BEAAquaLogic Data Services Platform

Administration Guide

Note: Product documentation may be revised post-release and made available from the following BEA e-docs site:

http://e-docs.bea.com/aldsp/docs25/index.html

Version: 2.5
Document Date: June 2005
Revised: March 2007

Contents

3. Deploying AquaLogic Data Services Platform Applications

4. Using the AquaLogic Data Services Platform Console

5. Configuring AquaLogic Data Services Platform Applications

6. Securing AquaLogic Data Services Platform Resources

7. Configuring the Query Results Cache

8. Viewing Metadata

9. Working With Audit and Log Information

Administration Guide

Overview of AquaLogic Data Services Platform Administration

This chapter introduces AquaLogic Data Services Platform administration. The chapter also introduces the concept of WebLogic domains, and explains how to create new WebLogic domains for AquaLogic Data Services Platform or add AquaLogic Data Services Platform to an existing WebLogic domain.

The chapter contains the following sections:

- AquaLogic Data Services Platform Administration Tasks
- Understanding WebLogic Domains and Administration
- License Key Updates

Note: AquaLogic Data Services Platform was previously named Liquid Data. Some artifacts of the original name remain in the product, installation path, and components.

AquaLogic Data Services Platform Administration Tasks

AquaLogic Data Services Platform is integration software that unifies data programming through the use of data services. Since it is deployed to a WebLogic Server, you can administer AquaLogic Data Services Platform through the underlying WebLogic Platform. Administrative tasks that you can perform through WebLogic include deployment, starting and stopping the server, configuring connection pools and data sources, logging, and others. The WebLogic Platform provides extensive tools and capabilities for configuring and maintaining a large-scale, production-level integration platform.

However, there are several administrative tasks that are specific to the AquaLogic Data Services Platform. Generally these arise from AquaLogic Data Services Platform's role as data integration software and include managing applications that use AquaLogic Data Services Platform data services, and configuring data caching and access control for data services.

This document introduces you to general WebLogic administration and describes several common tasks. However, its primary focus is on AquaLogic Data Services Platform-specific tasks. For complete information on WebLogic administration, see *Configuring and Managing WebLogic Server* at:

http://e-docs.bea.com/wls/docs81/adminguide/index.html

Securing Data

AquaLogic Data Services Platform leverages the security model of the WebLogic Platform to ensure data security. WebLogic uses security policies that control access to deployed resources based on user credentials or other factors.

AquaLogic Data Services Platform extends WebLogic security to enable you to apply policies to its data resources at a range of levels, from the application to individual data elements. In addition, you can secure resources based on data values (called instance-level security). For example, you can secure objects if an element value exceeds a specific threshold.

For details, see Chapter 6, "Securing AquaLogic Data Services Platform Resources."

Caching Query Results

AquaLogic Data Services Platform can cache query results for data service functions to enhance overall AquaLogic Data Services Platform performance. Caching data alleviates the burden on back-end resource and improves data request response times from the client's perspective. If you want to cache data service function results, you must explicitly enable results caching in the AquaLogic Data Services Platform Console.

For more information, see Chapter 5, "Configuring AquaLogic Data Services Platform Applications."

Data Service Metadata

Traditionally, enterprises have lacked a universal mechanism for advertising availability of data resources across source types, or for communicating information about those resources. AquaLogic Data Services Platform provides this capability through dynamically generated metadata.

Data service metadata serves these primary purposes:

- It helps developers create client applications that use the information made available by AquaLogic Data Services Platform by revealing what data is available and how to use it.

- It helps administrators maintain AquaLogic Data Services Platform by providing a mechanism to gauge effects of changes in underlying data sources upon a data service deployment.

Metadata provides information on data services such as their public functions, datatypes, data lineage, and more. It also provides *where used* information, showing dependencies between data services.

For more information, see Chapter 8, "Viewing Metadata."

Understanding WebLogic Domains and Administration

A WebLogic *domain* is a collection of WebLogic resources managed as a single unit. A WebLogic domain includes one or more instances of a WebLogic Server and may include WebLogic Server clusters. For more information about domains, see "WebLogic Server Domains" in *Configuring and Managing WebLogic Server*.

The WebLogic Administration Console is a web-based interface for configuring and monitoring a WebLogic domain. In cases when the domain has more than one server, one of the servers is designated as the *Administration Server* for the domain. The Administration Server then serves as the central point of control for an entire domain. If there is only one server in a domain, that server is the Administration Server in addition to the other functions it provides. Any other servers in a domain are *Managed Servers*.

The Administration Console enables you to perform most of the configuration tasks for domains and servers. It is also where you deploy the AquaLogic Data Services Platform application to your domain.

AquaLogic Data Services Platform supplements the WebLogic Administration Console with the AquaLogic Data Services Platform Administration Console (named *ldconsole*). The AquaLogic Data Services Platform Console gives you access to configuration settings specific for AquaLogic Data Services Platform, such as caching and data resource security controls as well as metadata information.

Understanding the Relationship of AquaLogic Data Services Platform to WebLogic Domains

AquaLogic Data Services Platform is an application and a set of associated resources that are deployed in a WebLogic domain. Starting, stopping, and managing AquaLogic Data Services Platform is accomplished by starting the WebLogic Server in the domain in which AquaLogic Data Services

Platform is deployed, and using the Administration Console for that server to configure and manage AquaLogic Data Services Platform resources for that domain.

Creating a New Domain

AquaLogic Data Services Platform applications work with WebLogic domains that have been provisioned for AquaLogic Data Services Platform. You can use the BEA WebLogic Configuration Wizard to create such domains.

To create a new domain provisioned with AquaLogic Data Services Platform:

1. On Windows systems, choose Programs $\rightarrow$ BEA WebLogic Platform 8.1 $\rightarrow$ Configuration Wizard.

2. In the wizard, choose Data Service Platform Domain as the domain type.

3. Follow the on-screen instructions to complete the initial configuration of the domain.

For more information on creating domains, see "Creating a New WebLogic Domain" in the WebLogic Platform documentation.

Provisioning an Existing Domain for AquaLogic Data Services Platform

In cases when you have an existing WebLogic Server domain in which you want to use AquaLogic Data Services Platform, you simply need to provision the domain for AquaLogic Data Services Platform. You can do this through the Configuration Wizard by following these steps:

1. Open the Configuration Wizard:

    Start $\rightarrow$ Programs $\rightarrow$ BEA Weblogic Platform 8.1 $\rightarrow$ Configuration Wizard

2. Select the option: Extend an existing WebLogic configuration.

3. Select the domain you wish to enable for AquaLogic Data Services Platform (such as: weblogic81/samples/domains/portal).

4. Select the AquaLogic Data Services Platform extension.

For information on selecting domain setting options see Tutorials: Using the Configuration Wizard (http://edocs.bea.com/platform/docs81/confgwiz/tutorials.html).

Once a domain is provisioned with AquaLogic Data Services Platform, you can deploy applications that contain AquaLogic Data Services Platform projects.

For additional information see Chapter 3, "Deploying AquaLogic Data Services Platform Applications."

Understanding Console Users

The AquaLogic Data Services Platform Administration Console is targeted for two types of users:

- Client developers

- AquaLogic Data Services Platform administrators

Configuration features of the console can be disabled based on the role of the user, so that caching and security controls, for example, are not displayed to the developer user. The administrative user, on the other hand, can access all pages in the console.

For more information, see Chapter 6, "Securing AquaLogic Data Services Platform Resources."

License Key Updates

AquaLogic Data Services Platform requires a valid product license to run. The AquaLogic Data Services Platform license is included as a component in the WebLogic Server license file, `license.bea`. If you need to apply or update a AquaLogic Data Services Platform license file (known as a *Liquid Data license file*), use the BEA UpdateLicense utility to update the `license.bea` file.

For details about BEA product licensing, see Installing and Updating WebLogic Platform License Files in *Installing WebLogic Platform* of the WebLogic Server documentation.

Using the WebLogic Server Console

This chapter introduces the WebLogic Server Administration Console, and explains how to start and stop the WebLogic Server.

The chapter contains the following sections:

- Using the Administration Console to Manage AquaLogic Data Services Platform-enabled Applications

- Starting the WebLogic Server

- Launching the Administration Console

- Exploring the Administration Console

- Stopping the WebLogic Server

Using the Administration Console to Manage AquaLogic Data Services Platform-enabled Applications

When deployed on a AquaLogic Data Services Platform provisioned domain, AquaLogic Data Services Platform-enabled applications become *managed resources* known to the WLS JMX management framework. This means that you can manage many of the runtime properties of a deployed AquaLogic Data Services Platform application using the WebLogic Administration Console.

Before you can configure or manage a AquaLogic Data Services Platform application, you must start the WebLogic Server on which it is deployed. When you run the `startWebLogic.cmd` (Windows) or `startWebLogic.sh` (UNIX) command for a domain, WebLogic Server is started, and the AquaLogic

Data Services Platform applications and resources specified in the configuration file for the domain are automatically deployed on the server.

Note: The instructions that follow are tailored for starting the WebLogic Server in conjunction with AquaLogic Data Services Platform. For general information on starting the WebLogic Server, see Starting and Stopping WebLogic Servers (`http://edocs.bea.com/wls/docs81/ConsoleHelp/startstop.html`) in the WebLogic Server documentation.

Starting the WebLogic Server

The instructions in this section describe how to start WebLogic Server (WLS) in a standalone WebLogic domain.

Note: If you are already running an instance of WebLogic Server that uses the same listener port as the one to be used by the server you are starting, you must stop the first server before starting the second server.

To start the server:

1. At the command prompt, navigate to the domain directory.

   The domain directory is `BEA_HOME`/`user_projects`/`domain_name`. An example could be `c:\bea\user_projects\mydomain`.

2. Run the server startup script: `startWebLogic.cmd` (Windows) or `startWebLogic.sh` (UNIX).

   The startup script displays a series of messages, finally displaying a message similar to the following:

   ```
   <Dec 8, 2004 3:50:42 PM PDT> <Notice> <WebLogicServer> <000360> <Server
   started in RUNNING mode>
   ```

After starting the server, you can start the WebLogic Administration Console.

Launching the Administration Console

The Administration Console is the web-based management interface for a WebLogic domain.

To launch the Administration Console:

1. Start the WebLogic Server in the WebLogic domain in which AquaLogic Data Services Platform is deployed.

   For more information, see "Starting the WebLogic Server."

2. Using a web browser, open the following URL:

   `http://`*`hostname`*`:`*`port`*`/console`

   Where

   – *`hostname`* is the machine name or IP address of the host server

   – *`port`* is the address of the port on which the host server is listening for requests (7001 by default)

   For example, to start the Administration Console for a local instance of WebLogic Server (running on your own machine), type the following URL in a Web browser address field:

   `http://localhost:7001/console/`

   If you started the Administration Server using Secure Socket Layer (SSL), you must add `s` after `http`, as follows:

   `https://`*`hostname`*`:`*`port`*`/console`

3. When the login page appears, enter the user name and password you used to start the Administration Server.

If you have your browser configured to send HTTP requests to a proxy server, then you may need to configure your browser so that it does not send Administration Server HTTP requests to the proxy. When the Administration Server is on the same machine as the browser, ensure that requests sent to localhost or 127.0.0.1 are not sent to the proxy.

Exploring the Administration Console

The WebLogic Administration Console uses the following panes to enable you to navigate and display information about entities in a WebLogic domain:

- **Navigation pane.** Enables you to browse servers, clusters, deployments, applications, and more.

- **Content pane.** Displays detailed information about entities selected in the Navigation pane.

Figure 2-1 illustrates the WebLogic Administration Console user interface.

Figure 2-1 Home Page of the WebLogic Server Administration Console
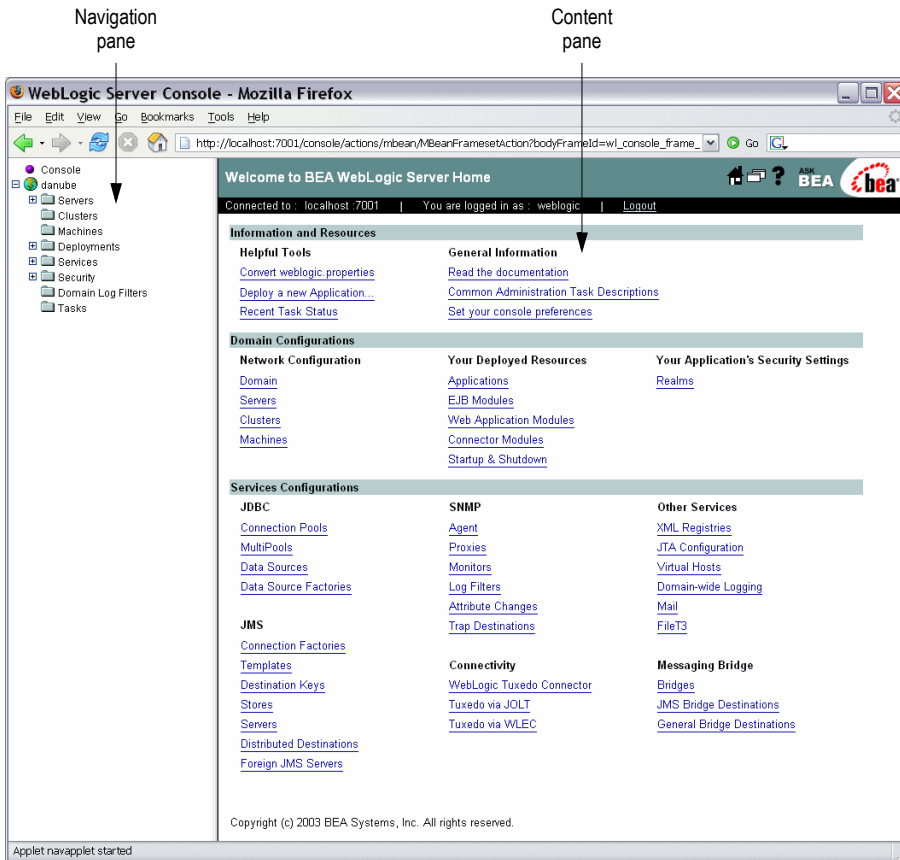
When you start WebLogic Administration Console, the general administration page is shown in the Content pane, as illustrated in Figure 2-1. You can use the topic links on the home page initially to navigate to top level resource nodes, or use the Navigation pane which contains a hierarchical tree — a domain tree — for navigating to tables of data, configuration pages and monitoring pages, or accessing logs.

Selecting an item in the domain tree enables you to display a table of data for resources of a particular type (such as WebLogic Servers) or configuration and monitoring pages for a selected resource.

You can expand and collapse nodes in the tree by clicking the + and - signs next to the nodes as follows:

- A plus sign is (+) next to a node indicates that the node contains subnodes; it is expandable. To expand a collapsed container node, click on the + beside it. Its next level subnodes appears.

- A minus sign (-) next to a node indicates that the node is a container that is fully expanded. To collapse an expanded container node, click on the - beside it.

- A node with neither - or + beside is either an empty folder with no resources as yet or a fixed resource with no subnodes. As you add resources to folders, these will become expandable containers.
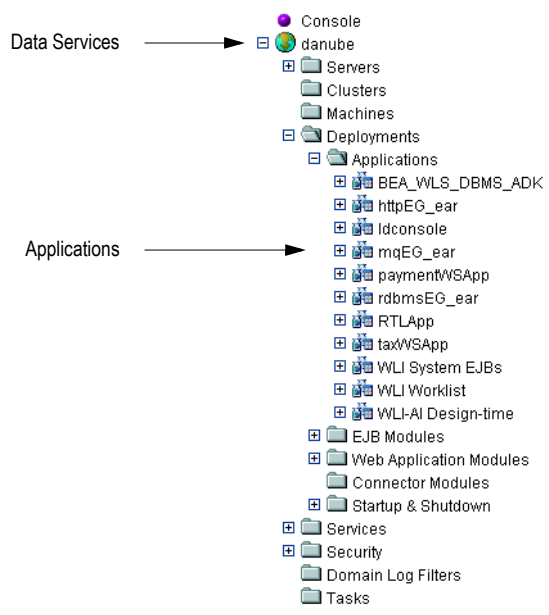
To manage AquaLogic Data Services Platform, you will need to access and use console pages for standard WebLogic Server resources as well as console pages specific to AquaLogic Data Services Platform resources.

For a detailed overview on using the Administration Console, see Starting the Administration Console (`http://e-docs.bea.com/wls/docs81/adminguide/overview.html#start_admin_con sole`) in the WebLogic Server documentation.

Finding the AquaLogic Data Services Platform Application Node

AquaLogic Data Services Platform applications appear under the Deployment → Applications node of the domain in the Navigation pane of the WebLogic Administration Console. Figure 2-2 illustrates deployed applications in the domain.

Figure 2-2 AquaLogic Data Services Platform Resources in the WebLogic Administration Console

Stopping the WebLogic Server

You can stop a WebLogic Server running a AquaLogic Data Services Platform application from the WebLogic Administration Console.

Note: It is recommended that you use the Administration Console to shut down the server gracefully rather than shutting down from a DOS window or UNIX shell.

To stop the WebLogic Server:

1. Start the Administration Console in a web browser by opening the following URL:

   `http://<HostName>:<Port>/console`

   For example, to start the Administration Console for a local instance of WebLogic Server (running on your own machine), type the following URL in a web browser address field:

   http://localhost:7001/console/

2. Expand the Servers node under the domain in which the AquaLogic Data Services Platform application runs, and click the name of the server that you want to stop.

3. Click the Control tab.

    The Start/Stop page appears, as illustrated in Figure 2-3.

