



BEA WebLogic Adapter for RDBMS

User Guide

Release 7.1
Released: June 2003
Revised: April 2004

Copyright

Copyright © 2004 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

BEA WebLogic Adapter for RDBMS User Guide

Part Number	Date
N/A	Released: June 2003 Revised: April 2004

Contents

About This Document

What You Need to Know	viii
e-docs Web Site	viii
Related Information	ix
Contact Us!	ix
Documentation Conventions	x

1. Introducing the BEA WebLogic Adapter for RDBMS

About the BEA WebLogic Adapter for RDBMS	1-1
Overview	1-2
Supported Integration Capabilities	1-2
Key Concepts	1-3
Application Views	1-3
Services	1-4
Events	1-5
Schemas	1-6

2. Defining Application Views for the Adapter for RDBMS

Before You Begin	2-1
Defining and Deploying an Application View	2-2
Step 1. Log On to the Application View Console	2-3
Step 2. Add a Folder	2-4
Step 3. Define an Application View	2-4
Step 4. Configure an RDBMS Connection	2-5
Step 5. Add Services and Events to an Application View	2-13
Step 6. Deploy an Application View	2-25
Step 7. Test Services and Events	2-27

Editing an Application View	2-29
3. Services and Events of the Adapter for RDBMS	
Browse Metadata	3-1
Services	3-7
Standard SQL	3-8
Parametrized SQL	3-8
Stored Procedure	3-8
Arbitrary SQL.....	3-9
Events	3-9
Trigger Event.....	3-10
Select By Timestamp Event	3-10
Select Then Delete Event	3-11
Handling Null Values	3-11
Null Values in Events.....	3-11
Null Values in Services	3-13
A. Schema Formats for Services and Event Schemes	
Schemas for Services	A-1
Standard SQL	A-2
Parametrized SQL	A-3
Stored Procedure	A-5
Arbitrary SQL.....	A-6
Schemas for Event Schemes.....	A-8
B. Supported Data Types	
C. Commonly Used Drivers	
D. Logging	
About Logging.....	D-1
Levels of Logging.....	D-2
Logging and Performance	D-2
Creating or Modifying the Logging Level for an Event.....	D-3

E. Error Messages and Troubleshooting

Index



About This Document

This document explains how to use the BEA WebLogic Adapter for RDBMS to develop connections between a WebLogic Integration client and a Relational Database Management System (RDBMS).

It is organized as follows:

- [Chapter 1, “Introducing the BEA WebLogic Adapter for RDBMS,”](#) introduces the BEA WebLogic Adapter for RDBMS and provides an overview of the adapter’s functionality.
- [Chapter 2, “Defining Application Views for the Adapter for RDBMS,”](#) describes how to define and deploy application views.
- [Chapter 3, “Services and Events of the Adapter for RDBMS,”](#) describes the services and events.
- [Appendix A, “Schema Formats for Services and Event Schemes,”](#) lists the XML schema formats for the services and events that the adapter offers.
- [Appendix B, “Supported Data Types,”](#) lists the data types that the adapter supports.
- [Appendix C, “Commonly Used Drivers,”](#) lists the names and JDBC URLs for the commonly used drivers with which the adapter has been tested.
- [Appendix D, “Logging,”](#) provides information on creating and maintaining log information.
- [Appendix E, “Error Messages and Troubleshooting,”](#) explains the various errors you might encounter while using the adapter, and how you can troubleshoot them.

What You Need to Know

This document is written for system integrators who develop client interfaces between an RDBMS and third-party Enterprise Information System (EIS) applications. It describes how to use the BEA WebLogic Adapter for RDBMS to develop connections between a WebLogic Integration client and an RDBMS.

It is assumed that you know Web technologies and have a general understanding of Microsoft Windows and UNIX systems, as well as the following:

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

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

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on product documentation or go directly to the e-docs product documentation page at <http://edocs.bea.com>.

Related Information

The following documents contain information that is relevant to using this product.

- *BEA WebLogic Adapter for RDBMS Installation and Configuration Guide* and *BEA WebLogic Adapter for RDBMS Release Notes* at <http://edocs.bea.com/wladders/rdbms/docs71/index.html>
- BEA WebLogic Server installation and user documentation, at http://edocs.bea.com/more_wls.html
- BEA WebLogic Integration installation and user documentation, at http://edocs.bea.com/more_wli.html
- Specific RDBMS installation and user documentation
- *Using the WebLogic Integration Studio* at <http://edocs.bea.com/wli/docs70/studio/index.htm>
- *Learning to Use BPM with WebLogic Integration* at <http://edocs.bea.com/wli/docs70/bpmtutor/index.htm>

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 the BEA WebLogic Adapter for RDBMS 7.1 release.

If you have any questions about this version of BEA WebLogic Adapter for RDBMS, or if you have problems installing and running the product, 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.
<i>italics</i>	Indicates emphasis or book titles.
monospace text	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard. <i>Examples:</i> <pre>#include <iostream.h> void main () the pointer psz chmod u+w * \tux\data\ap .doc tux.doc BITMAP float</pre>

Convention	Item
monospace boldface text	Identifies significant words in code. <i>Example:</i> void commit ()
<i>monospace</i> <i>italic</i> <i>text</i>	Identifies variables in code. <i>Example:</i> String <i>expr</i>
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. <i>Examples:</i> 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. <i>Example:</i> buildobjclient [-v] [-o name] [-f <i>file-list</i>]... [-l <i>file-list</i>]...
	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
...	Indicates one of the following in a command line: <ul style="list-style-type: none"> ■ 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. <i>Example:</i> buildobjclient [-v] [-o name] [-f <i>file-list</i>]... [-l <i>file-list</i>]...
.	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.



1 Introducing the BEA WebLogic Adapter for RDBMS

This section introduces the BEA WebLogic Adapter for RDBMS and explains the key concepts. It contains the following topics:

- [About the BEA WebLogic Adapter for RDBMS](#)
- [Key Concepts](#)

About the BEA WebLogic Adapter for RDBMS

This section gives an overview of what the BEA WebLogic Adapter for RDBMS does, and how it does it in the context of a workflow. It contains the following topics:

- [Overview](#)
- [Supported Integration Capabilities](#)

Overview

Since most custom and packaged applications are built with relational databases, RDBMS systems must be taken into consideration in any enterprise integration strategy. The BEA WebLogic Adapter for RDBMS incorporates in-depth knowledge of relational database query access and transaction.

The BEA WebLogic Adapter for RDBMS enables integration with RDBMS systems by functioning as a service for accessing a database and as an event for listening to a database. In both cases, the query or stored procedure call is expressed in XML. This provides a convenient and simple method for integrating databases with enterprise applications using WebLogic Integration.

Key features of the BEA WebLogic Adapter for RDBMS include:

- Asynchronous, bi-directional message interactions between applications and databases, including Oracle, Microsoft SQL Server, IBM DB2 UDB, and Sybase Adaptive Server.
- Integration of requests and table event (outbound) operations in workflows
- JDBC 2.1 standard SQL operations (DELETE, INSERT, SELECT, and UPDATE) and the execution of stored procedures and functions

Supported Integration Capabilities

The adapter has been tested against the following databases and versions:

- Oracle 8i
- Oracle 9i
- SQL Server 2000
- DB2 UDB V7
- Sybase Adaptive Server 12.5 (Version of the adapter included with WebLogic Integration 7.0 SP5 only)

The Adapter for RDBMS has been built strictly to the JDBC standard. It does not take advantage of any proprietary extensions offered by these databases.

Key Concepts

This section explains the key concepts you should know while using the BEA WebLogic Adapter for RDBMS. It contains the following topics:

- [Application Views](#)
- [Services](#)
- [Events](#)
- [Schemas](#)

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. Application views define:

- 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 the 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.

An application view is typically configured for a single business purpose and contains only the services or events required for that business purpose.

Application views provide a layer of abstraction between applications and the EIS, making it easier for developers and non-programmers to access and maintain the services and events exposed by the adapter.

Services

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 to the client. Responses can be either synchronous or asynchronous.

A service processes SQL statements embedded in XML requests and forwards them to an RDBMS. The RDBMS returns the data to the adapter, which returns the data to the client.

Services receive an XML request document from a client and invoke a specific function in the target RDBMS. The adapters receive request messages and provide responses depending on the request. There are two possible methods for invoking an RDBMS service:

- Asynchronous. The client application issues a service request and then processes it without waiting for the response.
- Synchronous. The client waits for the response before proceeding with further processing.

WebLogic Integration supports both of these service invocations, relieving you from having to provide this functionality within your own application code.

The service can:

- Receive service requests from an external client.
- Transform the XML request document into the RDBMS data format. The request document conforms to the request XML schema for the service, which is based on metadata in the RDBMS.
- Invoke the underlying function in the RDBMS and wait for its response.
- Transform the response from the RDBMS data format to an XML document that conforms to the response XML schema for the service, which is based on metadata in the RDBMS.

Events

Events are asynchronous, one-way messages received from an EIS. For example, the adapter can receive a message from an RDBMS or route the EIS message to the appropriate software

Events are captured from applications that write to a relational table. It captures the data and performs operations based on the content of the rows. The event 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 then 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 as an application view and sent as an event to WebLogic Integration.

The event can:

- Monitor data changes by repeatedly performing an SQL query.
- Monitor data changes by querying on timestamp columns.
- Be configured to operate one row at a time or to operate on sets of rows, sending events to a business process workflow whenever one or more rows, not exceeding the maximum number of rows specified by the user, become available in the source RDBMS. When the rows exceed the maximum number of rows, a new event is triggered as multiples of the maximum rows. Separate events will be posted with data packaged with the size of maximum rows. For instance, if there are 10 rows of interest, and the maximum number of rows is 3, 4 events will be posted with 3 rows in each of the first 3 events, and the last event will contain the remaining row.
- Allow the configuration of an optional SQL post-query statement, which deletes/retains after the adapter has sent the rowset (formatted as XML) to a business process workflow. The default operation is to delete the rows that have been transferred to the workflow.

Schemas

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 have been defined at design-time to map the data between XML and the EIS format:

- 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 (as defined in the application view) 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.
- For event notifications, the inbound 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 (as defined in the application view) to translate the response to an XML document that the subscribed application handles.

