BEA WebLogic Adapter for RDBMS

User Guide

Version 8.1.1
Document Revised: October 2003

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

About This Document

1. Introducing the BEA WebLogic Adapter for RDBMS

2. Defining Application Views for RDBMS

A. Service and Event Examples

About This Document

This document describes how to use the BEA WebLogic Adapter for RDBMS. This document is organized as follows:

- Chapter 1, "Introducing the BEA WebLogic Adapter for RDBMS," describes the adapter, how it relates to both RDBMS business objects and WebLogic Integration.
- Chapter 2, "Defining Application Views for RDBMS," describes application views and how to use them to configure events and services.
- Appendix A, "Service and Event Examples," contains sample XML documents based the request and response schemas for services and the event schemas for events.
- Appendix B, "Supported Data Types," lists the data types for which the Adapter for RDBMS has been tested.
- Appendix C, "Error Messages and Troubleshooting," explains the various error messages you might encounter while using the Adapter for RDBMS, and how you can troubleshoot them.

Who Should Read This Documentation

This document is intended for the following members of an integration team:

- Integration Specialists—Lead the integration design effort. Integration specialists have expertise in defining the business and technical requirements of integration projects, and in designing integration solutions that implement specific features of WebLogic Integration.

The skills of integration specialists include business and technical analysis, architecture design, project management, and WebLogic Integration product knowledge.

- Technical Analysts—Provide expertise in an organization's information technology infrastructure, including telecommunications, operating systems, applications, data repositories, future technologies, and IT organizations. The skills of technical analysts include technical analysis, application design, and information systems knowledge.
- Enterprise Information System (EIS) Specialists—Provide domain expertise in the systems that are being integrated using WebLogic adapters. The skills of EIS specialists include technical analysis and application integration design.
- System Administrators—Provide in-depth technical and operational knowledge about databases and applications deployed in an organization. The skills of system administrators include capacity and load analysis, performance analysis and tuning, deployment topologies, and support planning.

Additional Information

To learn more about the software components associated with the adapter, see the following documents:

- *BEA WebLogic Adapter for RDBMS Release Notes*

  http://edocs.bea.com/wladapters/rdbms/docs811/pdf/relnotes.pdf
- *BEA WebLogic Adapter for RDBMS Installation and Configuration Guide*

  http://edocs.bea.com/wladapters/rdbms/docs811/pdf/install.pdf
- *Introduction to the BEA WebLogic Adapters*

  http://edocs.bea.com/wladapters/docs81/pdf/intro.pdf
- BEA WebLogic Adapters 8.1 Dev2Dev Product Documentation

  http://dev2dev.bea.com/products/wladapters/index.jsp
- Application Integration documentation

  http://edocs.bea.com/wli/docs81/aiover/index.html

  http://edocs.bea.com/wli/docs81/aiuser/index.html
- BEA Application Explorer Installation and Configuration Guide

  http://edocs.bea.com/wladapters/bae/docs811/pdf/install.pdf
- BEA WebLogic Integration documentation

http://edocs.bea.com/wli/docs81/index.html

- BEA WebLogic Platform documentation

  http://edocs.bea.com/platform/docs81/index.html

- Documentation from your RDBMS vendor

How to Use This Document

This document is designed to be used in conjunction with *Using the Application Integration Design Console*, available at the following URL:

http://edocs.bea.com/wli/docs81/aiuser/index.html

Using the Application Integration Design Console describes, in detail, the process of defining an application view, which is a key part of making an adapter available to process designers and other users. What *Using the Application Integration Design Console* does *not* cover is the specific information about Adapter for RDBMS that you need to supply to complete the application view definition. You will find that information in this document.

At each point in *Using the Application Integration Design Console* where you need to refer to this document, you will see a note that directs you to a section in your adapter user guide, with a link to the edocs page for adapters. The following roadmap illustration shows where you need to refer from *Using the Application Integration Design Console* to this document.

Figure 1 Information Interlock with *Using the Application Integration Design Console*

Contact Us!

Your feedback on the BEA WebLogic Adapter for RDBMS documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the BEA WebLogic Adapter for RDBMS documentation.

In your e-mail message, please indicate that you are using the documentation for BEA WebLogic Adapter for RDBMS and the version of the documentation.

If you have any questions about this version of BEA WebLogic Adapter for RDBMS, or if you have problems using the BEA WebLogic Adapter for RDBMS, contact BEA Customer Support through BEA WebSUPPORT at www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

Convention	Item

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

|	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

Convention	Item

. . .	Indicates one of the following in a command line:
	• That an argument can be repeated several times in a command line
	• That the statement omits additional optional arguments
	• That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER 1

Introducing the BEA WebLogic Adapter for RDBMS

This section introduces the BEA WebLogic Adapter for RDBMS and describes how the adapter enables integration with RDBMS business objects and WebLogic Integration.

It includes the following topics:

About Adapters and BEA WebLogic Integration

The BEA application integration solution uses adapters and application views to help you integrate applications in your enterprise. At its most fundamental level, an *adapter* is software that connects an enterprise information system (EIS) and an integration server. This bi-directional connection consists of *services*—interactions that originate in the adapter (and may require a response from the EIS)—and *events*, interactions that originate in the EIS.

Most EIS systems make selected information and functions available to other applications by way of specialized integration APIs. An adapter connects to the EIS through its integration API, or through database or system calls, and exposes the available functions from the EIS. However, rather than exposing the intricacies of APIs to users, WebLogic Integration incorporates *application views*—business-oriented interfaces that provide a layer of abstraction between an adapter and the EIS capabilities the adapter exposes.

Figure 1-1 Application View in an Integration Solution

Application views contain definitions for the services and events used by business processes to communicate with an EIS. They also contain connection information and XML schema that define inputs and outputs for services and events. After an adapter is deployed, you can use its Web-based interface to define as many application views as you need, and other WebLogic Integration components and applications can use that adapter to access data on the EIS.

To learn more about BEA WebLogic Integration in the BEA WebLogic Workshop environment, see the WebLogic Integration site at the following URL:

`http://edocs.bea.com/wladapters/docs81/index.html`

To learn more about the role of adapters in application integration architecture, see "Key Components of Integration Solutions" on page 1-3.

To learn more about adapters in general, see the *Introduction to the BEA WebLogic Adapters* at the following URL:

`http://edocs.bea.com/wladapters/docs81/index.html`

Key Components of Integration Solutions

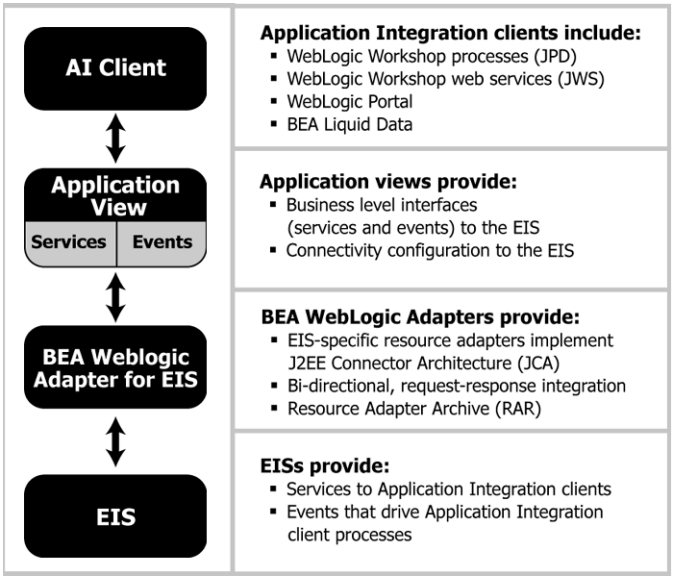
This section describes some of the key concepts you need to be familiar with before you work with an adapter.

- Basic WebLogic Integration Architecture
- Enterprise Information Systems
- Resource Adapters
- Application Views
- Service Clients and Event Consumers
- Schema

Basic WebLogic Integration Architecture

Adapters are used in conjunction with the Application Integration component of BEA WebLogic Integration. This component provides a systematic, standards-based architecture for hosting business-oriented interfaces to enterprise applications.

Figure 1-2 Adapters in the Application Integration Architecture



For general information about Application Integration, see the following documents:

- *Introducing Application Integration* at the following URL:

  `http://edocs.bea.com/wli/docs81/aiover/index.html`

- *Using the Application Integration Design Console* at the following URL:

  `http://edocs.bea.com/wli/docs81/aiuser/index.html`

Enterprise Information Systems

An *enterprise information system* (EIS) is software that provides the information infrastructure for an enterprise. An EIS offers a set of services to its clients, which are made available to clients via local and/or remote interfaces. An integration solution involves integration with one or more EISs.

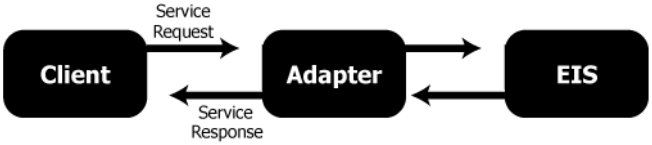
Resource Adapters

A *resource adapter* (or simply *adapter*) is a BEA software component that acts as a connector between an EIS and a J2EE application server (such as BEA WebLogic Server). Each adapter provides bi-directional, request-response integration with a specific application or technology.

Adapters handle two general types of operations:

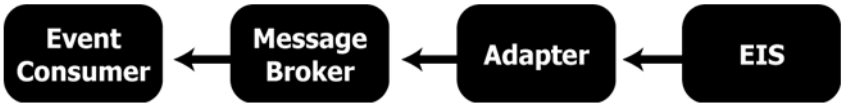
- *Services* are request/response communications with the EIS. Client applications submit service requests to the EIS via the adapter, and the adapter returns the EIS response back to the client. For example, a business process might invoke a service to execute a SELECT statement on an RDBMS. Responses are either synchronous or asynchronous.

Figure 1-3 Service Invocations



- *Events* are asynchronous, one-way messages received from an EIS. For example, the adapter can receive data from an RDBMS in response to an event triggered. The adapter routes the EIS message to the appropriate software component via the WebLogic Integration Message Broker and the Application Integration JMS infrastructure.

Figure 1-4 Event Notifications



In effect, a service is a *request for some work to be done* and an event is a *notification that some work has been done*.

For more information about the specific services and events supported by Adapter for RDBMS, see "About the BEA WebLogic Adapter for RDBMS" on page 1-9.

To learn more about the WebLogic Integration Message Broker and the Application Integration JMS infrastructure, see "Introducing Application Integration" at the following URL:

`http://edocs.bea.com/wli/docs81/aiover/index.html`

Application Views

An *application view* is a business oriented interface to objects and operations within an EIS. Application views include the information needed to communicate with the EIS as well as configurations for services and events.