Figure 2-3 Graceful Shutdown of a Server

4. Click the Graceful shutdown of this server link.

5. Click Yes to confirm.

Deploying AquaLogic Data Services Platform Applications

This chapter describes how to deploy AquaLogic Data Services Platform applications to an Administration Server, Managed Server, or to a cluster. The chapter also describes how to deploy AquaLogic Data Services Platform applications from development to production mode.

The chapter contains the following sections:

- Introduction

- Deploying AquaLogic Data Services Platform Applications to an Administration Server

- Deploying AquaLogic Data Services Platform Applications to a Managed Server

- Deploying AquaLogic Data Services Platform Applications to a Cluster

- Deploying AquaLogic Data Services Platform Applications from Development to Production Mode

- Checking the AquaLogic Data Services Platform Version Number

Introduction

During development, you can deploy applications to a WebLogic Server directly from Workshop (or from other IDEs such as Eclipse with a WebLogic plug-in). Following development, however, applications are more typically deployed to production WebLogic Servers using the Administration Console.

In most production scenarios, there are multiple WebLogic instances in a given domain. Using the Administration Console, you can deploy applications to an Administration Server, a Managed WebLogic Server, or to a cluster.

Note: You can deploy a AquaLogic Data Services Platform application to only a single target, which can be either a server or a cluster.

The Administration Console further enables you to upgrade applications or shut down application modules on a WebLogic Server without interrupting other running applications. For general information about deploying applications, see Deploying WebLogic Platform Applications at:

http://e-docs.bea.com/platform/docs81/deploy/index.html

Deploying AquaLogic Data Services Platform Components

AquaLogic Data Services Platform-enabled applications can only run in a domain that has been *provisioned* for AquaLogic Data Services Platform. For information on such provisioning see "Provisioning an Existing Domain for AquaLogic Data Services Platform" on page 1-4.

The WebLogic Configuration Wizard automatically transfers the required items to the target server. These include the AquaLogic Data Services Platform project artifacts, including configuration files and binary files, as well as WebLogic components such as data source connections and pools.

You need to make sure, however, that any data sources configured in the development environment are available from the production environment.

Table 3-1 lists the contents of a compiled AquaLogic Data Services Platform project.

Table 3-1 Contents of a AquaLogic Data Services Platform Provisioned Application EAR file

Component	Description
`ld-server-app.jar`	Compiled components and executables for the AquaLogic Data Services Platform runtime engine.
Project JAR files	Individual JAR files for each AquaLogic Data Services Platform project in the EAR file.

Deploying AquaLogic Data Services Platform Applications to an Administration Server
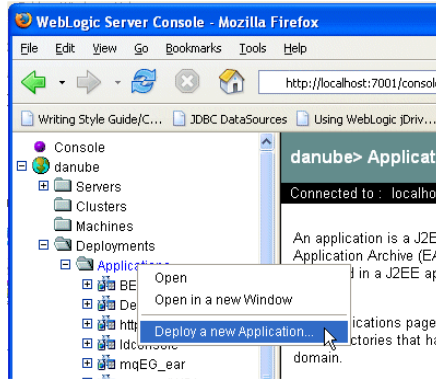
An Administration Server is the central configuration repository for the set of WebLogic Servers in a domain. Once the AquaLogic Data Services Platform application is deployed to the Administration Server, you can deploy it to all of the managed servers in the domain.

To deploy an application to WebLogic using the Administration Console:

1. Start the Administration Console for the Administration Server of the WebLogic domain.

   For more information, see Chapter 2, "Using the WebLogic Server Console."

2. Right-click the Application node under Deployments in the Navigation pane, and choose Deploy a new Application from the menu.

   Figure 3-1 illustrates the Application node context-sensitive menu.

Figure 3-1 Deploy Application Menu Selection

3. Using the Location links, navigate to the directory where the EAR file, JAR, or EJB is located.

4. Click the radio button for the application you want to deploy, and click Continue.

Figure 3-2 Deploy an Application page
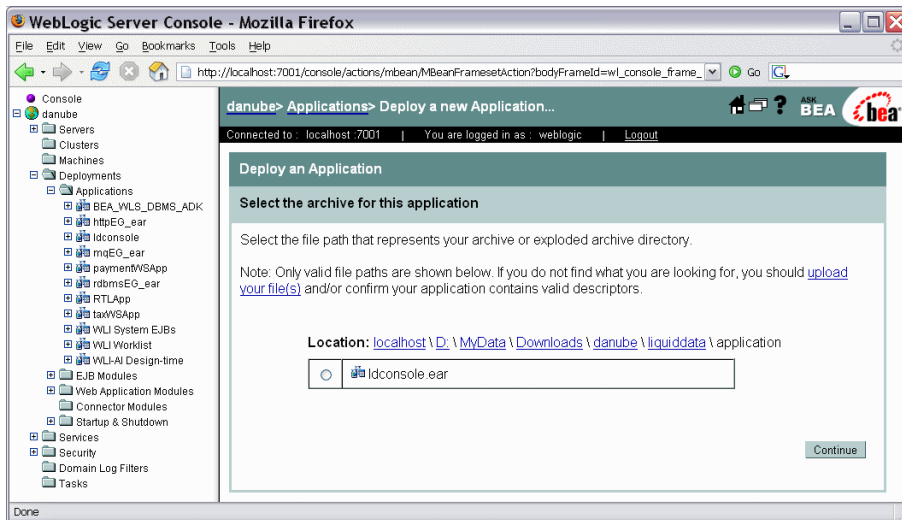
5. After reviewing the deployment information, click Deploy.

The deployment status of the application appears. Also, the application appears in the list of Applications in the Navigation pane. From there you can manage the application and deploy it to other servers in the domain.

Deploying AquaLogic Data Services Platform Applications to a Managed Server
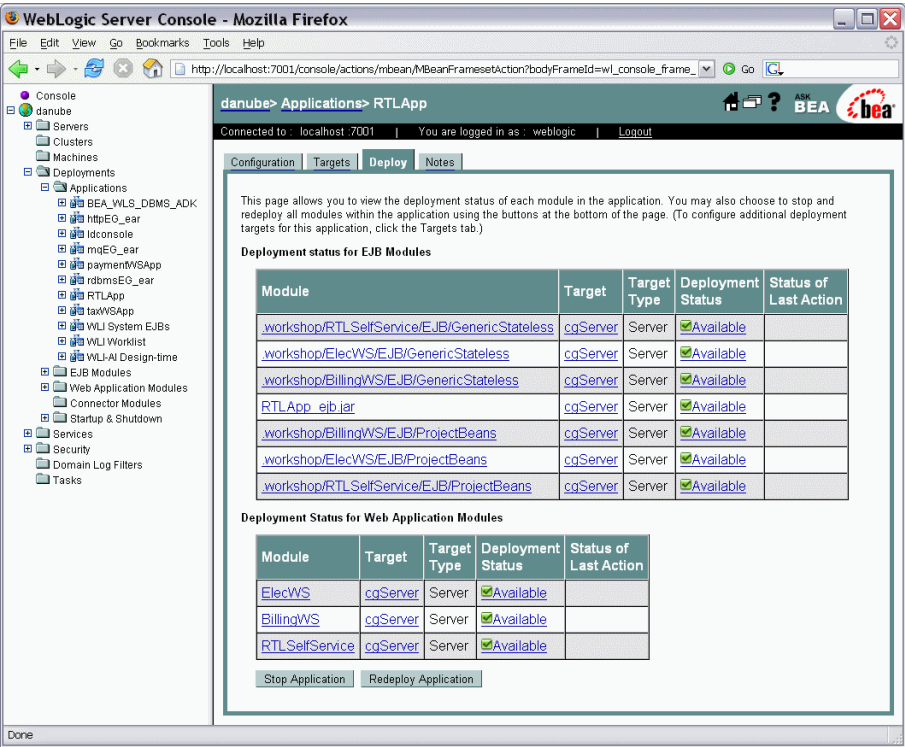
You can deploy applications to Managed Servers in the WebLogic domain using the Administration Console.

To deployed applications to a Managed Server:

1. Start the Administration Console for the Administration Server of the WebLogic domain.

   For more information, see Chapter 2, "Using the WebLogic Server Console."

2. Select the node for the AquaLogic Data Services Platform application in the Navigation pane.

3. Click the Deploy tab in the Contents pane.

   The Administration Console displays the AquaLogic Data Services Platform Deploy tab.

Figure 3-3 Deploy Tab for a AquaLogic Data Services Platform Node in the Administration Console

4. Click Redeploy Application.

   The console shows the status of the redeploy action, and displays Success for each module when the redeploy operation has completed.

Deploying AquaLogic Data Services Platform Applications to a Cluster

A cluster is multiple WebLogic Server instances running simultaneously and working together to provide increased scalability and reliability. A cluster appears to clients to be a single WebLogic Server instance.

To deploy a AquaLogic Data Services Platform application to a cluster:

1. Start the Administration Console for the Administration Server of the WebLogic domain.

For more information, see Chapter 2, "Using the WebLogic Server Console."

2. Right-click the Application node under Deployments in the Navigation pane, and choose Deploy a new Application from the menu.

3. Using the Location links, navigate to the directory where the EAR file, JAR, or EJB is located.

Figure 3-4 illustrates the screen for selecting an application to deploy to a cluster.

Figure 3-4 Selecting an Application to Deploy to a Cluster
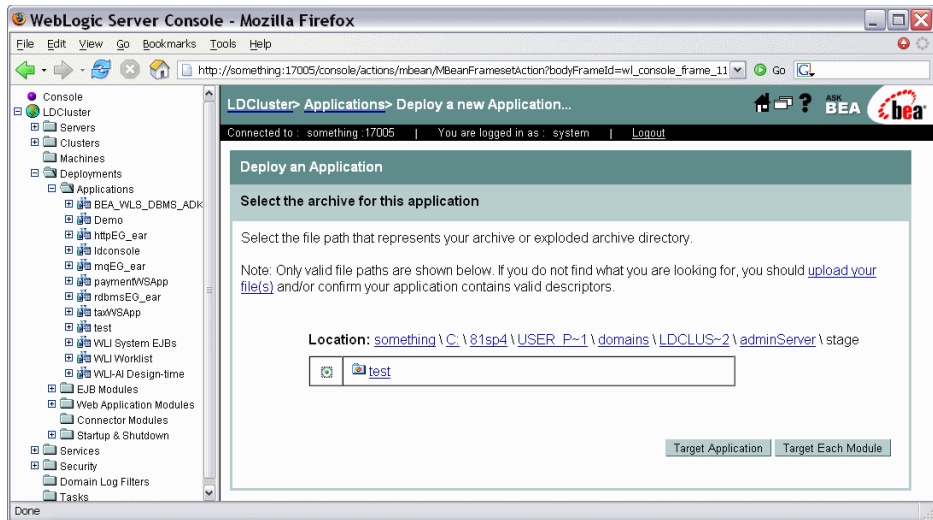
4. Click the radio button for the application you want to deploy, and click Target Application.

The console displays the available clusters, as illustrated in Figure 3-5.

Figure 3-5 Selecting a Target for the Application
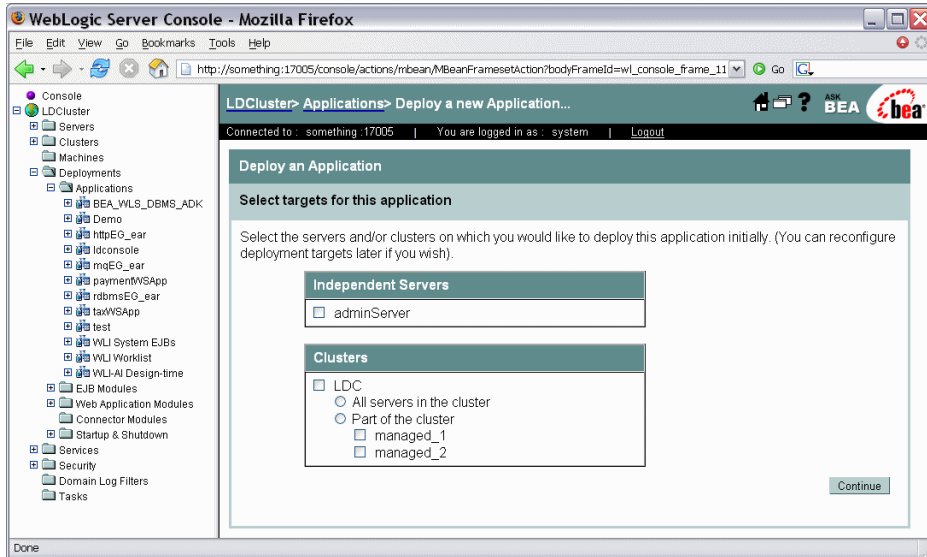
5. Click the radio button corresponding to the cluster or part of cluster to which you want to deploy the AquaLogic Data Services Platform application, and click Continue.

6. After reviewing the deployment information, click Deploy.

Deploying AquaLogic Data Services Platform Applications from Development to Production Mode

AquaLogic Data Services Platform applications are typically developed and tested in development mode, which offers a relaxed security configuration and enables auto-deployment of applications. Once the application is available in its final form, you can deploy the application to production mode which offers full security and may use clusters or other advanced features.

This section describes the following methods for migrating AquaLogic Data Services Platform applications from development to production mode:

- Migrating applications using configuration templates

- Manually migrating applications

Migrating AquaLogic Data Services Platform Applications Using Configuration Templates

You can migrate AquaLogic Data Services Platform applications from development to production mode by creating a configuration template using the WebLogic Configuration Template Builder, and then choosing the template when creating a new domain using the WebLogic Configuration Wizard.

This section highlights steps specific to migrating AquaLogic Data Services Platform applications. For complete information about using the Configuration Template Builder and Configuration Wizard, see the following:

- Creating Configuration Templates Using the WebLogic Configuration Template Builder (http://e-docs.bea.com/platform/docs81/confgwiz/tempbuild.html)

- Creating WebLogic Configurations Using the Configuration Wizard (http://e-docs.bea.com/platform/docs81/confgwiz/newdom.html)

To migrate AquaLogic Data Services Platform applications using configuration templates:

1. Start the Configuration Template Builder by choosing Start → Programs → BEA WebLogic Platform 8.1 → Other Development Tools → Configuration Template Builder.

   Complete the following:

   a. Choose to Create a Configuration Template, and click Next.

   b. Select the WebLogic configuration directory for the domain in development mode, and click Next.

   c. Enter descriptive information about the template you are creating, and click Next.

   d. Choose the AquaLogic Data Services Platform applications to add to the template, including the ldconsole application, and click Next.

   e. Add the `liquiddata` folder to the `<Domain Root Directory>` of the Current Template View, and click Next.

   f. Add SQL scripts, as required, and click Next.

   g. Configure the Administration Server, and click Next.

   h. Configure the managed servers and clusters, as required, and click Next.

   i. Edit the JDBC connection pools, updating the database configuration, and click Next.

      Maintain the JDBC connection pool names unchanged.

j. Continue through the rest of the wizard, configuring options as required.

k. Click Create to create the template, and click Done to exit the Configuration Template Builder.

    By default, the Configuration Template Builder stores the new template in the `<BEA_HOME>/user_templates` directory on the development server.

2. Start the Configuration Wizard by choosing Start → Programs → BEA WebLogic Platform 8.1 → Configuration Wizard.

    Complete the following:

    a. Choose Create a new WebLogic configuration, and click Next.

    b. Click Browse and choose the directory in which the template resides. Choose the template in the Templates pane, and click Next.

    c. Continue through the rest of the wizard, configuring options as required.

    d. Click Create to create the domain, and click Done to exit the Configuration Wizard.

Manually Migrating Applications from Development to Production Mode

You can manually deploy AquaLogic Data Services Platform applications from development to production mode, if required.

To manually deploy an application from development to production mode:

1. Create a AquaLogic Data Services Platform domain in production mode with the same JDBC connection pool and data source information as the development domain.

2. Copy the `liquiddata` folder which contains `<app_name>LDconfig.xml` file from the development domain to the production domain.

3. Copy the EAR file of the AquaLogic Data Services Platform application from the development domain to the production domain.

    The EAR file resides in the `applications` folder of the domain.

4. Edit the `config.xml` file of the production domain, and add application elements which belong to the AquaLogic Data Services Platform application and AquaLogic Data Services Platform Administration Console (ldconsole).

    You can cut and paste this information from the `config.xml` file in the development domain.

5. Migrate the WebLogic security data from the development domain to the production domain.

   Export the security policies for the WebLogic Authorization provider, and import the policies into the new security realm. For more information about migrating WebLogic Security data, see the WebLogic documentation at:

   `http://e-docs.bea.com/wls/docs81/secmanage/security_data_migration.html`

6. Migrate the AquaLogic Data Services Platform security policies from the development domain to the production domain.

   Export the AquaLogic Data Services Platform security policies in the development domain and import them into the production domain. For more information about exporting AquaLogic Data Services Platform security policies, see "Exporting Access Control Resources" on page 6-21.

7. If you are using Data Service controls in any of your applications, migrate the `ldcontrol.properties` file from development to the production domain.

   Each domain that runs AquaLogic Data Services Platform Control applications has a single ldcontrol.properties file, which stores the connection information for *all* AquaLogic Data Services Platform Control applications running in the domain.

   The `ldcontrol.properties` file is located at the root directory of your domain where the application EAR file is deployed that uses a Data Service control. There is an entry in the ldcontrol.properties file for each control you have created in each of your applications.

   The entries in the `ldcontrol.properties` file are of the following form:

   ```
   AppName.ProjectName.FolderName.jcxName=t3\://hostname\:port
   ```

Table 3-2 provides additional details.

