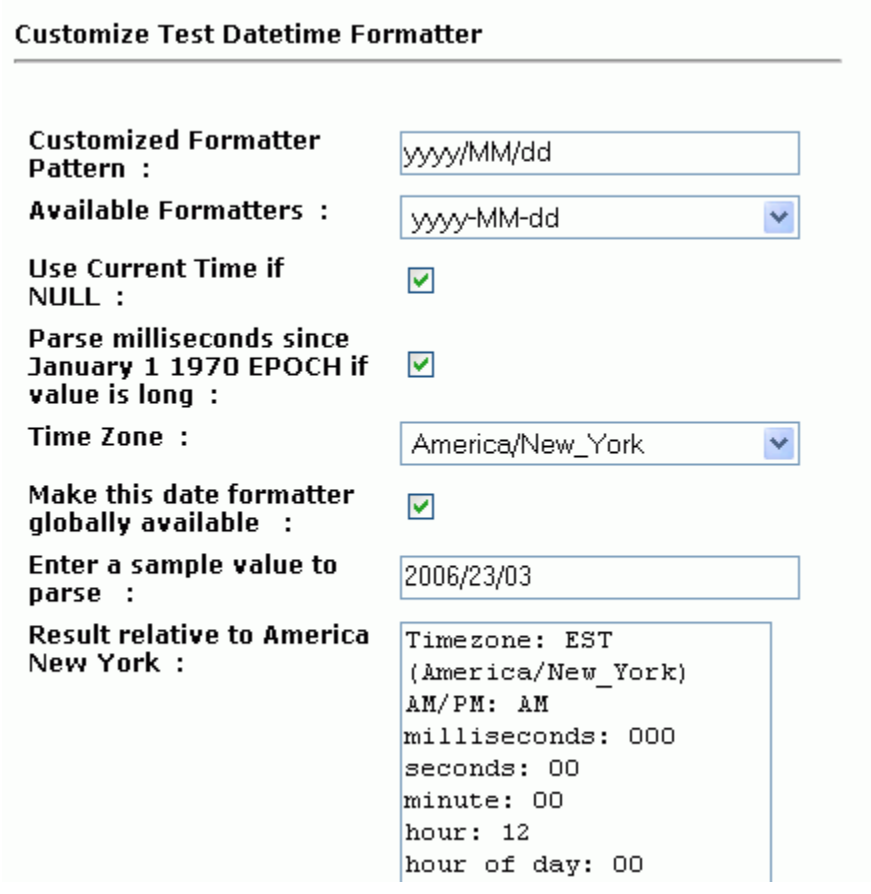
e. Select the time zone from the Time Zone drop-down list.

f. Select *Make this date formatter globally available* if you want this formatter available for use by other services.

g. To test a sample date value, click *Test* and enter a sample value to parse in the *Enter a sample value to parse* text entry box.

h. Click *Test* to test the sample date value.

The following image shows sample test results.



4. Click *Apply* to commit the changes.

Executing an SQL Statement, Stored Procedure, or Table Function Multiple Times

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

- **Connection Resource Utilization.** One connection or thread is established to the back-end database. This minimizes resource consumption. Note that the number of cursors for select statements created is based on the number of parameter sets in the submitted request. If there are 3 sets of parameters for a select statement, there will be 3 cursors created and closed sequentially.

- **Logical Unit of Work (LUW).** A logical unit of work (LUW) is established that will roll back all of the updates or inserts that were issued by the SQL statement of prior executions. A transaction starts at the first set of parameters. The sets of parameters are executed in order from top to bottom of the message sequentially. For example, if an insert statement is submitted along with 3 sets of parameters, and the third set of parameters fails to insert, the previous 2 inserts will be rolled back.
- **Integration of Data From an Outside Source.** For integration scenarios where external data is used to "feed" SQL or stored procedures, this feature allows external data to be mapped to one XML execution block for the adapter to execute.

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

Generating a Schema for a Prepared Statement

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

- *How to Create a Regular SQL Statement* on page 55.
- *How to Create a Parameterized SQL Statement* on page 59.
- *How to Create a Batch Statement* on page 76.

Procedure: How to Generate a Schema for a Prepared Statement

To generate a schema for a prepared statement:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Click the *Statements* node.
3. Click the node containing the prepared statement for which you want to generate a schema.
4. In the right pane, move the pointer over *Operations* and select *Generate Schema*.

   A table that lists the available schemas appears.
5. To view a schema, click the ellipsis symbol in the Schema column.

The schema is generated and ready to use. You can use the generated request schema to create a sample XML document to be used by the adapter. To add a schema to a business service, see *Understanding iWay Business Services* on page 101.

Understanding and Creating Batch Statements

Batch statements enable you to execute multiple SQL and/or parameterized SQL statements within one transaction. This section provides an overview of batch statements and describes how to create them using iWay Explorer.

Creating a Batch Statement and an Iterative Process Using EDIT Batch

Iterative processes are batch type processes that reference other processes, including stored procedures, statements, and table functions. In an iterative batch statement, each statement, procedure, or function can be called multiple times. This differs from a regular batch statement, in which a statement, procedure, or function can be called in a single batch. Instead, an iterative batch statement allows you to iterate the batch itself.

All iterative processes reside within the Batches node. All prepared statements created in the Statements node are automatically imported into the Iterate container. However, you can import processes from Tables or stored procedures as well.

When editing a batch statement and iterative process, full schemas are produced. These schemas include the statement name, location, parameters, and datatypes. This is useful if the batch schema is used in a transformation.

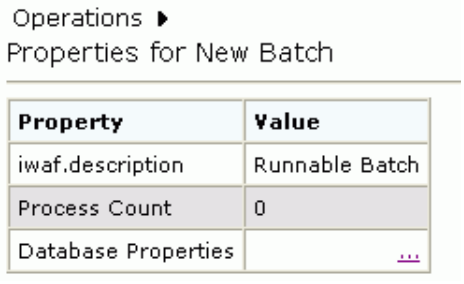
Procedure: How to Create a Batch Statement

To create a batch statement:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. In the left pane, select the *Batches* node.
3. In the right pane, move the pointer over *Operations* and select *Create A Batch*.

   The Create A Batch pane opens on the right.
4. Type a name for the new Batch and click *Create*.

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



5. Move the pointer over *Operations* and select *Edit Batch*.

   The Node Type drop-down selection appears in the right pane.

6. From the drop-down list, select the statement, table, or procedure and click *Next*.
7. Select *Iterate* if you want to make the procedure, statement or table iterative.
8. To add more statements or procedures, select the batch node in the left pane, and then select the appropriate option from the Operations menu in the right pane.

Procedure: How to Create An Iterative Process

To import tables or processes into an iterative process:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Expand the Batches node.
3. Select *Iterate*.

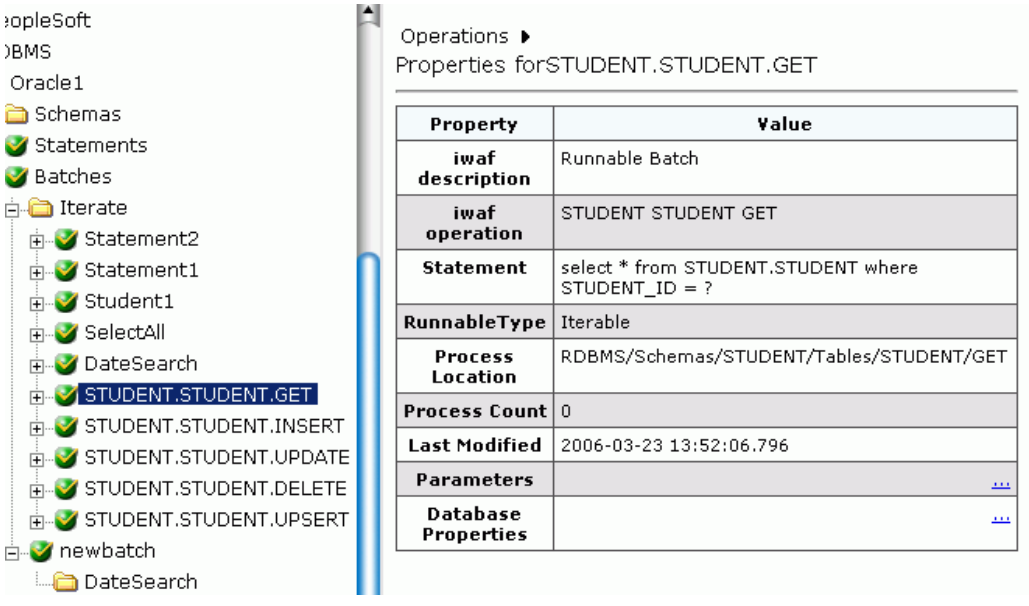
   By default, the Iterate node contains all the statements created in the Statements node.

4. In the right pane, move the mouse pointer over *Operations* and select *Import Process*

   The Import Process pane appears on the right.

5. Select either Tables or Procedures to import and click *Next*.
6. Select schema and click *Next*.
7. Select the tables or procedures you want to import and click *Next*.

If you import tables, the GET, INSERT, UPDATE, DELETE, and UPSERT (but not CURSOR) functions are imported for each table you select. The following image shows the addition of the table functions for the STUDENT table added to the Iterate node:



The stored procedures you import will appear the same way.

Example: Creating an Iterative Process

The following is an example of the input XML for a batch statement named batchtest. Two parameterized SQL statements were added to this statement, one named ParamSTMT1, with an INSERT accessing TableA, and another named ParamSTMT2, with an INSERT accessing TableB. When the SQL statements were added to the batch statement, the *Iterate* check box was selected.

In this input XML document, two different statements are being called, with two different sets of values for each statement.

Note: Added statements or procedures to a batch must be included in the input document.

```
<?xml version="1.0" encoding="UTF-8"?>
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="C:\schemas\ora013_request.xsd">
    <BATCH location="RDBMS/Batches/batchtest">
     <ParamSTMT1 location="RDBMS/Batches/Iterate/ParamSTMT1">
       <PARAMS>
         <lname>Chunnulal</lname>
         <age>30</age>
       </PARAMS>
       <PARAMS>
         <lname>Doe</lname>
         <age>31</age>
       </PARAMS>
     </ParamSTMT1>
     <ParamSTMT2 location="RDBMS/Batches/Iterate/ParamSTMT2">
       <PARAMS>
         <fname>Neena</fname>
         <empid>4</empid>
       </PARAMS>
       <PARAMS>
         <fname>John</fname>
         <empid>5</empid>
       </PARAMS>
     </ParamSTMT2>
    </BATCH>
</RDBMS>
```

The following is an example of the input XML for a regular ITERATE process.

```
<RDBMS>
    <ITERATE location=”RDBMS/Batches/Iterate/ParamSQL1">
          <PARAMS>
               <param0>a</param0>
               <param1>b</param1>
          </PARAMS>
          <PARAMS>
               <param0>c</param0>
               <param1>d</param1>
          </PARAMS>
    </ITERATE>
<RDBMS>
```

Procedure: How to Add Processes to a Batch Process

You can add a new process to a batch statement after you create it. The adapter also allows you to add a process to a batch even if the batch already contains it, allowing you to make limitless calls to the same process.

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Expand the Batches node.

3. Select the batch statement you want to edit.
4. In the right pane, move the mouse pointer over *Operations* and select *Edit Batch*.

   The Edit Batch pane appears on the right.
5. Select *Add Process*.
6. Select the type of process you want to add from the Node Type drop-down box, and click *Next*.

   a. If you select **Tables**, select the schema and then the particular table, and then the table function, clicking *Next* after each selection. After selecting the table function, select *Iterate* to iterate the table function, and then click *Add*.

      The iterate setting determines whether the table function you choose will be iterative. If so, the adapter searches for the process node in the Iterate node. If not, the adapter will search for it within the Schemas node.

      You can only add table functions if the table is listed under the Schemas node. If the table is not listed, you must first find it, and then edit the batch to add the function. This same rule applies for stored procedures.

   b. If you select **Statements**, select the statement you want to add from the Prepared Statement drop-down box, select *Iterate* if you want the prepared statement to be iterative, and click *Add*.

      You have the option of viewing the details of the statement you are adding by clicking *View Details*.

      The iterate setting determines whether the prepared statement you choose will be iterative. If so, the adapter searches for the process node in the Iterate node. If not, the adapter will search for it within the Statements node.

   c. If you select **Procedures**, select the schema from the Schema drop-down list, and click *Get Stored Procedures*. Select the stored procedure you want from the Stored Procedure drop-down box, select *Iterate* if you want the Procedure to be iterative, and click *Add Stored Procedure*.