To learn more about using application views in business processes, see "Application View Control" at the following URL:

`http://edocs.bea.com/workshop/docs81/doc/en/integration/controls/controlsAppView.html`

You typically define an application view for a specific business process. Therefore, you might have multiple application views defined for a single adapter, each designed to meet a specific requirement.

An application view defines:

- **Communication with the EIS**, including connection settings, login credentials, and so on.
- **Service invocations**, including the information that the EIS requires for the request, as well as any service request and response schemas associated with the service.
- **Event notifications**, including the information that the EIS publishes and the event schemas for inbound messages.

You create application views in either of two ways:

- Using the Application View Console. For detailed information about application views, see "What Are Application Views?" in "Understanding Application Integration" in *Using the Application Integration Design Console* at the following URL:

  `http://edocs.bea.com/wli/docs81/aiover/2intfra.html`

- Writing custom code. For more information, see "Using Application Views by Writing Custom Code" in *Using the Application Integration Design Console* at the following URL:

  `http://edocs.bea.com/wli/docs81/aiuser/4usrcust.html`

An application view for Adapter for RDBMS provides these features:

- Standards-based data representation. All events, requests, and responses are represented as standards-based XML.
- Abstraction from the details of the EIS. Application views offer a level of abstraction from the details of the underlying EIS, freeing the developers to concentrate on the business processes and data and not on the configuration and details of that system.

To learn more about application views, see Chapter 2, "Defining Application Views for RDBMS."

Service Clients and Event Consumers

In an integration solution, there are clients that invoke services, and consumers for event notifications.

Service Clients

A variety of clients can invoke services on an EIS via an application view. They include BEA WebLogic Workshop business processes, web services, and portals; queries and BEA Liquid Data; and custom Java applications.

For more information, see the following topics in the BEA WebLogic Workshop Help System:

- "Building Integration Applications"
- "Building Web Services"
- "Building Portal Applications"

at the following URL:

`http://edocs.bea.com/workshop/docs81/doc/en/core/index.html`

In addition, see "Using Application Views With Business Processes" in *Using the Application Integration Design Console* at the following URL:

http://edocs.bea.com/wli/docs81/aiuser/3usruse.html

Event Consumers

Adapters deliver events using the WebLogic Integration Message Broker, which provides business processes with a channels-based publish and subscribe communication mechanism. For more information, see Message Broker Controls at the following URL:

http://edocs.bea.com/workshop/docs81/doc/en/integration/controls/controlsBrokerOview.html

Consumers can include BEA WebLogic Workshop business processes, web services, and portals, as well as custom Java applications. For more information, see the WebLogic Workshop help system at the following URL:

http://edocs.bea.com/workshop/docs81/doc/en/core/index.html

In addition, see "Receiving Events" in "Using Applications With Business Processes" in *Using the Application Integration Design Console* at the following URL:

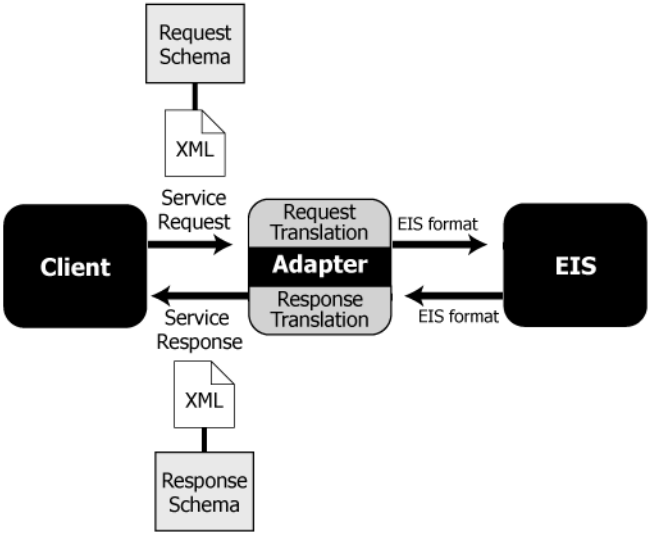
http://edocs.bea.com/wli/docs81/aiuser/3usruse.html

Schema

At run-time, the EIS and the adapter exchange service requests, service responses, and events via XML documents. The adapter handles the data translation between XML documents and the EIS format, using *schemas* that map the data between XML and the EIS format. The Adapter for RDBMS automatically generates the schema when you add services and events to an application view.

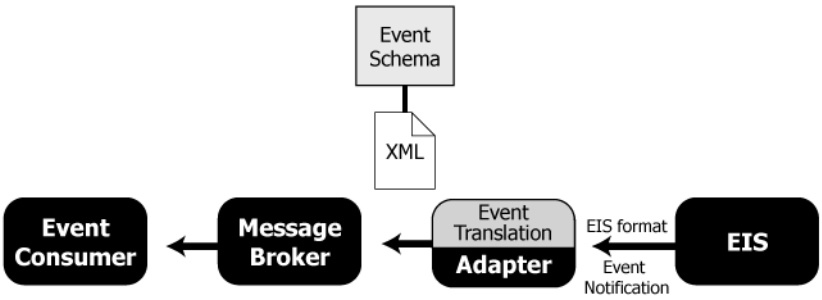
- For service requests, the request arrives at the adapter in the form of an XML document. The adapter uses the *request schema* associated with the service to translate the request to the format that the EIS expects. Similarly, when the adapter receives the response back from the EIS, it uses the *response schema* associated with the service to translate the response to an XML document that the requesting application handles.

Figure 1-5 Adapter Translation for Service Invocations



- For event notifications, the message arrives at the adapter in the format that the EIS uses to publish the event. The adapter uses the *event schema* associated with the event to translate the response to an XML document that the subscribed application handles.

Figure 1-6 Adapter Translation for Event Notifications



About the BEA WebLogic Adapter for RDBMS

The BEA WebLogic Adapter for RDBMS connects to your RDBMS system so that you can easily use your RDBMS data and functions within your business processes. The adapter provides scalable, reliable, and secure access to your RDBMS system.

This section includes the following topics:

- Supported RDBMS Operations for Application Integration
- Supported Services
- Supported Events
- Benefits of the Adapter for RDBMS

Supported RDBMS Operations for Application Integration

The Adapter for RDBMS enables integration with RDBMS by providing bi-directional access to a database. Services and events act, therefore, as plugable components that can be used in business processes to interact with the database.

The input and output of a query or stored procedure invocation are expressed in XML. This provides a convenient and simple method for integrating databases with enterprise applications using WebLogic Integration.

The Adapter for RDBMS supports the following operations:

- Asynchronous, bi-directional message interactions between applications and databases, including Oracle, Microsoft SQL Server, IBM DB2 UDB, Informix Dynamic Server, and Sybase Adaptive Server

- Integration of requests and table event (outbound) operations in processes
- JDBC 2.1 standard SQL operations (DELETE, INSERT, SELECT, and UPDATE)
- Execution of stored procedures

Supported Services

The Adapter for RDBMS supports the following service types:

- Standard SQL, which invokes a standard static SQL statement without any parameters for the statement defined at design time
- Parametrized SQL, which invokes a parametrized SQL statement for the statement defined at design time
- Stored Procedure, which invokes a stored procedure defined in the RDBMS
- Arbitrary SQL, which invokes an SQL statement without any parameters for the statement defined at run time

Supported Events

Events are asynchronous, one-way messages received from an RDBMS. Events post an XML response document from an RDBMS whenever a specific event of interest is triggered. The event could contain data about a Select, Insert, Update or Delete operation that has occurred on a table in the RDBMS.

An event:

1. Polls the RDBMS for an event of interest.
2. Transforms the data of interest (response) from the RDBMS data format to an XML document that conforms to the response XML schema for the event, which is based on metadata in the RDBMS.

The Adapter for RDBMS supports the following event types:

- Trigger, which notifies that an Insert, Update, or Delete event has occurred in a database table
- Select By Timestamp, which notifies an Insert or Update event in a database table with the corresponding timestamp columns

- Select Then Delete, which notifies records of interest based on a Select query on a database table

Benefits of the Adapter for RDBMS

The combination of the adapter and WebLogic Integration supplies everything you need to integrate your business processes and enterprise applications with your RDBMS system. The Adapter for RDBMS provides these benefits:

- Integration can be achieved without custom coding.
- Business processes can be started by events generated by RDBMS.
- Business processes can request and receive data from your RDBMS system using services.
- Adapter events and services are standards-based. The adapter services and events provide extensions to the *J2EE Connector Architecture* (JCA) version 1.0 from Sun Microsystems, Inc. For more information, see the Sun JCA page at the following URL:

  `http://java.sun.com/j2ee/connector/`
- The adapter and WebLogic Integration solution is scalable. The BEA WebLogic Platform provides clustering, load balancing, and resource pooling for a scalable solution. For more information about scalability, see the following URL:

  `http://e-docs.bea.com/wls/docs81/cluster/index.html`
- The adapter and WebLogic Integration solution benefits from the fault-tolerant features of the BEA WebLogic Platform. For more information about high availability, see the following URL:

  `http://edocs.bea.com/wli/docs81/deploy/index.html`
- The adapter and WebLogic Integration solution is secure, using the security features of the BEA WebLogic Platform and the security of your RDBMS system. For more information about security, see the following URL:

  `http://edocs.bea.com/wls/docs81/secintro/index.html`

Getting Started With the Adapter for RDBMS

This section gives an overview of how to get started using the BEA WebLogic Adapter for RDBMS within the context of an application integration solution. Integration with RDBMS involves the following tasks:

- Step 1: Design the Application Integration Solution

- Step 2: Determine the Required RDBMS Functions
- Step 3: Define Application Views and Configure Services and Events
- Step 4: Integrate with Other BEA Software Components
- Step 5: Deploy the Solution to the Production Environment

Step 1: Design the Application Integration Solution

The first step is to design an application integration solution, which includes (but is not limited to) such tasks as:

- Defining the overall scope of application integration.
- Determining the business process(es) to integrate.
- Determining which WebLogic Platform components will be involved in the integration, such as business processes or portals built in the BEA WebLogic Workshop environment.
- Determining which external systems and technologies will be involved in the integration, such as RDBMS systems and other EISs.
- Determining which BEA WebLogic Adapters will be required, such as the BEA WebLogic Adapter for RDBMS. An application integration solution can involve multiple adapters.

This step involves the expertise of business analysts, system integrators, and EIS specialists (including RDBMS specialists). Note that an application integration solution can be part of a larger integration solution.

To learn more about designing an application integration solution, see *Designing WebLogic Integration Solutions* at the following URL:

`http://edocs.bea.com/wli/docs81/design/index.html`

Step 2: Determine the Required RDBMS Functions

Within the larger context of an application integration project, you must determine which specific RDBMS tables and stored procedures you need to access.