Table 3-2 Description of ldcontrol.properties File Options

Name	Description
AppName	The name of the WebLogic Workshop application.
ProjectName	The name of the WebLogic Workshop Project which contains the AquaLogic Data Services Platform Control.
FolderName	The name of the folder which contains the AquaLogic Data Services Platform Control.

Table 3-2 Description of ldcontrol.properties File Options

Name	Description
jcxName	The name of the AquaLogic Data Services Platform Control file (without the `.jcx` extension). For example, if the control file is named `myLDControl.jcx`, the entry in this file is `myLDControl`.
hostname	The hostname or IP address of the AquaLogic Data Services Platform Server for this control.
port	The port number for the AquaLogic Data Services Platform Server for this control.

Note: The colons (`:`) in the URL must be escaped with a backslash (`\`) character.

If the URL value is missing, the AquaLogic Data Services Platform Control uses the connection information from the domain `config.xml` file.

The following is a sample `ldcontrol.properties` file.

```
#Fri Oct 31 15:30:36 PST 2003
myTest.myTestWeb.myFolder.Untitled=t3\:myLDServer\:7001
myTest.myTestWeb.myFolder.myControl=
SampleApp.LiquidDataSampleApp.Controls.RTLControl=t3\:myLDServer\:7001
SampleApp.Untitled.NewFolder.Untitled=t3\:yourLDServer\:7001
testnew.Untitled.NewFolder.ldc=
test.testWeb.NewFolder.Untitled=
```

8. Update the WebLogic Workshop configuration settings by adding:

```
-Djavax.xml.rpc.ServiceFactory="weblogic.webservice.core.rpc.
ServiceFactoryImpl"
```

to the following file:

> `<WL_HOME>\workshop\workshop.cfg`

9. Start the WebLogic Server and verify that the AquaLogic Data Services Platform application is working properly.

Checking the AquaLogic Data Services Platform Version Number

You can determine which version of AquaLogic Data Services Platform you are through the WebLogic Administration Console.

To determine the version number (which appears associated with the name *Liquid Data*):

1. Start the Administration Console for the Administration Server of the WebLogic domain.

   For more information, see Chapter 2, "Using the WebLogic Server Console."

2. Click Console in the Navigation pane.

3. Click the Versions tab in the Contents pane.

   A page displaying the version information appears.

Using the AquaLogic Data Services Platform Console

This chapter describes how to use the AquaLogic AquaLogic Data Services Platform Console to manage applications on a domain that has been provisioned for AquaLogic Data Services Platform.

Note: For information on provisioning WebLogic domains for AquaLogic Data Services Platform see "Understanding the Relationship of AquaLogic Data Services Platform to WebLogic Domains" on page 1-3.

The chapter contains the following sections:

- Introducing the AquaLogic Data Services Platform Console

- Launching the AquaLogic Data Services Platform Console

- Navigating the AquaLogic Data Services Platform Console

- Controlling Access to the AquaLogic Data Services Platform Console

Introducing the AquaLogic Data Services Platform Console

The AquaLogic Data Services Console (accessed under the name *ldconsole*) is a web-based interface specifically designed for managing and using AquaLogic Data Services Platform applications. You can use the AquaLogic Data Services Console to set security and caching policies for data services, and configure AquaLogic Data Services Platform runtime settings such as thread usage and logging levels.

The AquaLogic Data Services Console also provides access to the Data Services Metadata Browser. The Metadata Browser provides information useful to both AquaLogic Data Services Platform

administrators and application developers. Developers can see what data services are available, what information they provide, how to call them, and more. Administrators can determine the effects of changes to the data source layer in the console.

Note: For more information, see Chapter 8, "Viewing Metadata."

Figure 4-1 shows the main page of the AquaLogic Data Services Platform Console.

Figure 4-1 AquaLogic Data Services Platform Console
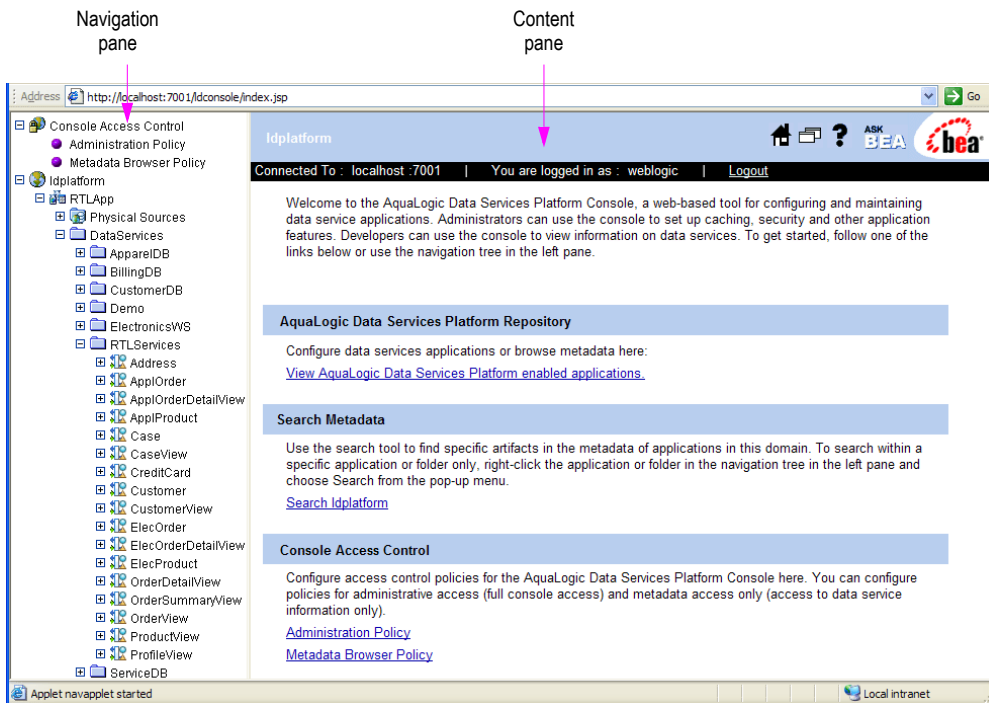
Launching the AquaLogic Data Services Platform Console

The AquaLogic Data Services Platform Console is a web-based interface that enables you to administer and manage AquaLogic Data Services Platform applications, access metadata, and configure security and caching policies.

To launch the AquaLogic Data Services Console:

1. Start the WebLogic Server in the WebLogic domain in which AquaLogic Data Services Platform is deployed.

   For more information, see "Starting the WebLogic Server."

2. Using a web browser, open the following URL:

   `http://hostname:port/ldconsole`

   Where:

   - `hostname` is the machine name or IP address of the host server
   - `port` is the address of the port on which the host server is listening for requests (7001 by default)

   For example, to start the AquaLogic Data Services Console on a local instance of WebLogic Server (running on your own machine), navigate to the following URL:

   `http://localhost:7001/ldconsole/`

3. When the login page appears, enter the appropriate user name and password.

   The defaults user name and password is weblogic/weblogic, respectively.
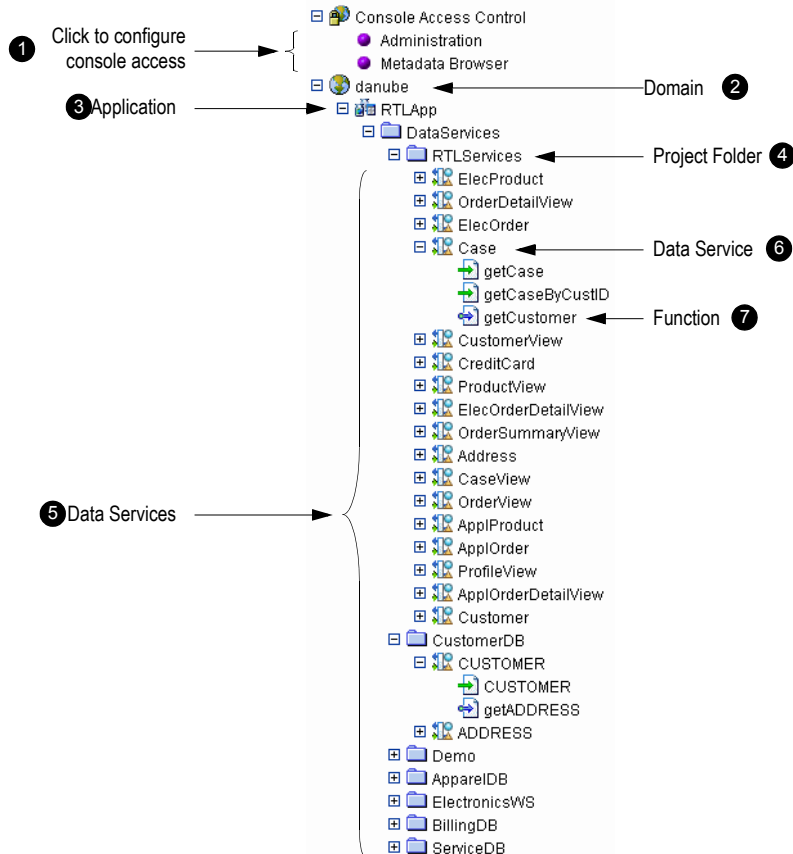
Note: The discussion and examples in the remainder of this section assume that you have:

   - Installed a current version of AquaLogic Data Services Platform
   - Either opened the RTLApp sample application or created a sample application based on the AquaLogic Data Services Platform Samples Tutorial.
   - Build your application as described in "Data Services Platform Projects and Components" in the *Building Queries and Data Views*. Building an application or project automatically deploys it and any data services it contains on your currently running WebLogic Server.

Navigating the AquaLogic Data Services Platform Console

You can navigate to the various pages in the AquaLogic Data Services Platform Console using the tree in the Navigation pane. Pages are organized by application and data service, as shown inFigure 4-2.

Figure 4-2 Console Tree Panel

The following describes the actions you can perform using the Navigation pane:

❶ **Console Access Control.** Enables you to configure the access control policies that specifies who can access particular console features. Clicking Administration or Metadata Browser displays the Policy Editor, enabling you to specify Policy Statements defining access. For more information, see "Using the WebLogic Policy Editor" on page 6-5.

❷ **Domain.** Expand to display the AquaLogic Data Services Platform-enabled applications in the domain. Alternatively, you can click a domain name to display the list of such applications in the Content pane. Right-click and choose Search in the context-sensitive menu to search metadata in the domain (see "Searching Metadata" on page 8-13).

3 **Applications.** Expand to display the Data Services folder. Alternatively, you can click the application name to display the general application settings in the Content pane. For more information, see "General Application Settings" on page 5-1. Right-click and choose Search in the context-sensitive menu to search metadata in the application (see "Searching Metadata" on page 8-13).

4 **Data Services.** Expand to display the data service project folders in the application. Alternatively, you can click the Data Services folder to display the list of project folders in the Content pane. Right-click and choose Search in the context-sensitive menu to search metadata in the data services (see "Searching Metadata" on page 8-13).

5 **Project Folder.** Expand to display specific data services contained in the project folder. Alternatively, you can click a project folder to display the list of data services in the Content pane. For more information, see "Displaying a AquaLogic Data Services Platform-enabled Application's Data Sources and Data Services" on page 4-6. Right-click and choose Search in the context-sensitive menu to search metadata in the project folder (see "Searching Metadata" on page 8-13).

6 **Specific Data Service.** Expand to display the functions that comprise the data service. Alternatively, you can click a specific data service to display the administration screen for the functions in the Content pane. For more information, see "Examining Data Service Functions" on page 4-9.

7 **Function.** Click to display information about the function in the Content pane, including general information, dependencies, where the function is used, properties, and the return type. For more information, see "Displaying Function Details" on page 4-10. Right-click and choose Define Security Policy in the context-sensitive menu to create a security policy for the function using the WebLogic Policy Editor (see "Understanding Security Policies" on page 6-4).
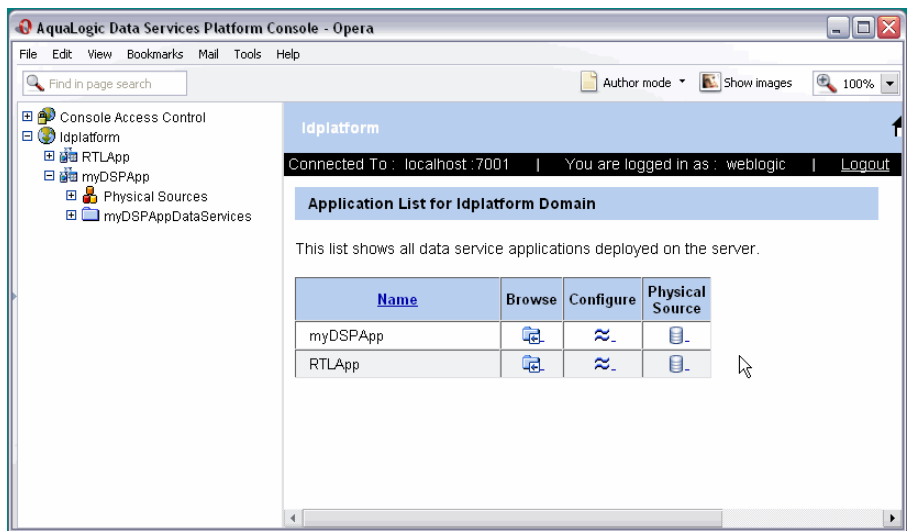
Displaying a Domain's AquaLogic Data Services Platform-Enabled Applications

The AquaLogic Data Services Platform Console lists the applications in your current WebLogic Server domain that are both:

- Enabled for AquaLogic Data Services Platform and

- Deployed to a WebLogic Server

Once deployed, applications appear in the Navigation pane.

Figure 4-3 AquaLogic Data Services Platform-Enabled Applications in a AquaLogic Data Services Platform-Provisioned Domain

For each application there are several navigation icon options, as shown in Figure 4-3:

- **Browse.** The option allows you to invoke the Data Services Metadata Browser for your application. These are described in detail in Chapter 8, "Viewing Metadata."

- **Configure.** This is a shortcut to the configuration options available for each application. These are described in detail in Chapter 5, "Configuring AquaLogic Data Services Platform Applications."

- **Physical Source.** Displays the types of physical sources used in your application.

Displaying a AquaLogic Data Services Platform-enabled Application's Data Sources and Data Services

You can display the data sources and data services available to an application, along with information about each. Details related to inspecting sources and services can be found in Chapter 8, "Viewing Metadata."

Physical sources are organized by type: relational, Web services, XML, and delimited. Some applications may include only one or several types of data sources.

To display the data sources associated with an application:

- In the Navigation Pane expand the Physical Sources folder within your application. The data sources used in the application appear in the Navigation pane (see Figure 4-4).

- Alternatively, select a specific data source folder in the Navigation pane.

Figure 4-4 Data Sources Available to the RTLApp
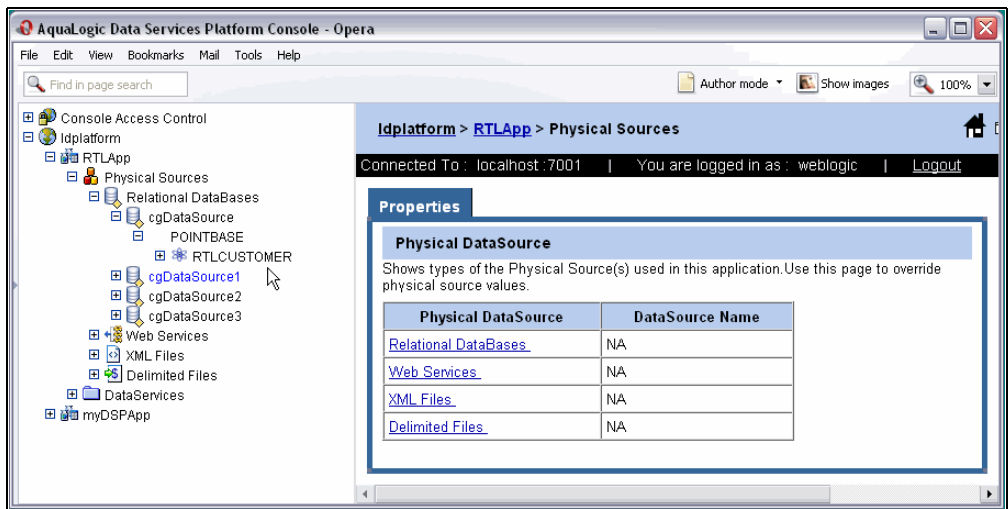
To display the data services associated with an application:

- Expand a Data Services project folder within an application in the Navigation pane. The data services contained in the project folder appear in the Navigation pane.

- Alternatively, select a specific folder in the Navigation pane.

   The list of data services contained in the folder appears in the Content pane, as illustrated in Figure 4-5.