The Adapter for RDBMS creates the schema you need when you define services and events. You can view the schemas that the adapter creates. For examples of schema, see [Appendix A, “Schema Formats for Services and Event Schemes.”](#)

2 Defining Application Views for the Adapter for RDBMS

This section explains how to define and maintain application views for RDBMS integration. It contains the following topics:

- [Before You Begin](#)
- [Defining and Deploying an Application View](#)
- [Editing an Application View](#)

Before You Begin

When you define an application view, you are creating an XML-based interface between WebLogic Server and a particular relational database management system (RDBMS) within your enterprise. Once you create the application view, a business analyst can use it to create business processes that use the application. You can create any number of application views for an adapter, each of which may contain any number of services and events.

Before you define an application view, make sure the following prerequisite is satisfied: You have determined which business processes need to be supported by the application view you are configuring. 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.

Note: This procedure shows how to install and configure an Oracle implementation of the RDBMS event and service. When configuring other JDBC drivers for other RDBMS, the parameters you enter will be different.

Defining and Deploying an Application View

This section explains how to define and maintain application views using the Adapter for RDBMS.

For details on defining application views and using them in workflows, see *Using Application Integration* at the following URL:
<http://edocs.bea.com/wli/docs70/aiuser>.

Note: Before performing the following steps, ensure that the WebLogic Integration Server is running on your system.

Defining an Application View consists of the following steps:

- [Step 1. Log On to the Application View Console](#)
- [Step 2. Add a Folder](#)
- [Step 3. Define an Application View](#)
- [Step 4. Configure an RDBMS Connection](#)
- [Step 5. Add Services and Events to an Application View](#)
- [Step 6. Deploy an Application View](#)
- [Step 7. Test Services and Events](#)

Step 1. Log On to the Application View Console

The Application View Console displays all the application views in your WebLogic Integration environment, organized in folders.

To log on to the Application View Console:

1. Open a new browser window.
2. Enter the URL for your system's Application View Console. The actual URL you enter depends on your system. It should conform to the following format:

`http://localhost:port/wlai`

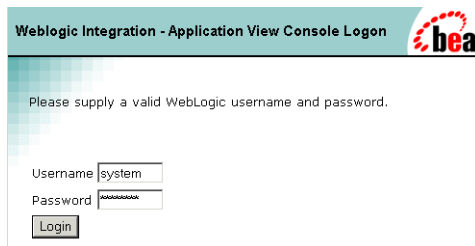
Here, *localhost* is the IP address or DNS name of the machine on which WebLogic Server is running, and *port* is the port number on which the server is listening.

For example, `http://localhost:7001/wlai` or

`http://172.19.138.44:7001/wlai`

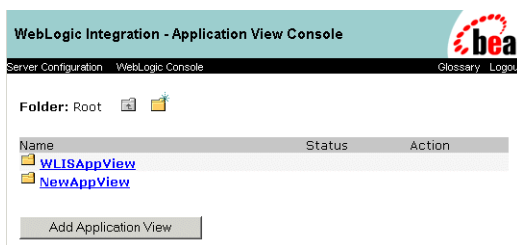
The WebLogic Integration Application View Console Login page appears.

Figure 2-1 WebLogic Integration - Application View Console Logon

The image shows a web browser window displaying the login page for the WebLogic Integration Application View Console. The title bar of the browser window reads "Weblogic Integration - Application View Console Logon". The page has a light blue header with the BEA logo on the right. Below the header, the text "Please supply a valid WebLogic username and password." is displayed. There are two input fields: "Username" with the value "system" and "Password" with masked characters. A "Login" button is located below the password field.

3. Enter your WebLogic Server username and password, then click Login. The WebLogic Integration - Application View Console page appears.

Figure 2-2 WebLogic Integration - Application View Console



Step 2. Add a Folder

You organize the application views in folders. A single folder may contain both application views and other subfolders. Once you create a folder, you cannot move it to another folder. Before you can remove a folder, you must first remove all application views and subfolders. Once you create an application view in a folder, you can remove the application view, but you cannot move it to another folder. You can add an application view to an existing folder.

To add a folder, do the following:

1. On the Application View Console page, click the New Folder icon. The Add Folder window appears.
2. In the New Folder field, enter a name. Valid characters include a-z, A-Z, 0-9, and the _ (underscore) character.

Note: The name Root is a reserved word, and cannot be used for a folder name. If you use Root as a name, you cannot import or export the folder using the import/export utility.

3. Click Save.

Step 3. Define an Application View

You define an application view before adding services and events.

To define an application view:

1. Click Add Application View.

Note: Make sure you are working in the appropriate folder before performing this step. Once you define an application view, you cannot move it to another folder.

The Define New Application View page appears.

Figure 2-3 Define New Application View

Define New Application View

Server Configuration > WebLogic Console

This page allows you to define a new application view

Folder: [NewAppView](#)

Application View Name: *

Description:

Associated Adapter:

Note: An red asterisk (*) indicates a mandatory field.

2. Enter a name for the application view in the Application View Name field. Each application view must be unique to its adapter.
3. Enter notes that may be helpful to users using this application view in workflows created in the WebLogic Integration Studio, in the Description field.
4. Select the adapter that should be associated with this application view (WebLogic RDBMS Adapter), from the Associated Adapter drop-down list.
5. Click OK. The Configure Connection Parameters page appears.

Step 4. Configure an RDBMS Connection

You should create a connection pool and a data source before configuring an RDBMS connection.

Step 4A. Create a Connection Pool

To create a connection pool for an Oracle database, do the following:

1. Open the WebLogic Server Console. The WebLogic Server Administration Console can be found at the following URL:

`http://localhost:port/console`

Here, *localhost* is the IP address or DNS name of the machine on which WebLogic Server is running, and *port* is the port number on which the server is listening.

2. In the left pane, choose Services→JDBC→Connection Pools from the navigation tree. In the page that appears, click Configure a new JDBC Connection Pool. The following page appears.

Figure 2-4 Create a New JDBC Connection Pool

The screenshot shows the 'Create a new JDBCConnectionPool...' page in the WebLogic Server Console. The page has a breadcrumb trail: 'samples> JDBC Connection ...> Create a new JDBCConnectionPool...'. The top status bar indicates 'Connected to localhost:7001', 'Active Domain: samples', and the date 'May 20, 2003 2:27:51 PM IST'. The 'Configuration' tab is active, with sub-tabs for 'General', 'Connections', 'Monitoring', and 'Notes'. The 'General' sub-tab is selected, displaying several configuration fields, each with a yellow warning icon and a question mark:

- Name:** OracleConnectionPool
- URL:** jdbc:oracle:thin:@172.19.139.58:15
- Driver Classname:** oracle.jdbc.driver.OracleDriver
- Properties (key=value):** user=beauser, password=beauser
- ACLName:** (empty field)
- Password:** (empty field)
- Open String Password:** (empty field)

A 'Create' button is located at the bottom right of the form.

3. On the Configuration General tab, enter the required details.
4. On the Connections tab enter the Initial Capacity, Maximum Capacity, and Capacity Increment as required. Set appropriate values based on the number of connections you require.

Note: If an XA driver is used to create the physical database connections, enable the Supports Local Transaction check box.

Figure 2-5 Connections Tab

The screenshot shows the 'Connections' tab in a configuration window. The window has tabs for 'Configuration', 'Targets', 'Monitoring', and 'Notes'. Under 'Configuration', there are sub-tabs for 'General', 'Connections', and 'Testing'. The 'Connections' sub-tab is active. It contains several configuration options, each with a yellow warning icon and a question mark:

- Initial Capacity: 10
- Maximum Capacity: 15
- Capacity Increment: 5
- Login Delay Seconds: 0 seconds
- Refresh Period: 0 minutes
- ☐ Supports Local Transaction
- ☒ Allow Shrinking
- Shrink Period: 15 minutes
- Prepared Statement Cache Size: 5

An 'Apply' button is located at the bottom right of the configuration area.

5. Click Create. Select the server and use the arrow buttons to move it to the Chosen list.

Figure 2-6 Targets Tab

The screenshot shows the 'Targets' tab in a configuration window. The window has tabs for 'Configuration', 'Targets', 'Monitoring', and 'Notes'. Under 'Configuration', there are sub-tabs for 'General', 'Connections', and 'Testing'. The 'Targets' sub-tab is active. It contains a 'Targets-Server' section with two lists: 'Available' and 'Chosen'. The 'Chosen' list contains the entry 'myserver'. There are arrow buttons between the two lists. An 'Apply' button is located at the bottom right of the configuration area.

6. Click Apply.

Step 4B. Configure Policy for the Connection Pool

With WebLogic Integration 7.0, to allow the administrative server user access to this connection pool, do the following:

1. In the left pane, click Services→JDBC→Connection Pools.
2. Right-click the newly created connection pool, and select Define Policy.

The Select Authorizer provider page appears.

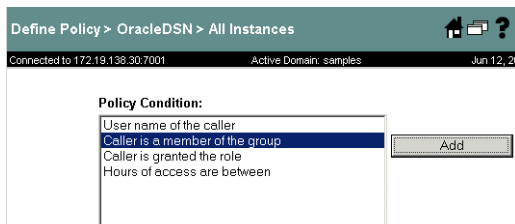
Figure 2-7 Select Authorizer provider



Name	Description
RealmAdapterAuthorizer	Realm Adapter for Authorization
DefaultAuthorizer	Weblogic Authorization Provider

3. Click the DefaultAuthorizer link. The Define Policy page appears.

Figure 2-8 Define Policy



Define Policy > OracleDSN > All Instances

Connected to 172.19.138.30:7001 Active Domain: samples Jun 12, 20

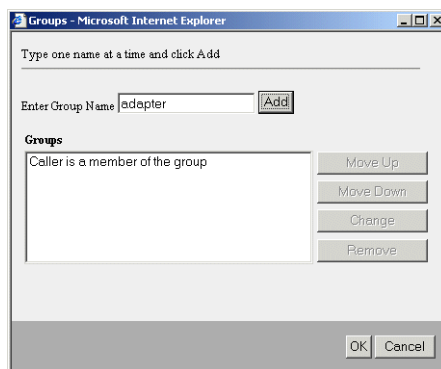
Policy Condition:

- User name of the caller
- Caller is a member of the group
- Caller is granted the role
- Hours of access are between

Add

4. In the Policy Condition list, select Caller is a Member of the Group and click Add. The Groups window appears.

Figure 2-9 Groups



Groups - Microsoft Internet Explorer

Type one name at a time and click Add

Enter Group Name Add

Groups

- Caller is a member of the group

Move Up
Move Down
Change
Remove

OK Cancel

5. Enter the group name and click Add. The group specified is added to the Groups list below.

Note: You can add only an existing group. For details on creating a group, see section "Create the Adapter Group and Add the User Name," in *BEA WebLogic Adapter for RDBMS Installation and Configuration Guide for WebLogic Integration 7.0* at the following URL:

http://edocs.bea.com/wladders/rdbms/docs71/pdf/install_wli70.pdf

6. Click OK. The Define Policy page is updated as shown in the following figure.

Figure 2-10 Policy Statement

Policy Condition:

- User name of the caller
- Caller is a member of the group**
- Caller is granted the role
- Hours of access are between

Policy Statement:

- Caller is a member of the group adapter

Buttons: Add, Move Up, Move Down, Change, Edit..., Remove

Now the administrative server user has access to the connection pool.