Factors to consider include (but are not limited to):

- Access requirements for your RDBMS
- RDBMS transactions involved in business processes

- RDBMS operations you need to perform
- Whether operations are, from the adapter point of view:
  - services, which notify the RDBMS with a request for action, and, in addition, whether such services should be processed synchronously or asynchronously
  - events, which are notifications from the RDBMS system that trigger business processes.

This step involves the expertise of RDBMS specialists, including analysts and administrators.

Step 3: Define Application Views and Configure Services and Events

You create an application view that provides an XML-based interface between WebLogic Integration and a particular RDBMS system within your enterprise. If you are accessing multiple RDBMS systems, you define a separate application view for each RDBMS system you want to access. To provide different levels of security access (such as "guest" and "administrator"), define a separate application view for each security level.

Once you define an application view, you can configure events and services in that application view that employ the XML schemas that the adapter automatically generates for you.

To learn more about defining application views, see Chapter 2, "Defining Application Views for RDBMS" in conjunction with *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

Step 4: Integrate with Other BEA Software Components

Once you have configured and published one or more application views for RDBMS integration, you can integrate these application views into other BEA software components, such as business processes or portals built in the BEA WebLogic Workshop environment.

For more information, see *Using the Application Integration Design Console*, particularly "Using Application Views with Business Processes," at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

Step 5: Deploy the Solution to the Production Environment

After you have designed, built, and tested your application integration solution, you can deploy it into a production environment. The following list describes some of the tasks involved in deploying an application integration:

- Design the deployment.
- Deploy the required components of the BEA WebLogic Platform.
- Install and deploy the BEA WebLogic Adapter for RDBMS as described in *BEA WebLogic Adapter for RDBMS Installation and Configuration Guide*.
- Deploy your application views and schemas for RDBMS integration.
- Verify business processes in the production environment.
- Monitor and tune the deployment.

To learn more about deploying your application integration solution, see *Deploying WebLogic Integration Solutions* at the following URL:

`http://edocs.bea.com/wls/docs81/deployment/index.html`

CHAPTER 2

Defining Application Views for RDBMS

An application view is a business-oriented interface to objects and operations within an EIS.

This section presents the following topics:

How to Use This Document

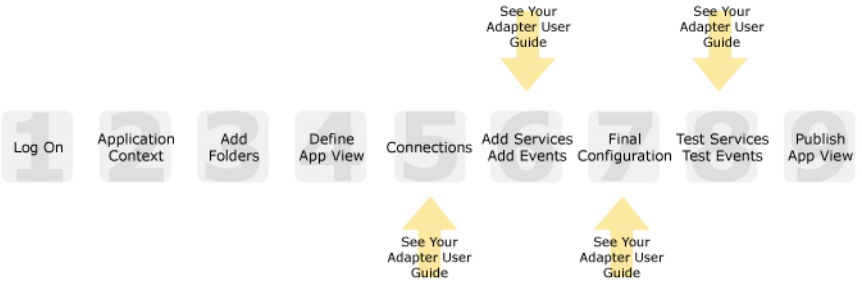
This document is designed to be used in conjunction with *Using the Application Integration Design Console*, available at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

Using the Application Integration Design Console describes, in detail, the process of defining an application view, which is a key part of making an adapter available to process designers and other users. What *Using the Application Integration Design Console* does *not* cover is the specific information—about connections to your RDBMS, as well as supported services and events—that you must supply as part of the application view definition. You will find that information in this section.

At each point in *Using the Application Integration Design Console* where you need to refer to this document, you will see a note that directs you to a section in your adapter user guide, with a link to the edocs page for adapters. The following road map illustration shows where you need to refer to this document from *Using the Application Integration Design Console*.

Figure 2-1 Information Interlock with *Using the Application Integration Design Console*



Before You Begin

Before you define an application view, make sure you have:

- Installed and deployed the adapter according to the instructions in *BEA WebLogic Adapter for RDBMS Installation and Configuration Guide*.
- Determined which business processes need to be supported by the application view. The required business processes determine the types of services and events you include in your application views. Therefore, you must gather information about the application's business requirements from the business analyst. Once you determine the necessary business

processes, you can define and test the appropriate services and events. For more information, see "Getting Started With the Adapter for RDBMS" on page 1-11.

- Gathered the connection information for your RDBMS system.

About Application Views

An application view defines:

- Connection information for the EIS, including login information, connection settings, and so on.
- Service invocations, including the information the EIS requires for this request, as well as the request and response schemas associated with the service.
- Event notifications, including the information the EIS publishes and the event schema for inbound messages.

Typically, an application view is configured for a single business purpose and contains only the services and events required for that purpose. An EIS might have multiple application views, each defined for a different purpose.

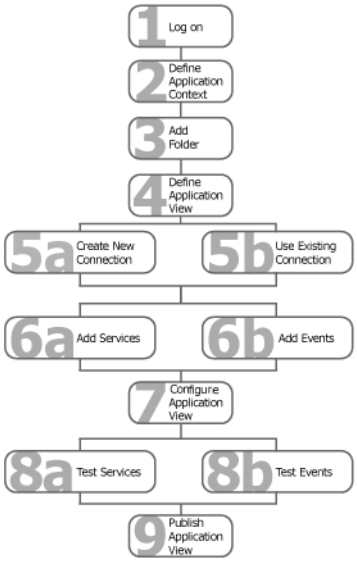
About Defining Application Views

Defining an application view is a multi-step process described in *Using the Application Integration Design Console*, available at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

The information you enter depends on the requirements of your business process and your EIS system configuration. Figure 2-2 summarizes the procedure for defining and configuring an application view.

Figure 2-2 Process for Defining and Configuring an Application View



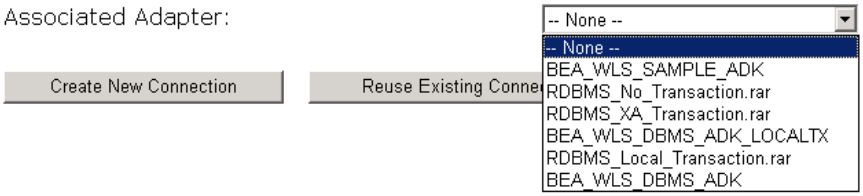
To define an application view:

1. Log on to the WebLogic Integration Application View Console.
2. Define the application context by selecting an existing application or specifying a new application name and root directory.

   This application will be using the events and services you define in your application view. The application view works within the context of this application.
3. Add folders as required to help you organize application views.
4. Define a new application view for your adapter.

Note: The Adapter for RDBMS supports the following transaction levels: No Transaction, Local Transaction, and XA Transaction. Each transaction level exposes a connector. When you define an application view, you must choose a connector (`RAR` file). The file you choose depends on your integration requirements. You must then select an appropriate driver to connect to the RDBMS.

The following figure displays an example menu of connectors for different transaction levels, which are available when you define an application view:



For details on how to choose connectors, see the following table.

Table 2-1 Choosing Connectors Based on the Required Transaction Support Level

Transaction Support Level	Description	Choose the Connector...	Choose Appropriate Driver...
No Transaction	The adapter operates in auto-commit mode, that is, every service is committed after it is successfully invoked.	`RDBMS_No_Transaction.rar`	Non-XA
Local Transaction	The container manages the transaction for the adapter. The transaction demarcation is the business process rather than individual services.	`RDBMS_Local_Transaction.rar`	Non-XA
XA Transaction	The adapter takes part in an XA transaction. Use this when you want XA behavior.	`RDBMS_XA_Transaction.rar`	XA for the database

5. Add a new connection service or select an existing one.

   If you are adding a new connection service, see "Defining Service Connection Parameters" on page 2-6 for details about RDBMS requirements.

6. Add the events and services for this application view.

   See the following sections for details about RDBMS requirements:

   – "Setting Service Properties" on page 2-7

– "Setting Event Properties" on page 2-13

7. Perform final configuration tasks.

   If you are adding an event connection, see "Defining Event Connection Parameters" on page 2-17 for details about RDBMS requirements.

8. Test all services and events to make sure they can properly interact with the target RDBMS system.

   See the following sections for details about RDBMS requirements:

   – "Testing Services" on page 2-21

   – "Testing Events Using a Service" on page 2-21

   – "Testing Events Manually" on page 2-22

9. Publish the application view to the target WebLogic Workshop application.

   This is the application you specified in step 2. Publishing the application view allows business process developers within the target application to interact with the newly published application view using an Application View control.

Defining Service Connection Parameters

This information applies to "Step 5A, Create a New Browsing Connection" in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

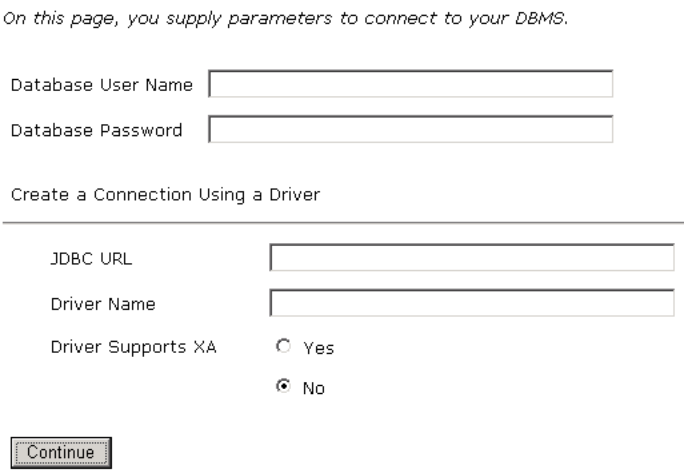
After you enter a connection name and description, you use the Configure Connection Parameters page to specify connection parameters for a connection factory.

To configure connection parameters:

1. In the Create New Browsing Connection page, enter a connection name and description as described in *Using the Application Integration Design Console*.

2. Click Define.

   The Configure Connection Parameters page appears, to allow you to configure the newly created connection factory within the new adapter instance.



Note: A red asterisk (*) indicates that a field is required.

3. Enter your database user name and password.
4. Enter the JDBC URL and the driver name. Select the appropriate Driver Supports XA setting. Click Continue.

   You return to the Create New Browsing Connections, where you can specify connection pool parameters and logging levels. For more information, see *Using the Application Integration Design Console* at the following URL:

   `http://edocs.bea.com/wli/docs81/aiuser/index.html`

Setting Service Properties

1 2 3 4 5 **6** 7 8 9

This information applies to "Step 6A, Add a Service to an Application View" in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

The Adapter for RDBMS uses services to make requests of the RDBMS system. A service consists of both a request and a response.