Figure 4-5 Data Services Available from the RTLApp
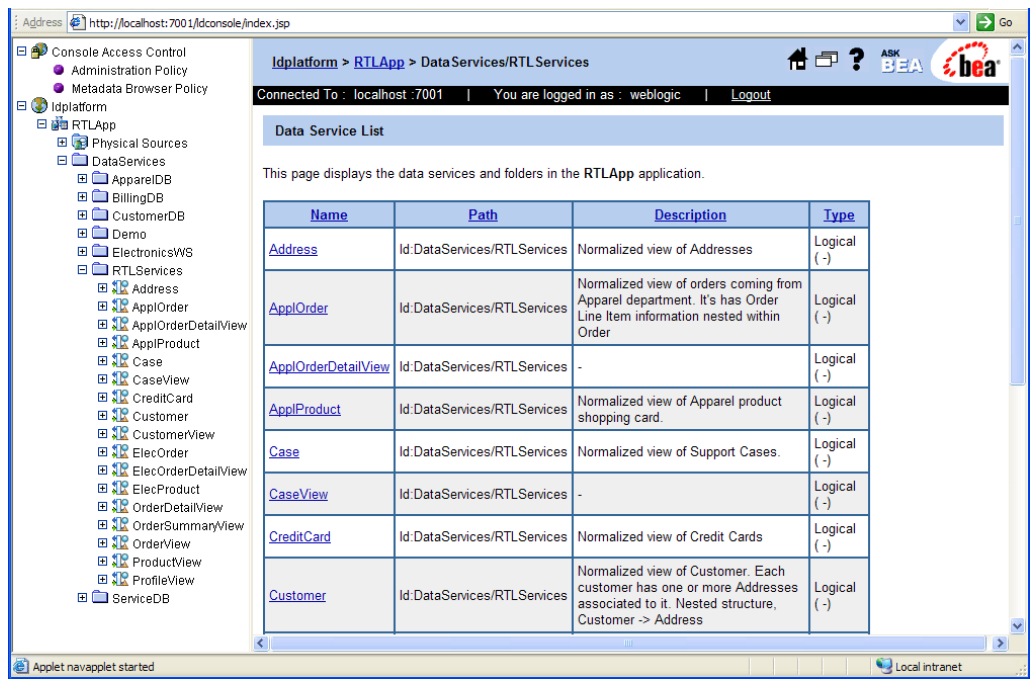
Table 4-1 describes the information presented for each data service.

Table 4-1 Data Service Information

Column	Description
Name	The name of the data service.
Path	The physical location of the data service.
Description	An optional description of the data service.
Type	Data services can be physical or logical. A physical data service represents an actual data source, such as a database table. The specific data source type, such as Relational, Web Service, and so on, is displayed for physical data services.
	A logical data service is a manually created data service that aggregates or filters data in some way.

Examining Data Service Functions

You can examine the functions that comprise a data service, and manage the cache and security settings, as required. You can also view metadata associated with a data service.

To display the functions that comprise a data service:

- Expand a specific data service within a project folder in the Navigation pane.

  The functions that comprise a data service appear in the Navigation pane.

- Alternatively, select a data service within a project folder in the Navigation pane.

  An administration screen for the functions in the data service appears in the Content pane, as illustrated in Figure 4-6. For more information about administering data service functions, see"Setting Up Caching" on page 7-3, "Securing Data Service Functions" on page 6-11, and "Introspecting Data Service Metadata" on page 8-7.

Figure 4-6 Data Service Functions
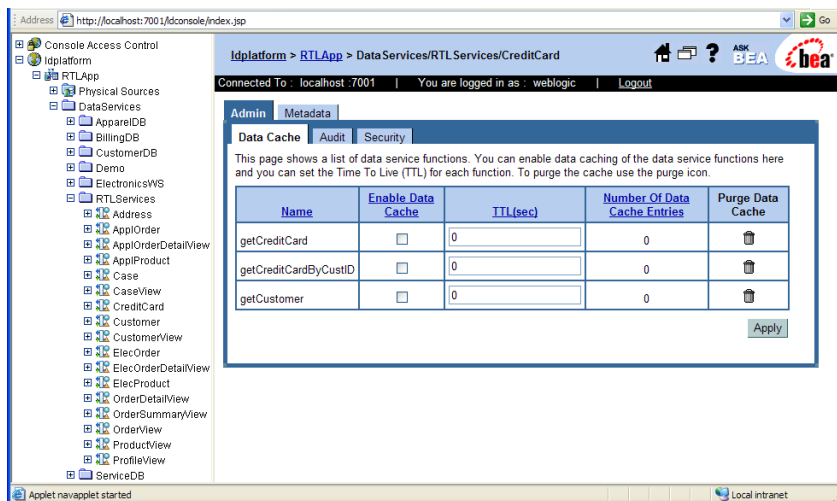
There are two types of functions identified in the Navigation tree, as described in Table 4-2.

Table 4-2 Function Types

Icon	Function Type
	Navigation function, which return data from a related data service.
	Read function, which return data in the form of the data service type.

Displaying Function Details

You can display information about specific functions, including general information, dependencies, where the function is used, properties, and the return type. To display details about a function:

- Select the specific function in the Navigation pane.

  Metadata associated with the function appears in the Content pane, as illustrated in Figure 4-7. For more information, see "Metadata Browser Interface for Data Service Functions" on page 8-9.

Figure 4-7 Function Details
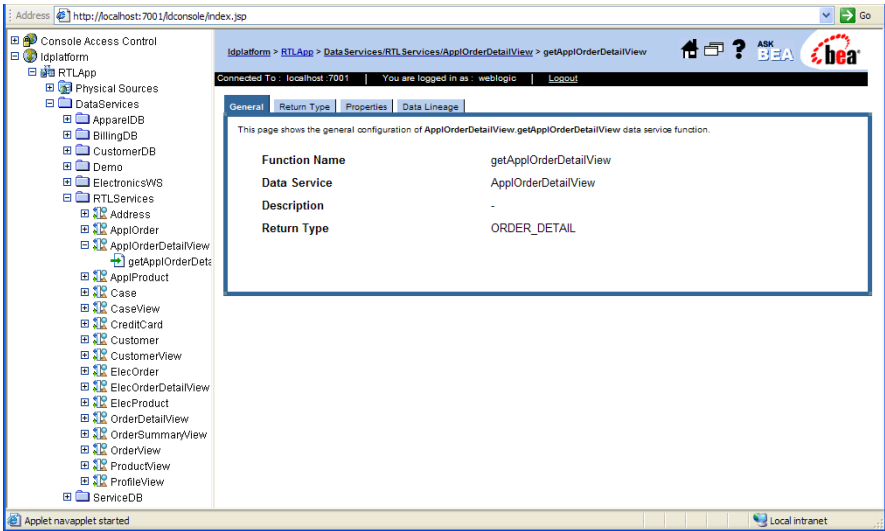
Controlling Access to the AquaLogic Data Services Platform Console

The AquaLogic Data Services Platform Console is a securable resource from the perspective of WebLogic Security. You can set access control policies that defines who can view and use particular pages in the console. The features are distinguished by two functional categories:

- **Administrative.** This includes security and cache settings.

- **Informational.** Displays metadata on data services, such as return types, functions, relationships, and so on.

For information on controlling resource access, see Chapter 6, "Securing AquaLogic Data Services Platform Resources."

Configuring AquaLogic Data Services Platform Applications

This chapter describes how to configure application-level settings for AquaLogic Data Services Platform. The chapter contains the following sections:

- General Application Settings

- Modifying Data Source End Points

- SQL Statement Substitution

- Guidelines for Setting Server Thread Count

- Monitoring Applications

- Terminating an Executing Query

- Using Administrative Properties

- Setting the Transaction Isolation Level

General Application Settings

You can view and configure runtime settings for AquaLogic Data Services Platform-enabled applications, including access control, cache settings, server resources (including thread usage), and log levels.

Note: For details on accessing the Data Services Platform Console (named *ldconsole*) see "Launching the AquaLogic Data Services Platform Console" on page 4-2.

To specify general application settings:

1. Click the application name in the Navigation pane of the Data Services Platform Console.

   The General settings page appears, as illustrated in Figure 5-1. Note that you must be logged into the console using a user name with administrator privileges.

Figure 5-1 General Application Settings Page
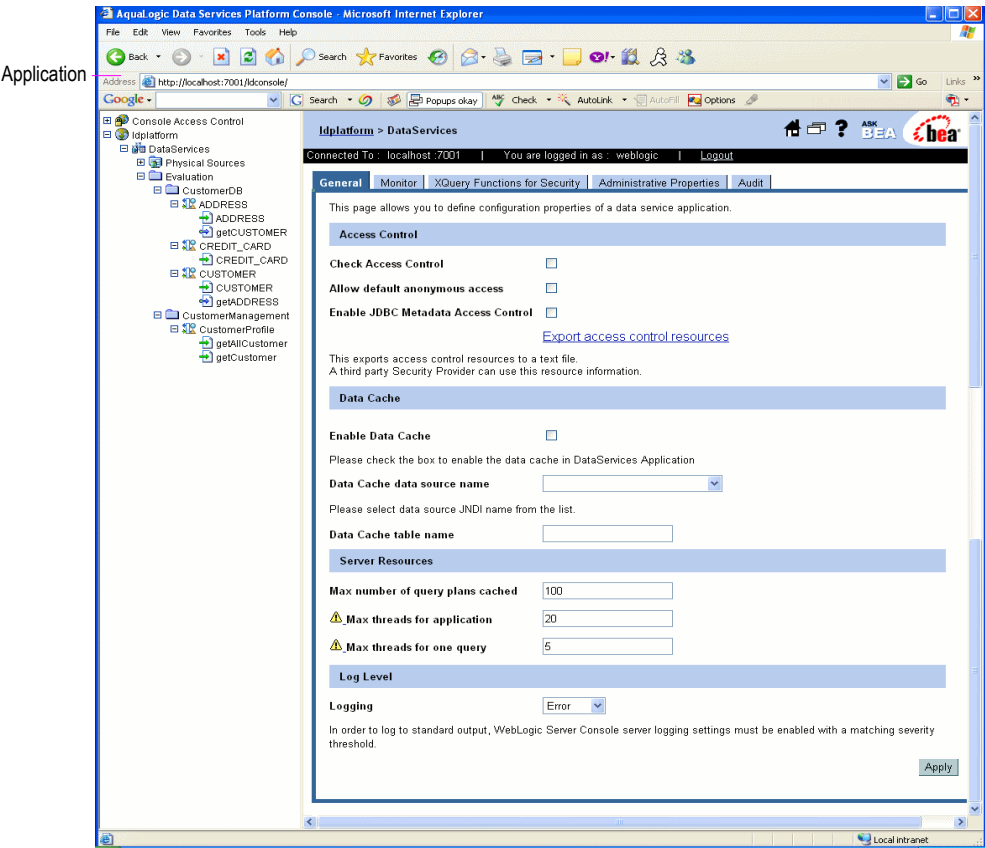
2. Specify settings, as appropriate.

3. Click Apply to save the settings.

Table 5-1 lists the application settings available under the General tab.

Table 5-1 AquaLogic Data Services Platform Server Configuration Settings

Section	Field	Description
Access Control	**Check Access Control**	Specifies whether the configured security policy settings will be enforced for the application.
	Allow default anonymous access	Enables access to the application by default (unless a more specific policy blocks it). If enabled, all users can access resources by default, even unauthenticated users.
		Disallowing default anonymous access disables access to the application by default (unless a more specific policy permits it). The anonymous access option works only with the WebLogic Authorization provider.
Cache	**Enable Cache**	Enables or disables (default) the caching of query results for stored queries.
		• To enable results caching, enable (check) this check box.
		• To disable results caching, clear (uncheck) this check box.
		For more information about caching, see Chapter 7, "Configuring the Query Results Cache."
	Cache data source name	The JNDI data source name for the database where the cache is stored.
	Cache table name	The name of the database table where cached data is stored. The default table name is *<appName>*_CACHE.

Table 5-1 AquaLogic Data Services Platform Server Configuration Settings (Continued)

Section	Field	Description
Server Resources	**Max number of query plans cached**	A query plan is a compilation of a query. The optimal number of query plans cached depends on the size of the queries. You will need to monitor the memory usage and performance of your server to determine whether to change this setting.
	Max threads for application	The maximum number of threads in the AquaLogic Data Services Platform server pool used to handle query requests.
		The default setting is 20. The minimum setting is 1. If the specified value is invalid, the server uses the default value of 20.
		Note: The maximum threads value that you specify here *does not* affect the WebLogic Server server thread pool. The value specified here applies only to the thread pool created and used by the AquaLogic Data Services Platform query engine for processing requests on application view, web service, or custom function data sources.
		For more information on configuring thread counts, see "Guidelines for Setting Server Thread Count."
	Max threads for one query	The maximum number of threads allowed for a single query. Use this to limit the number of threads spawned by a single query. The actual number of threads used will not exceed the maximum number of threads specified in Maximum Threads, regardless of the Maximum Number of Threads Per Query setting.
		The default setting is 4. The minimum setting is 1. If the specified value is invalid, the server uses the default value of 4.
		Note: The maximum threads value that you specify here *does not* affect the WebLogic Server server thread pool. The value specified here applies only to the thread pool created and used by the AquaLogic Data Services Platform query engine for processing requests on application view and web service data sources.
		For more information on configuring thread counts, see "Guidelines for Setting Server Thread Count."

Table 5-1 AquaLogic Data Services Platform Server Configuration Settings (Continued)

Section	Field	Description
Log Level	**Logging**	The verbosity of the events logged. The options include the following: • **Error.** Runtime exceptions. • **Notice.** Possible errors that do not affect runtime operation, as well as error level events. • **Information.** Start/stop events, unsuccessful access attempts, query execute times, and so on, as well as error and notice level events. The log file is in the following location: *<BeaHome>*\user_projects\domains*<domainName>*\ *<domainName>*.log

Modifying Data Source End Points

It is frequently desirable to change the location of data sources or names of other artifacts as you move applications from development to staging to production. For example, if you are using "dummy" data sources during development in order to protect confidential or otherwise secured information, you will at some point need to substitute a new data source with the actual data for the test version. You can make these changes through the Data Services Platform Console.

In modifying end points you are not limited to the name and location of a data source. It is also possible to change the target names of subordinate artifacts. In the case of relational sources this includes catalog name, schema names, package names, table names, and stored procedure names.

Note: Once set, end point modifications are effective until they are further modified or reverted to the original name. To assign the end point name its original value, simply click **Reset to original value**. This option will not revert the value to the previous setting, it will directly revert it to the original name. So, if you have assigned a few names over time, the moment you click **Reset to original value**, the values revert to the same as those in the **Original Value** column.

Figure 5-2 Setting End Points for Relational Sources
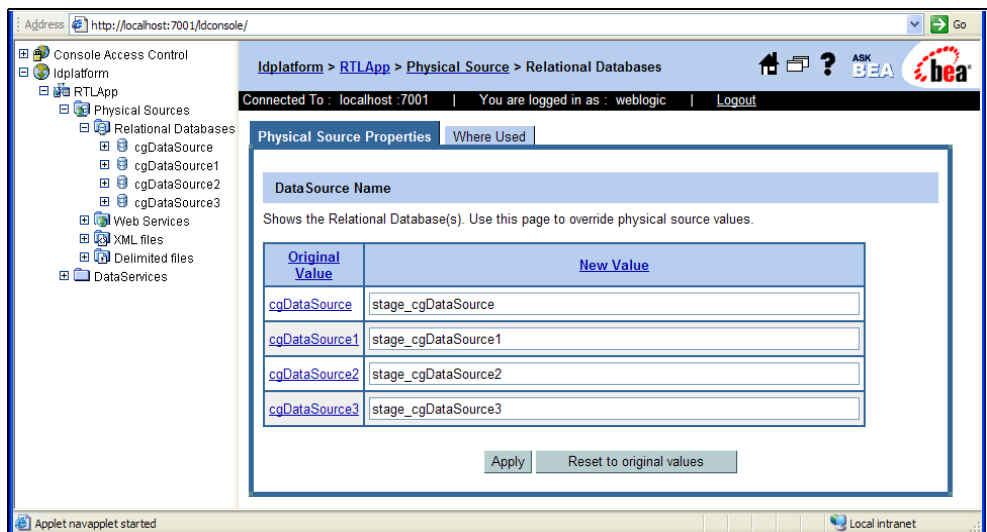
Note: Whenever you change the end point for an artifact you need to ensure that the intrinsic aspects of that artifact remain identical with the old source. In the case of a relational source properties such as Vendor Type and Version must be identical.

When you change the end point of a particular object, the new end point appears in brackets next to the original name. Figure 5-3 below displays the original data source name, and the new data source name (in square brackets) adjacent to it.

Figure 5-3 End Point Settings Reflected in the Navigation Pane
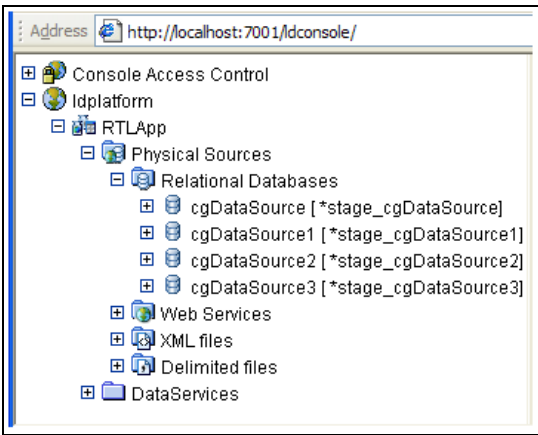
Table 5-2 identifies the artifacts whose end point settings can be changed.

Table 5-2 Artifacts for Which End Points Can be Modified Through the AquaLogic Data Services Platform Console

Data Source Type	Artifact
Relational	Data source name and location
	Catalog
	Schema
	Package
	Table
	Stored procedure
Web Service	Web service name and location
	Service
	Port
	Operation
XML Content	Data source name and location
Delimited File Content	Data source name and location

Physical Data Source Locations

You can view a list of data services and function libraries that use the defined relational databases. Click the **Where Used** tab to view the list of data services and their specific paths (Figure 5-4).