      You have the option of viewing the details of the procedure you are adding by clicking *Show Details*.

      The iterate setting determines whether the Procedure you choose will be iterative. If so, the adapter searches for the process node in the Iterate node. If not, the adapter will search for it within the Procedures node

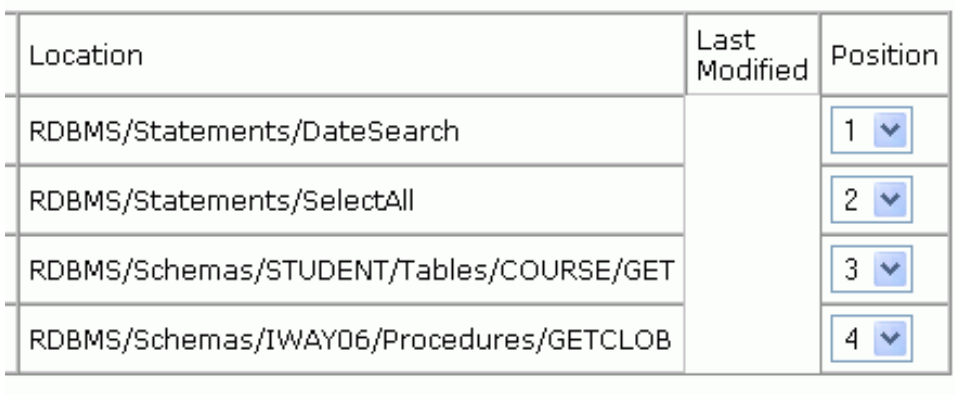
Procedure: How to Reorder Processes in a Batch Process

You can reorder the processes within a batch statement:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Expand the Batches node.
3. Select the batch statement you want to edit.
4. In the right pane, move the mouse pointer over *Operations* and select *Edit Batch*.

   The Edit Batch pane appears on the right.
5. In the right pane, click *Reorder Processes*.

   A table listing the processes appears in the right pane. Each row includes a position column.



Location	Last Modified	Position
RDBMS/Statements/DateSearch		1
RDBMS/Statements/SelectAll		2
RDBMS/Schemas/STUDENT/Tables/COURSE/GET		3
RDBMS/Schemas/IWAY06/Procedures/GETCLOB		4

6. Select the position for each row as appropriate and click *Reorder* to review the new order.
7. Click *Commit* to save the changes.

Procedure: How to Test a Batch Process

To test a batch process:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Expand the Batches node.
3. Select the batch statement you want to edit.
4. In the right pane, move the mouse pointer over *Operations* and select *Test Run*.

The Test Run pane appears on the right. You can edit parameter names and data types for the test.

5. Click *Test*.

   The results appear in the right pane.

LUW for Batch Iterate

There are times when a section of the iterated process fails. For example, if you have inserts into two separate tables, the second process in the second statement can fail. In this case, you may want to roll back the entire Batch. To do so, you will need to set the following Java Property in the iWay Service Manager Administration Console:

```
iwaf.rdbms.iterate.commit=false
```

For more information on setting Java Settings in the console, see the *iWay Service Manager User's Guide*.

After this property is set, you must restart iWay Service Manager for the change to take effect. Once this is done, the iWay Technology Adapter for RDBMS looks for this property and the iterate will not commit if it is set to false. This occurs only if the iterate is called within a batch. If the iterate is called directly, this property is ignored. If you do not want the transaction to be rolled back, do not add this Java setting. By default, Edit Batch does not roll back.

Creating a Batch Statement Using the AnyBatch Process

The AnyBatch process is used to execute a batch if you require flexibility with your batch process. Instead of editing the batch at design-time, you can modify the batch process at run-time, which provides additional flexibility. AnyBatch still enables you to execute multiple transactions in a given request; however, with AnyBatch these transactions can change according to your needs at runtime.

When using AnyBatch, a batch name is the only thing that is required during design-time. If the batch is going to use statements, you must ensure that all statements are created prior to creating the batch name. During run-time, each request is added for each transaction. As a result, if you want to iterate a single Statement, you must add that statement multiple times. This same rule applies for stored procedures and table functions.

Using AnyBatch, an AnyBatch schema is generated, which is a scaled-down schema. However, the transactions that are added to the batch during run-time must adhere to a valid request schema. In addition, with AnyBatch, LUW is built into the process, which means that the entire batch is considered as a single transaction. If an error occurs, the entire transaction will be rolled back.

Procedure: How to Create an AnyBatch Process

To create an AnyBatch process:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Expand the *Batches* node.
3. In the right pane, move the pointer over *Operations* and click *Create A Batch*.
4. Provide a name for the batch.

   Note: You must ensure that all statements, procedures, and table functions that will be used in the batch at runtime are available before the batch is created.

Example: Creating an AnyBatch Process

You can generate an AnyBatch request from its corresponding request schema by using an XML editor, such as XMLSpy. The request will have the following format:

```
<RDBMS xmlns="RDBMS/Batches/Any:55012SP2:Request">
     <BATCH  location="RDBMS/Batches/Any:55012SP2">
          <probably_invalid_for_strict_wildcard/>
     </BATCH>
</RDBMS>
```

Note: In this example, *Any* is the name of the batch.

After you modify your batch request for run-time execution, it has a structure as shown in the following example.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2008 rel. 2
(http://www.altova.com)-->
<RDBMS xmlns="RDBMS/Batches/Any:55012SP2:Request">
       <BATCH  location="RDBMS/Batches/TestAny:55012SP2">
           <AdapterParams location="RDBMS/Statements/x_insert_char">
               <param0>f</param0>
           </AdapterParams>
           <AdapterParams location="RDBMS/Statements/x_insert_char">
               <param0>g</param0>
           </AdapterParams>
           <AdapterParams location="RDBMS/Schemas/SCOTT/Procedures/Packages/
MY_PKG/raisePrice">
           <prod>Soup</prod>
           <times>2</times>
           </AdapterParams>
           <AdapterParams location="RDBMS/Schemas/SCOTT/Tables/CHARS/
INSERT">
           <FLDA>a</FLDA>
           </AdapterParams>
       </BATCH>
</RDBMS>
```

In this example, the *x_insert_char* statement is being iterated since it is called twice in the batch. The *raisePrice* stored procedure and *INSERT* table function for the *CHARS* table is also being called. If you need to iterate the stored procedure or the table Function, simply add everything from *AdapterParams* for that process again within the *BATCH* element. Any modifications to the batch must be made within the *BATCH* element and must conform to a valid schema. The above insertions are all generated from their respective request schemas. The location attribute must also be present.

Request and Response Documents

You can generate request document schemas using iWay Explorer, as described in *Creating an SQL Statement and Generating Schemas* on page 54. 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
- Table Functions
- Batch Statement
- Stored Procedures

Regular SQL Statements

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

Example: **Regular SQL Request Schema**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2005-04-28T22:18:27Z -->
<xs:schema targetNamespace="urn:iwaysoftware:ibse:feb2004:Request"
  xmlns:m1="urn:iwaysoftware:ibse:feb2004:Request"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="AdapterParams">
    <xs:complexType>
      <xs:attribute name="location" type="xs:string" use="optional"
        default="RDBMS/Statements/test1"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example: Regular SQL Request Instance Document

```
<!--Sample XML file generated by XMLSPY v5 rel. 4 U
(http://www.xmlspy.com)-->
<AdapterParams xmlns="urn:iwaysoftware:ibse:feb2004:Request"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:iwaysoftware:ibse:feb2004:Request
  C:\temp\SP3\test1_request.xsd" location="RDBMS/Statements/test1"/>
```

Example: Regular SQL Response Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2005-04-28T22:18:27Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RESULT">
   <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="test1">
        <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="CHAR_" type="xsd:string"/>
                          <xsd:element name="VARCHAR2_5"
type="xsd:string"/>
                           <xsd:element name="DATE_TIME"
type="xsd:dateTime"/>
                        </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+"UT-8?>
<RESULT xmlns:xsi="http://www.w3.org/2005/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\stock price
  select_response.xsd"
  <stock_price_select
    <RESULTSET_1>
      <ROW>
        <CHAR_>xxxxx</CHAR_>
        <VARCHAR2_5>1111</VARCHAR2_5>
        <DATE_TIME>1978-09-22T00:00:00Z</DATE_TIME>
      </ROW>
      <ROW>
        <CHAR_>bob</CHAR_>
        <VARCHAR2_5>hit</VARCHAR2_5>
        <DATE_TIME>2005-04-05T08:34:23Z</DATE_TIME>
      </ROW>
      <ROW>
        <CHAR_>NEW</CHAR_>
        <VARCHAR2_5>NEW</VARCHAR2_5>
        <DATE_TIME>1978-09-22T00:00:00Z</DATE_TIME>
      </ROW>
      <ROW>
        <CHAR_>NEW</CHAR_>
        <VARCHAR2_5>NEW</VARCHAR2_5>
        <DATE_TIME>1978-09-22T00:00:00Z</DATE_TIME>
      </ROW>
    </RESULTSET_1>
  </test1>
</RESULT>
```

Parameterized SQL Statements

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

Example: **Parameterized SQL Request Statement**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2005-04-28T22:35:12Z -->
<xs:schema targetNamespace="urn:iwaysoftware:ibse:feb2004:Request"
  xmlns:m1="urn:iwaysoftware:ibse:feb2004:Request"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="AdapterParams">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="param0" type="xs:string" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="location" type="xs:string" use="optional"
        default="RDBMS/Statements/test4"/>
    </xs:complexType>
  </xs:element>
</xs: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)-->
<AdapterParams xmlns="urn:iwaysoftware:ibse:feb2004:Request"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:iwaysoftware:ibse:feb2004:Request
  \\g4d7001\temp\sp3\test4.xsd" location="RDBMS/Statements/test4">
  <param0>john</param0>
</AdapterParams>
```

Example: **Parameterized SQL Response Schema**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated by the iBSE 2005-04-28T22:35:12Z -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="RESULT">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="test4">
          <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="FNAME" type="xsd:string"/>
                          <xsd:element name="MINIT" type="xsd:string"/>
                          <xsd:element name="LNAME" type="xsd:string"/>
                          <xsd:element name="SSN" type="xsd:double"/>
                          <xsd:element name="BDATE" type="xsd:dateTime"/>
                          <xsd:element name="ADDRESS" type="xsd:string"/>
                          <xsd:element name="SEX" type="xsd:string"/>
                          <xsd:element name="SALARY" type="xsd:double"/>
                          <xsd:element name="SUPERSSN" type="xsd:double"/>
                          <xsd:element name="DNO" 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: **Parameterized SQL Response Instance Document**