Step 4C. Create a Data Source

You can create a JNDI data source for both normal and XA connections.

To create a normal JNDI data source, do the following:

1. In the left pane of the WebLogic Server Console, choose Services→JDBC→Data Sources from the navigation tree. In the page that appears, click Configure a new JDBC Data Source. The following page appears.

Figure 2-11 Create a New JDBCDataSource

The screenshot shows the 'Create a new JDBCDataSource...' dialog box. The 'Configuration' tab is selected. The fields are as follows:

Field	Value
Name	OracleDSN
JNDI Name	OracleDSN
Pool Name	OracleConnectionPool
Row Prefetch Enabled	<input type="checkbox"/>
Row Prefetch Size	48
Stream Chunk Size	256 bytes

The 'Create' button is located at the bottom right of the dialog.

2. On the Configuration tab, enter the required information.
3. Click Create. Select the server and use the arrow buttons to move it to the Chosen list.

Figure 2-12 Targets Tab

The screenshot shows the 'Targets Tab' for the 'OracleDSN' data source. The 'Targets-Server' section is active. It contains two lists: 'Available' and 'Chosen'. The 'Chosen' list contains the server 'myserver'. The 'Apply' button is at the bottom right.

4. Click Apply.

Once created, the Data Source JNDI Name can be specified in Application View Service.

Note: For more information on connection pools and data sources, see section “JDBC Components-Connection Pools, Data Sources, and MultiPools,” in “Managing JDBC Connectivity” in the *WebLogic Server Administration Guide*:

- For WebLogic Server 7.0, see the following URL:
<http://edocs.bea.com/wls/docs70/adminguide/jdbc.html>

- For WebLogic Server 6.1, see the following URL:

<http://edocs.bea.com/wls/docs61/adminguide/jdbc.html>

To create a JNDI data source for an XA connection, do the following:

In the left pane of the WebLogic Server Console, choose Services→JDBC→Tx Data Sources from the navigation tree. In the page that appears, click Configure a new JDBC Tx Data Source. Now, follow the same procedure as you would for a normal connection. See step 2. on page 10.

Step 4D. Configure Connection Parameters

To configure connection parameters, do the following:

1. On the Configure Connection Parameters page, set the following properties.

Figure 2-13 Configure Connection Parameters

Configure Connection Parameters

Application View Console WebLogic Console Glossary Logout

Select Connection Type
• **Configure Connection**
Administration
Add Service
Add Event
Deploy Application View

On this page, you supply parameters to connect to your DBMS.

WebLogic / Database User Name*

WebLogic / Database Password*

A valid Weblogic user name and password should be given when creating a connection using Data Source JNDI Name. And a valid Database user name and password should be given when creating a connection using a driver.

Create a DataSource Using WLS DataSource

☒ Data Source JNDI Name

Create a Connection Using a Driver

☐ JDBC URL

Driver Name

Driver Supports XA ☐ Yes ☐ No ☐ NA

Note: An asterisk (*) indicates a mandatory field.

2. Enter your WebLogic user name in the WebLogic/Database User Name field, based on the data source option you plan to choose.

2 *Defining Application Views for the Adapter for RDBMS*

- If you select DataSource JNDI Name option, enter your WebLogic Server user name.
 - If you select the JDBC URL option, enter your RDBMS user name.
3. Enter the WebLogic password in the WebLogic/Database Password field, for the user name you entered.
 4. To create a data source using an existing WebLogic Server DataSource, select the DataSource JNDI Name option button. This JNDI data source could be for either a normal or an XA connection.
 5. Enter the DataSource JNDI Name for the JDBC connection pool to use for connecting to the RDBMS system.

Note: For details on how to create a connection pool, see [“Step 4A. Create a Connection Pool” on page 2-6](#). For details on how to create a data source, see [“Step 4C. Create a Data Source” on page 2-9](#).
 6. To create a connection using a driver, select the JDBC URL option button.
 7. Enter the JDBC URL, Driver Name, and whether the driver supports XA.

Note: For details of commonly used drivers, see [Appendix C, “Commonly Used Drivers.”](#)
 8. Click Continue. The Application View Administration page appears.

Figure 2-14 Application View Administration

Application View Administration for NewAppView.NewAppView

Application View Console WebLogic Console

Select Connection Type
Configure Connection
Administration
Add Service
Add Event
Deploy Application View

This page allows you to add events and/or services to an Application View.

Description: No description available for NewAppView.NewAppView. [Edit](#)

Connection Criteria

Message Bundle Base:	BEA_RDBMS_7_1
Data Source Name:	OracleDSN
JDBC URL:	
Additional Log Category:	NewAppView
Username:	system
Root Log Category:	BEA_RDBMS_7_1
Connection Info Type:	DSN
Log Configuration File:	BEA_RDBMS_7_1.xml
Driver Name:	

[Reconfigure connection parameters for NewAppView](#)

Events

--

Services

--

Save

Step 5. Add Services and Events to an Application View

You can add services and events that support specific business processes. An application view can have multiple services and events. The required business processes determine the types of services and events you include in your application view.

This section contains the following topics:

- [Step 5A. Add a Service](#)
- [Understanding Service Types](#)
- [Step 5B. Add an Event](#)
- [Understanding Events](#)

Step 5A. Add a Service

The Adapter for RDBMS offers the following services:

- Standard SQL

- Parametrized SQL
- Stored Procedure
- Arbitrary SQL

To add a service to the application view, do the following:

1. On the Application View Administration page, click Add in the Services row. The Add Service page appears.

Figure 2-15 Add Service

Add Service

Application View Console WebLogic Console

Select Connection Type
Configure Connection
Administration
Add Service
Add Event
Deploy Application View

On this page, you add services to your Application View.

Unique Service Name:*

Description:

Service Details

Service Type:*

Isolation Level:

Enter SQL Statement (If you have chosen the Service Type For Parametrised Query & Standard Query)

SQL Statement:

Syntax Help: 1. Use fully qualified table name (i.e. catalog.schema.table); 2. to gather user input, bracket the column name and type as follows: "[ColumnName ColumnType]". Hint: browse to cut & paste ColumnName and ColumnType into your sql.

Parameters created in SQL only support JDBC types. To view the supported types follow the link. [JDBC Types](#)

Enter the Stored Procedure Name (If You have chosen the Service Type as Stored Procedure)

Stored Procedure Name:

(This field is mandatory for service type is 'Stored Procedure')

[Browse DBMS to view Tables, Views and Stored Procedures...](#)

2. In the Unique Service Name field, enter a name. The name should describe the function performed by this service. Each service name must be unique to its application view. Valid characters include a-z, A-Z, 0-9, and the _ (underscore) character. This is a mandatory field.

3. In the Description field, enter any notes which may be helpful to people using this application view in workflows created in the WebLogic Integration Studio.
4. Under Service Details, select the Service Type from the drop-down list. This is a mandatory field. You can select one of the following:
 - Standard SQL - This service invokes a standard SQL statement, without any parameters.
 - Parametrized SQL - This service invokes a parametrized SQL statement, with parameters given at design time.
 - Stored Procedure - This service invokes a stored procedure defined in an RDBMS.
 - Arbitrary SQL - This service invokes an arbitrary SQL statement at run time.For details, see section [“Understanding Service Types” on page 2-16](#).
5. Select the Isolation Level from the drop-down list. For details, see the following table.

Table 2-1 Isolation Levels

Isolation Level	Description
None	This level 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, and multiple reads yield the same committed result. 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.

6. Enter the SQL statement, if you have selected Standard SQL or Parametrized SQL from the Service Type drop-down list.

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

For details, see section [“Understanding Service Types” on page 2-16](#).

7. Enter the Stored Procedure Name, if you have selected Stored Procedures from the Service Type drop-down list.

For example, `BEAUSER.RAISESAL(?, ?)`

For details, see section [“Understanding Service Types” on page 2-16](#).

Do not enter an SQL statement or a Stored Procedure Name if you have selected Arbitrary SQL from the drop-down list.

Note: Click the Browse DBMS to view Tables, Views, and Stored Procedures link to select the Stored Procedure Name. For details, see section [“Browse Metadata” on page 3-1](#).

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

Figure 2-16 Added Service

Events		Add
EveTrigger	Edit Remove Event View Summary View Event Schema	

Services		Add
ServStdSQL	Edit Remove Service View Summary View Request Schema View Response Schema	

Continue Save ?

9. Do one of the following:
 - Click Add to continue adding services or events.
 - Click Save to save the application view.
 - Click Continue to save and deploy the application view. For details, see [“Step 6. Deploy an Application View” on page 2-25](#).

Understanding Service Types

The Adapter for RDBMS offers the following service types:

- [Standard SQL](#)
- [Parametrized SQL](#)
- [Stored Procedure](#)
- [Arbitrary SQL](#)

Standard SQL

This service invokes a Standard SQL statement, without any parameters. You give the SQL statement, without any parameters, at design time.

The SQL executed can be either a DML (SELECT, INSERT, UPDATE, or DELETE) statement or a DDL (CREATE, ALTER, or DROP) statement. The resultset is converted into the response XML document, based on the XML schema formed during design time.

Note: While entering the SQL statement, note the following:

- The table name should be fully qualified with the SCHEMA/CATALOG name as per the requirement in the underlying RDBMS. The table name is case sensitive—use upper case. For example, in Oracle, it should be `SCHEMA.TABLENAME`.
- Insert/Update statements containing `clob/blob` data type will not be supported in this service type. Use Parametrized SQL to insert or update `clob/blob` data types in a table. The output from a table with a column of the type `clob/blob` will be decoded to base64 before forming the output XML document.