The Adapter for RDBMS supports the following services:

- Standard SQL: This service invokes a static standard SQL statement without any parameters. You give the SQL statement, without parameters, at design time. The SQL

executed can be either a Data Manipulation Language (DML) - SELECT, INSERT, UPDATE, or DELETE statement or a Data Definition Language (DDL) - CREATE, ALTER, or DROP statement. The list of resultant records is converted into a response XML document, based on the XML schema formed during design time.

This service type does not support Insert/Update statements containing Character Large Objects (`CLOB`) or Binary Large Objects (`BLOB`) data types.

- Parametrized SQL: This service invokes a parametrized SQL. You give the SQL statement, with parameters, at design time. The SQL executed can only be a DML (SELECT, INSERT, UPDATE, or DELETE) statement. This service type receives an XML document that gives the values for the parameters specified. Based on the data type of the input provided, the adapter binds the parameter values. The prepared SQL statement is executed, and the list of resultant records is converted into a response XML document, based on the XML schema formed during design time.

  This service type supports statements containing `CLOB/BLOB` data type. Such parameters should be base64 encoded. The adapter decodes the input and binds it to the statement before execution. The output from a table with a column of the type `CLOB/BLOB` will be encoded to base64 before forming the output XML document.

- Stored Procedure: This service invokes a stored procedure defined in an RDBMS. The stored procedure executed can have `IN`, `IN-OUT`, and `OUT` parameters and a `RETURN` value. This service type receives an XML document giving the input values for the parameters of the type `IN` and `IN-OUT`. Based on the data type of the input provided, the adapter binds the parameter values. The stored procedure is executed and the list of resultant records is converted into a response XML document, based on the XML schema formed during design time.

  Note:

  - Because DBMS manufacturers produce JDBC agents with varying degrees of functionality, stored procedures can be executed in several different ways within the service adapter.
  - Oracle subprograms (stored procedures) can be called whether they reside as functions, stored procedures, or stored procedures located within packages.
  - For this service type, Oracle JDBC drivers do not support `boolean` data type.
  - Informix stored procedures do not have `IN-OUT` parameters.

- Arbitrary SQL: This service invokes an SQL statement. You give the SQL statement, with parameters, at run time. The SQL executed can either be a DML (SELECT, INSERT, UPDATE, or DELETE) or a DDL (CREATE, ALTER, or DROP) statement. The list of

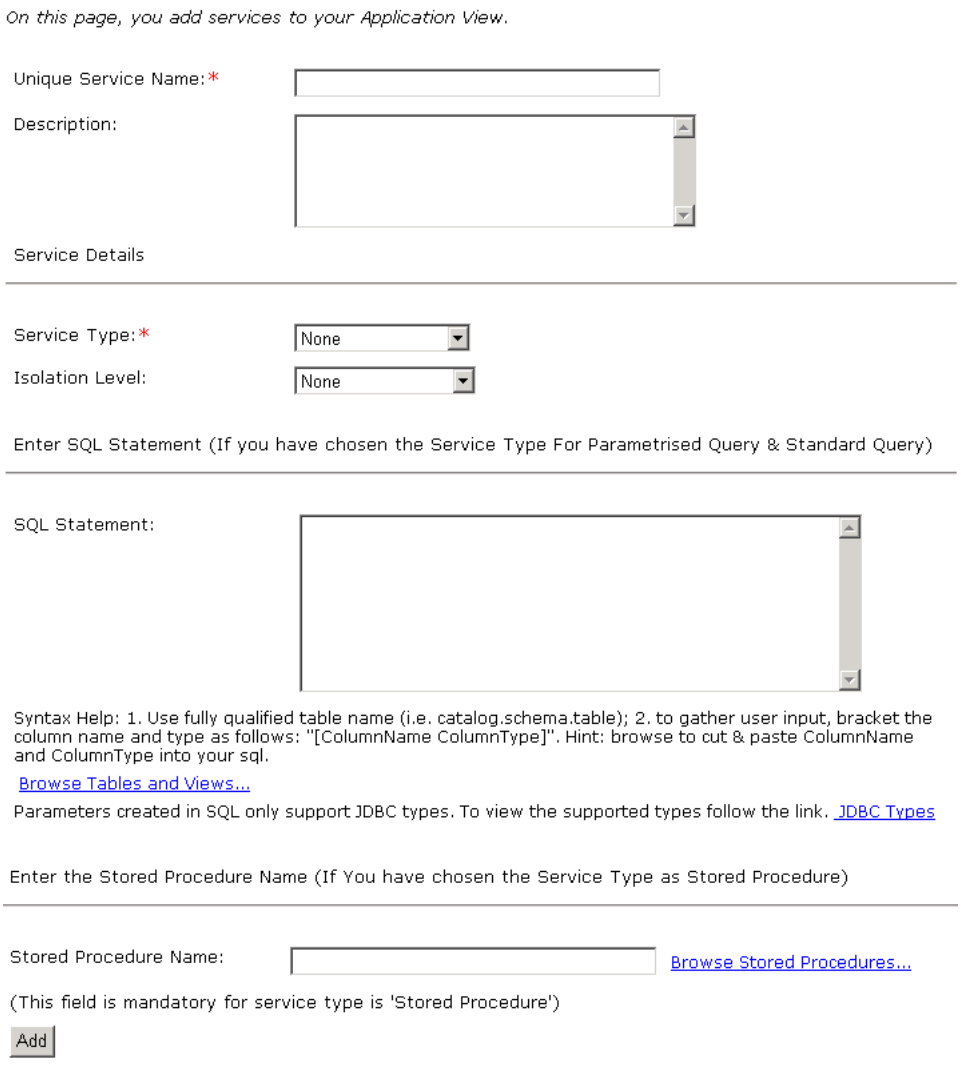
resultant records is converted into the response XML document, based on the type of statement executed.

Note: In Standard SQL, the statement is given at design time, whereas in Arbitrary SQL, it is given at run time. Otherwise, the same rules apply.

To configure a service:

1. Enter a unique service name that describes the function the service performs and a description.

   The service name must be unique to its application view. Valid characters include a-z, A-Z, 0-9, and the _ (underscore) character.



Note: A red asterisk (*) indicates that a field is required.

2. Select the Service Type from the Service Type list. You can select one of the following:
   - Standard SQL, which invokes a static standard SQL statement without any parameters for the statement defined at design time
   - Parametrized SQL, which invokes a parametrized SQL statement for the statement defined at design time
   - Stored Procedure, which invokes a stored procedure defined in the RDBMS
   - Arbitrary SQL, which invokes an SQL statement for the statement defined at run time
3. Select the Isolation Level.

Table 2-2 Isolation Level Settings

Isolation Level	Description
None	Does not specifically set an isolation level for the connection. If you select None, the default setting for the driver is taken.
Read Committed	Limits a transaction to reading only data that has already been committed by other transactions. Prohibits a transaction from reading a row with uncommitted changes in it.
Read Uncommitted	Permits a transaction to read data that has been updated but not yet committed by other transactions. Does not prevent read violations.
Repeatable Read	Permits a transaction to read only data that has been committed by another transaction. Multiple reads of the same data yield the same committed result within a transaction. Uncommitted data is not read.
Serializable	Requires transactions to be run serially to achieve maximum data integrity. Yields the slowest performance and least concurrency.

Note: The transaction isolation levels should be same for all the services participating in a transaction.

4. Enter the SQL statement for Standard SQL or Parametrized SQL services.

   For both service types, the table name should be fully qualified with the SCHEMA/CATALOG name as the database requires. The table name is case sensitive. In addition, Oracle requires the use of all upper-case letters in table names. For example, in Oracle, it should be `SCHEMA.TABLENAME`.

To gather user input for Parametrized SQL, enter the column name and column type within brackets. For example, `[ParamName ColumnType]`. The `ParamName` is used to define an element in the input schema. This name should be unique in the parametrized SQL statement. The `ColumnType` should be a JDBC data type. To see the list of supported types, click the JDBC Types link.To see the list of data types tested and supported by this adapter on various RDBMS, see Appendix B, "Supported Data Types."

Examples of SQL statements:

- Standard SQL: `SELECT * FROM BEAUSER.BONUS`
- Parametrized SQL: `SELECT * FROM BEAUSER.BONUS WHERE SAL>[SALARY INTEGER]`

5. Enter the Stored Procedure Name for Stored Procedures service.

   The stored procedure name should be in the format `ProcedureName(?, ?, ...)`. For example, `BEAUSER.RAISESAL(?, ?)`

   The ?s indicate the number of `IN`, `IN-OUT`, and `OUT` parameters. The ProcedureName must be qualified with the catalog and/or schema name as per the requirement of the database.

Table 2-3 Stored Procedure Name Format for Databases

RDBMS	Stored Procedure Name Format
Oracle	• For a packaged procedure, *SchemaName*.*PackageName*.*ProcedureName*(?, ?, ...) • For a non-packaged procedure, *SchemaName*.*ProcedureName*(?, ?, ...)
SQL Server	*CatalogName*.*SchemaName*.*ProcedureName*(?, ?, ...)
DB2 UDB	*SchemaName*.*ProcedureName*(?, ?, ...)
Informix Dynamic Server	*SchemaName*.*ProcedureName*(?, ?, ...);*ProcedureSignature* **Note:** Because you can overload stored procedures in Informix, the procedure signature is included in the stored procedure name format. However, you do not have to enter the signature for the procedure in this field. It will be appended automatically.
Sybase Adaptive Server	*DatabaseName*.*SchemaName*.*ProcedureName*(?, ?, ...)

Note: Click the Browse DBMS to view Tables, Views, and Stored Procedures link to select the stored procedure name. You can click the option button next to the stored procedure name to fill it in the field.

6. When finished, click Add. The service is added to the list.

For examples of design-time and run-time input values for services, see Appendix A, "Service and Event Examples."

Setting Event Properties

1 2 3 4 5 **6** 7 8 9

This information applies to "Step 6B, Add an Event to an Application View" in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

An event defines how your application responds to events generated by the RDBMS. The Adapter for RDBMS supports the following event types:

- Trigger: This event type notifies an Insert, Update, or Delete event occurring in a database table using the trigger mechanism.

  Note: For Sybase and Informix, the table should contain a primary key.

- Select By Timestamp: This event type notifies an Insert and Update event occurring in a database table. The table of interest should have columns that represent Created Timestamp and Updated Timestamp. You can configure Insert event only if Created Timestamp is maintained as a part of the table, and Update event only if Updated Timestamp is maintained as a part of the table.

  Note: Ensure that the system time of the machine running the EIS is in sync with the system time of the machine running the WebLogic Server, so that the events do not time out.

- Select Then Delete: This event type notifies records of interest based on a select query given on a database table. The event polling mechanism looks for records based on the select query, packages the result into one or more rows, and creates an XML document representing the records of interest. You can choose to either delete or retain the selected records after the events are processed.

  Note: For Informix, the table should contain a primary key and the Select statement should retrieve the primary key column.