```
<RESULT>
  <test4>
    <RESULTSET_1>
      <ROW>
        <FNAME>JOHN</FNAME>
        <MINIT>B</MINIT>
        <LNAME>SMITH</LNAME>
        <SSN>123456789</SSN>
        <BDATE>1955-01-09T00:00:00Z</BDATE>
        <ADDRESS>731 FONDREN, HOUSTON, TX</ADDRESS>
        <SEX>M</SEX>
        <SALARY>330000</SALARY>
        <SUPERSSN>333445555</SUPERSSN>
        <DNO>5</DNO>
      </ROW>
      <ROW>
        <FNAME>JOHN</FNAME>
        <MINIT>B</MINIT>
        <LNAME>SMITH</LNAME>
        <SSN>123456789</SSN>
        <BDATE>2005-01-09T00:00:00Z</BDATE>
        <ADDRESS>731 FONDREN, HOUSTON, TX</ADDRESS>
        <SEX>M</SEX>
        <SALARY>30000</SALARY>
        <SUPERSSN>333445555</SUPERSSN>
        <DNO>5</DNO>
      </ROW>
      <ROW>
        <FNAME>JOHN</FNAME>
        <MINIT>B</MINIT>
        <LNAME>SMITH</LNAME>
        <SSN>123456789</SSN>
        <BDATE>2005-01-09T00:00:00Z</BDATE>
        <ADDRESS>731 FONDREN, HOUSTON, TX</ADDRESS>
        <SEX>M</SEX>
        <SALARY>30000</SALARY>
        <SUPERSSN>333445555</SUPERSSN>
        <DNO>5</DNO>
      </ROW>
    </RESULTSET_1>
  </test4>
</RESULT>
```

Table Functions

The iWay Technology Adapter for RDBMS includes extensive table functions that simplify the ability to update and query a table by creating commonly used prepared statements.

These are basic SQL statements that allow the user to execute basic SQL functionality. These table functions are predefined and cannot be edited. For more complex queries, you can write your own SQL under SQL statements. SQL update functions may not appear in the list of available functions if the table does not have a primary key.

When you create web services from a table function, the adapter builds the SQL request and incorporates it into the web service. You can also export the schema and use it to generate XML instance request documents.

Existing APIs into which these functions can fit are the J2EE design pattern data access object (DAO), Java data objects (JDOs), and J2EE entity beans, all of which abstract and encapsulate access to a data source.

The functions include the following:

- CURSOR
- GET
- INSERT
- UPDATE
- DELETE
- UPSERT
- COUNT
- AVERAGE
- MAX
- MIN
- SUM
- UPSERT

Procedure: How to Use the CURSOR Function

CURSOR is a query function that allows you to scroll through a result set without having an open cursor within the adapter.

1. Select the table node in which you are interested.
2. Select *CURSOR*.

3. Move the mouse pointer over *Operations* and select *Create iWay Business Service*.

   For detailed instructions on creating and testing iWay Business Services, see *Understanding iWay Business Services* on page 101. If you are using Swing iWay Explorer you can also Export the Schema by right-clicking the node in which you are interested and selecting *Export Schemas*. When using the JSP version of iWay Explorer, you can generate schemas by moving the mouse over *Operations* and selecting *Generate Schema*. You can use the schema to create instance XML request documents.

   The web service or schema that is created incorporates the SQL statement for the CURSOR function. This function requires five parameters. Only ROW_COUNT and ROW_REFERENCE require a value. The parameters are listed and defined in the table below.

   Note: If ROW_COUNT is not provided in the input XML document, all records will be retrieved.

Parameter	Description
ROW_COUNT (required)	The number of rows that you want the function to return. If you a supply a value of -1, all rows will be returned.
ORDERBY_COLUMN	The list of columns for the table. It is an enumeration and can have only values in its enumeration list. The column name passed to this parameter sets the “order by” clause in the dynamic statement generated by this function.
ROW_REFERENCE (required)	The row and all of its values from which the returned result set starts or ends. The function dynamically creates a select statement and determines the next set of rows to be sent based on the parameters sent.
ASCENDING	The boolean input parameter that determines if the “order by” is ascending or descending.
NEXT	The boolean input parameter that determines if the result returned is the next or previous set of rows.

Procedure: How to Use Test Run for the CURSOR Function

Using Test Run for the CURSOR function provides an opportunity to use the function. To use Test Run for the CURSOR function:

1. Ensure that the CURSOR function is selected.
2. In the right pane, move the pointer over *Operations* and select *Test Run*.

   The Test Run information appears in the right pane.
3. Specify the information required:
   a. For Row count, enter the number of rows you want to be returned.
   b. From the Column drop-down box, select the column by which you want the result to be sorted.
   c. Select *Ascending* if you want to order rows in ascending order.
4. Click *Get Rows*.

   The result appears in the right pane.
5. Click *Previous* or *Next* to return the previous rows or next rows, respectively.

   The number or rows returned by clicking Previous or Next is the same number specified in the Row count parameter in the Test Run dialog box.

Procedure: How to Use the Additional Table Functions

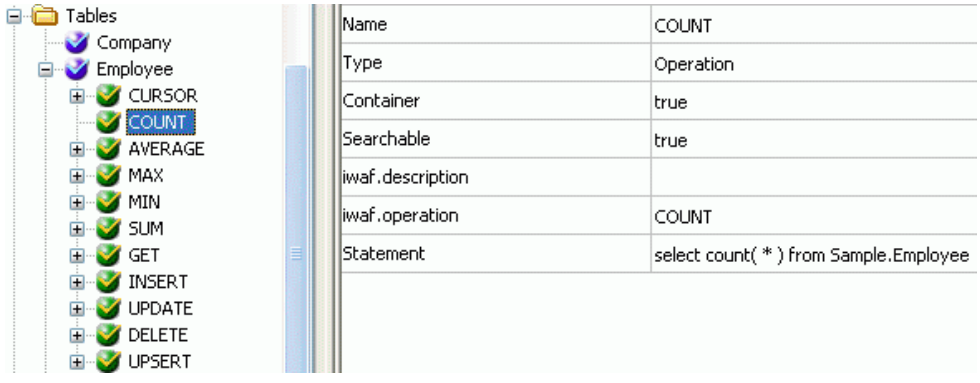
The additional table functions provided by the iWay Technology Adapter for RDBMS provide a standard way to store, update, and retrieve data from any database. The GET, INSERT, UPDATE, DELETE, and UPSERT functions operate on a single row at a time. The COUNT, AVERAGE, MIN, MAX, and SUM functions operate on a single table at a time.

Note: UPDATE functions are not displayed unless the table contains a primary key.

To use the additional table functions:

1. After connecting to a target, expand *Schemas* and then *Tables*, and then select a table.
2. Select the function in the left pane.

All the table functions can be used to create web services and to export schemas to create instance XML request documents. You can review the SQL statement in the right pane when you select one of the functions. The following image shows the Count function selected and the SQL statement displayed in the right pane.



The following table lists and describes the available table functions.

Function	Description
GET	Retrieves one row at a time.
INSERT	Inserts one row into a table.
UPDATE	Updates non-primary keys and operates on a single row at a time. The primary keys sent in the request are used to populate the where clause in the SQL statement. Updating primary keys is logically equivalent to deleting the row by primary key and then inserting a new row with a new ID with the same non-primary key values. This logic fits the DAO, JDO, or EJB frameworks while simply updating the primary keys does not.
DELETE	Deletes one row at a time. The parameters are the primary keys.
UPSERT	Combines the INSERT and UPDATE functions. This function checks if a row already exists. If it does not exist, the request is forwarded to the INSERT function; this is determined by performing a count query with the table's primary keys as part of the where clause, as shown in the properties in the right pane.

Function	Description
COUNT	The COUNT table function is used to return the number of entries in a table. The query does not accept any input parameters.
AVERAGE	This query executes an avg() function on a table and accepts an input parameter representing the column name. The JDBC API is not used to set the question mark in the statement. This process recreates the statement string replacing the question mark with the value in the input parameter in the request. This is performed before the statement is prepared.
MAX	This query executes a max() function on a table and accepts an input parameter representing the column name.
MIN	This query executes a min() function on a table and accepts an input parameter representing the column name.
SUM	This query executes a sum() function on a table and takes in an input parameter representing the column name.

3. In the left pane, select the function you want to use.
4. In the right pane, move the mouse pointer over *Operations* and select either *Create iWay Business Service* or *Generate Schema*.

   You can also use Search to locate data quickly.

Batch Statements

The following is an example of schema for a batch statement.

Example: **AnyBatch Statement Request Schema**

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--  Generated by the iBSE 2009-01-22T17:15:24Z   -->
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="RDBMS/Batches/Any:55012SP2:Request">
- <xsd:element name="RDBMS">
 - <xsd:complexType>
  - <xsd:sequence>
   - <xsd:element name="BATCH">
    - <xsd:complexType>
     - <xsd:sequence>
        <xsd:any namespace="##any" maxOccurs="unbounded" />
       </xsd:sequence>
       <xsd:attribute type="xsd:string" use="required"
         fixed="RDBMS/Batches/Any:55012SP2" name="location" />
      </xsd:complexType>
     </xsd:element>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Example: **AnyBatch Statement Response Schema**

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--  Generated by the iBSE 2009-01-22T17:15:24Z -->
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="RDBMS/Batches/Any:55012SP2:Response">
  - <xsd:element name="RESULTS">
    - <xsd:complexType>
      - <xsd:sequence>
         <xsd:any namespace="##any" maxOccurs="unbounded" />
       </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
```

Example: EditBatch Statement Request Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--  Generated by the iBSE 2009-01-09T20:27:51Z   -->
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="RDBMS/Batches/EditBatch:55012SP2:Request">
  - <xsd:element name="RDBMS">
   - <xsd:complexType>
    - <xsd:sequence>
     - <xsd:element name="BATCH">
      - <xsd:complexType>
       - <xsd:sequence>
        - <xsd:element name="x_insert_char">
         - <xsd:complexType>
          - <xsd:sequence>
              <xsd:element type="xsd:string" name="param0" />
            </xsd:sequence>
           </xsd:complexType>
         </xsd:element>
        </xsd:sequence>
        <xsd:attribute type="xsd:string" use="required"
          fixed="RDBMS/Batches/EditBatch:55012SP2" name="location" />
       </xsd:complexType>
      </xsd:element>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
  </xsd:schema>
```

Example: EditBatch Statement Response Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--  Generated by the iBSE 2009-01-09T20:27:51Z   -->
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="RDBMS/Batches/EditBatch:55012SP2:Response">
  - <xsd:element name="RESULTS">
   - <xsd:complexType>
    - <xsd:sequence>
     - <xsd:element name="x_insert_char">
      - <xsd:complexType>
       - <xsd:sequence>
           <xsd:element type="xsd:integer" name="COUNT" />
         </xsd:sequence>
        </xsd:complexType>
       </xsd:element>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
  </xsd:schema>
```

Example: **EditBatch Statement With Iterate Request Schema**

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--  Generated by the iBSE 2009-01-09T20:29:01Z   -->
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="RDBMS/Batches/EditBatchIter:55012SP2:Request">
  - <xsd:element name="RDBMS">
   - <xsd:complexType>
    - <xsd:sequence>
     - <xsd:element name="BATCH">
      - <xsd:complexType>
       - <xsd:sequence>
        - <xsd:element name="x_insert_char">
         - <xsd:complexType>
          - <xsd:sequence>
           - <xsd:element name="PARAMS" maxOccurs="unbounded">
            - <xsd:complexType>
             - <xsd:sequence>
                <xsd:element type="xsd:string" name="param0" />
              </xsd:sequence>
             </xsd:complexType>
            </xsd:element>
           </xsd:sequence>
          </xsd:complexType>
         </xsd:element>
        </xsd:sequence>
        <xsd:attribute type="xsd:string" use="required"
          fixed="RDBMS/Batches/EditBatchIter:55012SP2" name="location" />
       </xsd:complexType>
      </xsd:element>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
  </xsd:schema>
```

Example: EditBatch Statement With Iterate Response Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--  Generated by the iBSE 2009-01-09T20:29:01Z   -->
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="RDBMS/Batches/EditBatchIter:55012SP2:Response">
  - <xsd:element name="RESULTS">
     - <xsd:complexType>
       - <xsd:sequence>
         - <xsd:element name="x_insert_char" maxOccurs="unbounded">
           - <xsd:complexType>
             - <xsd:sequence>
                 <xsd:element type="xsd:integer" name="COUNT" />
               </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
  </xsd:schema>
```

Example: Batch Statement XML Input Document

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!-- Sample XML file generated by XMLSpy v2005 rel. 3 U
(http://www.altova.com)-->
- <RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="C:\schemas\ORA015_request.xsd">
  - <BATCH location="RDBMS/Batches/ORA015">
     <ORA015A/>
     <ORA015B/>
     <ORA015C/>
   </BATCH>
  </RDBMS>
```

Example: **AnyBatch Statement XML Input Document**

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2008 rel. 2
(http://www.altova.com)-->
<RDBMS xmlns="RDBMS/Batches/Any:55012SP2:Request">
  <BATCH  location="RDBMS/Batches/TestAny:55012SP2">
    <AdapterParams location="RDBMS/Statements/x_insert_char">
      <param0>f</param0>
    </AdapterParams>
    <AdapterParams location="RDBMS/Statements/x_insert_char">
      <param0>g</param0>
    </AdapterParams>
    <AdapterParams location="RDBMS/Schemas/SCOTT/Procedures/Packages/MY_PKG/
raisePrice">
      <prod>Soup</prod>
      <times>2</times>
    </AdapterParams>
    <AdapterParams location="RDBMS/Schemas/SCOTT/Tables/CHARS/INSERT">
      <FLDA>a</FLDA>
    </AdapterParams>
  </BATCH>
</RDBMS>
```

Stored Procedures

The following examples are based on schemas created for stored procedures.

Note: Only positional parameters are supported for Legacy Stored Procedures (FEX).

Example: **Stored Procedure Request Schema**

```
<?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

```
<?xml version="1.0" encoding="UTF-8"?>
<RPCIn xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="D:\iway\RDBMS\RPCVSM_request.xsd"
  name="RPCVSM">
  <1>String</1>
</RPCIn>
```

Example: Stored Procedure Response Schema

```
<?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

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

Understanding iWay Business Services

iWay Explorer provides web developers with a simple, consistent mechanism for extending the capabilities of the adapter. The iWay Business Services Provider (iBSP) exposes functionality as web services. It serves as a gateway to heterogeneous back-end applications and databases.