Parametrized SQL

This service invokes a parametrized SQL defined in an RDBMS. You should 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 giving the values for the parameters specified.

Based on the data type of the input provided, appropriate binding of parameter values is done. The prepared SQL statement is executed, and the resultset is converted into the response XML document, based on the XML schema formed during design time.

Note: While entering the SQL statement, note the following:

- The table name should be fully qualified with the SCHEMA/CATALOG name as per the requirement in the underlying RDBMS. The table name is case sensitive—use upper case. For example, in Oracle, it should be `SCHEMA.TABLENAME`.
- To gather user input, enter the column name and column type within brackets. For example, `[ParamName ColumnType]`. The ParamName provided here will be used to define an element in the input schema. This name needs to be unique in the parametrized statement given as input during design time.
- The ColumnType specified in the parametrized query 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.”](#)

Note: Parameters of the Type `clob` and `blob` given as input in the Parametrized SQL should be base64 encoded. The adapter would decode the input and bind it to the statement before execution. The output from a table with a column of the type `clob/blob` will be decoded to base64 before forming the output XML document.

Note: At run time, enter the values for the parameters specified during design time.

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. The stored procedure 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, appropriate binding of parameter values is done. The stored procedure is executed and the resultset is converted into a response XML document, based on the XML schema formed during design time.

Because DBMS manufacturers produce JDBC agents with varying degrees of functionality, stored procedures can be executed in several different ways within the service.

Note: Oracle subprograms (stored procedures) can be called whether they reside as functions, stored procedures, or stored procedures located within packages.

Note: Enter the stored procedure name in the format `ProcedureName(?, ?, ...)`. The ?s denote 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.

- In Oracle, it should be `SchemaName.PackageName.ProcedureName(?, ?, ...)`. If the procedure is not packaged, it should be `SchemaName.ProcedureName(?, ?, ...)`
- In SQL Server, it should be `CatalogName.SchemaName.ProcedureName(?, ?, ...)`
- In DB2, it should be `SchemaName.ProcedureName(?, ?, ...)`
- In Sybase, it should be `DatabaseName.SchemaName.ProcedureName(?, ?, ...)`

Note: At run time, enter the values of the type IN and IN-OUT parameters. These will be validated against the input schema.

Arbitrary SQL

This service invokes an arbitrary SQL statement at run time.

The SQL given at run time can either be a DML (SELECT, INSERT, UPDATE, or DELETE) or a DDL (CREATE, ALTER, or DROP) statement. The resultset is converted into the response XML document, based on the type of statement executed.

Note: While entering the SQL statement, note the following:

- The table name should be fully qualified with the SCHEMA/CATALOG name as per the requirement in the underlying RDBMS. The table name is case sensitive—use upper case. For example, in Oracle, it should be `SCHEMA.TABLENAME`.
- Insert/Update statements containing clob/blob data type will not be supported in this service type. Use Parametrized SQL to insert or update clob/blob data types in a table. The output from a table with a column of the type clob/blob will be decoded to base64 before forming the output XML document.
- The SQL statement should not be parametrized.

Step 5B. Add an Event

You can add services and events that support specific business processes. An application view can have multiple services and events. The required business processes determine the types of services and events you include in your application view.

The Adapter for RDBMS offers the following events:

- Trigger
- Select By Timestamp
- Select Then Delete

To add an event to the application view, do the following:

1. On the Application View Administration page, click Add in the Events row. The Add Event page appears.

Figure 2-17 Add Event

Add Event

Application View Console WebLogic Console Glossary Logo

Select Connection Type
Configure Connection
Administration
Add Service
Add Event
Deploy Application View

On this page, you add events to your Application View.

Unique Event Name:* EveTrigger

Description:

Event Details

Event Polling Interval (in secs):* 6

Maximum Event Records:* 5

Table Name:* BEAUSER.BONUS [Browse DBMS...](#)

Syntax Help... ORACLE: SCHEMA.TABLENAME, MS SQLSERVER: catalog.schema.tablename, SYBASE: catalog.schema.tablename, DB2: SCHEMA.TABLENAME

☒ Trigger

Please select the type of event to create:

☒ Insert Event
☐ Update Event
☐ Delete Event
☐ Select_By_Timestamp

Please select the type of event to create:

☒ Insert Event
☐ Update Event
☐ Select_Then_Delete

Updated TimeStamp Column [Browse Columns...](#)

Created TimeStamp Column [Browse Columns...](#)

Enter SQL Query

Delete Required ☐ Yes ☒ No

2. In the Unique Event Name field, enter a name. Each event name must be unique to its application view. Valid characters include a-z, A-Z, 0-9, and the _ (underscore) character. This is a mandatory field.
3. In the Description field, enter any notes that may be helpful to people using this application view in workflows created in the WebLogic Integration Studio.

4. Enter the interval in seconds at which the database table is to be monitored for changes, that is, add/update/delete of rows, in the Event Polling Interval field. This is a mandatory field.

Note: This value should be greater than 4 seconds. The polling will happen only at times that are proper multiples of 4 seconds, that is, 4, 8, 12, and so on. There will be an error of 0-3 seconds, based on the polling interval specified.

5. Enter the maximum number of data records to be processed as an event in the Event Polling Interval specified, in the Maximum Event Records field. This is a mandatory field.

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.

6. Enter the name of the database table in the Table Name field. This is a mandatory field. Use the given syntax for the following databases:

Table 2-2 Syntax for Database Table Names

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

Click the Browse DBMS link to see the schemas and table names.

7. Select the option button for the type of event to add—Trigger, Select_By_Timestamp, or Select_Then_Delete.

For details, see section [“Understanding Events” on page 2-23](#).

8. Do the following, based on the event:
 - If the event is a Trigger, select one of the following event types: Insert Event, Update Event, Delete Event.

- If the event is a `Select_By_Timestamp`, select one of the following events: Insert Event, Update Event. If you select Insert Event, enter the Created Timestamp Column name. If you select Update Event, enter the Updated Timestamp Column name.
 - If the event is a `Select_Then_Delete`, enter the SQL Query. Select the Delete Required Yes option button if you want the selected records to be deleted.
9. When finished, click Add. The event is added to the list.

Figure 2-18 Added Event

The screenshot shows a web interface with two main sections: 'Events' and 'Services'. The 'Events' section has a table with one row containing 'EveTrigger' and several links: 'Edit', 'Remove Event', 'View Summary', and 'View Event Schema'. The 'Services' section has a table with one row containing 'ServStdSQL' and several links: 'Edit', 'Remove Service', 'View Summary', 'View Request Schema', and 'View Response Schema'. At the bottom of the interface are three buttons: 'Continue', 'Save', and a question mark icon.

10. Do one of the following:
- Click Add to continue adding services or events.
 - Click Save to save the application view.
 - Click Continue to save and deploy the application view. For details, see [“Step 6. Deploy an Application View” on page 2-25](#).

Understanding Events

The Adapter for RDBMS offers the following events:

- [Trigger Event](#)
- [Select By Timestamp Event](#)
- [Select Then Delete Event](#)

The adapter provides the following event types for the events:

Table 2-3 Event Types for Events

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

The event supports the polling of relational tables for incoming (inserted), modified (updated) and deleted data. This highly configurable event, reads one or more rows from the table of interest specified and creates an XML document representing the data pertaining to the record of interest.

Trigger Event

This event can be used to notify an Insert, Update, or Delete event occurring in a database table.

Select By Timestamp Event

This event can be used to notify an Insert and Update event occurring in a database table. The table of interest should have columns representing `Created Timestamp` and `Updated Timestamp`. Insert event can be configured only if `Created Timestamp` is maintained as a part of the table, and Update event can be configured 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 EVENT

This event can be used to notify records of interest based on a select query given on a table of interest. The event polling mechanism looks for records based on the select query and packages the result into one or more rows and creates an XML document representing record of interest. The user can opt to either delete or retain the records selected after the processing of events.

Step 6. Deploy an Application View

You may deploy an application view when you have added at least one event or service to it. You must deploy an application view before you can test its services and/or events or use the application view in the WebLogic Server environment. Application view deployment places relevant metadata about its services and events into a run-time metadata repository. Deployment makes the application view available to other WebLogic Server clients. This means business processes can interact with the application view, and you can test the application view's services and events.

You can undeploy an application view if required.

To deploy the application view, do the following:

1. After you have added the required services and events, click Continue on the Application View Administration page.

Figure 2-19 Application View Administration

Application View Administration for NewAppView.NewAppView

Application View Console WebLogic Console

Select Connection Type
Configure Connection
Administration
Add Service
Add Event
Deploy Application View

This page allows you to add events and/or services to an Application View.

Description: No description available for NewAppView.NewAppView. [Edit](#)

Connection Criteria

Message Bundle Base:	BEA_RDBMS_7_1
Data Source Name:	OracleDSN
JDBC URL:	
Additional Log Category:	NewAppView
Username:	system
Root Log Category:	BEA_RDBMS_7_1
Connection Info Type:	DSN
Log Level:	DEBUG
Log Configuration File:	BEA_RDBMS_7_1.xml
Driver Name:	

[Reconfigure connection parameters for NewAppView](#)

Events

Services

ServStdSQL [Edit](#) [Remove Service](#) [View Summary](#) [View Request Schema](#) [View R](#)

?

The Deploy Application View to Server page appears.

Figure 2-20 Deploy Application View to Server

Deploy Application View NewAppView.NewAppView to Server

Application View Console WebLogic Console

On this page you deploy your Application View to the application server.

Required Event Parameters

Event Router URL*

Connection Pool Parameters

Use these parameters to configure the connection pool used by this Application View.

Minimum Pool Size*

Maximum Pool Size*

Allow Pool to Shrink? ☒

Log Configuration

Set the log verbosity level for this Application View.

Log warnings, errors, and audit messages

Configure Security

[Restrict Access to NewAppView using J2EE Security](#)

Deploy ☒ Deploy persistently? Save

Note: An asterisk (*) after the name indicates a mandatory field.