Figure 5-4 Physical Data Services Relational Dependencies
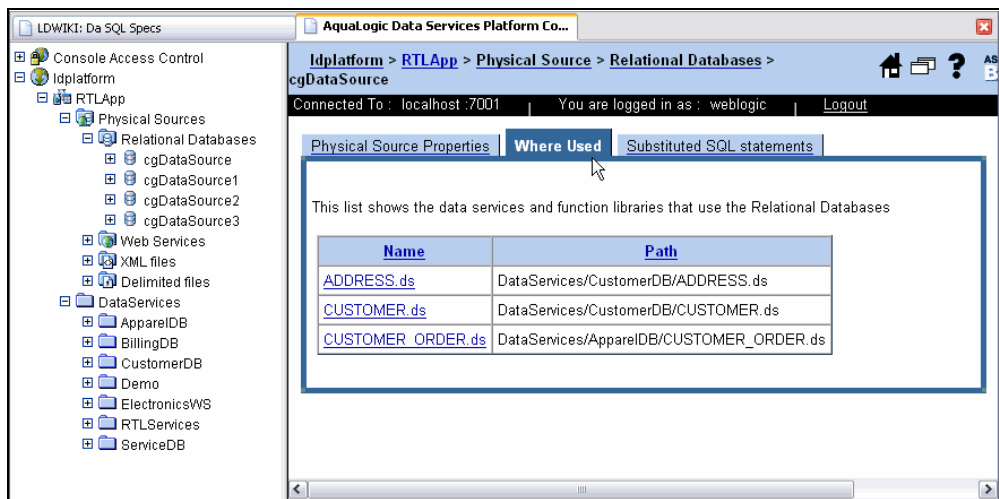
SQL Statement Substitution

AquaLogic Data Services Platform uses SQL language to access relational data sources. At the compilation time built-in query optimizer determines the best execution strategy for backend sources. Then SQL queries are generated and submitted to underlying databases.

SQL queries generated by the relational wrapper are specific to each underlying database. While the SQL queries that are generated typically produce good results, there are cases however when further optimization of the generated queries is desirable. In most RDBS systems such optimization is done through execution *hints*.

SQL statement substitution allows you to add hints to generated SQL queries by providing edited SQL statements that will be executed instead of the query that is generated by default by AquaLogic Data Services Platform.

WARNING: Unlike SQL statements generated by AquaLogic Data Services Platform, substituted SQL statements are passed to the underlying database in an unvalidated form. For this reason users are strongly advised against using this feature for any purpose other than providing hints to the database. It is also recommended that prior to deployment any substituted SQL statement be tested against its generated counterpart to make sure that the expected performance advantage is being obtained.

How SQL Statement Substitution Works

AquaLogic Data Services Platform server maintains a substitution table between the original generated SQL queries and any replacement queries supplied by the user. Only SQL queries specified by user will be substituted.

The AquaLogic Data Services Platform administrator defines and maintains substitution queries through the AquaLogic Data Services Console.

The replacement query is executed instead of the original SQL query. The AquaLogic Data Services Platform runtime engine reads the SQL result set using type/column information of the original query. Incorrect substitution which violates the conditions listed in Requirements for SQL Statement Substitution might lead to the following problems:

- Incorrect result returned by the XQuery (e.g. incorrect data, no result at all, incorrect order of the result, are among the possible unwanted outcomes)

- Runtime engine error during SQL statements execution (problems with parameter binding, reading of the result, and so forth.

Supporting Externalized End Points in Substituted Queries

In both the generated and substitute queries, a special syntax is used to support externalized end points (see "Modifying Data Source End Points" on page 5-5 for details). The following substituted queries shows such this special syntax (emphasis added):

```
SELECT /*+ FIRST_ROWS (10)*/ t1."BILL_TO_ID" AS c1, t1."C_ID" AS c2,
t1."DATE_INT" AS c3, t1."ESTIMATED_SHIP_DT" AS c4,
t1."HANDLING_CHRG_AMT" AS c5, t1."ORDER_DT" AS c6, t1."ORDER_ID" AS c7,
t1."SALE_TAX_AMT" AS c8,
t1."SHIP_METHOD_DSC" AS c9, t1."SHIP_TO_ID" AS c10, t1."SHIP_TO_NM" AS c11,
t1."STATUS" AS c12,
t1."SUBTOTAL_AMT" AS c13, t1."TOTAL_ORDER_AMT" AS c14, t1."TRACKING_NO" AS c15
FROM {RTLAPPLOMS}.{CUSTOMER_ORDER} t1
```

Note: If you are adding SQL fragments (such as string literals) in your substituted SQL statement, you also need to use the convention of doubling opening curlie braces.

For example:

```
SELECT t1.ID FROM CUSTOMER() WHERE $i/ID > 'a{bee}c' return $i/ID
```

is translated to:
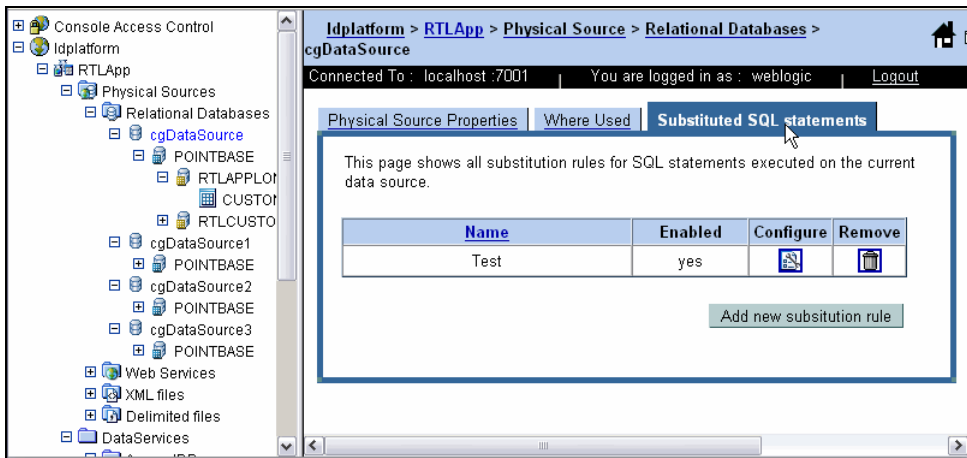
```
SELECT t1.ID FROM {CUSTOMER} t1 WHERE t1.ID > 'a{{bee}c'
```

As needed you should specify replacement queries using the same name placeholders as the original query. At the end of the SQL generation stage these original names will be replaced with the current end point names. The original name will be used if no endpoint setting is found.

Managing Substitute SQL Statements

Substitute SQL statements are created and registered in the Data Services Platform Console using the Substituted SQL Statements option (see Figure 5-4).

Figure 5-5 Substituted SQL Statement Dialog Box

The options available include:

- **Name.** The name you select for your substitute SQL statement.

- **Enabled.** Optionally enable or disable the substitute query.

- **Configure.** Activates a dialog box where you can modify your substitute SQL query.

- **Remove.** A mechanism for deleting substitute SQL queries.

- **Add new substitution rule.** Activates a dialog box where you can create a new substitution rule and its substitute query.

Creating Substitute SQL Query Statements

When you create a substitute SQL query you need to provided the following items of information:

- Name you assign to the substitute query.

- Whether the substitute query is enabled or not.

- Your description of the query.

- The SQL statement generated by AquaLogic Data Services Platform.

- The substituted statement.

Entries must be made in all fields but the description, which is optional.

The system automatically tracks creation and last modified dates.

Requirements for SQL Statement Substitution

There are several requirements regarding the substituted SQL query:

- The query must return same data, with same number of columns and column types.

- Columns must be listed in the same order as the original query.

- The query must have the same number of parameters, in the same order, as the original query.

- The expected parameter types must match that of the original query.

- Alias column names must be exactly the same as in the original query.

  Note: For queries using sub-queries only the outermost subquery must preserve column aliases, inner subqueries need not do so.

- If the original query contained an ORDER BY clause, the same ordering the result must be required.

Example: SQL Statement Substitution

The order in which SQL statement substitutions are established is not fixed. Thus the example in this section and the steps involved are only one approach to creating and testing SQL statement substitution.

1. Setup your environment with these actions:

- WebLogic Workshop is open to a AquaLogic Data Services Platform project that has been successfully built and deployed.

- The WebLogic Server is running.

- Your AquaLogic Data Services Console is open. In the sample application the URI is:

  `http://localhost:7001/ldconsole`
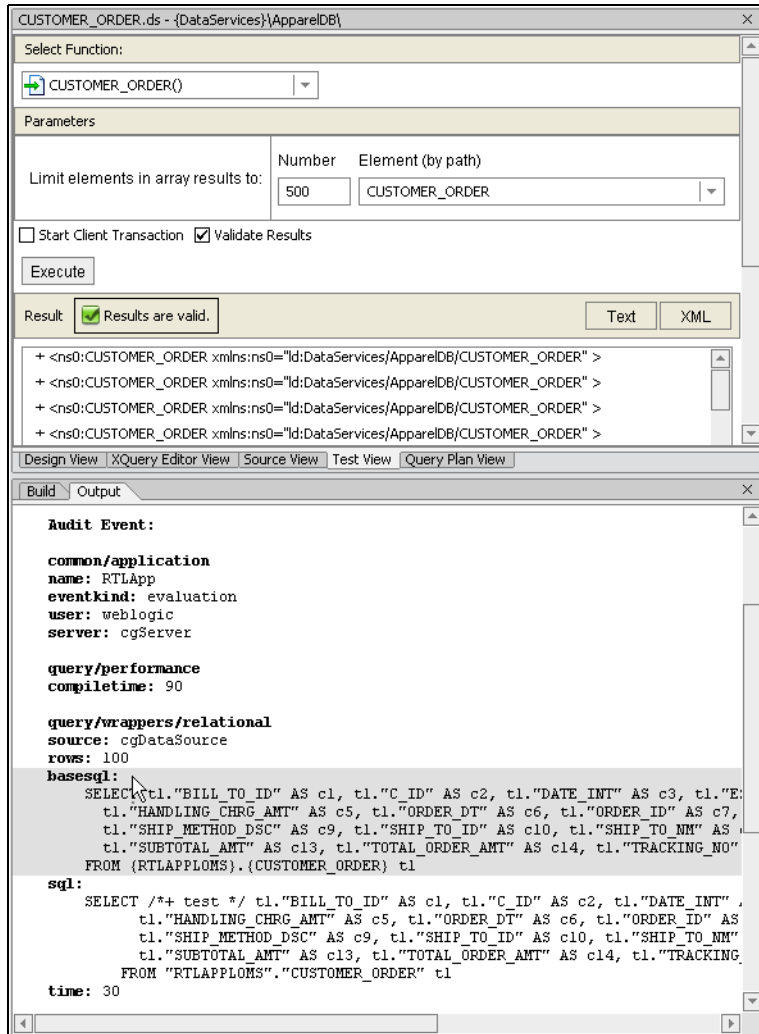
- Auditing is enabled. (For details on activating and using auditing see Chapter 9, "Working With Audit and Log Information.")

2. Your application's audit property supporting the *base SQL* statement generated by AquaLogic Data Services Platform needs to be set to Always (Figure 5-6), meaning that the base SQL statement will always be returned. (See also "Setting Individual Auditing Properties" on page 9-4.)

Figure 5-6 Setting the basesql Property to Always be Returned
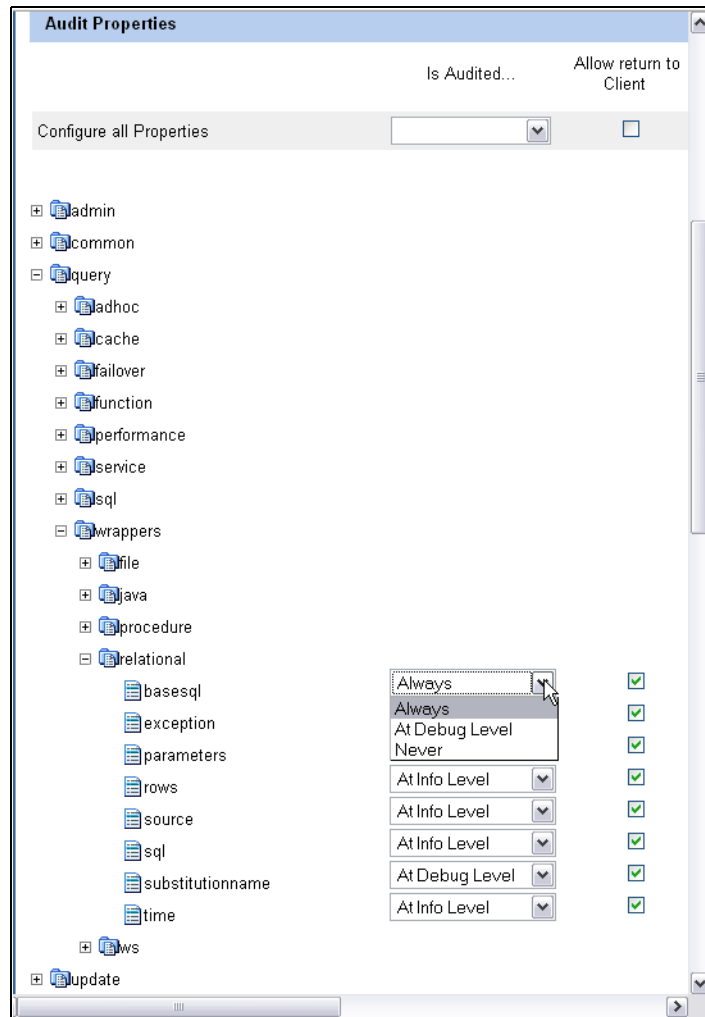
3. Select your relational data source in the AquaLogic Data Services Console (Figure 5-5).

4. Select the Substituted SQL statements option.

5. Click the Add new substitution rule option.

6. Enter the following in the resulting dialog box:

- Name you want to assign to your substitute query.

- An option description.

- Enable (or disable) the substitution logic for the query you are about to create using the Enabled checkbox.

7. In your Workshop application run your query (such as CUSTOMER_ORDER) in Workshop Test View. Notice (Figure 5-7) that a basesql version of generated SQL statement is created.

Figure 5-7 Output from RTLApp CUSTOMER_ORDER() Query with basesql Result Highlighted

8. In the Output pane scroll down until you locate the basesql version of the query you just generated (also shown in Figure 5-7). Copy this version of the query to your clipboard. A sample query appears below:

```
SELECT t1."BILL_TO_ID" AS c1, t1."C_ID" AS c2, t1."DATE_INT" AS c3,
t1."ESTIMATED_SHIP_DT" AS c4,
t1."HANDLING_CHRG_AMT" AS c5, t1."ORDER_DT" AS c6, t1."ORDER_ID" AS c7,
```

```
t1."SALE_TAX_AMT" AS c8,
t1."SHIP_METHOD_DSC" AS c9, t1."SHIP_TO_ID" AS c10, t1."SHIP_TO_NM" AS c11,
t1."STATUS" AS c12,
t1."SUBTOTAL_AMT" AS c13, t1."TOTAL_ORDER_AMT" AS c14, t1."TRACKING_NO" AS c15
FROM {RTLAPPLOMS}.{CUSTOMER_ORDER} t1
```
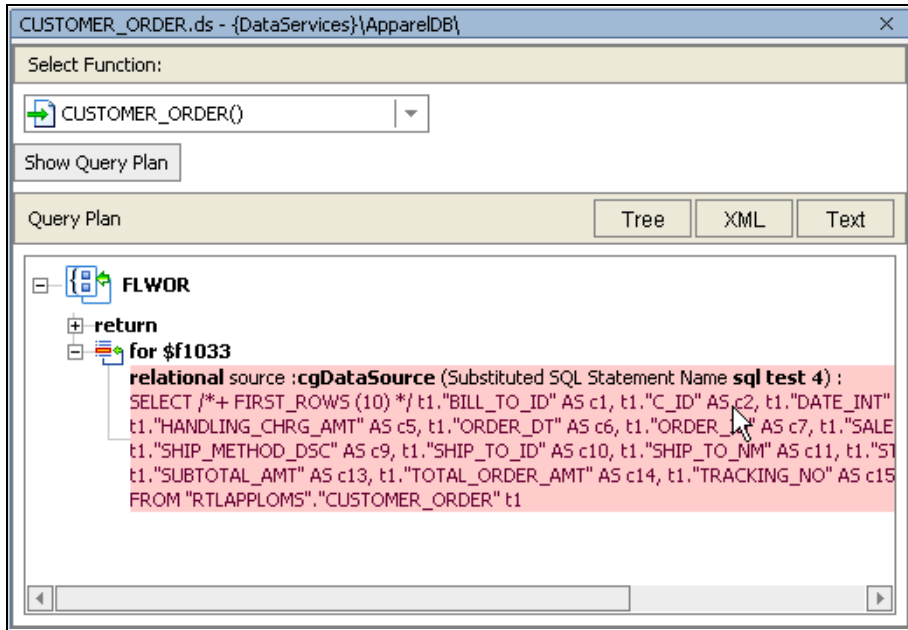
9. Return to the AquaLogic Data Services Console, Substituted SQL Statements area and paste the basesql statement into the field labeled Generated SQL Statement.