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

Creating a Business Service

You can create a business service for an SQL statement, stored procedure, table function, or batch. A request and response schema must be available before a business service can be generated.

Procedure: How to Generate a Business Service

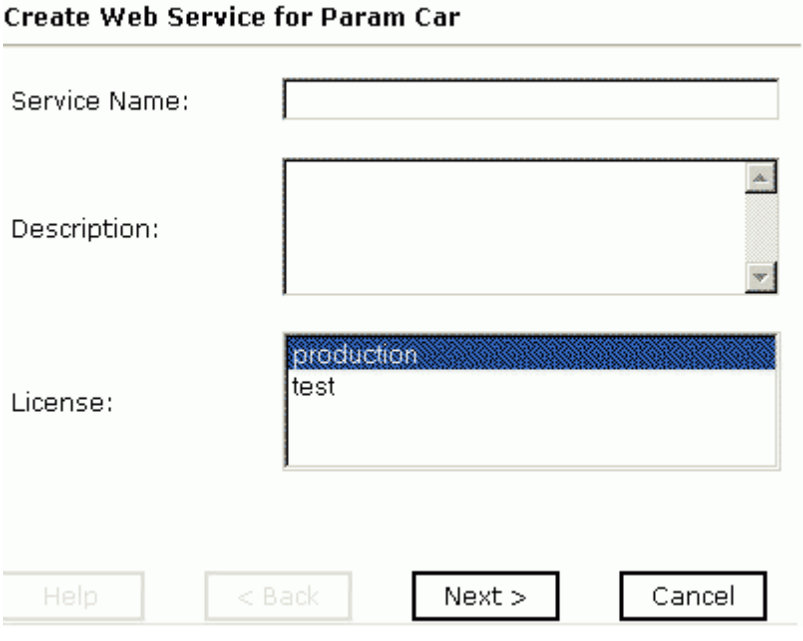
To generate a business service:

1. If you are not connected to a defined target, connect to one, as described in *How to Connect to a Defined Target* on page 33.
2. Expand the node to display the statements or procedures.
3. Click the SQL statement or stored procedure for which you want to create a business service.
4. In the right pane, move the pointer over *Operations* and select *Create iWay Business Services*.

   The Create Web Service information appears in the right pane.
5. Choose whether to create a new service or use an existing service.

   If you select **Use an existing service**, a drop-down list appears from which you must select the service.

   If you select **Create a new service**, the Create Web Service pane opens on the right as shown in the following image.



   a. In the Service Name field, type a name to identify the web service (under the Service node in the left pane of the iWay Business Services tab).

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

c. In the License field, select the license(s) with which you want to associate this business service. To select more than one, hold down the *Ctrl* key and click the licenses.

6. Click *Next*.

   Another pane with the Method Name and Description fields opens.

   a. In the Method Name field, type a name to specify the name of the SQL statement or stored procedure to be added to the business service.

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

7. Click *Finish*.

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

Testing a Business Service

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

Procedure: How to Test a Business Service

To test a business service:

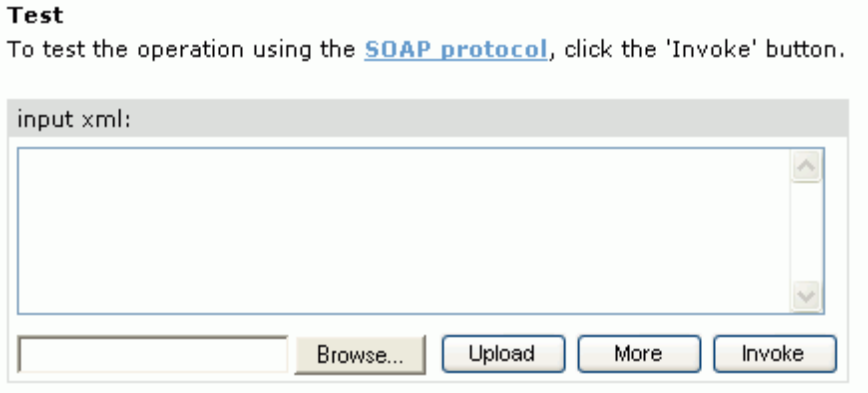
1. If you are not on the iWay Business Services tab of iWay Explorer, click the tab to access business services.
2. If it is not expanded, expand the list of business services under iWay Business Services.
3. Expand the *Services* node.
4. Select the name of the business service you want to test.

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

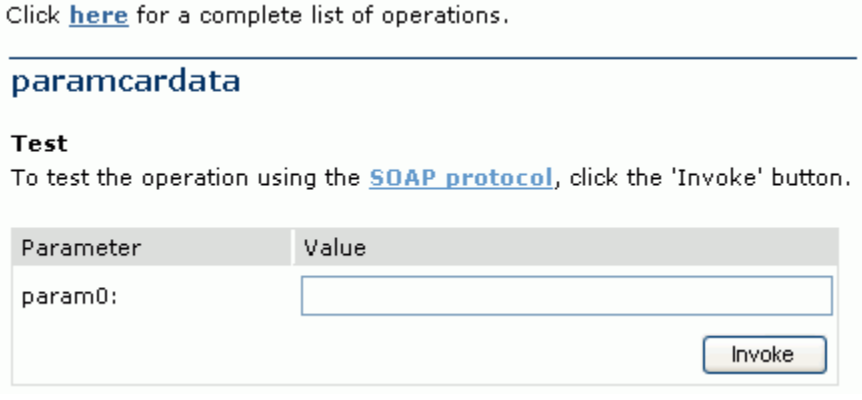
5. In the right pane, click the named business services link.

   The test option appears in the right pane.

If you are testing a web service that requires XML input, an input field appears as shown in the following illustration. Options to browse, upload, view additional information, or invoke the input are available through buttons.



If you are testing a web service for a parameterized SQL statement, an input area appears where you can enter the parameter value, as shown in the following illustration.



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

iWay Explorer displays the results in the right pane as shown in the following illustration.

Generating WSDL From a Web Service

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

Procedure: How to Generate WSDL From a Web Service

To generate WSDL from a web service:

1. Click the *iWay Business Services* tab to access business services.

2. In the left pane, expand the list of services to display the web service for which you want to generate WSDL.
3. Click the web service.

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

   Note: The file extension must be .wsdl.
6. Click *Save*.

Example: Viewing WSDL Generated from a Web Service

The following is an example of a WSDL file for a web service called MPS generated from a parameterized SQL statement against Legacy Data.

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

Identity Propagation

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

The following is a sample SOAP header that is included in the WSDL file for a web service:

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

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

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

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

Chapter 3

Listening for Database Events

This section describes how to use the iWay Technology Adapter for RDBMS, deployed to a 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.

In this chapter:

Understanding iWay Event Functionality

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

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

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

- Port

  A port associates a particular business object exposed by an adapter with a particular disposition. A disposition defines the protocol and resulting location of the event data. The port defines the end point of the event consumption. For more information, see *Creating, Editing, or Deleting an Event Port* on page 110.

- Channel

  A channel represents configured connections to particular instances of back-end or other types of systems. A channel binds one or more event ports to a particular listener managed by an adapter. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 124.

Creating, Editing, or Deleting an Event Port

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

Creating an Event Port From the iWay Events Tab

The following procedures describe how to create an event port from the iWay Events tab for various dispositions. You can switch to an iBSP by using the drop-down menu in the upper right of iWay Explorer.

The following dispositions are available when using iWay Explorer in conjunction with an **iBSP** deployment:

- File
- iBSP
- MSMQ
- JMSQ
- SOAP
- HTTP
- MQSeries

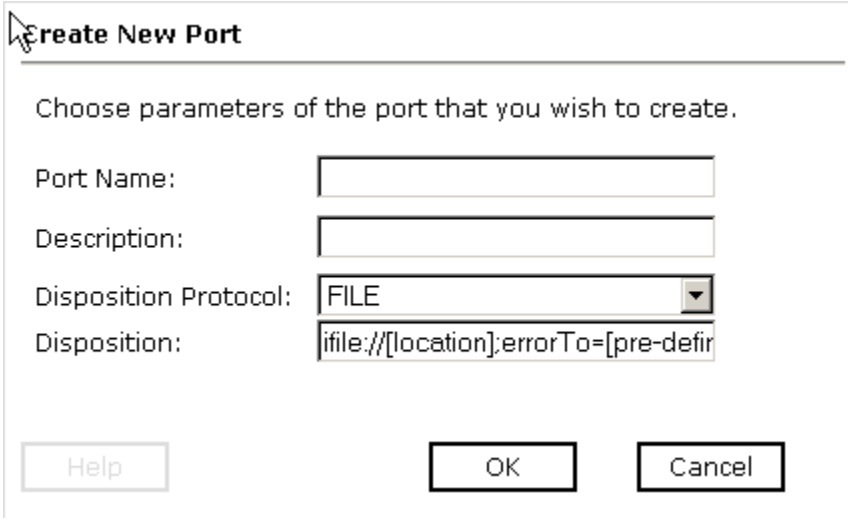
Note: The MAIL disposition option will be supported in a future release.

Procedure: How to Create an Event Port for File

To create an event port for File:

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

The Create New Port pane opens on the right as shown in the following image.



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

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

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

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

When pointing iWay Explorer to an iBSP deployment, specify the destination file using the following format:

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

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

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

5. Click *OK*.

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

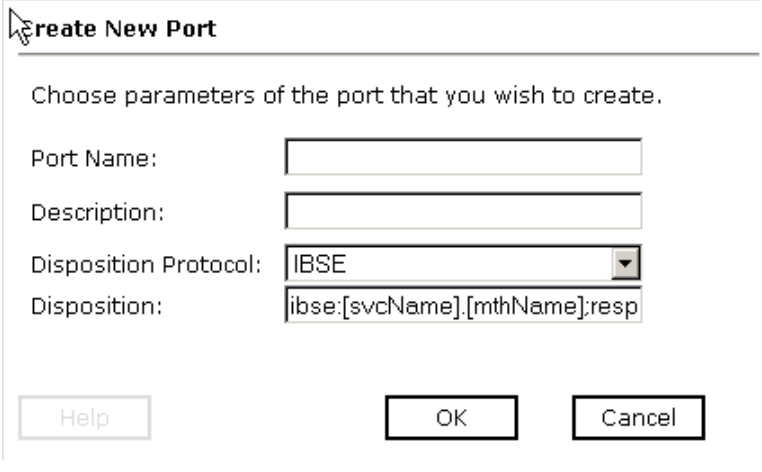
   You are ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 124.

Procedure: How to Create an Event Port for iBSP

To create an event port for iBSP:

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

   The Create New Port pane opens on the right as shown in the following image.



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

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

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

   d. In the Disposition field, type an iBSP destination using the following format:

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

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

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

5. Click *OK*.

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

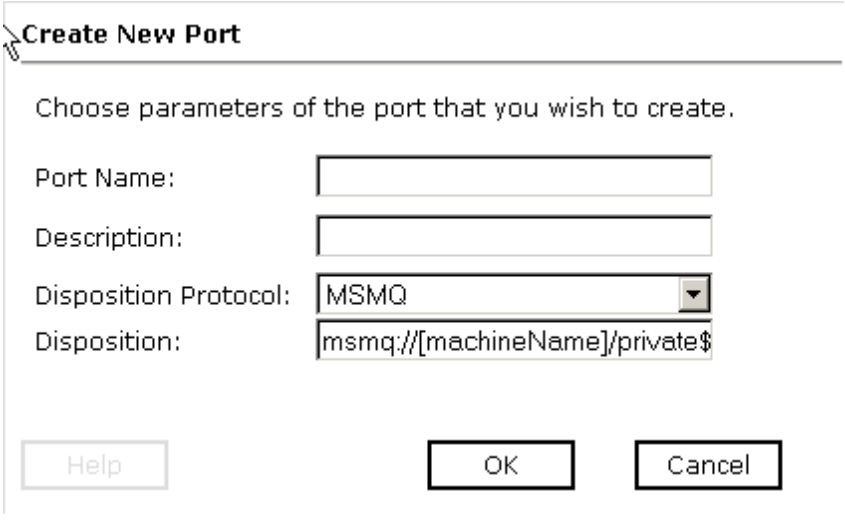
   You are ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 124.

Procedure: How to Create an Event Port for MSMQ

To create an event port for MSMQ:

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

The Create New Port pane opens on the right as shown in the following image.



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

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

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

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

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

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

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

Parameter	Description
machineName	Machine name where the Microsoft Queuing system is running.
qName	Name of the private queue where messages are placed.