2. Enter the URL of the event router. The default is
`http://localhost:port/DbmsEventRouter/EventRouter`
3. Enter the minimum size of the connection pool used by this application view.
4. Enter the maximum size of the connection pool used by this application view.
5. Select the Allow Pool to Shrink check box if the size of the connection pool should be allowed to reduce.
6. Select what should be logged for this application view, from the drop-down list. The options are as follows:
 - Log errors and audit messages
 - Log warnings, errors, and audit messages
 - Log informationals, warnings, errors, and audit messages
 - Log all messages

Note: For maximum logging, select Log all Messages. This is recommended to obtain optimum debugging information for BEA support personnel. This causes all generated messages to be written to the log. Logs are created the next time the service is invoked or the event occurs. Logs are output to a

file named `BEA_RDBMS_7_1.log` in the WebLogic Domain home directory. For details on logging, see [Appendix D, “Logging.”](#)

7. Click the Restrict Access using J2EE Security link to configure the security for this application view by granting or revoking permissions for users or groups.
8. Select the Deploy persistently check box if you want this application view to be redeployed whenever the application server is restarted.
9. Click Deploy. The application view is deployed, and the Summary page is displayed.

Step 7. Test Services and Events

The purpose of testing an application view service or event is to evaluate whether the service or event interacts properly with the RDBMS.

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

Step 7A. Test a Service

To confirm that a deployed application view service is correctly configured:

1. In the left pane, select Summary. The Summary for Application View page appears.
2. In the Services area of the Events and Services tab, find the appropriate service and click Test. The Test Service page appears.
3. If necessary, enter the required data in the appropriate fields.
4. Click Test. If the application view service correctly processes the input data that you provided in step 3, the test is successful. The Test Result page, on which all input and output documents are listed, appears.
5. Repeat the test procedure (steps 1-4) for each service you want to test.
6. After you finish testing the application view's services, you may keep the application view deployed or, if you want to edit it, you may undeploy it.

Step 7B. Test an Event

You can use one of the application view's own services to generate an event, or log on to an RDBMS and perform the appropriate event-generating function.

To confirm that a deployed application view service is correctly configured:

1. In the left pane, click Summary. The Summary for Application View page appears.
2. In the Events area on the Events and Services tab, find the appropriate event and click Test. The Test Event page appears.
3. Select Service when you want to use one of the application view's own services to generate a canned event. Then do the following:
 - a. From the Service menu, select a service that triggers the event you are testing.
 - b. In the Time field, enter a reasonable period of time to wait, specified in milliseconds. (One minute = 60,000 milliseconds) If the specified period elapses before the event succeeds, the test times out and a failure message appears.
 - c. Click Test. The triggering service is executed. If the service requires input data, an input page appears.
 - d. If service input data is required, enter it in the appropriate fields, and click Test. The service is executed. If the test succeeds, the Test Result page appears, showing the event document, the service input document, and the service output document. If the test fails, the Test Result page displays only a Timed Out message.
 - e. If the test fails, edit the event definition, or contact the system administrator or application manager.
 - f. If the test succeeds, repeat the test procedure for each remaining event you want to test.
 - g. When finished, save the application view.
4. Select Manual when you want to generate the event by logging on to an RDBMS and performing the appropriate event-generating function. Then do the following:
 - a. In the Time field, enter a reasonable time to wait, specified in milliseconds. (One minute = 60,000 milliseconds) If this period elapses before the event succeeds, the test times out and a failure message appears.

- b. If the application you will use to trigger the event is not already open, open it now.
- c. Click Test. The test waits for an event to trigger it.
- d. Using the triggering application, perform an action that executes the service that, in turn, tests the application view event. If the test succeeds, the Test Result page appears. This page, in turn, displays the event document from the application, the service input document, and the service output document. If the test fails or takes too long, the Test Result page appears, showing a Timed Out message.
- e. If the test fails, edit the event definition, or contact the system administrator or application manager.
- f. If the test succeeds, repeat the test procedure for each remaining event you want to test.
- g. When you are finished, save the application view.

If the application view event responds correctly before the specified amount of time elapses, the test is successful.

Editing an Application View

When you define an application view, you configure its connection parameters. After you add services and events, you may want to reconfigure the connection parameters or remove services and events.

Note: You cannot edit an application view after it is deployed. You have to undeploy it, edit it, and then deploy it again.

To undeploy and edit an application view:

1. Open the Application View. The Summary for Application View page appears.
2. Click Undeploy and then Confirm. The Summary for Application View page with Edit/Remove options appears.

Figure 2-21 Summary for Application View

The screenshot shows the 'Summary for Application View NewAppView.NewAppView' page. The page has a dark sidebar on the left with the 'Summary' tab selected. The main content area has a light blue header with the title and a breadcrumb trail: 'Application View Console > Server Configuration > WebLogic Console'. Below the header, a message states: 'This page shows the events and services defined for the NewAppView.NewAppView Application View.' The main content area is divided into two sections: 'Name: NewAppView' and 'Description:'. Below these, the 'Status: Not Deployed' is shown. The 'Available Actions:' section includes links for 'Edit', 'Remove', and a 'Deploy' button. A checkbox labeled 'Deploy Persistently?' is also present. At the bottom, there are tabs for 'Connection', 'Security', 'Deploy', 'Events and Services', and 'Summary'. The 'Events' and 'Services' sections are currently empty, showing only the 'ServStdSQL' service.

Summary for Application View NewAppView.NewAppView

Application View Console Server Configuration WebLogic Console

This page shows the events and services defined for the NewAppView.NewAppView Application View.

Name: NewAppView

Description:

Status: Not Deployed

Available Actions:

Edit

Remove

Deploy

Deploy Persistently?

Connection Security Deploy Events and Services

Events

Services

ServStdSQL

View Summary View Request Schema View Response Schema

3. Click Edit. The Application View Administration page appears.

Figure 2-22 Application View Administration

The screenshot shows the 'Application View Administration for NewAppView.NewAppView' page. The page has a dark sidebar on the left with the 'Administration' tab selected. The main content area has a light blue header with the title and a breadcrumb trail: 'Application View Console > WebLogic Console'. Below the header, a message states: 'This page allows you to add events and/or services to an Application View.' The main content area is divided into two sections: 'Description:' and 'Connection Criteria'. The 'Description:' section shows 'No description available for NewAppView.NewAppView.' with an 'Edit' link. The 'Connection Criteria' section includes fields for 'Message Bundle Base', 'Data Source Name', 'JDBC URL', 'Additional Log Category', 'Username', 'Root Log Category', 'Connection Info Type', 'Log Level', 'Log Configuration File', and 'Driver Name'. Below these fields, there is a link to 'Reconfigure connection parameters for NewAppView'. At the bottom, there are tabs for 'Events' and 'Services'. The 'Events' and 'Services' sections are currently empty, showing only the 'ServStdSQL' service. At the bottom of the page, there are 'Continue' and 'Save' buttons.

Application View Administration for NewAppView.NewAppView

Application View Console WebLogic Console

Select Connection Type

Configure Connection

Administration

Add Service

Add Event

Deploy Application View

This page allows you to add events and/or services to an Application View.

Description: No description available for NewAppView.NewAppView. Edit

Connection Criteria

Message Bundle Base: BEA_RDBMS_7_1

Data Source Name: OracleDSN

JDBC URL:

Additional Log Category: NewAppView

Username: system

Root Log Category: BEA_RDBMS_7_1

Connection Info Type: DSN

Log Level: DEBUG

Log Configuration File: BEA_RDBMS_7_1.xml

Driver Name:

Reconfigure connection parameters for NewAppView

Events

Services

ServStdSQL

Edit Remove Service View Summary View Request Schema View R

Continue Save

4. To reconfigure the application view's connection parameters, select Configure Connection in the left pane or the Reconfigure connection parameters link on the page. The Configuration Connection Parameters page appears. Follow the instructions in [“Step 4. Configure an RDBMS Connection” on page 2-5](#).
5. To add services or events, click Add Service or Add Event, respectively. Follow the instructions in [“Step 5A. Add a Service” on page 2-13](#) or [“Step 5B. Add an Event” on page 2-20](#).

3 Services and Events of the Adapter for RDBMS

This section provides information on the services and events of the Adapter for RDBMS. It contains the following topics:

- [Browse Metadata](#)
- [Services](#)
- [Events](#)
- [Handling Null Values](#)

For details on how to add a service or an event to an application view, see [“Step 5. Add Services and Events to an Application View” on page 2-13](#).

Browse Metadata

This is a design-time feature that the BEA WebLogic Adapter for RDBMS provides. Using this service, while you are adding services and events, you can browse the metadata information of the RDBMS through the Application View Console. You can copy and paste the required information into the fields, wherever applicable, ensuring correctness. The information you select will be used to build the schema definitions.

Using this service, you can browse the following RDBMS information:

- Schemas
- Tables and Columns
- Views and Columns
- Procedures and Parameters
- JDBC Data Types

It contains the following topics:

- [Browse Metadata While Adding a Service](#)
- [Browse Metadata While Adding an Event](#)

Browse Metadata While Adding a Service

- To browse DBMS while adding a service, do the following:
 1. On the Add Service page, click the Browse DBMS to view Tables, Views and Stored Procedures link. The list of schemas appears.

Figure 3-1 Browse Schemas

DBMS Schemas For Catalog:

[ADAMS](#)
[AURORA\\$ORB\\$UNAUTHENTICATED](#)
[BEAUSER](#)
[BLAKE](#)
[CLARK](#)
[COMMONPROD](#)
[CTXSYS](#)
[DBSNMP](#)
[DOCUMGR](#)
[EXP_IMP](#)
[FINTEL](#)
[FINTELN8](#)
[JONES](#)
[MDSYS](#)
[MISPROD](#)

2. Click the required schema. The table types appear.

Figure 3-2 Browse Table Types

DBMS Table Types:

[TABLE](#)
[VIEW](#)
[PROCEDURES](#)

3. Do one of the following:

- To browse tables and then the columns, do the following:
 - a. Click Table. The list of tables for the selected schema appears.

Figure 3-3 Browse Tables

Tables For: .BEAUSER

Table Name:

[ABBA](#)
[ABBAC](#)
[ACENTRIES](#)
[ACTIVECOLLABORATOR](#)
[ACTIVECONVDEF](#)
[ACTIVECONVERSATION](#)
[ACTIVECONVSTATE](#)
[ACTIVECPA](#)
[ACTIVECSpace](#)
[ACTIVEHUB](#)
[ACTIVEKEY](#)
[ACTIVEMESSAGE](#)
[ACTIVEMESSAGEEDATA](#)
[ACTIVEMESSAGEENVELOPE](#)
[ACTIVEMESSAGESTORE](#)

- b. Click the required table name. The table information for the selected table, containing the Column Name, Column Type, and Column Size, appears. You can copy the required information from this page and paste it into the fields, where applicable.