10. Also paste the basesql statement into the field labeled Substituted SQL statement.

11. Edit the substituted statement based on supported hints provided by the underlying database. A sample edited query — restricting results to the first 10 rows in an Oracle database (emphasis added) — appears below:

```
SELECT /*+ FIRST_ROWS (10)*/ t1."BILL_TO_ID" AS c1, t1."C_ID" AS c2,
t1."DATE_INT" AS c3, t1."ESTIMATED_SHIP_DT" AS c4,
t1."HANDLING_CHRG_AMT" AS c5, t1."ORDER_DT" AS c6, t1."ORDER_ID" AS c7,
t1."SALE_TAX_AMT" AS c8,
t1."SHIP_METHOD_DSC" AS c9, t1."SHIP_TO_ID" AS c10, t1."SHIP_TO_NM" AS c11,
t1."STATUS" AS c12,
t1."SUBTOTAL_AMT" AS c13, t1."TOTAL_ORDER_AMT" AS c14, t1."TRACKING_NO" AS c15
FROM {RTLAPPLOMS}.{CUSTOMER_ORDER} t1
```

> **WARNING:** Unlike SQL statements generated by AquaLogic Data Services Platform, substituted SQL statements are passed to the underlying database in an unvalidated form. For this reason users are strongly advised against using this feature for any purpose other than providing hints to the database. It is also recommended that prior to deployment any substituted SQL statement be tested against its generated counterpart to make sure that the expected performance advantage is being obtained.

12. Return to your Workshop application and re-run your query in Test View. Notice in the Output pane that your substitute query appears in the SQL Statement area.

13. Select the CUSTOMER_ORDER( ) query in Query Plan View. Click the Show Query Plan button. Notice that the resulting plan contains the substituted SQL as well as the named of the substituted SQL statement.

Figure 5-8 Query Plan Displaying Substituted SQL Query

Guidelines for Setting Server Thread Count

The optimal thread count settings you configure depends on the physical resources of the machine on which you deploy AquaLogic Data Services Platform, the anticipated load, and the type of application you are deploying. Increasing the number of threads can accelerate processing, but since each thread consumes memory, you must achieve a balance based on the available resources.

Use the following general guidelines for settings the thread count:

- The maximum threads set for an application should not exceed the WebLogic Server thread count.

- The total maximum application thread counts for all deployed applications should not be significantly greater than the total WebLogic Server thread count.

AquaLogic Data Services Platform only uses the thread pool for acquiring web service calls; threads are only spawned when web services are invoked by queries. Therefore, an application that does not rely on web service content can have a relatively low thread count setting.

For more information on tuning performance for the WebLogic Server and applications, see the following:

http://e-docs.bea.com/wls/docs81/perform/index.html

Monitoring Applications

You can view statistics and status information for a AquaLogic Data Services Platform application, particularly relating to query activities, using the Monitor tab. You can also monitor active application processes, displaying information such as the user who initiated the process, the time is has been running, and the number of cached entries for the process type.

To monitor an application:

1. Click the name of the application node in the Navigation pane of the Data Services Platform Console.

   The General settings page appears. Note that you must be logged into the console using a user name with administrator privileges.

2. Click the Monitor tab.

   The monitoring information for the application appears, as illustrated in Figure 5-9.

Figure 5-9 AquaLogic Data Services Platform Console Application Monitor Tab
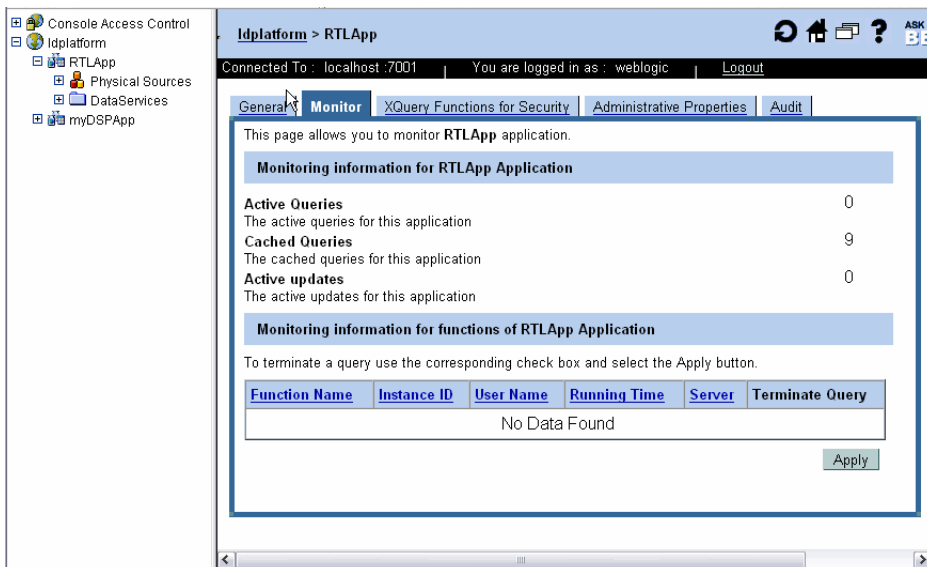
Table 5-3 describes the information displayed in the Monitor tab.

Table 5-3 Monitoring Statistics for the Liquid Data Server

Section	Field	Description
Monitoring information for... Application	Active Queries	The number of query instances currently running.
	Cached Queries	The total number of XQuery plans currently cached in memory. A cache entry is made for each distinct invocation of the named function with different input parameters.
	Active Updates	The number of update functions currently running.
Monitoring information for functions of... Application	Function Name	The name of the function for which the statistics apply.
	Instance ID	The unique identifier assigned to the process by the AquaLogic Data Services Platform runtime components.
	User Name	For secured data services, the name of the user that invoked the service.
	Running Time	The amount of time the query has been running in milliseconds.
	Server	
	Terminate Query	Checkbox option allowing you to terminate an executing query associated with a function.

Terminating an Executing Query

Once invoked, a data service function runs until either it gets a result or a time-out expires (assuming a time-out period is set). The time-out setting enables you to specify, in the query, the maximum time a query should wait for unresponsive data sources.

In some cases, it may be necessary to cancel the execution of a function. The Monitor tab enables you to view and cancel currently running queries. The page also displays the user associated with the query and cache information.

When you terminate a process, the operation in progress finishes, then the process completes without executing subsequent nodes.

Note: The submit query is rolled back only in cases when you are using the XA driver.

To terminate function execution:

1. Click the name of the application in the Navigation pane.

   The General settings page appears. (Note that you must be logged into the console using a user name with administrator privileges.)

2. Click the Monitor tab.

   The list of functions currently running appears in the functions table.

3. Select the check box in the Terminate Query column for the appropriate function, and click Apply to terminate the query.

   A confirmation dialog box is displayed.

4. Click OK to confirm, or Cancel to dismiss the dialog and cancel the action.

Note: Terminating a query triggers a weblogic.xml.query.exceptions.XQuerySystemException on the client.

Using Administrative Properties

An administrative property is a user-defined property that you can configure using the AquaLogic Data Services Console. The value of an administrative property can be used in XQuery functions, either in data service functions or security XQuery functions.

Note: For information on security XQuery functions, see Chapter 6, "Securing AquaLogic Data Services Platform Resources."

An administrative property is a convenient way of having function parameters that can be easily changed by the administrator, without having to modify the body of either the data service function or security XQuery function.

The administrative property has application scope — any data service in the application can use the property value. The property value can be accessed using XQuery with the BEA function `get-property()`. The function takes the name of the property as an argument and returns the value as a string. It also takes an argument that serves as the default value for the parameter. This value is used if the property is not configured in the console.

The following shows a complete example of an XQuery Function Library function using an administrative property:

```
declare function f1:getMaximumAccountViewable() as xsd:decimal {
   let $amount := fn-bea:get-property("maxAccountValue", "1000.00")
                              cast as xsd:decimal
   return $amount
};
```

To manage administrative properties:

1. Click the name of the application in the Navigation pane. The General Settings page appears. (Note that you must be logged into the console using a user name with administrator privileges.)

2. Click the Administrative Properties tab. The list of property names currently defined appears in the table, as illustrated in Figure 5-10.

Figure 5-10 Administrative Properties Tab
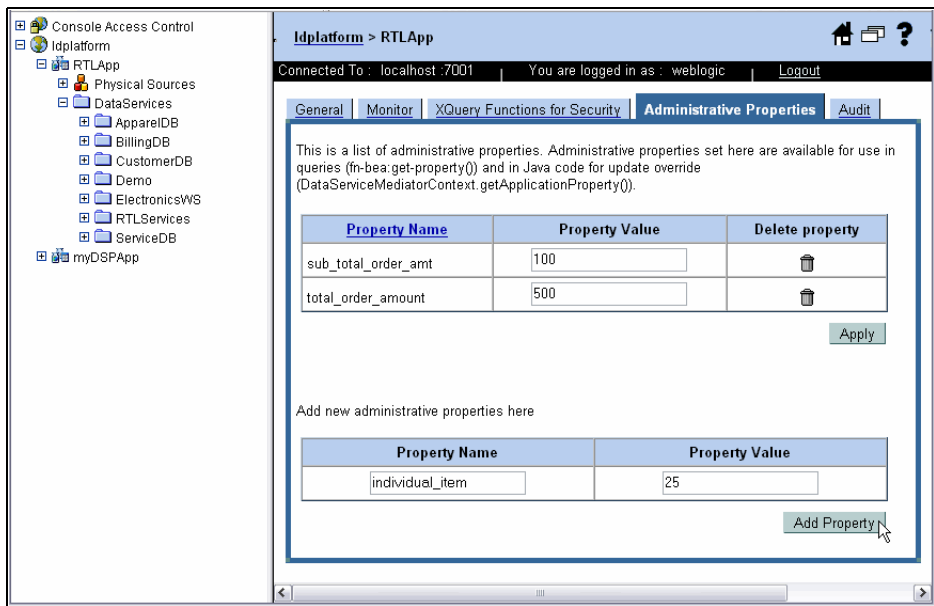
Table 5-4 describes the information displayed in the Administrative Properties tab:

Table 5-4 Administrative Properties

Column	Description
Property Name	The name of the administrative property.
Property Value	The current value of the property.
Delete Property	A Trash icon enabling you to delete the property.

3. To add a property, complete the following:

    a. Enter a name for the property in the Property Name field.

    The name must match the name property passed to the `get-property()` function used to access the properties value. For example:

    ```
    fn-bea:get-property("maxAccountValue", "1")
    ```

    b. Optionally, enter an initial value for the property.

    You can change this value later, if required.

    c. Click Add Property.

    The property appears in the list.

4. To change a property value:

    a. Enter a new value in the Property Value field (in the list of currently defined properties).

    b. Click Apply.

5. To delete a property:

    a. Click the delete icon (🗑) next to the property.

    b. Confirm the delete when prompted.

    Note that the default value for the property is used in any `get-property()` call using the deleted property.

Setting the Transaction Isolation Level

In some instances, AquaLogic Data Services Platform may not be able to read data from a database table because another application has locked the table, causing queries issued by AquaLogic Data Services Platform to be queued until the application releases the lock. To prevent this, you can set the transaction isolation to read uncommitted in the JDBC connection pool on your WebLogic Server.

To set the transaction isolation level:

1. Start the Administration Console in a web browser by opening the following URL:

   ```
   http://<HostName>:<Port>/console
   ```

   For example, to start the Administration Console for a local instance of WebLogic Server (running on your own machine), type the following URL in a web browser address field:

   ```
   http://localhost:7001/console/
   ```

2. Expand Services → JDBC → Connection Pools under the domain in which the AquaLogic Data Services Platform application runs, and click the name of the connection pool you want to configure.

   The Connections tab appears, as illustrated in Figure 5-11.

Figure 5-11 Connections Tab
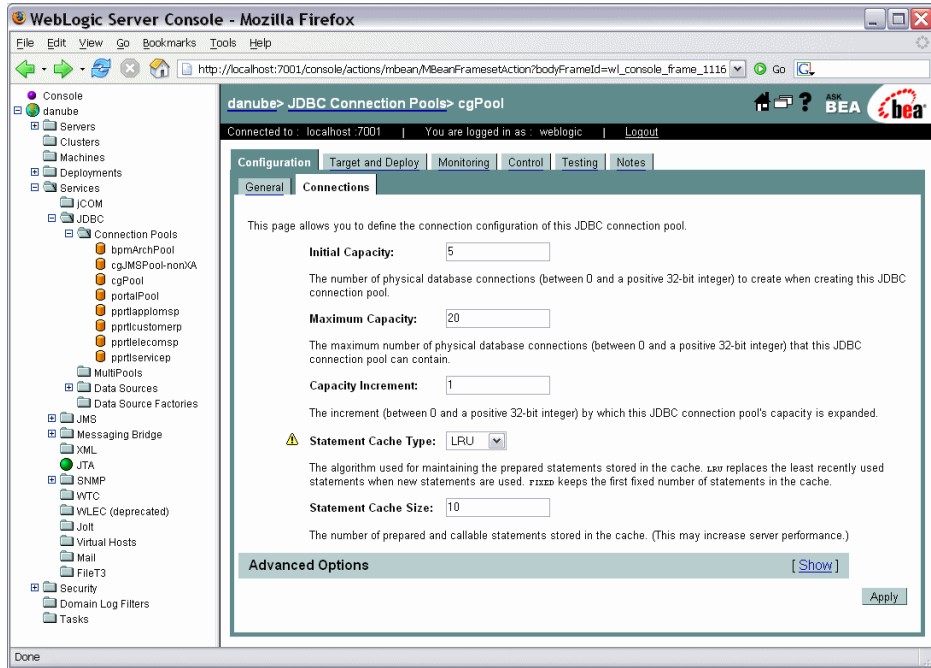
3. Click Show in the Advanced Options section of the page.

   The page expands to include the Advanced Options section.

4. Scroll to the bottom of the section, and enter the following in the Init SQL field:

   ```
   SQL SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
   ```

5. Click Apply.

Securing AquaLogic Data Services Platform Resources

This chapter describes how to secure AquaLogic Data Services Platform resources, in particular, how to control access to those resources.

The chapter contains the following sections:

- Introducing AquaLogic Data Services Platform Security
- What is a Securable Resource?
- Understanding Security Policies
- Securing Data Services Platform Resources
- Securing Access to the Data Services Platform Console
- Exporting Access Control Resources

Introducing AquaLogic Data Services Platform Security

AquaLogic Data Services Platform uses the security features of the underlying WebLogic platform to ensure the security of the information it provides. Specifically, AquaLogic Data Services Platform uses role-base security policies to control access to data resources.

For a secured resource, a requesting client must meet the condition of the security policy applicable to that resource, whether accessing the resource through the typed mediator API, an ad hoc query, or any data access interface. A typical condition is based on the role of the user identified by the credentials passed by the client. But other types of conditions are possible as well, including policies based on time of day or user identity.

AquaLogic Data Services Platform exposes its deployed artifacts as resources that can be secured through WebLogic role-based security policy control. With AquaLogic Data Services Platform, you can apply security policies at various levels, from the application to individual data elements. This range gives you significant flexibility. For example, you can control access to an entire AquaLogic Data Services Platform deployment or just to a credit card number element in an order.

When a request comes to AquaLogic Data Services Platform for a secured resource, AquaLogic Data Services Platform passes an identifier for the resource to WebLogic. WebLogic, in turn, passes the resource identifier, user name, and other context information to the authorization provider. The provider evaluates the policy that applies to the resource given the information passed by WebLogic. As a result of the evaluation, access to the resource is either permitted or blocked.

If the user does not satisfy the requirements of an element-level policy, the element is *redacted* from the result object—it does not appear.

Figure 6-1 Data Redaction
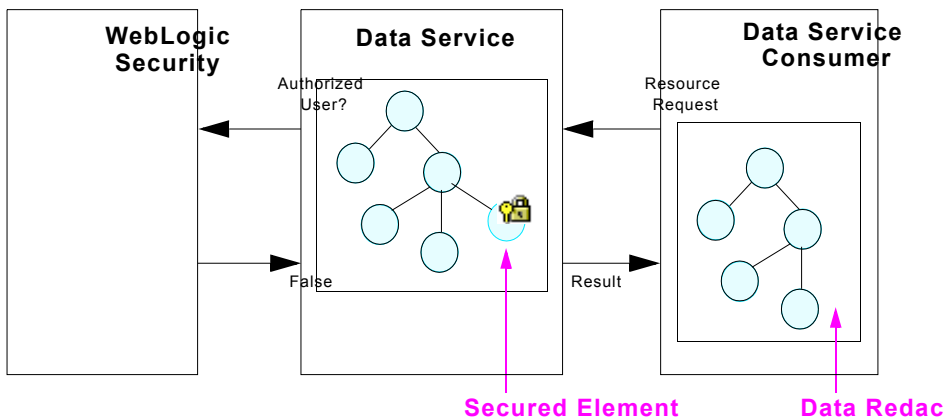
Note: By default, WebLogic security uses the ATZ authorization provider module. ATZ keeps policies in an LDAP system. Other authenticators can use any external resource necessary to implement the policy evaluation.

Setting up AquaLogic Data Services Platform security in the AquaLogic Data Services Console involves one or more of these tasks:

- Turning on access control checking for the application. Security policies are not applied unless this option is selected.

- Specifying the global, application-level default policy for anonymous users.

- Configuring security policies for data service functions.

- Identifying data elements that you want to secure and then configuring either security policies or custom XQuery security functions for the elements.

Note: Keep in mind that AquaLogic Data Services Platform directly supports the application of role-based security policies to its resources. The WebLogic Platform supports extensive security features that can be applied to your implementation as well, including encryption-based, transport-level security.

For information on WebLogic Server security, see "Managing WebLogic Security" in the WebLogic Server documentation.

You can also apply access controls to the AquaLogic Data Services Console interface itself. You can control user access to specific functionality in the console, for example, limiting developer access to the Metadata Browser portion of the console.

What is a Securable Resource?

A securable resource is a AquaLogic Data Services Platform artifact, such as a data element or function, to which you can apply a security policy. The resources you can protect with role-based security include:

- **Functions.** The policy applies to individual data service functions in an application.