Parameter	Description
errorTo	Location to which error logs are sent. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

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

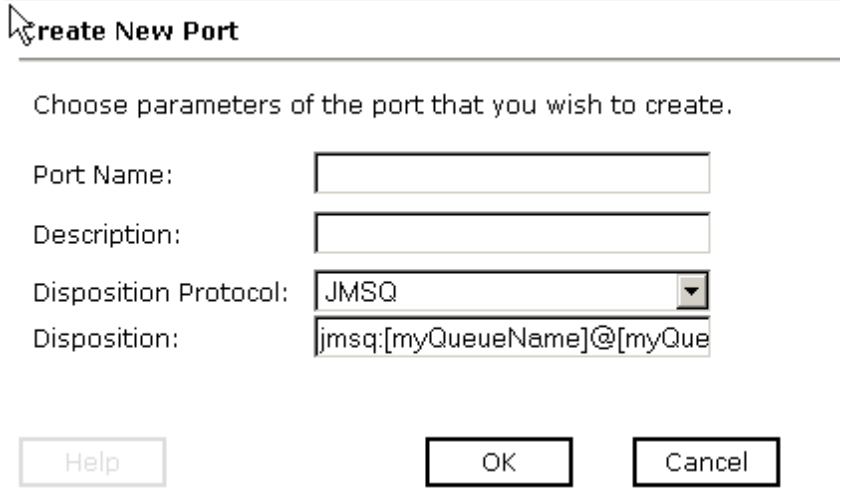
   You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 124.

Procedure: How to Create an Event Port for JMSQ

To create an event port for JMSQ:

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

   The Create New Port pane opens on the right as shown in the following image.



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

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

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

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

When pointing iWay Explorer to an iBSP deployment, specify the destination using the following format:

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

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

Parameter	Description
myQueueName or jmsqueue	JNDI name of a queue to which events are emitted.
myQueueFac or jmsfactory	Resource that contains information about the JMS Server.
jndiurl	URL to use to contact the JNDI provider. The syntax of this URL depends on the JNDI provider being used. This value corresponds to the standard JNDI property, `java.naming.provider.url`
jndifactory	Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider.
user	Valid user name required to access a JMS server.
password	Valid password required to access a JMS server.
errorTo	Location to which error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

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

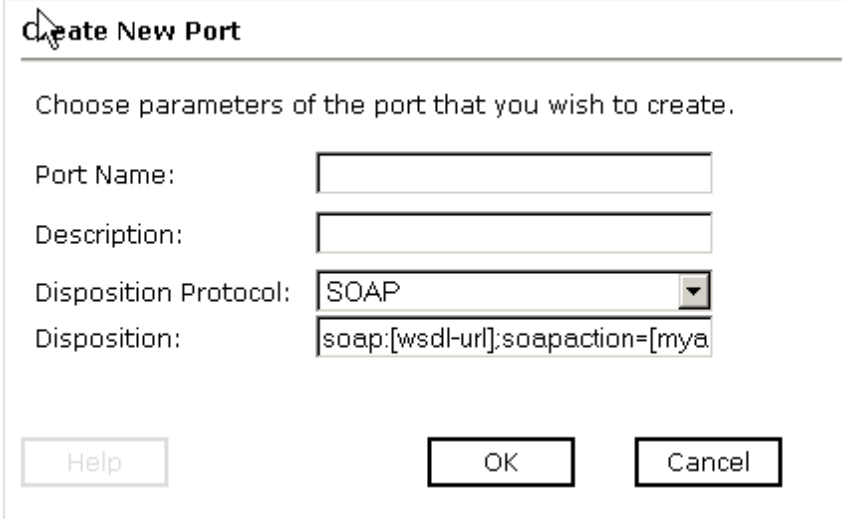
You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 124.

Procedure: How to Create a Port for SOAP

To create a port for SOAP:

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

   The Create New Port pane opens on the right as shown in the following image.



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

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

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

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

Parameter	Description

soapaction	The method that will be called by the SOAP disposition. For example:
	`webservice.method@test@@`
	where:
	`webservice`
	Is the name of the web service you created using iWay Explorer.
	`method`
	Is the method being used.
	`test`
	Is the license that is being used by the web service.
	This value can be found by navigating to the iWay Business Services tab and opening the Service Description link in a new window. Perform a search for *soapAction*.
	You can also open the WSDL file in a third party XML editor (for example, XMLSpy) and view the SOAP request settings to find this value.

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

5. Click *OK*.

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

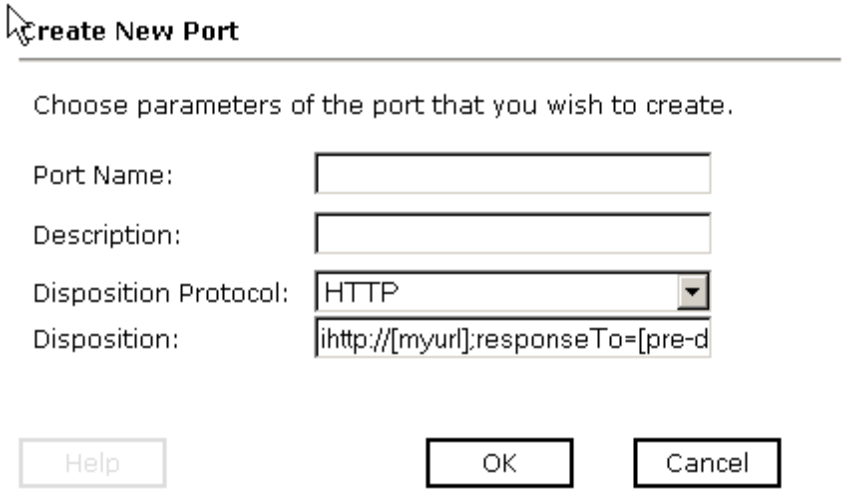
You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 124.

Procedure: How to Create an Event Port for HTTP

To create an event port for HTTP:

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

   The Create New Port pane opens on the right as shown in the following image.



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

      When pointing iWay Explorer to an iBSP deployment, specify the destination using the following format:

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

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

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

5. Click *OK*.

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

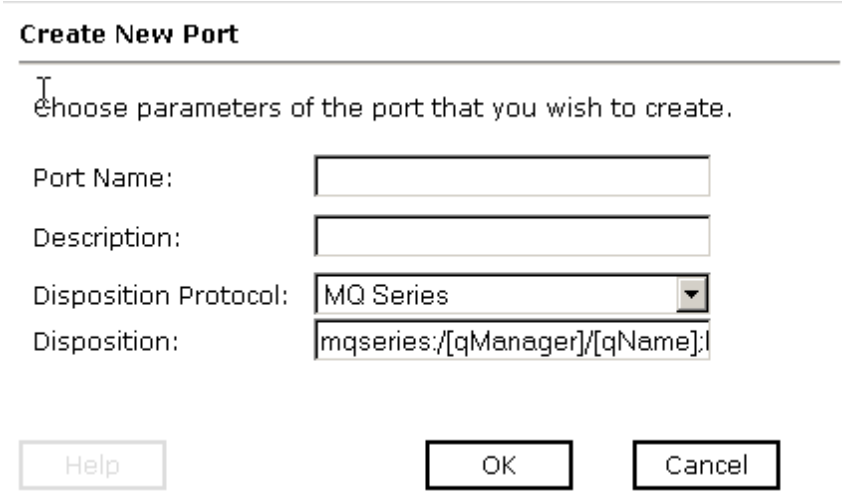
   You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 124.

Procedure: How to Create an Event Port for MQSeries

To create an event port for MQSeries:

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

The Create New Port pane opens on the right as shown in the following image.



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

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

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

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

When pointing iWay Explorer to an iBSP deployment, specify the destination using the following format:

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

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

Parameter	Description
qManager	Name of the queue manager to which the server must connect.
qName or respqueue	Name of the queue where messages are placed.

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

5. Click *OK*.

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

   You are now ready to associate the event port with a channel. For more information, see *Creating, Editing, or Deleting an Event Channel* on page 124.

Editing and Deleting an Event Port

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

Procedure: How to Edit an Event Port

To edit an event port:

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

   The Edit Port dialog box opens.
3. Make the required changes and click *OK*.

Procedure: How to Delete an Event Port

To delete an event port:

1. In the left pane, select the event port you want to delete.
2. In the right pane, move the pointer over *Operations* and select *Delete*.

   A confirmation dialog box opens.

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

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

Creating, Editing, or Deleting an Event Channel

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

Creating a Channel

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

Procedure: How to Create a Channel

To create a channel:

1. Click the *iWay Events* tab.

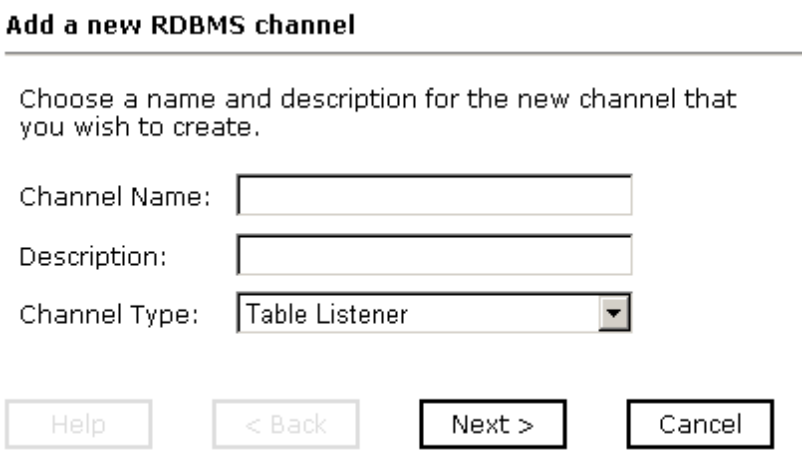
   The adapters that appear in the left pane support events.

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

   The ports and channels nodes appear in the left pane.

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

The Add a new RDBMS channel pane opens on the right as shown in the following image.



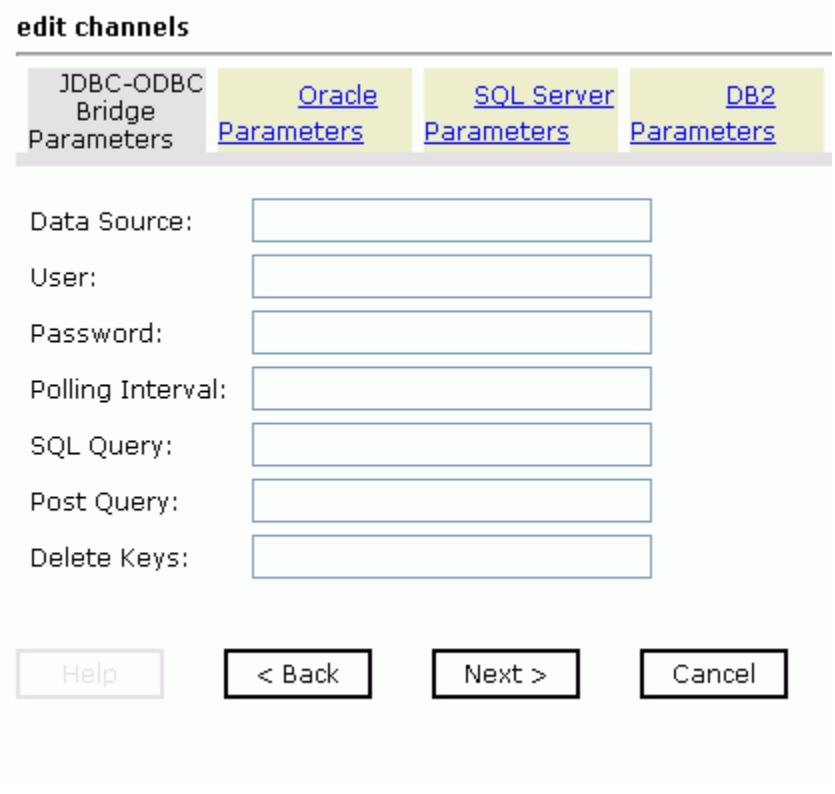
a. In the Channel Name field, type a name, for example, NewChannel.

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

c. From the Channel Type drop-down list, select a channel type.

5. Click *Next*.

The Edit channels pane opens on the right with six tabs; five representing listener parameters. The following image shows four of the tabs.



a. Select either an Oracle, SQL Server, DB2, EDA Server, or JDBC-ODBC Bridge Listener by clicking the appropriate tab.

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

   For information on **common listener parameters**, see *Common Listener Parameters* on page 128.

   For information on listener parameters for **JDBC-ODBC Bridge**, see *JDBC-ODBC Bridge Listener Parameters* on page 131.