Figure 3-4 Browse Columns for Selected Table

DBMS Columns For Table: BONUS

ColumnName:	ColumnType:	ColumnSize:
ENAME	VARCHAR2	10
JOB	VARCHAR2	9
SAL	NUMBER	22
COMM	NUMBER	22

- To browse views and then columns, do the following:
 - a. Click View. The list of table views for the selected schema appears.

Figure 3-5 Browse Views

Views For: .BEAUSER

View Name:

[BASIC_VIEW](#)
[BASIC_VIEW1](#)
[BASIC_VIEW3](#)

- b. Click the required view. The table information for the selected view, containing the Column Name, Column Type, and Column Size, appears. You can copy the required information from this page and paste it into the fields, where applicable.

Figure 3-6 Browse Columns for the Selected View

DBMS Columns For View: BASIC_VIEW

ColumnName:	ColumnType:	ColumnSize:
CHAR1	CHAR	10
VARCHAR1	VARCHAR2	10
NUMERIC1	NUMBER	6

- If you have selected Stored Procedure from the Service Type drop-down list, you must enter the Stored Procedure Name.

To view the procedures and to pick a procedure, do the following:

- a. Click Procedures. The list of procedures for the selected schema appears.

Figure 3-7 Browse Procedures

Procedures For: .BEAUSER

Procedure Name:

Fill procedure name with selected procedure

BEAUSER.LOADLOBS.APPENDPIECE(?, ?)	<input type="radio"/>
BEAUSER.LOADLOBS.BEGINLOAD(?)	<input type="radio"/>
BEAUSER.LOADLOBS.BEGINREAD(?, ?)	<input type="radio"/>
BEAUSER.SALMAN.BONUS(?)	<input type="radio"/>
BEAUSER.SAMPAT.BONUS(?)	<input type="radio"/>
BEAUSER.MANISH.EMPLOYEE(?)	<input type="radio"/>
BEAUSER.LOADLOBS.ENDLOAD(?)	<input type="radio"/>
BEAUSER.LOADLOBS.ENDREAD(?)	<input type="radio"/>
BEAUSER.EXAMPLEINFOPROC()	<input type="radio"/>
BEAUSER.FUNCLOB(?, ?, ?)	<input type="radio"/>
BEAUSER.FUNCCHAR(?, ?, ?)	<input type="radio"/>

- b. Select the option button next to the required procedure, and click Fill procedure name with selected procedure.

To view the parameter information, click the required procedure name. The column information, containing Parameter Name, Parameter Type, Data

Type, and Parameter Size, appears. You can copy the required information from this page and paste it into the fields, where applicable.

Figure 3-8 Browse Parameters for the Selected Procedure

DBMS Columns For Procedure: BEAUSER.FUNCDATE
(?, ?, ?)

ParamName:	ParamType:	DataType:	ParamSize:
	RETURN	DATE	
X	IN	DATE	
Y	IN OUT	DATE	
Z	OUT	DATE	

- To browse JDBC Data Types while adding a service, click the JDBC Types link. The list of the data types the Adapter for RDBMS supports, appears.

Figure 3-9 Browse JDBC Data Types

JDBC Data Types

The RDBMS adapter will only support the following JDBC column types. Use the JDBC type instead of a Database specific type.

Simple	Complex	Date	Variable
BIT	ARRAY	DATE	VARCHAR
CHAR	BINARY	TIME	VARBINARY
DOUBLE	BLOB	TIMESTAMP	LONGVARBINARY
FLOAT	CLOB		LONGVARCHAR
REAL	DISTINCT		
BIGINT	JAVA_OBJECT		
DECIMAL	OTHER		
INTEGER	REF		
NUMERIC	STRUCT		
TINYINT			
SMALLINT			

Browse Metadata While Adding an Event

To browse DBMS and pick values while adding an event, do the following:

1. On the Add Event page, click the Browse DBMS link. The list of schemas appears.

Figure 3-10 Browse Schemas

DBMS Schemas For Catalog:

[ADAMS](#)
[AURORA\\$ORB\\$UNAUTHENTICATED](#)
[BEAUSER](#)
[BLAKE](#)
[CLARK](#)
[COMMONPROD](#)
[CTXSYS](#)
[DBSNMP](#)
[DOCMGR](#)
[EXP_IMP](#)
[FINTEL](#)
[FINTELEB](#)
[JONES](#)
[MDSYS](#)
[MISPROD](#)

2. Click the required schema. The table types appear.

Figure 3-11 Browse Table Types

DBMS Table Types:

[TABLE](#)
[VIEW](#)
[PROCEDURES](#)

3. To browse or pick table names and then the column names, do the following:
 - a. Click Table. The list of tables for the selected schema appears.

Figure 3-12 Browse or Pick Tables

Tables For: .BEAUSER

Table Name: Select:

Fill table name with selected table

ABBA	<input type="radio"/>
ABBAC	<input type="radio"/>
ACENTRIES	<input type="radio"/>
ACTIVECOLLABORATOR	<input type="radio"/>
ACTIVECONVDEE	<input type="radio"/>
ACTIVECONVERSATION	<input type="radio"/>
ACTIVECONVSTATE	<input type="radio"/>
ACTIVECPA	<input type="radio"/>
ACTIVECSpace	<input type="radio"/>
ACTIVEHUB	<input type="radio"/>

- b. Do one of the following:
 - To pick a table, select the option button next to it, and click Fill table name with selected table.

- To view column information, click the required table name. The table information, containing Column Name, Column Type, and Column Size, appears. To pick a column name, select the option button next to it, and click Fill column name with selected value.

Figure 3-13 Browse and Pick Column Names for the Selected Table

DBMS Columns For Table: ABBA			
ColumnName:	ColumnType:	ColumnSize:	Select:
Fill column name with selected value			
COLA	VARCHAR2	256	<input checked="" type="radio"/>
COLB	VARCHAR2	256	<input type="radio"/>

Note: You need to browse for column names, and pick the columns of type Timestamp only if you have selected the Select_By_Timestamp event. You can also directly click the link Browse Columns to do the same.

Services

A Service receives an XML request document from a client and invokes a specific function in the target RDBMS. The adapter then waits for the response from the RDBMS, and transforms the response from its RDBMS data format to an XML document that conforms to the response XML schema for the service.

It contains the following topics:

- [Standard SQL](#)
- [Parametrized SQL](#)
- [Stored Procedure](#)
- [Arbitrary SQL](#)

You can add any of these services to an application view. For details, see section [“Step 5. Add Services and Events to an Application View” on page 2-13](#).

Standard SQL

This service invokes a Standard SQL statement, without any parameters. You give the SQL statement, without any parameters, at design time.

Note: In the Arbitrary SQL service type, you give the SQL statement at run time.

The SQL executed can be either a DML (SELECT, INSERT, UPDATE, or DELETE) statement or a DDL (CREATE, ALTER, or DROP) statement. The resultset is converted into the response XML document, based on the XML schema formed during design time.

Parametrized SQL

This service invokes a parametrized SQL defined in an RDBMS. You should 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 giving the values for the parameters specified.

Based on the data type of the input provided, appropriate binding of parameter values is done. The prepared SQL statement is executed, and the resultset is converted into the response XML document, based on the XML schema formed during design time.

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. The stored procedure 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, appropriate binding of parameter values is done. The stored procedure is executed and the resultset is converted into a response XML document, based on the XML schema formed during design time.

Because DBMS manufacturers produce JDBC agents with varying degrees of functionality, stored procedures can be executed in several different ways within the service.

Note: Oracle subprograms (stored procedures) can be called whether they reside as functions, stored procedures, or stored procedures located within packages.

Arbitrary SQL

This service invokes an arbitrary SQL statement at run time.

Note: In the Standard SQL service type, you give the SQL statement at design time.

The SQL given at run time can either be a DML (SELECT, INSERT, UPDATE, or DELETE) or a DDL (CREATE, ALTER, or DROP) statement. The resultset is converted into the response XML document, based on the type of statement executed.

Events

When you want to be notified of a particular occurrence in an RDBMS, you can register the notification request as an event in the application view. 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.

The event provides the following events:

- [Trigger Event](#)
- [Select By Timestamp Event](#)
- [Select Then Delete Event](#)

The event provides the following event types for the various events:

Table 3-1 Event Types for Events

Event	Event Type Insert	Event Type Update	Event Type Delete
Trigger	Yes	Yes	Yes

Table 3-1 Event Types for Events (Continued)

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

The event supports the polling of relational tables for incoming (inserted), modified (updated) and deleted data. This highly configurable event, reads one or more rows from the table of interest specified and creates an XML document representing the data pertaining to the record of interest.

You can add any of these events to an application view. For details, see section [“Step 5. Add Services and Events to an Application View”](#) on page 2-13.

Trigger Event

This event can be used to notify an Insert, Update, or Delete event occurring in a database table.

Select By Timestamp Event

This event can be used to notify an Insert and Update event occurring in a database table. The table of interest should have columns representing `Created Timestamp` and `Updated Timestamp`. Insert event can be configured only if `Created Timestamp` is maintained as a part of the table, and Update event can be configured only if `Updated Timestamp` is maintained as a part of the table.

Select Then Delete Event

This event can be used to notify records of interest based on a select query given on a table of interest. The event polling mechanism looks for records based on the select query and packages the result into one or more rows and creates an XML document representing record of interest. The user can opt to either delete or retain the records selected after the processing of events.

Handling Null Values

Relational databases support the notion of null values in a data field. On the event side, it is sometimes important to ascertain whether a field in the output contains a null value, or is simply an empty string. On the service side, it is important to properly set null values when performing an insert or update through a parametrized query or giving input to a stored procedure.

Null Values in Events

For events, null values in the data being read are denoted in the result document by the attribute `isNull='true'`. The following table depicts the behavior of the BEA WebLogic Adapter for RDBMS in handling null values for events.

Table 3-2 Handling Null Values for Events

Database Value	Database Nullable	XML Value	XML Attribute
Spaces	Yes	Spaces	<code>isNull='false'</code>
Spaces	No	Spaces	N/A
Null	Yes	None	<code>isNull='true'</code>
Null	No	N/A	N/A

Table 3-2 Handling Null Values for Events (Continued)

Database Value	Database Nullable	XML Value	XML Attribute
Text	Yes	Text	isNull='false'
Text	No	Text	N/A

The attribute `isNull` is used to indicate fields containing a null value. For the Field and Column formats, the `isNull` attribute is set to `'true'` in the case of 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.