The following table shows the events that each of the event types supports.

Table 2-4 Supported Events for Event Types

	Insert	Update	Delete
Trigger	Yes	Yes	Yes
Select By Timestamp	Yes	Yes	No
Select Then Delete	N/A	N/A	N/A

The application view publishes the event as an XML document during the specified occurrence, acting as a broker between the target application and the client.

Before you configure an event, ensure the following tables are created:

- EventPollingMetadataTable: This table stores the last polled timestamp for the application view and the event. The last polled timestamp is used to construct the next event payload for a Select By Timestamp event type.

  For example: If you specify a polling interval of 4 seconds, and the last polled timestamp is T1, the event generator scans the target table for any row whose timestamp column T satisfies the following condition: T1 < T <= T1 + 4.

- Trigger Tables: These tables are used when you configure a Trigger event. When an Insert/Update/Delete takes place in the target table, a trigger is invoked. This trigger creates entries in the event and event data tables, which contain event and event data respectively. The event generator then scans these tables for rows that indicate event occurrences. After the event is successfully invoked, the respective rows are deleted from the event and event data tables.

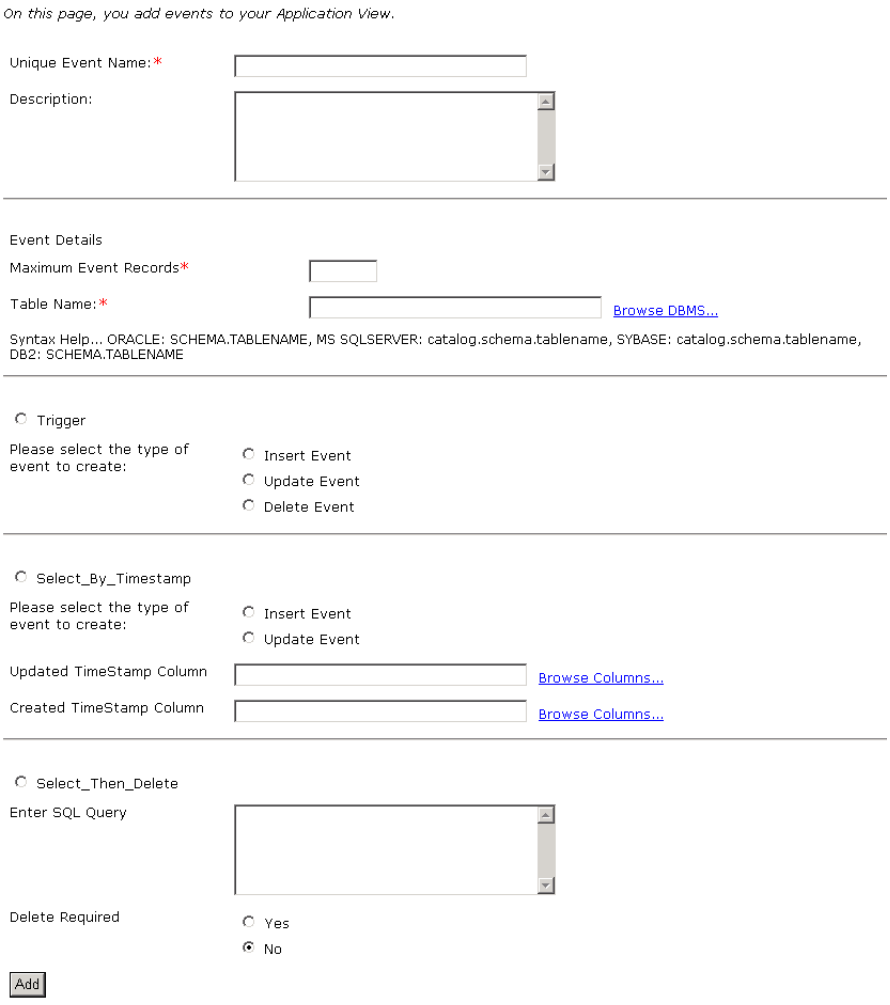
For details on how to set up and run the EventPollingMetadataTable and Trigger Tables, see the *BEA WebLogic Adapter for RDBMS Installation and Configuration Guide* at the following URL:

`http://edocs.bea.com/wladapters/docs81/index.html`

To configure an event:

1. Enter a unique event name that describes the function the event performs and a description.

   Each event name must be unique to its application view. Valid characters include a-z, A-Z, 0-9, and the _ (underscore) character.



Note: A red asterisk (*) indicates that a field is required.

2. Enter the maximum number of data records to be processed as an event.

   For example, if there are 10 records of interest, and the Maximum Event Records is 3, there will be 3 events with 3 records each, and an event with the remaining record. If there are 2 records of interest, and the Maximum Event Records is 3, there will still be an event with 2 records.

3. Enter the name of the database table.

   Use the corresponding syntax for the following databases:

Table 2-5 Syntax for Database Table Names

Database	Syntax for Table Name
Oracle	`SCHEMA.TABLENAME`
SQL Server	`catalog.schema.tablename`
DB2 UDB	`SCHEMA.TABLENAME`
Informix Dynamic Server	`SCHEMA.TABLENAME`
Sybase Adaptive Server	`catalog.schema.tablename`

Note: Click the Browse DBMS link to see the schemas and table names. You can click the option button next to the table name, and then click Fill Table Name for the selected table.

4. Select the option button for the type of event to add:
   - Trigger, which notifies an Insert, Update, or Delete event occurring in a database table
   - Select_By_Timestamp, which notifies an Insert or Update event in a database table with the corresponding timestamp columns
   - Select_Then_Delete, which notifies records of interest based on a select query given on a database table
5. Do one of the following, based on the type of event:
   - For a Trigger event, select one of the following actions: Insert Event, Update Event, Delete Event.

– For a Select_By_Timestamp event, select Insert Event or Update Event. For Insert Event, enter the Created Timestamp Column name. For Update Event, enter the Updated Timestamp Column name.

Note: Click the Browse Columns link to see the column names. You can click the option button next to the column name, and then click Fill Column Name to fill in the field.

– For a Select_Then_Delete event, enter the SQL Query. Select the Delete Required Yes option button to delete the selected records after the event is triggered.

6. When finished, click Add. The event is added to the list.

Defining Event Connection Parameters

1 2 3 4 5 6 7 8 9

This information applies to "Step 7, Perform Final Configuration Tasks" in *Using the Application Integration Design Console*, at the following URL:

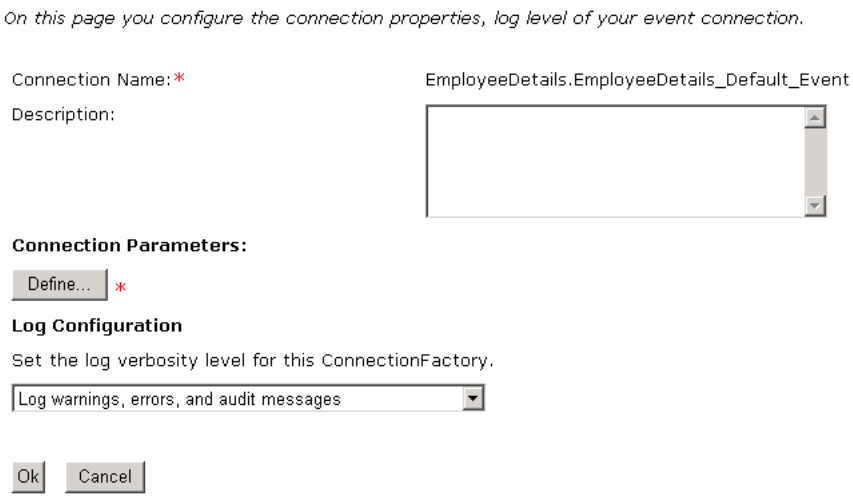
`http://edocs.bea.com/wli/docs81/aiuser/index.html`

Once you have finished adding services and events and have saved your application view, you must perform some final configuration tasks, including configuring event-delivery connections, before testing the services and events. Perform these configuration tasks from the Final Configuration and Testing page.

To define event connection parameters:

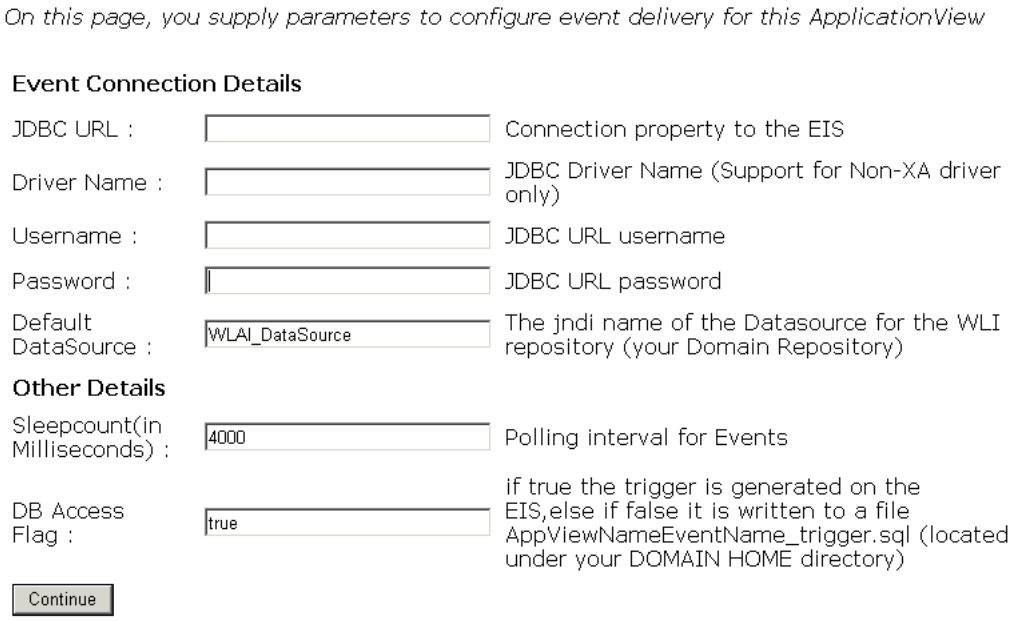
1. In the Connections area on the Application View Administration page, click Select/Edit.
2. In the Event Connection area, click the Select Existing link, then click Event in the Event Connection page to edit the default event connection.

The following page is displayed.



3. Click Define to enter the event-delivery parameters.

   The parameters you enter on this page enable connection to your RDBMS and are used when generating events. The parameters are specific to the associated adapter and are defined in the `weblogic-ra.xml` file within the base adapter.



4. Enter the JDBC URL and Driver Name required to connect to the database.
5. Enter the Username and Password required to connect to the database.
6. Enter the Default DataSource, the JNDI name of the Weblogic Integration domain datasource.