- **Data elements.** A policy can apply to individual items of information within a return type, such as the salary property of a customer.

Note: When using a custom Authorization provider (other than the default WebLogic Authorization provider) you can also configure policies for data services. A data service policy applies to any of the data service's functions and data elements. See "Exporting Access Control Resources" on page 6-21 for more information about using custom Authorization providers.

Once you have secured individual resources, you can enable or disable security for the application. Security policies are inherited. This means that security enabled at the application level applies to all functions and elements within the application. If several policies apply to a particular resource, the more specific policy prevails. Therefore, for example, a policy on an element supercedes a policy for the data service.
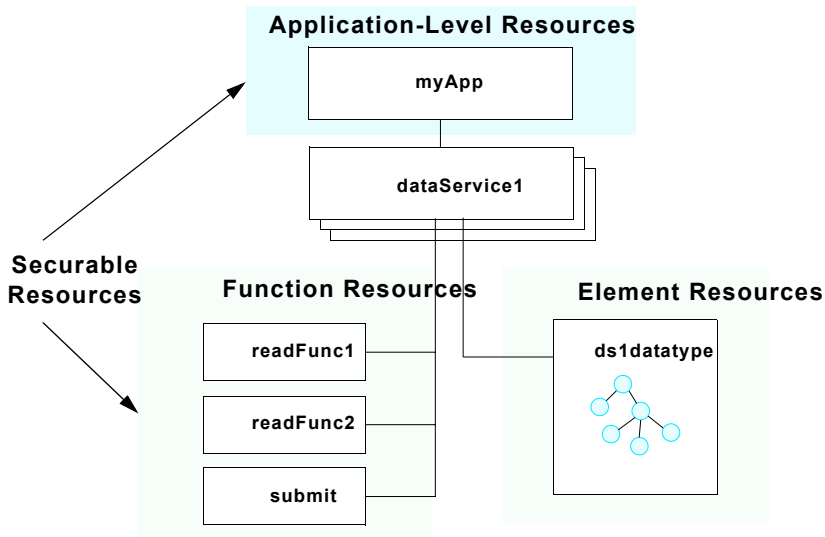
The hierarchy of AquaLogic Data Services Platform artifacts is as follows:

- Application

- Data service

- Function

- Data element

Figure 6-2 illustrates the securable resources in a AquaLogic Data Services Platform application.

Figure 6-2 Securable Resources

Enabling anonymous access is a special type of application-level setting. It enables you to either disable access to the application by default (unless a more specific policy permits it) or enable access (unless a more specific policy blocks it). If enabled, all users can access resources by default, even unauthenticated users. The anonymous access option works only with the WebLogic Authorization provider.

Note: Note that the AquaLogic Data Services Console itself constitutes an administrative resource you can secure with security policies.

Understanding Security Policies

A security policy is a condition that must be met for a secured resource to be accessed. If the outcome of condition evaluation is false—given the policy, requested resource, and user context—access to the resource is blocked and associated data is not returned.

Policies can be based on the following criteria:

- **User Name of the Caller.** Creates a condition for a security policy based on a user name. For example, you might create a condition indicating that only the user John Smith III can access the Customer data service.

- **Caller is a Member of the Group.** Creates a condition for a security policy based on a group. For example, you might create a condition indicating that only members of the finance group can access the Accounts data service.

- **Caller is Granted the Role.** Creates a condition based on a security role. A security role is a special type of user group for managing the common security needs of a group of users.

- **Hours of Access are Between.** Creates a condition for a security policy based on a specified time period.

- **Server is in Development Mode.** Creates a condition for a security policy based on whether the server is running in development mode.

The security policies you configure in the AquaLogic Data Services Console are intended to work with the default WebLogic Authorization provider. If you are using another provider, you will need to create policies using the facilities of the other provider. For more information, see "WebLogic Authorization Provider" in the *Administration Console Online Help* at:

http://e-docs.bea.com/wls/docs81/ConsoleHelp/security_defaultauthorizer_general.html

Using the WebLogic Policy Editor
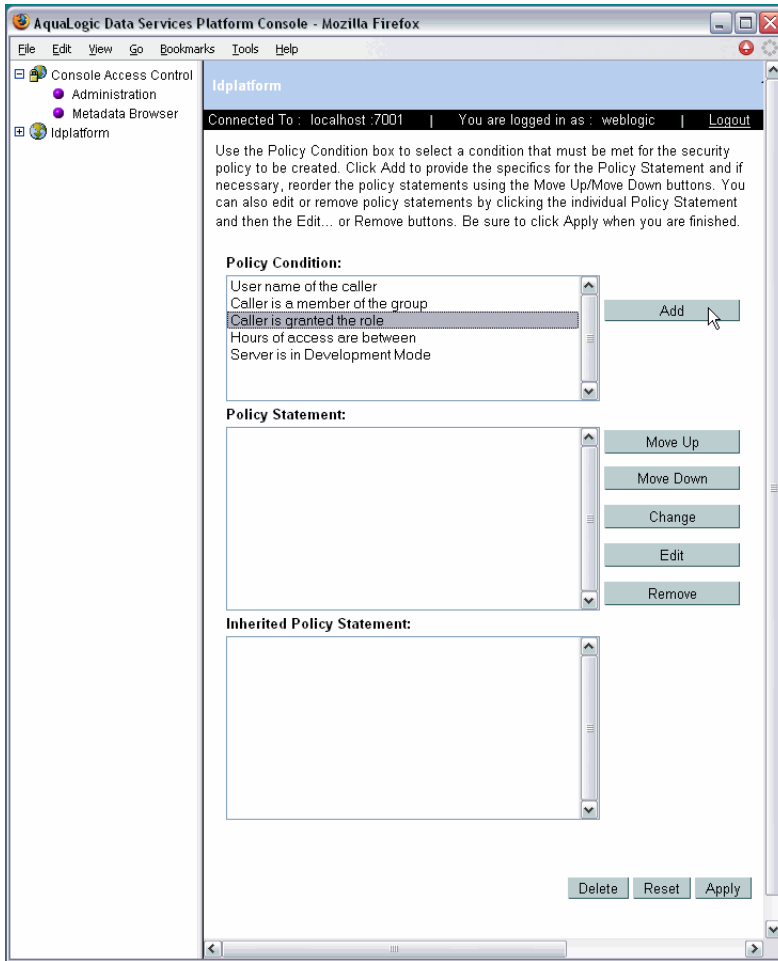
The AquaLogic Data Services Console incorporates the WebLogic Policy Editor interface for creating AquaLogic Data Services Platform security policies. You can use the policy editor for both AquaLogic Data Services Platform application resources — such as data elements and functions — and administrative resources.

To create a policy using the WebLogic Policy Editor:

1. In the Data Services Platform Console click on Administration Policies under Console Access Control.

2. Choose a condition from the Administration Policies list box.

   You can select any of the policy criteria listed, as shown in Figure 6-3.

Figure 6-3 Policy Condition Editor

3. Click Add.

    The window that appears depends on the condition you selected, as follows:

    – If you selected the Server is in Development Mode condition, no window appears. Instead
      the completed expression appears in the Policy Statement list box.

    – If you selected the Hours of Access are Between condition, use the Time Constraint window
      to select start and end times, and click OK. The window closes and an expression appears in
      the Policy Statement list box.

– If you selected one of the other conditions, use the Users, Groups, or Roles window to enter the name of a user, group, or security role, and click Add. An expression appears in the list box, as shown in Figure 6-4. Repeat this step to add more than one user, group, or security role, and click OK to add the expression to the policy statement. The window closes and an expression appears in the Policy Statement list box.

Figure 6-4 Policy Composition Window
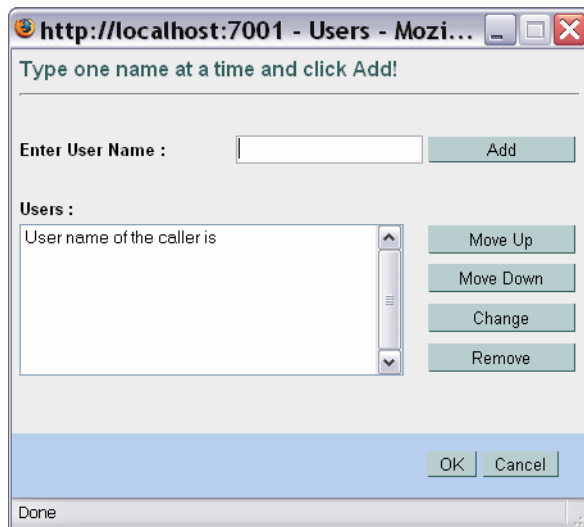
4. If needed, repeat steps 1 and 2 to add expressions based on different policy conditions.

5. After adding a policy, use the buttons located to the right of the Policy Statement list box to modify the expressions.

   The buttons enable you to do the following:

   – **Move Up/Move Down.** Changes the order of the highlighted expression, and therefore the order in which the expressions are evaluated.

   – **Change.** Toggles the compound operator that combines the selected expression and the previous expression between "and" and "or".

   – **Edit.** Reopens the edit window for the highlighted expression.

   – **Remove.** Deletes the highlighted expression.

6. Click Apply to save the security policies.

For more information on WebLogic security policies, see the WebLogic documentation at:

http://e-docs.bea.com/wls/docs81/secwlres/sec_poly.html

Securing Data Services Platform Resources

You can secure AquaLogic Data Services Platform resources by application, data service function, and element. An element-level security policy applies to all functions in the data service that use the data element.

To use element or function-level security, you must first specify access control checking for the application. Security policies are not applied to users unless access control checking is enabled.

This section describes the following topics:

- "Securing Applications" on page 6-8
- "Securing Data Service Functions" on page 6-11
- "Securing Data Elements" on page 6-12
- "Using Data-Driven Security Policies" on page 6-15

Securing Applications

Three optional checkboxes set security for your application (Figure 6-5). These are:

- Check Access Control
- Allow Default Anonymous Access
- Enable JDBC Metadata Access Control

These option are not mutually exclusive. In a deployed application, generally speaking, you would always want access control enabled.

Each of these options is described in this section.

Figure 6-5 Securing a AquaLogic Data Services Platform-Enabled Application
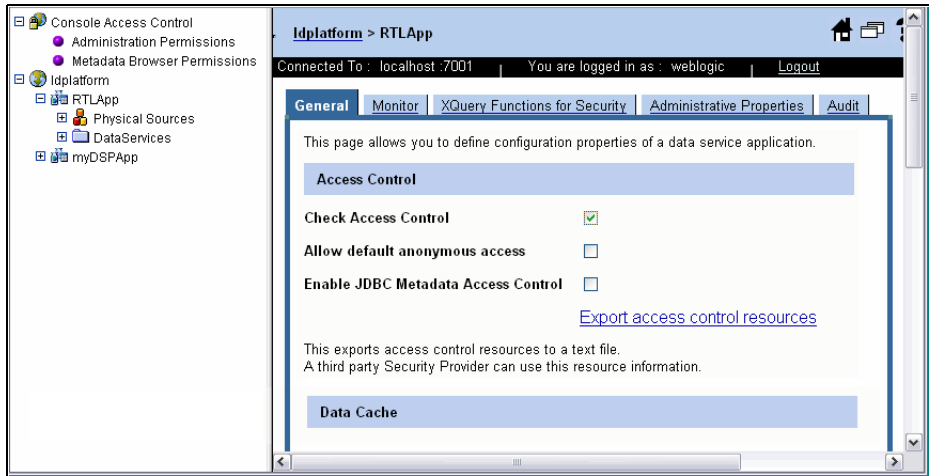
Enabling Security Access Control

Enabling access control checking activates the checking of policies throughout the application by the WebLogic Server authorization provider. Once access control checking is activated, access to any resource in the application is determined by the policy on that resource.

By default, access control is not enabled.

WARNING: If the access control option is not selected, none of the data in your application is secure.

Allowing Default Anonymous Access

For the default authorization provider, if access control is enabled and no specific overriding resource policy is defined, access will be denied.

You can "invert" access control policies by selecting the allow default anonymous access option. Or, put another way, if anonymous access is enabled, access to application resources is enabled unless a more specific policy blocks access.

Note: This option only applies to the default authorization provider in the WebLogic Server security framework. It works by defining a policy rule applied to a common parent resource of application resources.

By default, anonymous access is enabled.

Note: If you do not select this option, then you need to either selectively configure security policies on individual resources, or disable access control checks for all resources by clearing the Check Access Control option. The second option is not recommended.

Enabling JDBC Metadata Access Control

You can control metadata accessed through SQL by selecting the Enable JDBC Metadata Access Control option. This option allows AquaLogic Data Services Platform metadata access to users based on their access rights at the JDBC driver level. Selecting this option ensures that users are able to list only those tables and procedures which they are authorized to use.

By default, this option is not enabled.

Note: If an access policy is time-dependent or is changed and the metadata access control option is enabled, you may not be able to access the tables and procedures that had been listed.

Steps to Setting Security Policies for an Application

To set the access policy for a AquaLogic Data Services Platform-enabled Workshop application follow these steps:

1. Select the application node in the Navigation pane. (The security policy dialog box should appear similar to that shown in Figure 6-6.)

2. Establish whether access control is active or not. See "Enabling Security Access Control" on page 6-9.

   WARNING: If access control is not selected, then security is not enabled for your application.

3. Determine whether default anonymous access is allowed. See "Allowing Default Anonymous Access" on page 6-9.

4. Determine whether access policies are to apply to AquaLogic Data Services Platform metadata accessed through SQL. See "Enabling JDBC Metadata Access Control" on page 6-10 for details.

5. Click Apply at the bottom of the General Application settings.

6. Finally, you can set function or element level security policies, as well as control metadata access, on AquaLogic Data Services Platform resources. See "Securing Data Service Functions" on page 6-11 and "Securing Data Elements" on page 6-12.

Securing Data Service Functions

A data service typically has several functions, including one or more read functions, navigation functions, and a single submit function. A submit function allows you to update back-end data sources. Function-level security policies enable you to control:

- User access to data service functions. Enables you to set stricter controls on the ability to change data, for example, compared to the ability to read data.

- Access times to data service functions. Enables you to control the times when a particular function can or cannot be accessed.

WARNING: Be sure to configure policies on the data service resources that are accessed directly by the user. Security policies on data services that are used by other data services are not inherited by the calling data service. This means that if a data service with a secured resource is accessed through another data service, the policy is not evaluated against the caller.

WARNING: For the purposes of security, data service functions are identified by name and number of parameters. This means that if you modify the number of parameters, you will need to reconfigure the security settings for the function.

Creating Function Security Policies

To create a function security policy:

1. Expand the folder containing the data services for which you which to establish function security policies. This folder is located below server application folder in the Navigation pane (see Figure 6-6).

2. Select the data service you want to configure.

3. Select the Admin tab.

4. Select the Security tab. The functions in your data service appear as resource names.

Figure 6-6 Security Policy Function List
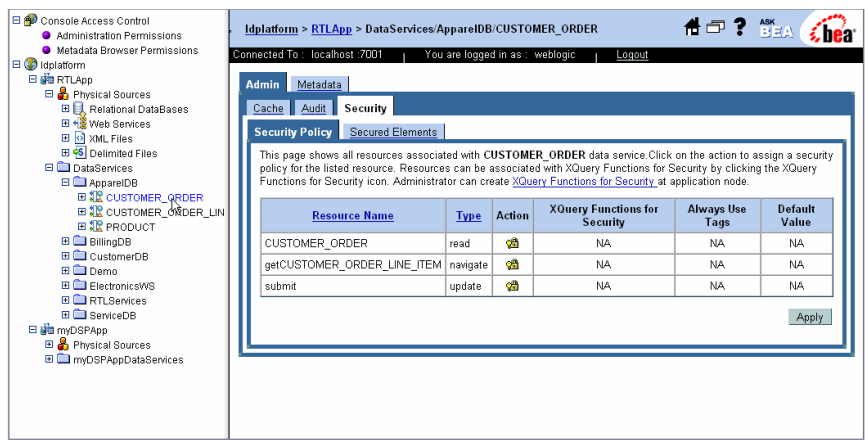
5. Click the Action icon ( ).

6. Use the WebLogic Policy Editor to create a policy for the function.

    For more information, see "Using the WebLogic Policy Editor" on page 6-5.

Note: You must enable access control for the application in order to have function-level security policies applied to users. For more information, see "Securing Applications" on page 6-8.

The other options shown in Figure 6-6 are described under "Creating Security Defaults for Data Elements" on page 6-14.

Securing Data Elements

Element-level security associates a security policy with a data element within a data service's return type. If the policy condition is not met, the corresponding data is not included in the result.

An element-level security policy applies across all functions of the data service but not to any other data services. In other words, a security policy set on a particular data service is not inherited. If the same data composes another data service, either from the source or as an inclusion of the data service on which the policy is configured, the policy does not apply to users of those data services.

When configuring element-level security, you first identify the element as a securable resource, then set a policy on the resource.

To configure a data element security policy:

1. Expand the data services folder under the application node in the Navigation pane.

2. Select the data service you want to configure, and click the Security tab.

   The functions in the data service appear.

3. Click the Secured Elements tab.

   A tree representing the data type appears, as illustrated in Figure 6-7.