Listing 3-1 XML Generated by an Event

```
<?xml version="1.0"?>
<!DOCTYPE Output>
<Output>
  <Rows>
    <Row>
      <COMM isNull="false">88</COMM> text/nullable
      <ENAME>name</ENAME> text/not nullable
      <JOB isNull="false"> </JOB> spaces/nullable
      <SAL isNull="false">9000</SAL>
    </Row>
    <Row>
      <COMM isNull="false">44</COMM>
      <ENAME> </ENAME> spaces/not nullable
      <JOB isNull="false">program</JOB>
      <SAL isNull="true"></SAL> null/nullable
    </Row>
  </Rows>
</Output>
```

Null Values in Services

For services, you need to properly designate null values in order to insert or update values into the RDBMS. The BEA WebLogic Adapter for RDBMS propagates null values into SQL under the following code.

```
<?xml version="1.0"?>
<!DOCTYPE Input>
<Input>
  <INPUT_VALUE isNull="true"></INPUT_VALUE>
</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 sent to the RDBMS is
`INSERT INTO ANOTHER_TABLE VALUES (NULL)`

A Schema Formats for Services and Event Schemes

This section lists the request and response XML schemas for services and the event XML schemas for events. It contains the following topics:

- [Schemas for Services](#)
- [Schemas for Event Schemes](#)

Schemas for Services

The Adapter for RDBMS provides four service types:

- [Standard SQL](#)
- [Parametrized SQL](#)
- [Stored Procedure](#)
- [Arbitrary SQL](#)

The following sections show the request and response schemas associated with each service type.

Standard SQL

This section shows the request and response schemas associated with the Standard SQL service type.

Request Schema Format

The request schema format would be as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Input" type="xsd:string"/>
</xsd:schema>
```

Response Schema Format

The response schema format would be as follows:

For SELECT :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Output">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Rows">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Row" maxOccurs="unbounded" minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="FIELD_NAME1" type="SOM_TYPE1"/>
                    eg;
                    <xsd:element name="ENAME" type="xsd:string"/>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Note: *FIELD_NAME1* and *SOM_TYPE1* are the field name and SOM type corresponding to the field selected.

For INSERT, UPDATE, DELETE :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Output">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="RowsAffected" type="xsd:int"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

For CREATE, ALTER, and DELETE :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Output">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Success" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Parametrized SQL

This section shows the request and response schemas associated with the Parametrized SQL service type.

Request Schema Format

The request schema format would be as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Input">
    <xsd:complexType>
      <xsd:sequence>
```

```
<xsd:element name="Row" maxOccurs="unbounded">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="PARAMETER_NAME">
        <xsd:complexType>
          <xsd:simpleContent>
            <xsd:extension base="SOM_Type">
              <xsd:attribute name="isNull" use="required" type="xsd:boolean"/>
            </xsd:extension>
          </xsd:simpleContent>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Note: *PARAMETER_NAME* is the parameter name specified in the parametrized SQL. *SOM_Type* is the schema type for the JDBC data type specified in the parametrized SQL. For example in the above schema, *PARAMETER_NAME* could be *employeeId* and the *SOM _Type* could be *xsd:int*.

Response Schema Format

The response schema format would be as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Output">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Rows">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Row" maxOccurs="unbounded" minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="FIELD_NAME" type="SOM_Type"/>
                    eg.,
                    <xsd:element name="ENAME" type="xsd:string"/>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```

    </xsd:complexType>
  </xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

Stored Procedure

This section shows the request and response schemas associated with the Stored Procedure service type.

Request Schema Format

The request schema format would be as follows:

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Input">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="IN_PARAMETER" type="SOM_Type" />
        <xsd:element name="IN_OUT_PARAMETER" type="SOM_Type" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

Response Schema Format

The response schema format would be as follows:

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Output">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Params" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="OUT" type="SOM_Type" />
              <xsd:element name="IN-OUT" type="SOM_Type" />
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

```
</xsd:complexType>
</xsd:element>
<xsd:element name="ReturnValue" type="xsd:dateTime" minOccurs="0"/>
<xsd:element name="RefCursor" minOccurs="0">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Row" maxOccurs="unbounded" type="xsd:string"
minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Note: The output will contain the following:

- OUT parameters
- IN-OUT parameters
- RETURN Value - In the case of functions
- REFCURSOR - If the CURSOR is returned by the stored procedure, this will have the resultant rows.

Arbitrary SQL

This section shows the request and response schemas associated with the Arbitrary SQL service type.

Request Schema Format

The request schema format would be as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Input">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Sql" type="xsd:normalizedString"/>
      </xsd:sequence>
    </xsd:complexType>
```

```

    </xsd:element>
  </xsd:schema>

```

Response Schema Format

The response schema format would be as follows:

For SELECT :

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Output">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Rows">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Row" maxOccurs="unbounded" minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="FIELD_NAME1" type="SOM_TYPE1"/>
                    eg;
                    <xsd:element name="ENAME" type="xsd:string"/>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

Note: *FIELD_NAME1* and *SOM_TYPE1* are the field name and SOM type corresponding to the field selected.

For INSERT, UPDATE, DELETE :