   This is the datasource that contains the EventPollingMetadata table created while you installed the adapter. The default is `WLAI_DataSource`.
7. Enter the Sleepcount, the number of seconds the adapter will wait between polling for events.

   The default is 4000 milliseconds, which means that the database is polled every four seconds for new events.
8. For Trigger event, enter True for the DB Access Flag if you have access to your customer's database. Otherwise, enter False.

   If true, the trigger is created on the database. If false, it is written to the file `DOMAIN_HOME\AppViewNameEventName_trigger.sql`. The default is True.

9. Click Continue to return to the Edit Event Connection page, and then click OK to return to the Final Configuration and Testing page.

   The Edit Event Connection page allows you to define event parameters and configure the information that will be logged for the connection factory. Select one of the following settings for the log:

   – Log errors and audit messages (recommended)

   – Log warnings, errors, and audit messages

   – Log informational, warning, error, and audit messages

   – Log all messages

   The table that follows describes the type of information that each logging message contains.

Table 2-6 Logging Message Categories

This type of message	Contains
Audit	Extremely important information related to the business processing performed by an adapter.
Error	Information about an error that has occurred in the adapter, which may affect system stability.
Warning	Information about a suspicious situation that has occurred. Although this is not an error, it could have an impact on adapter operation.
Information	Information about normal adapter operations.

Testing Services

This information applies to "Step 8A, Test an Application View's Services" in *Using the Application Integration Design Console*, at the following URL:

http://edocs.bea.com/wli/docs81/aiuser/index.html

The purpose of testing an application view service is to evaluate whether that service interacts properly with the target RDBMS. When you test a service, you supply any inputs required to start the service. For the Adapter for RDBMS, the input is in the form of a valid SQL statement that acts as input for the service.

Note: You can test an application view only if it is in Test mode and only if it contains at least one event or service.

To test a service:

1. In the Application View Administration page, click the Test link beside the service to be tested.

   The Test Services page appears.

2. In the Test Service window, enter the appropriate SQL statement to execute the service.

3. Click Test.

   The results appear in the Test Results window.

Testing Events Using a Service

This information applies to "Step 8B, Test an Application View's Events" in *Using the Application Integration Design Console*, at the following URL:

http://edocs.bea.com/wli/docs81/aiuser/index.html

The purpose of testing an application view event is to make sure that the adapter correctly handles events generated by the RDBMS. When you test an event, you can trigger the event using a service or manually.

Note: You can test an application view only if it is in Test mode and only if it contains at least one event or service.

To test an event:

1. In the Application View Administration page, click the Test link beside the event to be tested.

   The Test Events page appears.

2. Click Service and select a service that triggers the event you are testing.

3. In the Time field, enter a reasonable period of time to wait, specified in milliseconds, before the test times out (One second = 1000 milliseconds. One minute = 60,000 milliseconds.).

4. Click Test and enter the SQL statement needed to trigger the service.

   The service is executed.

   – If the test succeeds, the Test Result page displays the event document, the service input document, and the service output document.

   – If the event is not available within the specified time limit, the Test Result page displays only a Timed Out message.

Testing Events Manually

This information applies to "Step 8B, Test an Application View's Events" in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

To test an event manually:

1. In the Time field, enter a reasonable period of time to wait, specified in milliseconds, before the test times out (One second = 1000 milliseconds. One minute = 60,000 milliseconds.).

2. Click Test. The test waits for an event to trigger it.

3. Using the triggering RDBMS application, perform an action that executes the service that, in turn, tests the application view event.

   – If the test succeeds, the Test Result page displays the event document from the application.

   – If the test fails or takes too long, the Test Result page shows a Timed Out message.

Handling Null Values

RDBMS support null values in a data field. In services, you should properly set null values when performing an insert or update through a parametrized query or giving input to a stored procedure. In events, you may need to ascertain whether a field in the output contains a null value, or is simply an empty string.

Null Values in Services

The BEA WebLogic Adapter for RDBMS propagates null values into `SQL` through the following code.

```
<?xml version="1.0"?>
<ns:Input xmlns:ns="wlai/Test_Parametrized_request">
<ns:INPUT_VALUE isNull="true"></ns:INPUT_VALUE>
</ns:Input>
```

Here, the application view has a service that is defined with `Target_Nodes` set to `/INPUT_VALUE/` and `SQL` set to `INSERT INTO ANOTHER_TABLE VALUES([INPUT_VALUE VARCHAR])`

The resulting `SQL` is sent to the RDBMS is `INSERT INTO ANOTHER_TABLE VALUES(NULL)`

Null Values in Events

The BEA WebLogic Adapter for RDBMS handles null values for events as shown in the following table.

Table 2-7 Handling Null Values in Events

Database Value	Database Nullable	XML Value	XML Attribute
Spaces	Yes	Spaces	`isNull='false'`
Spaces	No	Spaces	N/A
Null	Yes	None	`isNull='true'`
Null	No	N/A	N/A
Text	Yes	Text	`isNull='false'`
Text	No	Text	N/A

For the Field and Column formats, the attribute `isNull` is set to `'true'` to indicate a nullable field containing a null value. The following is an example of XML, in the Field format, containing fields representing the valid combinations from the above table.

```
<?xml version="1.0"?>
<Rows xmlns="wlai/Test_BASE" xmlns:ns="wlai/Test_BASE">
<ns:Row>
<ns:COMM isNull="false">88</ns:COMM>
<ns:ENAME isNull="false">name</ns:ENAME>
<ns:JOB isNull="false"> </ns:JOB>
<ns:SAL isNull="false">9000</ns:SAL>
</ns:Row>
<ns:Row>
<ns:COMM isNull="false">44</ns:COMM>
<ns:ENAME> </ns:ENAME>
<ns:JOB isNull="false">program</ns:JOB>
<ns:SAL isNull="true"></ns:SAL>
</ns:Row>
</ns:Rows>
```

Service and Event Examples

At run-time, the EIS and the adapter exchange service requests, service responses, and events via XML documents. The adapter handles the data translation between XML documents and the EIS format, using *schemas* that map the data between XML and the EIS format. The Adapter for RDBMS automatically generates the schema when you add services and events to an application view. You can click the corresponding link on the Application View Administration page to view the automatically generated schemas.

This section gives examples of input values, input XML, and/or output XML for the services and events that the adapter offers.

The examples given assume the EIS to be MS SQL 2000 with the following schema:

```
CREATE TABLE adaptertesting.dbo.base (
       CHAR1 CHAR (10),
       VARCHAR1 VARCHAR (10),
       NUMERIC1 NUMERIC(6, 0),
       INTEGER1 INT,
       SMALLINT1 SMALLINT,
       REAL1 REAL,
       FLOAT1 FLOAT,
       DOUBLE1 DOUBLE PRECISION,
       DECIMAL1 DECIMAL(8, 2),
       CREATEDT DATETIME,
       MODDT DATETIME
)
```

where `adaptertesting` is the catalog name, `dbo` is the schema name, and `base` is the table name.

The stored procedure used is:

```
CREATE PROCEDURE PROCCHAR (@X CHAR(10),@Y  char(10) output,@Z char(10)
output) as BEGIN INSERT INTO BASE (CHAR1) VALUES (@X); SET @Y = 'THIS IS Y';
SET @Z = 'THIS IS @Z'; END
```

Services

Standard SQL

The following section gives sample input values and input and output XML for the Standard SQL service.

Design-Time Input

Standard SQL Statement: `SELECT * FROM adaptertesting.dbo.base`

Run-Time Input

None

Input XML

```
<?xml version="1.0"?>
<ns:Input xmlns:ns="wlai/Test_Standard_request"/>
```

Output XML

```
<?xml version="1.0"?>
<ns0:Output xmlns:ns0="wlai/Test_Standard_response">
<ns0:Rows>
<ns0:Row>
<ns0:Char1 isNull="false">A</ns0:Char1>
<ns0:CreateDt isNull="false">2003-02-02T00:00:00+05:30</ns0:CreateDt>
<ns0:Decimal1 isNull="false">1.00</ns0:Decimal1>
<ns0:Double1 isNull="false">1.0</ns0:Double1>
<ns0:Float1 isNull="false">1.0</ns0:Float1>
<ns0:Integer1 isNull="false">1</ns0:Integer1>
```

```
<ns0:ModDt
isNull="false">1998-10-22T00:00:00+05:30</ns0:ModDt>
<ns0:Numeric1 isNull="false">1</ns0:Numeric1>
<ns0:Real1 isNull="false">1.0</ns0:Real1>
<ns0:Smallint1 isNull="false">1</ns0:Smallint1>
<ns0:Varchar1 isNull="false">A</ns0:Varchar1>
</ns0:Row>
</ns0:Rows>
</ns0:Output>
```

Parametrized SQL

The following section gives sample input values and input and output XML for the Parametrized SQL service. It also explains why `CLOB/BLOB` datatypes require base64 encoding.

Design-Time Input

Parametrized SQL Statement:

```
SELECT * FROM adaptertesting.dbo.base WHERE
CHAR1=[CHAR1 CHAR]
AND VARCHAR1=[VARCHAR1 VARCHAR]
AND NUMERIC1=[NUMERIC1 NUMERIC]
AND INTEGER1=[INTEGER1 INTEGER]
AND SMALLINT1=[SMALLINT1 SMALLINT]
AND REAL1=[REAL1 REAL]
AND FLOAT1=[FLOAT1 FLOAT]
AND DOUBLE1=[DOUBLE1 DOUBLE]
AND DECIMAL1=[DECIMAL1 DECIMAL]
AND CREATEDT=[CREATEDT DATE]
AND MODDT=[MODDT DATE]
```

Run-Time Input

Parameter Values: `A , A , 1 , 1 , 1 , 1.0 , 1.0 , 1.0 , 1.0 , 2003-02-02 , 1998-10-22`

Input XML

```
<?xml version="1.0"?>
<ns:Input xmlns:ns="wlai/Test_Parametrized_request">
```

```
<ns:CHAR1 isNull="false">A</ns:CHAR1>
<ns:VARCHAR1 isNull="false">A</ns:VARCHAR1>
<ns:NUMERIC1 isNull="false">1</ns:NUMERIC1>
<ns:INTEGER1 isNull="false">1</ns:INTEGER1>
<ns:SMALLINT1 isNull="false">1</ns:SMALLINT1>
<ns:REAL1 isNull="false">1</ns:REAL1>
<ns:FLOAT1 isNull="false">1</ns:FLOAT1>
<ns:DOUBLE1 isNull="false">1</ns:DOUBLE1>
<ns:DECIMAL1 isNull="false">1</ns:DECIMAL1>
<ns:CREATEDT isNull="false">2003-02-02</ns:CREATEDT>
<ns:MODDT isNull="false">1998-10-22</ns:MODDT>
</ns:Input>
```

Output XML

```
<?xml version="1.0"?>
<ns0:Output xmlns:ns0="wlai/Test_Parametrized_response">
<ns0:Rows>
<ns0:Row>
<ns0:Char1 isNull="false">A</ns0:Char1>
<ns0:CreateDt isNull="false">2003-02-02T00:00:00+05:30</ns0:CreateDt>
<ns0:Decimal1 isNull="false">1.00</ns0:Decimal1>
<ns0:Double1 isNull="false">1.0</ns0:Double1>
<ns0:Float1 isNull="false">1.0</ns0:Float1>
<ns0:Integer1 isNull="false">1</ns0:Integer1>
<ns0:ModDt isNull="false">1998-10-22T00:00:00+05:30</ns0:ModDt>
<ns0:Numeric1 isNull="false">1</ns0:Numeric1>
<ns0:Real1 isNull="false">1.0</ns0:Real1>
<ns0:Smallint1 isNull="false">1</ns0:Smallint1>
<ns0:Varchar1 isNull="false">A</ns0:Varchar1>
</ns0:Row>
</ns0:Rows>
</ns0:Output>
```

Why Use Base64 Encoding for CLOB/BLOB Datatypes?