   For information on listener parameters for **Oracle**, see *Oracle Listener Parameters* on page 131.

   For information on listener parameters for **SQL Server**, see *SQL Server Listener Parameters* on page 132.

For information on listener parameters for **DB2**, see *DB2 Listener Parameters* on page 132.

For information on listener parameters for **EDA Server**, see *EDA Server Listener Parameters* on page 133.

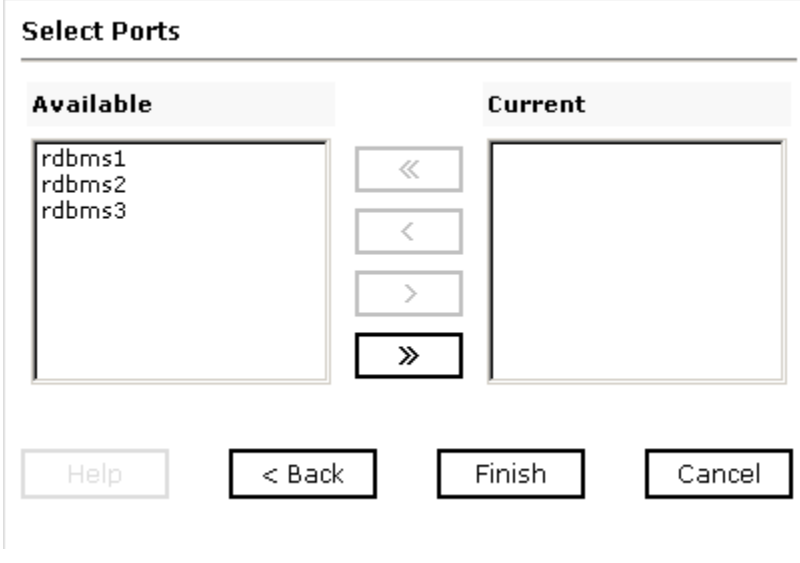
For information on listener parameters for **JDBC**, see *JDBC Listener Parameters* on page 134.

From the **Advanced** tab, you can access the Synctype drop-down list which defaults to REQUEST.

b. Type the system information that is specific to the database on which you are listening based on the descriptions in the previously referenced topics.

6. Click *Next*.

The Select Ports pane opens on the right with buttons to enable you to move ports from one area to the other as shown in the following image.

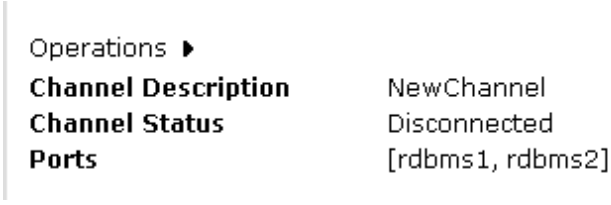


a. Select an event port from the list of available ports. To select more than one, hold down the *Ctrl* key and click the ports.

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

7. Click *Finish*.

Summary information appears as shown in the following image and includes the channel description, channel status, and current ports. All the information is associated with the channel you created in the right pane.



The channel appears under the channels node in the left pane with an X over the icon indicating that the channel is currently disconnected as shown in the following image.

You must start the channel to activate your event configuration.

Reference: Common Listener Parameters

The following table lists and describes the parameters common to all listeners. For parameters specific to your listener, see the reference topics in this section.

Parameter	Description
Polling Interval	Interval, in milliseconds, at which to check for new input.

Parameter	Description

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

Parameter	Description
Post Query	A SQL statement that is executed after each new record is read from the table. Case sensitive: the case used to specify the column names must match the case used in the SELECT statement that polled the table. If the SQL Query property was omitted so that a default SELECT statement polled the table, the case used to specify the column names must match the case used to define the columns in the DBMS native schema. If you do not specify a value for SQL Post-query, each record read from the table is deleted after it is read. How this happens depends on whether you specify the Delete Keys property. If you: **Specify the Delete Keys property,** by default the adapter issues a DELETE statement with a WHERE clause containing every key column specified for the Delete Keys property. At run time this is faster than if you had not specified the Delete Keys property if there is an index on the key or if there are fewer key columns than there are columns in the SELECT statement that polled the table. **Do not specify the Delete Keys property,** by default the adapter issues a DELETE statement with a WHERE clause that specifies every column from the SELECT statement that polled the table. You can choose to retain the table data after it is read by specifying a value for this parameter, as shown in the examples that follow. **Note:** The SQL Post-query and Delete Keys parameters are mutually exclusive, because Delete Keys applies to the default DELETE statement, and SQL Post-query overrides the default DELETE statement. You can provide a value for one or the other, but not for both. There are two field operators, ? and ^, that you can use in a post-query SQL statement. For more information, see *The Post-query Parameter Operators* on page 140. **Important:** When a SQL Query joins two or more tables, a SQL Post Query must be used. Also, do not use a semicolon at the end of a SQL statement for a SQL Query or a SQL Post Query.

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

Reference: JDBC-ODBC Bridge Listener Parameters

The following table lists and describes the parameters for the JDBC-ODBC Bridge listener.

Note: Common listener properties are defined in *Common Listener Parameters* on page 128.

Parameter	Description
Datasource	Name of the data source configured under the ODBC Driver Manager. For more information, see your ODBC Driver Manager documentation.
User	Database user ID to access the table.
Password	Database password associated with the user ID.

Reference: Oracle Listener Parameters

The following table lists and describes the parameters for the Oracle listener.

Note: Common listener properties are defined in *Common Listener Parameters* on page 128.

Parameter	Description
Host	Name or URL of the machine where the database is installed.
Port	Port on which the Host database is listening. The default is 1521.
SID	A unique name for the database service, chosen by the database administrator or the person who installed Oracle E-Business Suite.
User	Database user ID to access the table.
Password	Database password associated with the user ID.

Reference: SQL Server Listener Parameters

The following table lists and describes the parameters for the SQL Server listener.

Note: Common listener properties are defined in *Common Listener Parameters* on page 128.

Parameter	Description
Host	Name or URL of the machine where the database is installed.
Port	Port on which the Host database is listening. The default port is 1433
Database Name	Database name of the database where the table specified in the SQL statement is located.
User	Database user ID to access the table.
Password	Database password associated with the user ID.

Reference: DB2 Listener Parameters

The following table lists and describes the parameters for the DB2 listener.

Note: Common listener properties are defined in *Common Listener Parameters* on page 128.

Parameter	Description
Host	Name or URL of the machine where the database is installed.

Parameter	Description
Port	Port on which the Host database is listening.
Database Name	The name of the database.
DB2 Driver Type	Select from the following: ❑ app driver(type2) ❑ net driver(type3) ❑ jcc driver(type4)
User	The user ID used to access the database.
Password	The password associated with the user ID given.

Reference: **EDA Server Listener Parameters**

The following table lists and describes the parameters for the EDA Server listener.

Note: Common listener properties are defined in *Common Listener Parameters* on page 128.

Parameter	Description
Host	Name or URL of the machine where the database is installed.
Port	Port on which the Host database is listening.
Database Name	Database name of the database where the table specified in the SQL statement is located. **Note:** When you access a non-relational database, and the server component is an SSCTL server component, the database name must be the service name, and you must specify it. If the server component is installed on USS, you can leave the database field blank. For more information about the server component, see *Introducing the iWay Technology Adapter for RDBMS* on page 13.
User	Database user ID to access the table.
Password	Database password associated with the user ID.

Reference: JDBC Listener Parameters

The following table lists and describes the parameters for the JDBC listener.

Note: Common listener properties are defined in *Common Listener Parameters* on page 128.

Parameter	Description
Driver Class	Driver class for the JDBC listener.
Driver URL	Valid location of the JDBC driver.
User	Database user ID to access the table.
Password	Database password associated with the user ID.

Procedure: How to Start or Stop a Channel

To start or stop a channel:

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

   The X over the icon disappears as shown in the following image, and the channel starts.

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

Editing or Deleting a Channel

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

Procedure: How to Edit a Channel

To edit a channel:

1. Expand the *iWay Events* node.

2. Expand the *RDBMS* node.
3. In the left pane, select the channel you want to edit.
4. In the right pane, move the pointer over *Operations* and select *Edit*.

   The Edit channels dialog box opens.
5. Make the required changes to the channel configuration and click *Finish*.

Procedure: How to Delete a Channel

To delete a channel:

1. Expand the *iWay Events* node.
2. Expand the *RDBMS* node.
3. In the left pane, select the channel you want to delete.
4. In the right pane, move the pointer over *Operations* and select *Delete*.

   A confirmation dialog box opens.
5. To delete the channel you selected, click *OK*.

   The channel disappears from the list in the left pane.

Choosing a Listening Technique

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

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

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

You can poll a relational or non-relational database directly and send the results to the appropriate port. You also can use the following advanced techniques to listen to a database event.

- Standard event processing with row tracking

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

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

For more information, see *Standard Event Processing With Row Tracking* on page 136.

- Standard event processing with row removal

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

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

  For more information, see *Standard Event Processing With Row Removal* on page 142.

- Trigger-based event processing

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

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

  For more information, see *Trigger-based Event Processing* on page 146.

Standard Event Processing With Row Tracking

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

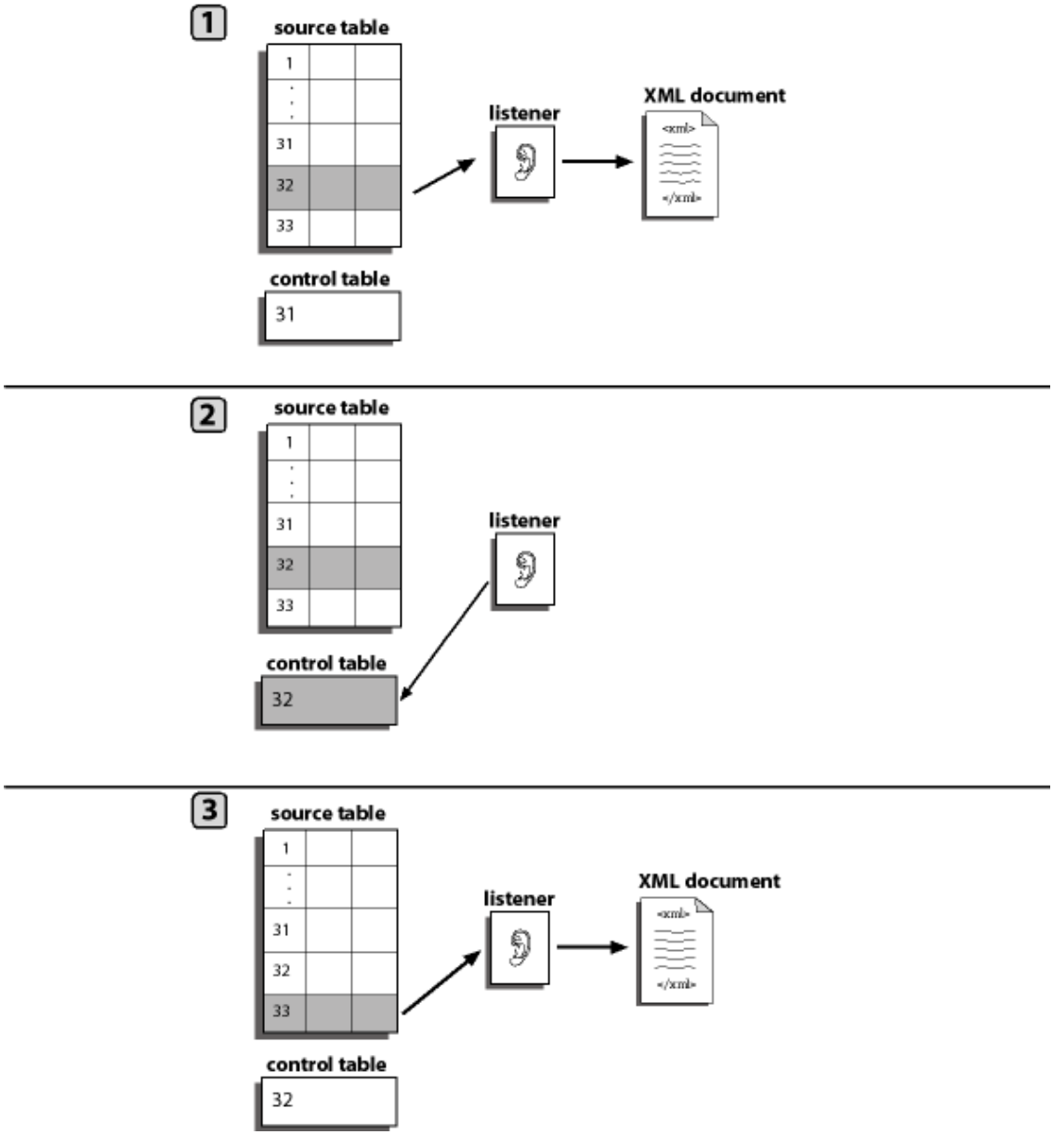
The single column of the control table corresponds to a column (or to a group of columns) in the source table that is unique, sortable, and indicates how recently the row was added to the source table relative to the other rows. For example, the first row added to the source table has the lowest value,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

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

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

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

The following figure illustrates standard event processing with row tracking.