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Output">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="RowsAffected" type="xsd:int"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

```
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

For CREATE, ALTER, and DELETE :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Output">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Success" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Schemas for Event Schemes

The Adapter for RDBMS provides three event schemes:

- Trigger
- Select By Timestamp
- Select Then Delete

The adapter provides the following event types for the event schemes:

Table A-1 Event Types for Event Schemes

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

The event schema format is the same for all event types. It is based on the fields of interest in the table specified in the event mechanism.

Event Schema Format

The event schema format would be as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Output">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Rows">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Row" maxOccurs="unbounded" minOccurs="0">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="FIELD_NAME1" type="SOM_TYPE1"/>
                    eg;
                    <xsd:element name="ENAME" type="xsd:string"/>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```


B Supported Data Types

The following are the data types that the Adapter for RDBMS supports.

Table 3-3 Supported Data Types

Database	Data Types Successfully Tested
Oracle	<ul style="list-style-type: none">■ char■ varchar■ real■ float■ date■ double■ integer■ numeric■ refcursor (Multiple resultsets not supported)■ smallint■ clob (Not supported in Stored Procedures)■ blob (Not supported in Stored Procedures)■ null

Table 3-3 Supported Data Types (Continued)

Database	Data Types Successfully Tested
SQL Server	<ul style="list-style-type: none">■ datetime■ numeric■ decimal■ real■ double■ float■ integer■ smallint■ char■ varchar■ longvarchar■ resultsets (Multiple)■ null
DB2 UDB	<ul style="list-style-type: none">■ timestamp■ date■ time■ real■ bit■ double■ float■ integer■ smallint■ char■ varchar■ clob (Not supported in Stored Procedures)■ blob (Not supported in Stored Procedures)■ null

Table 3-3 Supported Data Types (Continued)

Database	Data Types Successfully Tested
Sybase Adaptive Server	<ul style="list-style-type: none">■ binary■ bit■ char■ datetime■ decimal■ double precision■ float■ image■ integer■ nchar■ numeric■ nvarchar■ real■ smalldatetime■ smallint■ text (not for trigger events)■ timestamp■ tinyint■ varbinary■ varchar

C Commonly Used Drivers

This section lists the commonly used drivers with which the BEA WebLogic Adapter for RDBMS has been tested on the following operating systems:

- Sun Solaris 8 (with Sun recommended patches)
- Microsoft Windows 2000 (with Service Pack 2)
- Red Hat Linux 7.3, Enterprise Linux 2.1

It also lists the driver class names and JDBC URLs corresponding to the drivers.

Note: The Adapter for RDBMS should be used in conjunction with a JDBC 2.1 compliant driver.

Oracle Thin Driver

Driver Files	<code>classes12.zip</code>
--------------	----------------------------

For Normal Connection

Driver Class Name	<code>oracle.jdbc.driver.OracleDriver</code>
-------------------	--

JDBC URL	<code>jdbc:oracle:thin://hostname:port/schema_name</code>
----------	---

Example	<code>jdbc:oracle:thin:@172.19.139.58:1521:UAT02</code>
---------	---

C Commonly Used Drivers

Oracle Thin Driver (Continued)

For XA Connection

Driver Class Name	<code>oracle.jdbc.xa.client.OracleXADataSource</code>
JDBC URL	<code>jdbc:oracle:thin://hostname:port/schema_name</code>
Example	<code>jdbc:oracle:thin:@172.19.139.58:1521:UAT02</code>

MS SQL Server 2000 JDBC Driver

Driver Files	<ul style="list-style-type: none">■ <code>msutil.jar</code>■ <code>msbase.jar</code>■ <code>mssqlserver.jar</code>
--------------	--

For Normal Connection

Driver Class Name	<code>com.microsoft.jdbc.sqlserver.SQLServerDriver</code>
JDBC URL	<code>jdbc:microsoft:sqlserver://hostname:port</code>
Example	<code>jdbc:microsoft:sqlserver://172.19.138.24:1433</code>

For XA Connection

Driver Class Name	<code>com.microsoft.jdbcx.sqlserver.SQLServerDataSource</code>
JDBC URL	<code>jdbc:microsoft:sqlserver://hostname:port;SelectMethod=?;DatabaseName=?;ServerName=?</code>
Example	<code>jdbc:microsoft:sqlserver://172.19.138.24:1433;SelectMethod=cursor;DatabaseName=AdapterTesting;ServerName=itpl-025019</code>

DB2 JDBC Driver (JDBC 2.0 Compliant)

Driver Files	db2java.zip
--------------	-------------

For Normal Connection

Driver Class Name	COM.ibm.db2.jdbc.app.DB2Driver
-------------------	--------------------------------

JDBC URL	jdbc:db2:DatabaseName
----------	-----------------------

Example	jdbc:db2:DWCTRLDB
---------	-------------------

For XA Connection

Driver Class Name	COM.ibm.db2.jdbc.DB2XADataSource
-------------------	----------------------------------

JDBC URL	jdbc:db2:DatabaseName=?; SupportsLocalTransaction=?
----------	--

Example	jdbc:db2:DatabaseName=DWCTRLDB; SupportsLocalTransaction=true
---------	--

Sybase Driver by DataDirect Technologies Release 3.3 (JDBC 2.0 Compliant)

Driver Files	base.jar util.jar sybase.jar
--------------	------------------------------------

For Normal Connection

Driver Class Name	com.ddtek.jdbc.sybase.SybaseDriver
-------------------	------------------------------------

JDBC URL	jdbc:datadirect:sybase://hostname:port;databaseName=?
----------	---

Example	jdbc:datadirect:sybase://172.16.192.72:2050;databaseName=sybaseqa
---------	---

C *Commonly Used Drivers*

Sybase Driver by DataDirect Technologies Release 3.3 (JDBC 2.0 Compliant) (Continued)

For XA Connection

Driver Class Name	<code>com.ddtek.jdbcx.sybase.SybaseDataSource</code>
JDBC URL	<code>jdbc:datadirect:sybase://hostname:port;DatabaseName=?;ServerName=?;PortNumber=?;SelectMethod=cursor</code>
Example	<code>jdbc:datadirect:sybase://172.16.192.72:2048;DatabaseName=sybaseqa;ServerName=172.16.192.72;PortNumber=2048;SelectMethod=cursor</code>

D Logging

This section describes the logging for services and events. It contains the following topics:

- [About Logging](#)
- [Levels of Logging](#)
- [Logging and Performance](#)
- [Creating or Modifying the Logging Level for an Event](#)

About Logging

Logging is an essential feature of an adapter. Most adapters are used to integrate different applications and do not interact with end users while processing data. Unlike a front-end component, when an adapter encounters an error or warning condition, it cannot stop processing and wait for an end user to respond.

Moreover, many business applications that are connected by adapters are mission-critical. For example, an adapter might be required to keep an audit report of every transaction with an RDBMS. Consequently, adapter components should provide both accurate logging and auditing information. The adapter logging framework is designed to accommodate both logging and auditing.

Levels of Logging

Logging is provided by both the BEA adapter framework and by the BEA WebLogic Adapter for RDBMS.

The BEA WebLogic Integration framework provides five distinct levels of logging:

Table 3-4 Logging Settings

Level	Indicates
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.
DEBUG	Information used to determine how the adapter is working internally.

Logging and Performance

The additional logging capabilities provided by the adapter are not strictly hierarchic; rather they are categorized. These loggings are designed to provide debugging help with minimum effect on performance. All internal adapter loggings are controlled through the additional logging settings, and all additional settings route their output to the standard debug setting.

If you configure the adapter for additional settings and do not configure standard logging settings, the loggings are generated but never appear in output. This affects performance, as the production of the logging continues even though you receive no benefit of the additional logging information.

Creating or Modifying the Logging Level for an Event

To create or modify the WebLogic framework logging level for an event:

1. Open the WebLogic Server Console in a browser using the following URL:

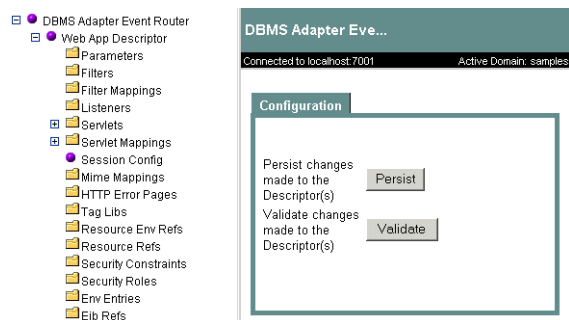
`http://localhost:port/console/`

Here, *localhost* represents the machine on which WebLogic Server is running and *port* represents the listening port.

For example, `http://localhost:7001/console/`

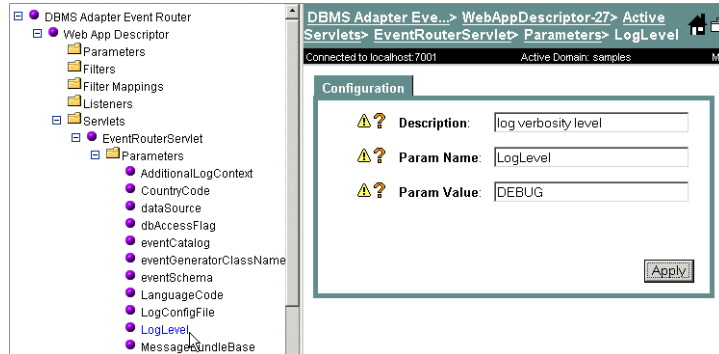
2. Enter the user name and password for the server. The WebLogic Server Console opens.
3. In the left pane, select Deployments→Web Applications.
4. Select the EventRouter corresponding to the adapter that should be logged. For example, if you require logs for an RDBMS event, select `BEA_RDBMS_7_1_EventRouter.war`.
5. Click Edit Web Application Deployment Descriptors. The following page appears.

Figure 3-14 Edit Web Application Deployment Descriptors



6. In the left pane, select Servlets→EventRouterServlet→Parameters→LogLevel. The following page appears with the default values.

Figure 3-15 Log Level



7. Enter a text description of the parameter in the Description field.
 8. Enter the name of the parameter in the Param Name field.
 9. Enter the parameter value in the Param Value field. For details, see [Table 3-4 “Logging Settings.”](#)
- Note:** For maximum logging, enter `DEBUG`. This is recommended to obtain optimum debugging information for BEA support personnel.
10. Click Apply to save the new log level.
 11. Click the EventRouter Servlet.
 12. Select Apply to persist the logging changes. This change need only be made once. It is set for all events associated with a given adapter.
 13. Return to the WebLogic Server Console.
 14. In the left pane, select Applications.
 15. Select the adapter whose EventRouter you have modified in the previous steps. The Deployment tab on the right pane displays the following adapter components:
 - `BEA_RDBMS_7_1.rar`
 - `BEA_RDBMS_7_1.Web.war`
 - `BEA_RDBMS_7_1_EventRouter.war`
 16. Redeploy the EventRouter by clicking the Redeploy button to the right of `BEA_RDBMS_7_1_EventRouter.war`.

E 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.

Error	weblogic.utils.NestedError: Unexpected Exception: detailsjava.lang.reflect.InvocationTargetException: java.security.PrivilegedActionException: java.sql.SQLException: weblogic.common.ResourceException: Access not allowed
Description	You may get this error while trying to configure connection parameters. The administrative server user name does not have access to the connection pool.
Action	For details on how to provide the administrative server user access to the connection pool, see section “Step 4A. Create a Connection Pool” on page 2-6.
Error	com.bea.wlai.client.DeploymentException: Exception thrown in managed resource while trying to invoke operation deployApplicationViewAtRuntime Start server side stack trace: --
Description	You may get this exception while deploying an event. The EVENT_POLLING_METADATA table is missing.

Error	com.bea.wlai.client.DeploymentException: Exception thrown in managed resource while trying to invoke operation deployApplicationViewAtRuntime Start server side stack trace: --
Action	Create the EVENT_POLLING_METADATA table. For details, see section “Prepare the Adapter for RDBMS,” in <i>BEA WebLogic Adapter for RDBMS Installation and Configuration Guide</i> for WebLogic Integration 7.0 or 2.1 at the following URL: http://edocs.bea.com/wlapters/rdbms/docs71/index.html
Error	Errors occurred while processing your request.
Description	You may get this exception while creating a connection pool. The driver files are not included in the CLASSPATH.
Action	Place the driver JAR files in the WebLogic Server CLASSPATH. For details, see section “Configure Connection,” in <i>BEA WebLogic Adapter for RDBMS Installation and Configuration Guide</i> for WebLogic Integration 7.0 or 2.1, at the following URL: http://edocs.bea.com/wlapters/rdbms/docs71/index.html
Error	Timed out
Description	Events may start timing out. It could be due to either of the following reasons: <ul style="list-style-type: none">■ The WebLogic Server might have been abnormally shut down 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.

Error	Timed out
Action	<ul style="list-style-type: none">■ If the WebLogic Server was abnormally shut down, then:<ol style="list-style-type: none">1. Undeploy the application views that have events defined in them.2. Redeploy these application views.■ 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.
Error	com.bea.wlai.client.DeploymentException: Error deploying application view AppViewName: could not find CCI ConnectionFactory instance with lookup to JNDIcontext:com.bea.wlai.connectionFactories.AppViewName_connectionFactoryInstance
Description	The BEA_RDBMS_7_1.jar file is not in the CLASSPATH.
Action	Ensure that the BEA_RDBMS_7_1.jar file is in the CLASSPATH. For details, see section “Step 4. Extract the JARs and Adjust the CLASSPATH,” in <i>BEA WebLogic Adapter for RDBMS Installation and Configuration Guide</i> for WebLogic Integration 2.1 at the following URL: http://edocs.bea.com/wlapters/rdbms/docs71/index.html

Index

A

Adapter for RDBMS

- about 1-1
- key concepts 1-3
- overview 1-2
- supported databases 1-2

Application View Console, logging on 2-3

application views

- about 1-3
- adding a folder 2-4
- adding event 2-20
- adding service 2-13
- before defining 2-1
- configuring RDBMS connection 2-5
- defining 2-2
- deploying 2-25
- editing 2-29

B

BEA

- contacting ix
- product documentation viii
- WebSupport ix

browse metadata 3-1

C

connection

- configuring parameters 2-11
- creating connection pool 2-6
- creating data source 2-9

conventions, documentation x

D

data types, supported 2-1

databases, supported 1-2

documentation conventions x

drivers, tested 3-1

E

e-docs web site viii

error messages 5-1

events

- about 1-5
- select by timestamp 2-24
- select then delete 2-24
- testing 2-28
- trigger 2-24
- understanding 2-23

H

handling null values 3-11

I

isolation levels 2-15

L

levels, isolation 2-15

logging

- about D-1

- creating level for event D-3
- levels D-2
- modifying level for event D-3
- performance D-2

T

- troubleshooting 5-1

M

- metadata, browsing 3-1

N

- null values
 - in events 3-11
 - in services 3-13

R

- related documents ix

S

- schemas
 - about 1-6
 - arbitrary SQL A-6
 - event schemes A-8
 - parametrized SQL A-3
 - services A-1
 - standard SQL A-2
 - stored procedure A-5
- services
 - about 1-4
 - arbitrary SQL 2-19
 - parametrized SQL 2-17
 - standard SQL 2-17
 - stored procedure 2-18
 - testing 2-27
 - understanding 2-16
- support, technical ix
- supported data types
 - DB2 UDB 2-2
 - Oracle 2-1
 - SQL Server 2-2
 - Sybase Adaptive Server 2-3