Base64 encoding is most commonly used to encode binary and text data that may contain non-ASCII characters, or characters that are considered special or reserved characters, so that this information can be safely transmitted across the Internet. All binary representations, including 0x00 - 0xFF, can be encoded in

Base64. It is particularly useful to users of XML, because the information one wishes to enclose within these documents may contain reserved or special characters.

CLOB Datatype: Sample Java Code Snippet to Encode to Base64

```
String str = new String("Sample String");
byte[] bytes = str.getBytes();
String strToSendInXml = new sun.misc.BASE64Encoder().encode(bytes);
```

CLOB Datatype: Sample Java Code Snippet to Decode from Base64

```
bytes = new sun.misc.BASE64Decoder().decodeBuffer(stringFromXml);
String decodedStr = new String(bytes);
```

where `//stringFromXml` is the Base64 encoded string obtained from the response document.

BLOB Datatype: Sample Java Code Snippet to Encode to Base64

```
Integer intToSend = new Integer(95);
ByteArrayOutputStream bos = new ByteArrayOutputStream();
ObjectOutputStream os = new ObjectOutputStream(bos);
os.writeObject(intToSend);
os.flush();
String sendThisInXml = new
sun.misc.BASE64Encoder().encode(bos.toByteArray());
```

where `//intToSend` is the Integer object to be sent to the RDBMS as a `BLOB` datatype.

BLOB Datatype: Sample Java Code Snippet to Decode from Base64

```
byte[] fromXml=new sun.misc.BASE64Decoder().decodeBuffer(stringFromXml);
ByteArrayInputStream bytestream = new ByteArrayInputStream(fromXml);
ObjectInputStream is = new ObjectInputStream(bytestream);
Integer originalInt = (Integer)is.readObject();
ois.close();
```

where `//stringFromXml` is the Base64 encoded string obtained from the response document.

Stored Procedure

The following sections give sample input values and input and output XML for the Stored Procedure service. The first section shows an example for Oracle, Microsoft SQL Server, DB2, and Sybase stored procedures. The second section shows an example for Informix stored procedures.

Note: Since Informix stored procedures can have multiple return values, the `response.xml` and schemas are slightly different than those for other databases.

Example for Oracle, Microsoft SQL Server, DB2, and Sybase Stored Procedures

Design-Time Input

Procedure Name: `AdapterTesting.dbo.PROCCHAR(?, ?, ?)`

where the `?`s represent the `IN`, `IN-OUT`, and `OUT` parameters respectively.

Run-Time Input

```
STRING1, STRING2, AND STRING3
```

Input XML

```
<?xml version="1.0"?>
<ns:Input xmlns:ns="wlai/Test_StoredProcedure_request">
<ns:X isNull="false">String1</ns:X>
<ns:Y isNull="false">String2</ns:Y>
<ns:Z isNull="false">String3</ns:Z>
</ns:Input>
```

Output XML

```
<?xml version="1.0"?>
<ns0:Output xmlns:ns0="wlai/Test_StoredProcedure_response">
<ns0:ReturnValue>0</ns0:ReturnValue>
<ns0:Params>
<ns0:Y isNull="false">THIS IS Y </ns0:Y>
<ns0:Z isNull="false">THIS IS @Z</ns0:Z>
</ns0:Params>
</ns0:Output>
```

Example for Informix Stored Procedures

Design-Time Input

Procedure Name: `informix.procout(?);create function informix.procout(integer) returning integer,date;`

where the `?`s represent the `IN` and `OUT` parameters respectively.

Run-Time Input

```
5
```

Input XML

```
<?xml version="1.0"?>

<ns:Input xmlns:ns="wlai/Informix_SP_request">

  <ns:Param_1>5</ns:Param_1>

</ns:Input>
```

Output XML

```
<?xml version="1.0"?>

<ns0:Output xmlns:ns0="wlai/Informix_SP_response">

  <ns0:ReturnValues>

    <ns0:ReturnValue>

      <ns0:ReturnValue_1>1</ns0:ReturnValue_1>

      <ns0:ReturnValue_2>2007-07-07+05:30</ns0:ReturnValue_2>

    </ns0:ReturnValue>

  </ns0:ReturnValues>

  <ns0:Params/>

</ns0:Output>
```

Arbitrary SQL

The following section gives sample input values and input and output XML for the Arbitrary SQL service.

Design-Time Input

None

Run-Time Input

SQL Statement: `SELECT * FROM adaptertesting.dbo.base`

Input XML

```
<?xml version="1.0"?>
<ns:Input xmlns:ns="wlai/Test_Arbitrary_request">
<ns:Sql>SELECT * FROM adaptertesting.dbo.base</ns:Sql>
</ns:Input>
```

Output XML

```
<?xml version="1.0"?>
<ns0:Output xmlns:ns0="wlai/Test_Arbitrary_response">
<ns0:Rows>
<ns0:Row>
<ns0:Char1 isNull="false">A</ns0:Char1>
<ns0:Varchar1 isNull="false">A</ns0:Varchar1>
<ns0:Numeric1 isNull="false">1</ns0:Numeric1>
<ns0:Integer1 isNull="false">1</ns0:Integer1>
<ns0:Smallint1 isNull="false">1</ns0:Smallint1>
<ns0:Real1 isNull="false">1.0</ns0:Real1>
<ns0:Float1 isNull="false">1.0</ns0:Float1>
<ns0:Double1 isNull="false">1.0</ns0:Double1>
<ns0:Decimal1 isNull="false">1.00</ns0:Decimal1>
<ns0:CreateDt isNull="false">2003-02-02T00:00:00+05:30</ns0:CreateDt>
<ns0:ModDt isNull="false">1998-10-22T00:00:00+05:30</ns0:ModDt>
</ns0:Row>
</ns0:Rows>
</ns0:Output>
```

Events

Trigger Insert

The following section gives sample input values and output XML for the Trigger Insert event.

Design-Time Input

Maximum Event Records: 5

Table Name: `adaptertesting.dbo.base`

Event: Insert

Run-Time Input

Time (in milliseconds): 20000

Trigger method: Manual or through a service

Output XML

```
<?xml version="1.0"?>
<Rows xmlns="wlai/TriggerInsert_BASE_insert"
xmlns:ns="wlai/TriggerInsert_BASE_insert">
<ns:Row>
<ns:Char1 isNull="false">STRING1   </ns:Char1>
<ns:CreateDt isNull="false">2003-07-16T15:21:02+05:30</ns:CreateDt>
<ns:Decimal1 isNull="false">1</ns:Decimal1>
<ns:Double1 isNull="false">1</ns:Double1>
<ns:Float1 isNull="false">1</ns:Float1>
<ns:Integer1 isNull="false">1</ns:Integer1>
<ns:ModDt isNull="false">2003-07-16T15:21:02+05:30</ns:ModDt>
<ns:Numeric1 isNull="false">1</ns:Numeric1>
<ns:Real1 isNull="false">1</ns:Real1>
<ns:Smallint1 isNull="false">1</ns:Smallint1>
<ns:Varchar1 isNull="false">STRING2</ns:Varchar1>
</ns:Row>
</Rows>
```

Trigger Update

The following section gives sample input values and output XML for the Trigger Update event.

Design-Time Input

Maximum Event Records: 5

Table Name: `adaptertesting.dbo.base`

Event: Update

Run-Time Input

Time (in milliseconds): 20000

Trigger method: Manual or through a service

Output XML

```
<?xml version="1.0"?>
<Rows xmlns="wlai/TriggerUpdate_BASE_update"
xmlns:ns="wlai/TriggerUpdate_BASE_update">
<ns:Row>
<ns:Old>
<ns:Char1 isNull="false">A</ns:Char1>
<ns:CreateDt isNull="false">2003-02-02T00:00:00+05:30</ns:CreateDt>
<ns:Decimal1 isNull="false">1</ns:Decimal1>
<ns:Double1 isNull="false">1</ns:Double1>
<ns:Float1 isNull="false">1</ns:Float1>
<ns:Integer1 isNull="false">1</ns:Integer1>
<ns:ModDt isNull="false">1998-10-22T00:00:00+05:30</ns:ModDt>
<ns:Numeric1 isNull="false">1</ns:Numeric1>
<ns:Real1 isNull="false">1</ns:Real1>
<ns:Smallint1 isNull="false">1</ns:Smallint1>
<ns:Varchar1 isNull="false">A</ns:Varchar1>
</ns:Old>
<ns:New>
<ns:Char1 isNull="false">STRING1   </ns:Char1>
<ns:CreateDt
isNull="false">2003-02-02T00:00:00+05:30</ns:CreateDt>
<ns:Decimal1 isNull="false">1</ns:Decimal1>
```

```
<ns:Double1 isNull="false">1</ns:Double1>
<ns:Float1 isNull="false">1</ns:Float1>
<ns:Integer1 isNull="false">1</ns:Integer1>
<ns:ModDt
isNull="false">1998-10-22T00:00:00+05:30</ns:ModDt>
<ns:Numeric1 isNull="false">1</ns:Numeric1>
<ns:Real1 isNull="false">1</ns:Real1>
<ns:Smallint1 isNull="false">1</ns:Smallint1>
<ns:Varchar1 isNull="false">A</ns:Varchar1>
</ns:New>
</ns:Row>
</Rows>
```

Trigger Delete

The following section gives sample input values and output XML for the Trigger Delete event.