In the previous figure:

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

The process repeats.

Procedure: How to Implement Standard Event Processing With Row Tracking

To implement standard event processing with row tracking:

1. Create a control table. For an example, see *Creating the Control Table for an RDBMS (Oracle) Event* on page 139.
2. Configure an RDBMS Table Listener in the iWay Web Console.

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

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

   For detailed instructions about configuring a listener, see *How to Create a Channel* on page 124. For information on post query parameters, see *The Post-query Parameter Operators* on page 140.

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

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

You can follow the steps in this example to create an Oracle E-Business Suite table named TEMP_NEW_YORK_ORDER_ENTITY that has a single field named WIP_ENTITY_ID. You specify this table when you configure the RDBMS Table Listener, as described in *The Post-query Parameter Operators* on page 140.

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_DISCETE_JOBS table.

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

Reference: The Post-query Parameter Operators

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

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

  The ? operator is useful in UPDATE statements:

  ```
  UPDATE table SET ?field WHERE ?field
  ```

  For example, the following statement

  ```
  UPDATE Stock_Prices_Temp SET ?PRICE WHERE ?RIC
  ```

  might be evaluated at run time as:

  ```
  UPDATE Stock_Prices_Temp SET PRICE=20 WHERE RIC = 'PG'
  ```

- ^fieldname is evaluated at run time as value.

  The ^ operator is useful in INSERT statements:

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

For example, the following statement

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

might be evaluated at run time as:

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

Example: Listening to trans_event Using the Row Tracking Technique

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

For more information on configuring a listener, see *How to Create a Channel* on page 124.

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

The table schemas for this example are:

```
SQL> describe trans_event
Name                            Null?         Type
---------------------------------------------------------------
EVENT_ID                         NOT NULL      NUMBER(38)
LAST_NAME                                      VARCHAR2(50)
TRANS_ID                                       CHAR(2)
SQL> describe last_trans
Name                            Null?        Type
-----------------------------------------------------------
EVENT_ID                                      NUMBER
```

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

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

1. Using a SQL query/data manipulation tool supplied by the database vendor, insert a record into the trans_event table based on the following information.

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

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

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

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

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

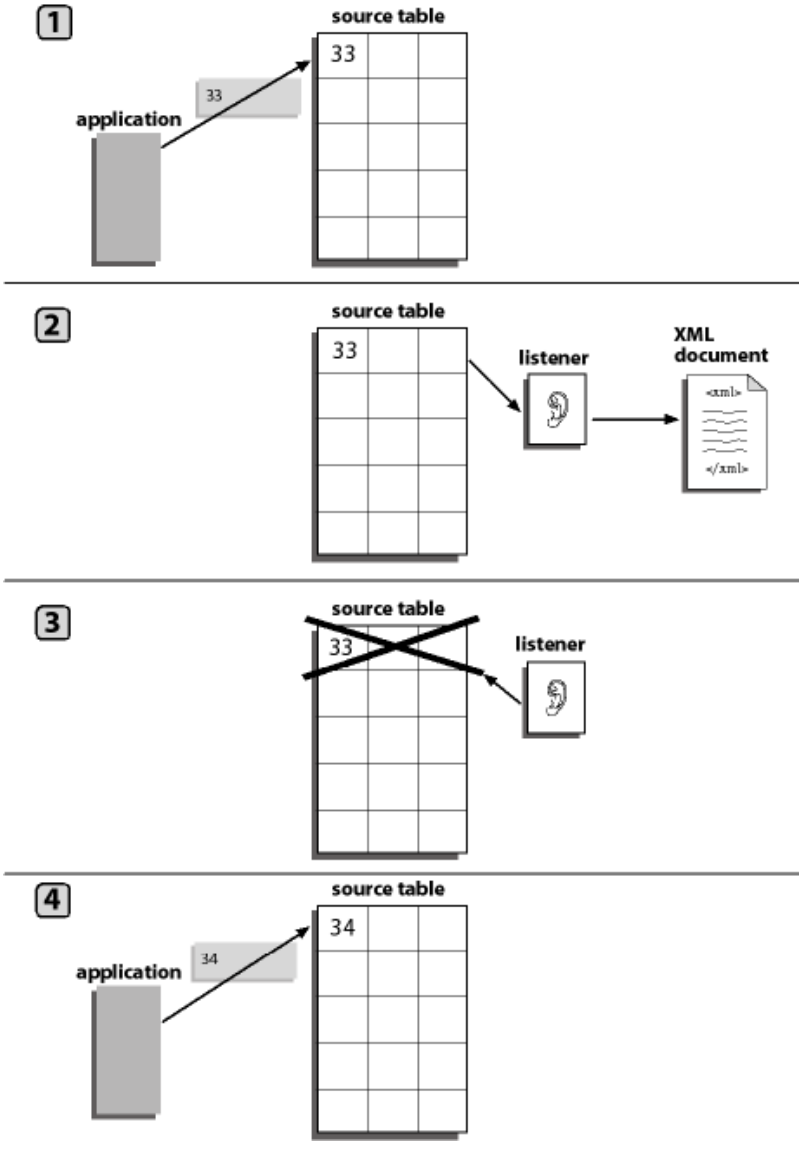
Parameter	Description
Host	Name or URL of the machine on which the database is installed.
Port	Port on which the Host database is listening.
User Name	User name that is registered with the back-end RDBMS.
Password	Password associated with the user name.
SQL Query	`SELECT * FROM TRANS_EVENT WHERE EVENT_ID>(select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

Standard Event Processing With Row Removal

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

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

The following figure illustrates standard event processing with row removal.



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

4. The application inserts a new row into the source table.

The process repeats itself.

Procedure: How to Implement Standard Event Processing With Row Removal

To implement the standard event processing with row removal technique:

1. Configure a Table Listener.
2. In addition to the required listener properties, provide values for the following optional properties:

   SQL Query: the SQL SELECT statement that identifies the source table to which the adapter listens and with which it queries the table.

   Post Query: the query that identifies the rows that the adapter automatically deletes from the table.

For detailed instructions about configuring a listener, see *Creating a Channel* on page 124. For information about Post query operators, see *The Post-query Parameter Operators* on page 140.

Example: Listening to stock_prices Using the Row Removal Technique

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

```
SQL> describe stock_prices
Name                                      Null?    Type
----------------------------------------- -------- -----------
RIC                                       NOT NULL VARCHAR2(6)
PRICE                                              NUMBER(7,2)
UPDATED                                            DATE
```

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

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

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

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

e. For the SQL Query, use select * from stock_prices.

f. For the Post Query, use delete from stock_price where ?RIC.

g. For Polling Interval, specify an interval in seconds.

For a description of these properties, see *The Post-query Parameter Operators* on page 140.

2. For more information on configuring a listener, see *Creating a Channel* on page 124.

Trigger-based Event Processing

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

The trigger-based technique provides the following benefits:

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

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

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

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

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

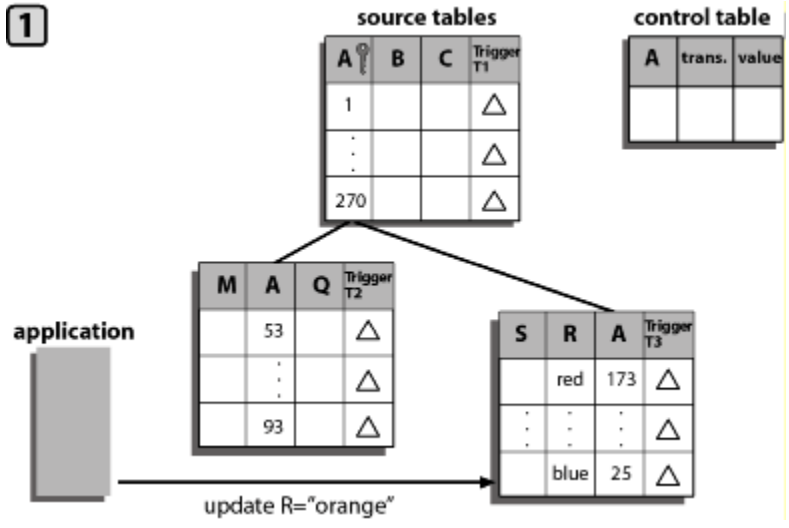
To use the trigger-based technique, you assign a trigger to each table that you want to monitor. When a value changes, it fires the corresponding trigger that writes data to a control table. The iWay Technology Adapter for RDBMS listens to the control table by running a query against it. When it finds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

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

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

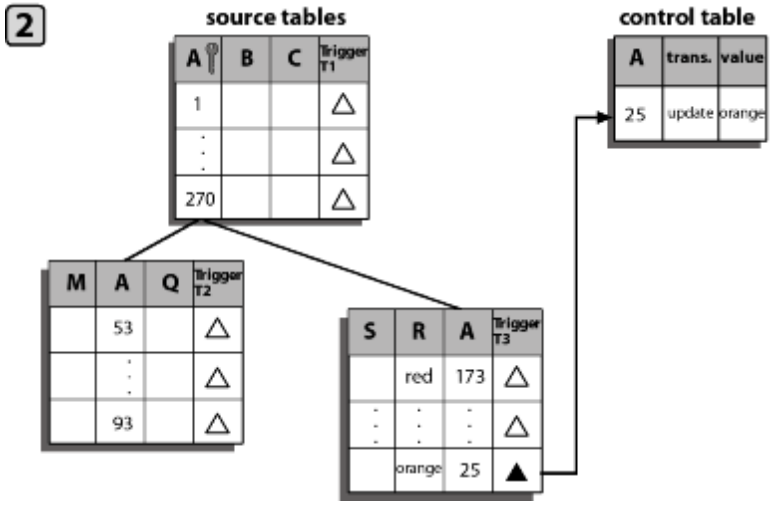
Note: Event processing may be limited for some non-relational databases due to the functionality of the database and its interaction with the iWay server component. For more information on the iWay server component, see the *iWay Data Adapter Administrator User's Guide* or consult with your DBA.

The following figures illustrate trigger-based event processing:



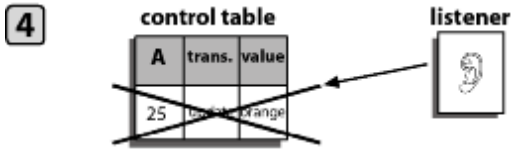
1. Your application updates a row in a group of related source tables as shown in the previous figure.

The update causes a row trigger to fire in the changed table as shown in the following figure.

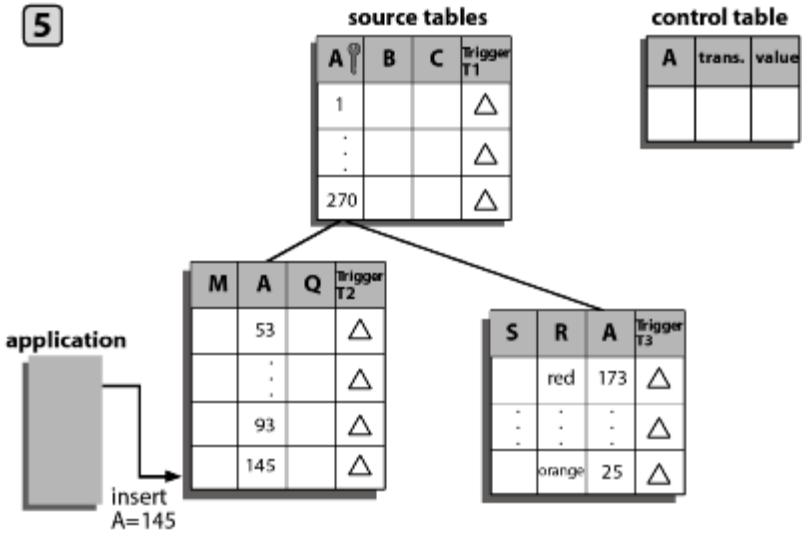


2. The trigger inserts a row into the control table, and the new control table row includes the key value (25), the type of transaction (update), and the new cell value (orange) as shown in the previous figure.
3. The listener queries the control table and copies the new row to an XML document. It sends the document to the Reply_to destination as shown in the following figure.