Design-Time Input

Maximum Event Records: 5

Table Name: `adaptertesting.dbo.base`

Event: Delete

Run-Time Input

Time (in milliseconds): 20000

Trigger method: Manual or through a service

Output XML

```
<?xml version="1.0"?>
<Rows xmlns="wlai/TriggerDelete_BASE_delete"
xmlns:ns="wlai/TriggerDelete_BASE_delete">
<ns:Row>
<ns:Char1 isNull="false">STRING1   </ns:Char1>
<ns:CreateDt
isNull="false">2003-07-16T15:21:02+05:30</ns:CreateDt>
<ns:Decimal1 isNull="false">1</ns:Decimal1>
<ns:Double1 isNull="false">1</ns:Double1>
```

```
<ns:Float1 isNull="false">1</ns:Float1>
<ns:Integer1 isNull="false">1</ns:Integer1>
<ns:ModDt
isNull="false">2003-07-16T15:21:02+05:30</ns:ModDt>
<ns:Numeric1 isNull="false">1</ns:Numeric1>
<ns:Real1 isNull="false">1</ns:Real1>
<ns:Smallint1 isNull="false">1</ns:Smallint1>
<ns:Varchar1 isNull="false">STRING2</ns:Varchar1>
</ns:Row>
</Rows>
```

Select By Insert Timestamp

The following section gives sample input values and output XML for the Select By Insert Timestamp event.

Design-Time Input

Maximum Event Records: 5

Table Name: `adaptertesting.dbo.base`

Event: Insert

Created Timestamp: A column of type `TIMESTAMP` or `TIME`

Run-Time Input

Time (in milliseconds): 20000

Trigger method: Manual or through a service

If the trigger is manual, insert `CURRENT TIME` for the column of type `TIMESTAMP` or `TIME`.

Output XML

```
<?xml version="1.0"?>
<Rows xmlns="wlai/TimeStampInsert_BASE"
xmlns:ns="wlai/TimeStampInsert_BASE">
<ns:Row>
<ns:Char1 isNull="false">STRING1</ns:Char1>
<ns:CreateDtisNull="false">2003-07-16T15:34:46.827+05:30</ns:CreateDt>
<ns:Decimal1 isNull="false">1.00</ns:Decimal1>
```

```
<ns:Double1 isNull="false">1.0</ns:Double1>
<ns:Float1 isNull="false">1.0</ns:Float1>
<ns:Integer1 isNull="false">1</ns:Integer1>
<ns:ModDt
isNull="false">2003-07-16T15:34:46.827+05:30</ns:ModDt>
<ns:Numeric1 isNull="false">1</ns:Numeric1>
<ns:Real1 isNull="false">1.0</ns:Real1>
<ns:Smallint1 isNull="false">1</ns:Smallint1>
<ns:Varchar1 isNull="false">STRING2</ns:Varchar1>
</ns:Row>
</Rows>
```

Select By Update Timestamp

The following section gives sample input values and output XML for the Select By Update Timestamp event.

Design-Time Input

Maximum Event Records: 5

Table Name: `AdapterTesting.dbo.BASE`

Event: Update

Updated Timestamp: A column of type `TIMESTAMP` or `TIME`

Run-Time Input

Time (in milliseconds): 20000

Trigger method: Manual or through a service

If the trigger is manual, update `CURRENT TIME` for the column of type `TIMESTAMP` or `TIME`.

Output XML

```
<?xml version="1.0"?>
<Rows xmlns="wlai/TimeStampUpdate_BASE"
xmlns:ns="wlai/TimeStampUpdate_BASE">
<ns:Row>
<ns:Char1 isNull="false">STRING1</ns:Char1>
<ns:CreateDt
```

```
isNull="false">2003-07-16T15:34:25.437+05:30</ns:CreateDt>
<ns:Decimal1 isNull="false">1.00</ns:Decimal1>
<ns:Double1 isNull="false">1.0</ns:Double1>
<ns:Float1 isNull="false">1.0</ns:Float1>
<ns:Integer1 isNull="false">1</ns:Integer1>
<ns:ModDt
isNull="false">2003-07-16T15:36:13.230+05:30</ns:ModDt>
<ns:Numeric1 isNull="false">1</ns:Numeric1>
<ns:Real1 isNull="false">1.0</ns:Real1>
<ns:Smallint1 isNull="false">1</ns:Smallint1>
<ns:Varchar1 isNull="false">STRING2</ns:Varchar1>
</ns:Row>
<ns:Row>
<ns:Char1 isNull="false">STRING1</ns:Char1>
<ns:CreateDt isNull="false">2003-07-16T15:34:46.827+05:30</ns:CreateDt>
<ns:Decimal1 isNull="false">1.00</ns:Decimal1>
<ns:Double1 isNull="false">1.0</ns:Double1>
<ns:Float1 isNull="false">1.0</ns:Float1>
<ns:Integer1 isNull="false">1</ns:Integer1>
<ns:ModDt
isNull="false">2003-07-16T15:36:13.230+05:30</ns:ModDt>
<ns:Numeric1 isNull="false">1</ns:Numeric1>
<ns:Real1 isNull="false">1.0</ns:Real1>
<ns:Smallint1 isNull="false">1</ns:Smallint1>
<ns:Varchar1 isNull="false">STRING2</ns:Varchar1>
</ns:Row>
</Rows>
```

Select Then Delete

The following section gives sample input values and output XML for the Select Then Delete event.

Design-Time Input

Maximum Event Records: 5

Table Name: `adaptertesting.dbo.base`

Delete Required: Yes/No

SQL Query: `SELECT * FROM adaptertesting.dbo.base`

Run-Time Input

Time (in milliseconds): 20000

Trigger method: Manual or through a service

Output XML

```
<?xml version="1.0"?>
<Rows xmlns="wlai/SelectThenDelete_BASE"
xmlns:ns="wlai/SelectThenDelete_BASE">
<ns:Row>
<ns:Char1 isNull="false">STRING1</ns:Char1>
<ns:CreateDt
isNull="false">2003-07-16T15:34:25.437+05:30</ns:CreateDt>
<ns:Decimal1 isNull="false">1.00</ns:Decimal1>
<ns:Double1 isNull="false">1.0</ns:Double1>
<ns:Float1 isNull="false">1.0</ns:Float1>
<ns:Integer1 isNull="false">1</ns:Integer1>
<ns:ModDt
isNull="false">2003-07-16T15:36:13.230+05:30</ns:ModDt>
<ns:Numeric1 isNull="false">1</ns:Numeric1>
<ns:Real1 isNull="false">1.0</ns:Real1>
<ns:Smallint1 isNull="false">1</ns:Smallint1>
<ns:Varchar1 isNull="false">STRING2</ns:Varchar1>
</ns:Row>
<ns:Row>
<ns:Char1 isNull="false">STRING1</ns:Char1>
<ns:CreateDt
isNull="false">2003-07-16T15:34:46.827+05:30</ns:CreateDt>
<ns:Decimal1 isNull="false">1.00</ns:Decimal1>
<ns:Double1 isNull="false">1.0</ns:Double1>
<ns:Float1 isNull="false">1.0</ns:Float1>
<ns:Integer1 isNull="false">1</ns:Integer1>
<ns:ModDt
isNull="false">2003-07-16T15:36:13.230+05:30</ns:ModDt>
<ns:Numeric1 isNull="false">1</ns:Numeric1>
<ns:Real1 isNull="false">1.0</ns:Real1>
```

```
<ns:Smallint1 isNull="false">1</ns:Smallint1>
<ns:Varchar1 isNull="false">STRING2</ns:Varchar1>
</ns:Row>
</Rows>
```

APPENDIX B

Supported Data Types

The BEA WebLogic Adapter for RDBMS supports the following data types for each supported RDBMS.

Table B-1 Supported Data Types

Oracle	SQL Server	DB2 UDB	Informix Dynamic Server	Sybase Adaptive Server
• char • varchar • real • float • date • double • integer • numeric • refcursor (Multiple resultsets not supported) • smallint • CLOB • BLOB • null	• char • varchar • longvarchar • real • float • datetime • double • integer • numeric • decimal • smallint • resultsets (Multiple) • null	• char • varchar • real • float • timestamp • date • time • bit • double • integer • smallint • CLOB • BLOB • null	• char • varchar • real • float • smallfloat • date • datetime • integer • smallint • decimal • CLOB • BLOB • null	• char • varchar • real • float • datetime • smalldatetime • timestamp (Not supported in Trigger events) • bit • tinyint • integer • decimal • numeric • nchar • nvarchar • text (Not supported in Trigger events) • image (Not supported in Trigger events or Parametrized Services) • binary (Not supported in Trigger events) • varbinary (Not supported in Trigger events) • null

APPENDIX C

Error Messages and Troubleshooting

This section explains the various error messages you might encounter while using the BEA WebLogic Adapter for RDBMS, and how you can troubleshoot them.

Table C-1 Error Messages and Troubleshooting

1	**ERROR: com.bea.connector.ResourceWrapperException: Could not start EventRouter in Resource Adapter missing properties Could not start EventRouter in Resource AdapterEventGenerator: CannotInit EVENT_POLLING_METADATA table does not exist or the jndi name for default DSN in web.xml is incorrect javax.resource.spi.ResourceAdapterInternalException**	
	Description	You may get this exception while deploying an event. The `EVENT_POLLING_METADATA` table is missing.
	Action	Create the `EVENT_POLLING_METADATA` table.
	See Also	Section "Prepare the Adapter for RDBMS," in *BEA WebLogic Adapter for RDBMS Installation and Configuration Guide* at the following URL: `http://edocs.bea.com/wladapters/docs811/index.html`
2	**ERROR: Timed out**	

Table C-1 Error Messages and Troubleshooting (Continued)

	Description	Events may start timing out. It could be due to either of the following reasons: • The WebLogic Server might have been shut down abnormally without undeploying the application views. • The system time of the machine running the EIS is not in sync with the system time of the machine running the WebLogic Server.
	Action	If the WebLogic Server was abnormally shut down, then: 1. Restart the WebLogic Integration Server. 2. Redeploy the application views with events. If the events time out without giving correct results, ensure that the system time of the machine running the EIS is in sync with the system time of the machine running the WebLogic Server.
	See Also	None
3	**ERROR: Encountering error "java.lang.UnsatisfiedLinkError: no db2jdbc in java.library.path" While configuring JDBC URL connection to DB2 Type 2 driver(COM.ibm.db2.jdbc.app.DB2Driver) on Solaris**	
	Description	You may get this error while configuring a DB2 connection pool in Solaris. The DB2 Type 2 driver is not found in the `CLASSPATH`.
	Action	In Solaris, for the user starting WebLogic Server, set the environment variable `LD_LIBRARY_PATH` to contain the DB2 Type 2 driver. The driver file `db2java.zip` is available in the DB2 instance user's `sqllib/lib` directory.
	See Also	None

Index

F

G

L

N

P

R

S

T

V