4. The listener deletes the control table row to ensure that the row is not read again when the listener next queries the table as shown in the following figure.



5. The application inserts a new row into one of the source tables as shown in the following figure.



The process repeats itself.

Procedure: How to Implement Trigger-based Event Processing

To implement the trigger-based event processing technique:

1. Create the control table.

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

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

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

2. Assign triggers to the source tables.

   The triggers you assign, and to which tables you assign them, is determined by what kind of change you want to monitor. The triggers implement event-processing logic. For a sample trigger, see *Trigger on WIP_ENTITY_NAME Column in an Oracle Table* on page 150.

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

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

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

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

   SQL Query: the SQL SELECT statement that identifies the control table to which the adapter listens and with which it queries the table to determine changes in the source tables.

   Post Query: the query that identifies the rows that the adapter automatically deletes from the control table.

   For detailed instructions about configuring a listener, see *Creating a Channel* on page 124. For information about Post query operators, see *The Post-query Parameter Operators* on page 140.

Example: Trigger on WIP_ENTITY_NAME Column in an Oracle Table

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

```
CREATE OR REPLACE TRIGGER IWAY.IWAY_PO_CDC_WE_TRG
AFTER INSERT OR DELETE OR UPDATE OF WIP_ENTITY_NAME
ON WIP.WIP_ENTITIES
FOR EACH ROW
BEGIN
IF INSERTING THEN
   INSERT INTO IWAY.IWAY_PO_CDC
      VALUES (
       :NEW.WIP_ENTITY_ID,
       :NEW.ORGANIZATION_ID,
            'UPDATE');
ELSE
   INSERT INTO IWAY.IWAY_PO_CDC
      VALUES (
         :OLD.WIP_ENTITY_ID,
         :OLD.ORGANIZATION_ID,
            'UPDATE');
END IF;
EXCEPTION
   WHEN DUP_VAL_ON_INDEX THEN
      NULL;         -- Record already exists
END;
```

Appendix A

Configuring the Adapter in an iWay Environment

After you successfully configure the adapter to represent a particular adapter target, the adapter can be assigned to an iWay Service Manager channel.

In this appendix:

Configuring the Adapter in iWay Service Manager

Before configuring the adapter in iWay Service Manager, you must first create a target, which represents a connection to a backend system, using iWay Explorer. For more information on configuring targets and connections using iWay Explorer, see *Creating XML Schemas or Business Services* on page 21 or the *iWay Explorer User's Guide*.

You configure the adapter in the iWay Service Manager console. The configuration process creates run-time connection and persistent data files within Service Manager. The configuration process interrogates the Service Manager repository entries that were built when the target and connection were created using iWay Explorer. The define adapter process creates the run-time repository based on the design-time repository.

Procedure: How to Define an Adapter

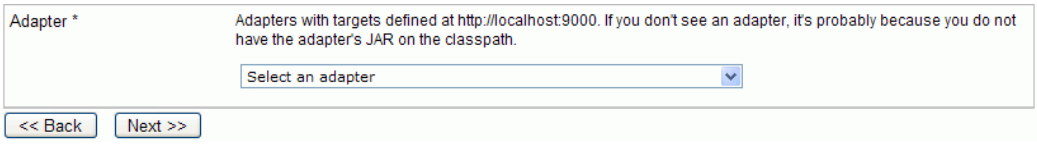
To define an adapter:

1. In the Service Manager console, select *Registry*, then *Adapters*.
2. Click *Add*.

   The iBSP URL pane opens, as shown in the following image.

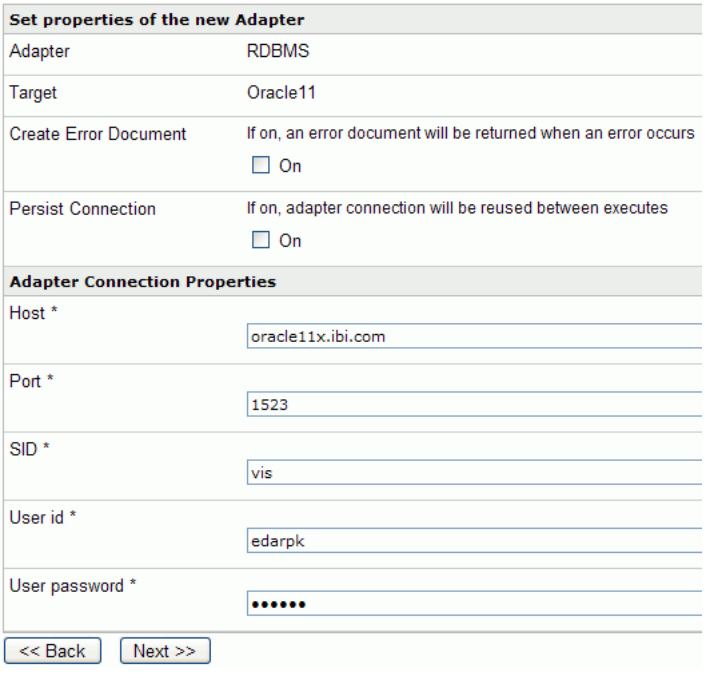
3. Enter your iBSP URL, which is the location of the Service Manager repository, for example, `http://localhost:9000`. This field is required.
4. Click *Next*.

An adapter selection pane opens, as shown in the following image.



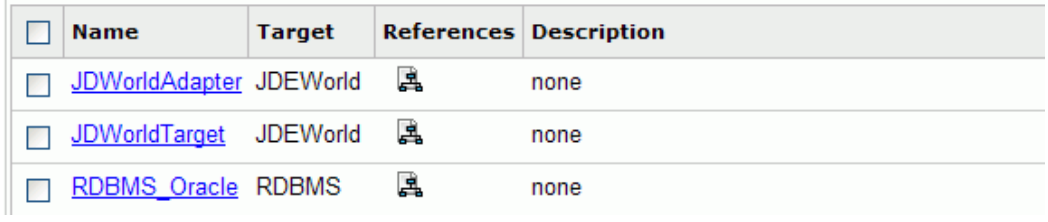
5. From the Adapter drop-down list, select the Adapter, then click *Next*.
6. From the Target drop-down list, select a target you configured for the adapter in iWay Explorer, then click *Next*.

   The connection information associated with the target selected is displayed.



   a. Select whether to return an error document when an error occurs.
   b. Select whether an adapter connection will be reused between executes.
   c. Review the connection information you specified in iWay Explorer. You can change or update any information.
7. Click *Next*.
8. Provide a name and, optionally, a description, for the adapter, and click *Finish*.

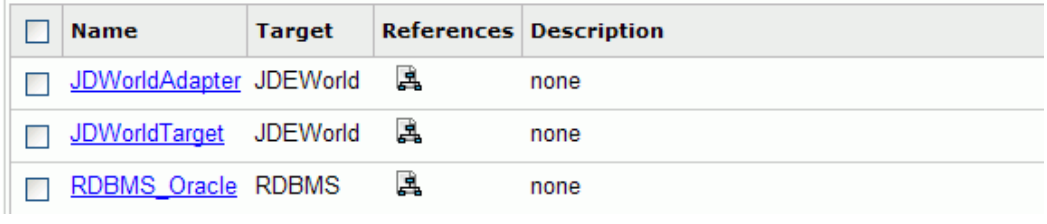
The adapter appears in the adapters list, as shown in the following image.



Name	Target	References	Description
JDWorldAdapter	JDEWorld		none
JDWorldTarget	JDEWorld		none
RDBMS_Oracle	RDBMS		none

Procedure: How to Modify or Update an Adapter Connection

The following image shows the Adapter Defines pane which displays the name of the adapter and the description (optional).



Name	Target	References	Description
JDWorldAdapter	JDEWorld		none
JDWorldTarget	JDEWorld		none
RDBMS_Oracle	RDBMS		none

To modify or update an adapter connection:

1. From the Adapters list, click the adapter reference you defined, in this example, *RDBMS_Oracle*.

   The pane that displays the target connection information opens. You cannot change the name of the adapter or the target, but you can edit the connection information.

2. After you modify the connection information, click *Update Connection Properties*.
3. After you make changes or additions to the adapter target in iWay Explorer, click *Update Adapter Data*.
4. Click *Finish*.

Appendix B

JDBC Drivers

This section lists the supported JDBC drivers for use with the iWay Technology Adapter for RDBMS.

In this appendix:

Copying and Collecting a JDBC File

The iWay Technology Adapter for RDBMS requires JDBC drivers when connecting to certain databases. The appropriate driver can be acquired from a respective website of a vendor or from the installation directory of the DBMS.

To enable these adapters:

1. Use the following table to determine the JDBC driver files required for your database.
2. Copy the required JDBC files into the iWay7\lib directory.

   The default location for this directory on Windows is:

   `C:\Program Files\iWay7\lib`

   On other platforms, use the corresponding location.

   The following table lists the iWay adapters and the required libraries or drivers.

iWay Adapter for...	Required Libraries or Drivers
CACHE	cachedb.jar, This file is typically located in *C*:\CacheSys\DEV\JAVA\lib
CICS	None

iWay Adapter for...	Required Libraries or Drivers

DB2 Net Driver
DB2 App Driver
JCC Driver
	JDBC driver for DB2 (db2java.zip) for connection to DB2 version 7 and DB2 version 8.
	Installed as part of the DB2 server. The default location on Windows is one of the following:
	`C:\SQLLIB\java\db2java.zip`
	`C:\Program Files\SQLLIB\java\db2java.zip`
	For JCC driver (connection to DB2 version 9), two files are needed: db2jcc.jar and db2jcc_license.jar. These are typically located in:
	`C:\Program Files\SQLLIB\java\`

IMS	None

Informix	JDBC driver for Informix (ifxjdbc.jar).
	Download the driver from the Informix website.
	For Informix 7:
	`ifxjdbc.jar`
	For Informix 9:
	`ifxjdbc.zip`

iWay Adapter for...	Required Libraries or Drivers
Oracle E-Business Suite	Oracle JDBC drivers (OJDBC14.jar). All calls to Oracle E-Business Suite occur through these drivers. If you do not have the appropriate JDBC driver, Oracle Technology Network (OTN) provides a download site at: `http://otn.oracle.com/software/tech/java/sqlj_jdbc/content.html` **Note:** To download the drivers, you require a logon ID. If you are using OCI drivers, you must install and configure Oracle Client on the machine with the iWay Adapter for Oracle E-Business Suite. To use iWay Concurrent Program request functionality, you must install and configure Oracle Client on the Oracle database that supports Oracle E-Business Suite.

Oracle	Oracle JDBC driver (OJDBC14.jar for support with Oracle 9i and higher).
	
	You can download this driver from the Oracle website:
	
	`http://otn.oracle.com/software/tech/java/sqlj_jdbc/content.html`
	
	Note: To download the drivers, you require a logon ID.
	For more information on Oracle JDBC issues, see the Oracle JDBC FAQ:
	
	`http://otn.oracle.com/tech/java/sqlj_jdbc/htdocs/jdbc_faq.htm`

SQL Server	SQL Server 2000 JDBC driver.
	
	The JDBC driver includes the following three files:
	
	`msbase.jar`
	`mssqlserver.jar`
	`msutil.jar`
	
	SQL Server 2005 JDBC driver version 1.1.
	SQL Server 2008 JDBC driver version 2.0.
	You can download the driver free of charge from:
	
	`http://microsoft.com`

iWay Adapter for...	Required Libraries or Drivers
Sybase	JDBC driver for Sybase servers (jConnect for JDBC) (jconn2.jar). You can download the driver from the Sybase downloads website: `http://www.sybase.com/downloads` Select *jConnect for JDBC* and review information on the jConnect for JDBC website. To extract the JDBC driver, obtain the jConnect.zip that corresponds to your version of Sybase and follow the steps described on the jConnect download page.

Feedback

Customer success is our top priority. Connect with us today!

Information Builders Technical Content Management team is comprised of many talented individuals who work together to design and deliver quality technical documentation products. Your feedback supports our ongoing efforts!

You can also preview new innovations to get an early look at new content products and services. Your participation helps us create great experiences for every customer.

To send us feedback or make a connection, contact Sarah Buccellato, Technical Editor, Technical Content Management at *Sarah_Buccellato@ibi.com.*

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at *Frances_Gambino@ibi.com.*

Information Builders

iWay

iWay Technology Adapter for RDBMS User's Guide

Version 7.0.x and Higher

DN3502308.0418

Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898