Figure 6-7 Secured Elements Tab
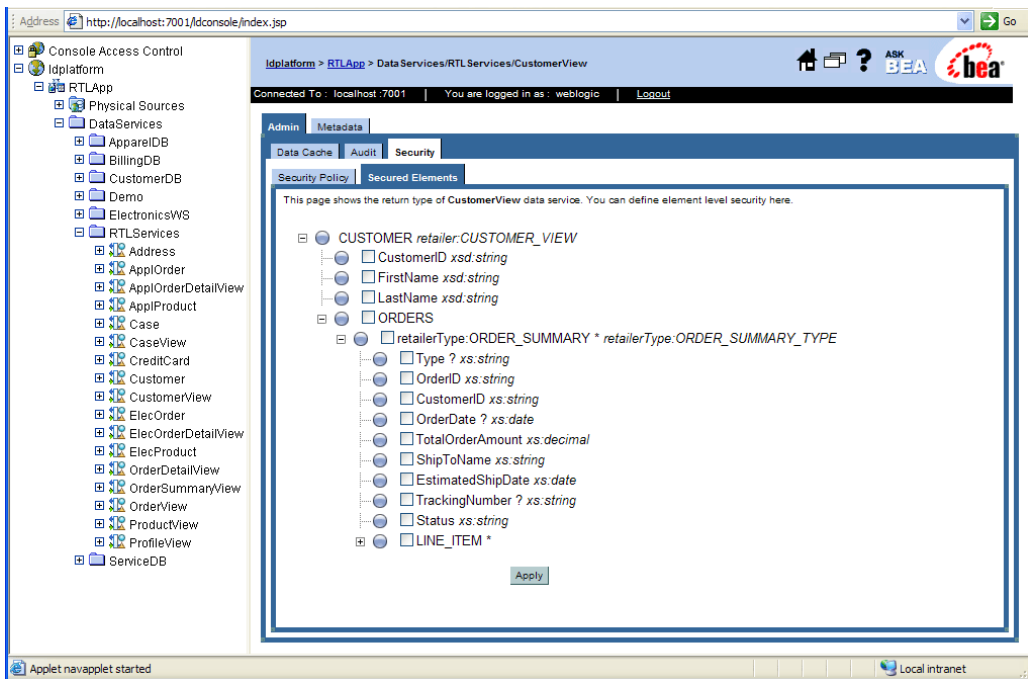
4. Select the check box next to the data elements you want to secure. Selecting a parent node includes all children of the parent.

5. Click Apply.

6. Click the Security Policy tab (Figure 6-6). The element now appears in the resources list as an element type.

7. Create a security policy or a custom security condition for the element.

Click the Action icon ( ) to create a security policy. Click the Security XQuery function icon
( ) to create a custom security condition.

For more information, see "Using the WebLogic Policy Editor" on page 6-5 or "Using Data-Driven Security Policies" on page 6-15.

Note: You must enable access control for the application to have the data element-level security policies applied to users. For more information, see "Securing Applications" on page 6-8.

Creating Security Defaults for Data Elements

The security defaults feature allows you to specify fixed values (or *mandatory elements*) for any data service fields with a return types. These values are used in cases where access control restricts access to the data service.

There are three ways to represent a secured field on which access is restricted:

- **Omit the field from returned data.** In this case the element or attribute is removed from the result. This can only be done when the element or attribute is optional.

- **Provide a default value for the field.** The element or attribute is assigned a constant value. Only primitive values are permitted; complex types cannot have default values.

- **Supply an empty value for the field.** This is the case where the default value supplied for the field is an empty string. This can only be used for types that allow an empty string as a valid value.

These settings are achieved through the data service security policy list (Figure 6-6). The options available are shown in Table 6-1.

Table 6-1 Data Service Security Policy List Options

Option	Meaning
Always Use Tags	If selected, the secured field will always be placed in the result and contain the default value if access is restricted.
	If the option is not selected, the secured field will be omitted if access is restricted. In this case the default value is never used.
	The check box is, by default, be selected for mandatory element/attributes. It is, by default, set to an unselected state for complete type elements and for optional elements and attributes.
Default Value	This field contains any default value you want to assign for attributes or elements. The default value is returned only if the Always Use Tag is also selected.
	By default, the contents of the field is an empty string.
	Note: No type or other validation is performed on the entered default value. Thus you should ensure the validity of enter values to avoid unexpected problems.

Using Data-Driven Security Policies

A security XQuery function enables you to specify custom security policies that can be applied to data elements. In particular, security XQuery functions are useful for creating data-driven policies (policies based on data values). For example, you can block access to an element if the order amount exceeds a given threshold.

Note that if both a standard security policy and a custom XQuery security function applies to a given data element, the results of the two policy evaluations must both be true for access to be permitted (a logical *and* is applied to the results).

You can apply security XQuery functions to any element resource. Applying data-driven security policies involves the following steps:

1. Identify the element as a secured element. (For more information, see "Securing Data Elements" on page 6-12.)

2. Create a security XQuery function to define the data-level security. (For more information, see "Creating a Security XQuery Function" on page 6-16.)

3. Apply a security XQuery function to a data element. (For more information, see "Applying a Security XQuery Function" on page 6-18.)

Creating a Security XQuery Function
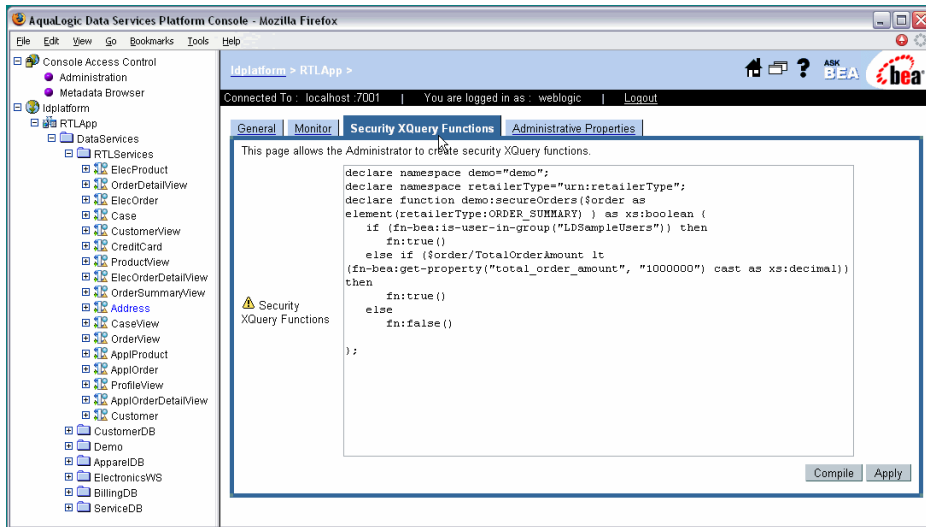
You can create one or more security XQuery functions to apply against data elements in an application. You define the functions in the Security XQuery Functions tab.

To create a security XQuery function:

1. Select the application node in the Navigation pane.

2. Click the Security XQuery Functions tab.

   Existing XQuery functions are displayed, as illustrated in Figure 6-8.

Figure 6-8 Security XQuery Functions

3. Add the XQuery function body in the text area of the tab.

   Add as many functions as required. The functions are applied to elements by qualified function name. The only requirement for the function is that it returns a Boolean value and that the name be qualified by a namespace.

4. After adding the function text, click Compile.

   An output window provides feedback on the compilation.

Note: For details on creating XQuery functions, see AquaLogic Data Services Platform *XQuery Developer's Guide*.

5. Click Apply when you have finished adding functions.

6. Redeploy the application from the WebLogic Administration Console for the changes to take effect.

   To redeploy the application:

   a. Open the WebLogic Administration Console.

   b. Select Deployments → Applications → *application_name* in the domain tree to open the application configuration page.

   c. Click the Redeploy tab, and click Redeploy Application.

The return value of the function determines whether access is granted as follows:

- **True.** Access is permitted to the element protected by the function.

- **False.** Access is blocked.

The following shows an example of a simple security XQuery function:

```
declare namespace demo="demo";
declare namespace retailerType="urn:retailerType";
declare function demo:secureOrders($order as
element(retailerType:ORDER_SUMMARY) ) as xs:boolean {

if (fn-bea:is-access-allowed("LimitAccess",
"ld:DataServices/RTLServices/OrderSummaryView.ds")) then

    fn:true()

else if ($order/TotalOrderAmount lt
(fn-bea:get-property("total_order_amount", "1000000") cast as
xs:decimal))

then

      fn:true()

else

      fn:false()

};
```

Note: A security XQuery function must be applied to a data element for it to take effect. For more information, see "Applying a Security XQuery Function" on page 6-18.

Notice that the function uses the BEA extension XQuery function `is-access-allowed()`. This function tests whether a user associated with the current request context can access the specified resource, which is denoted by a element name and a resource identifier.

AquaLogic Data Services Platform provides the following additional convenience functions for security purposes:

- `is-user-in-group ($arg as xs:string) as xs:boolean`

  Checks whether the current user is in the specified group.

- `is-user-in-role ($arg as xs:string) as xs:boolean`

  Convenience method that checks whether the current user is in the specified role.

- `userid() as xs:string`

  Returns the identifier of the user making the request for the protected resource.

Applying a Security XQuery Function

You can use security XQuery functions to control access to data elements. Once you have defined the security XQuery function, as described in "Creating a Security XQuery Function" on page 6-16, you must apply the function to a data element for it to take effect.

To apply a security XQuery function:

1. Select a data service in the Navigation pane, and click the Secured Elements tab.

2. Choose the data element to which you want to apply a custom function.

3. Click the Security Policy tab.

   The Security Policy page appears, as illustrated in Figure 6-9.

Figure 6-9 Applying Security XQuery Functions
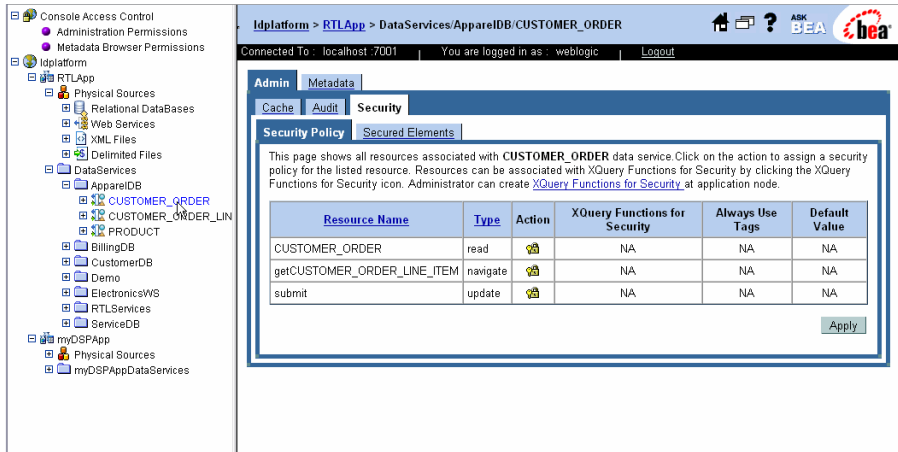
4. Click the security XQuery function icon ( ) corresponding to the data element you want to secure.

   Figure 6-10 illustrates the dialog that appears enabling you to add the qualified name of the security function.

Figure 6-10 Applying a Function to an Element
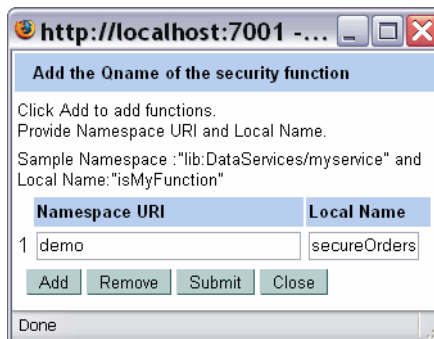
5. Click Add, and enter the Namespace URI and local name of the function to be applied to the data element.

6. Click Submit.

   Optionally, you can remove a function or add additional functions by clicking the Remove and Add buttons respectively.

7. Click Close.

8. Redeploy the application from the WebLogic Administration Console for the changes to take effect.

    To redeploy the application:

    a. Open the WebLogic Administration Console.

    b. Select Deployments $\rightarrow$ Applications $\rightarrow$ *application_name* in the domain tree to open the application configuration page.

    c. Click the Redeploy tab, and then Redeploy Application.

Securing Access to the Data Services Platform Console

Similar to the WebLogic Administration Console, the AquaLogic Data Services Console is itself an administrative resource for which you can control access using security policies. If a policy blocks a user from accessing a page, the page is omitted from the console.

Security policies control access by functional category of the page. The pages are divided into the following functional categories:

- **Administration pages.** Allows users to configure the deployment, for example, by setting cache and security policies.

- **Metadata pages.** Provide information on data services. They give users a read-only view of the type of information provided by data services, their names, data types, functions, and so on. You can specify policies that control who can access console pages based on this classification.

To create a policy:

1. Expand the Console Access Control node in the Navigation pane, and choose one of the following:

    – **Administration.** This enables you to specify policies for accessing AquaLogic Data Services Platform configuration pages in the console.

    – **Metadata Browser.** This enables you to specify policies for accessing the Metadata information tabs. The Metadata Browser is intended for AquaLogic Data Services Platform administrators and developers who want to use AquaLogic Data Services Platform services in their applications.

2. Add policy conditions for the resource, as appropriate.

    For more information on creating security policies, see "Understanding Security Policies."

3. Click Apply when finished.

Exporting Access Control Resources

Authorization is the process whereby the interaction between users and resources are limited to ensure integrity, confidentiality, and availability. WebLogic uses resource identifiers to identify deployed AquaLogic Data Services Platform artifacts, such as applications, data services, and functions. This identifier is used to associate a client request to any security policies configured for the requested resource.

Resource identifiers are managed for you when you use the default WebLogic Authorization provider and the AquaLogic Data Services Console to configure your policies. In particular, resource identifiers already exist for AquaLogic Data Services Platform applications, their data services, and data service functions. In addition, when you choose elements to be secured in the console, an identifier is generated for the element.

However, when using a custom authorizer, you will need to know the resource identifiers for your deployment and configure policies for the resources in the form expected by the other authorization module. This means that you will need to identify the element resources that you want to protect.

Note: The WebLogic security documentation provides details on how to connect another security authenticator to WebLogic. For more information, see "WebLogic Authorization Provider" in the *Administration Console Online Help* at:

http://e-docs.bea.com/wls/docs81/ConsoleHelp/security_defaultauthorizer_general.html

You can view the list of resource identifiers by exporting the access control resources from the AquaLogic Data Services Console.

To export the file:

1. Select the application node in the Navigation pane.

   The General application settings page appears.

2. Click the Export access control resources link.

   The File Save dialog appears.

3. Choose the location where you want to save the file, and click OK.

   An example of a portion of the file follows:

```
<ld type="app"><app>RTLApp</app></ld>
<ld type="service"><app>RTLApp</app><ds>ld:DataServices/ElectronicsWS/
```

```
                getProductList.ds</ds></ld>
<ld type="function"><app>RTLApp</app><ds>ld:DataServices/ElectronicsWS/
      getProductList.ds</ds><res>{ld:DataServices/ElectronicsWS/
      getProductList}getProductList:1</res></ld>
<ld type="submit"><app>RTLApp</app><ds>ld:DataServices/ElectronicsWS/
      getProductList.ds</ds><res>ld:submit</res></ld>
<ld type="service"><app>RTLApp</app><ds>ld:DataServices/RTLServices/
      OrderSummaryView.ds</ds></ld>
<ld type="custom"><app>RTLApp</app><ds>ld:DataServices/RTLServices/
      OrderSummaryView.ds</ds><res>ORDER_SUMMARY/ORDER_SUMMARY/
      LINE_ITEM</res></ld>
```

The format of a resource identifier is shown in Figure 6-11.

Figure 6-11 Resource Identifier Format

The resource can be any of the following:

- **Function.** A data service function, for example, `{ld:DataServices/ElectronicsWS/`
  `getProductList}getProductList:1`

- **Submit operation.** For example, `ld:submit`.

- **User defined or administrative entity.** A custom entity, such as a protected element or an arbitrary label defined in a data service that is used with fn-bea:is-access-allowed function, for example.

These are generated when you select an element in the Secured Element tab of the AquaLogic Data Services Console.

Configuring the Query Results Cache

This chapter describes how to set up and manage caching for data services in BEA AquaLogic Data Services Platform.

The chapter contains the following sections:

- Understanding Results Caching

- Setting Up Caching

- Purging Cache Entries

Note: Caching is only available for data service functions for which caching is allowed. For details see "Caching Functions" in the "Using Data Services Design View" chapter of the *Building Queries and Data Views*.

Caching is not available for ad-hoc queries or XQuery security functions.

Understanding Results Caching

By caching data returned by data service functions, you can improve response times for clients and reduce the processing burden on back-end systems.

Note: To use results caching, a database that has been certified for AquaLogic Data Services Platform caching support should be installed and running. Such DBMS systems are identified in the "Supported Configurations" section of AquaLogic Data Services Platform *Release Notes*.

When function caching has been authorized through Design View (see "Caching Functions" in the chapter of the Using Data Services Design View chapter of the *Building Queries and Data Views*) the

first time a data service function is run, AquaLogic Data Services Platform saves the results to a local *query results cache*. The next time the function is run with the same parameters, AquaLogic Data Services Platform checks the cache configuration and, if the results have not expired, retrieves the results from the cache rather than from the external source.

A cache entry exists for the results of each function invocation with distinct parameters. In cases when a cache-enabled function is invoked twice with two different parameters, two cache entries will be created.

By default caching is disabled. Once enabled, you can configure the cache and its time-to-live (TTL) for individual data service functions. Configuration tasks associated with caching include the following:

- Enabling caching for an application, and setting the cache data source and table names.

- Enabling caching of data service functions, and setting the cache time-to-live (which determines how long results are stored in cache).

- Monitoring and clearing the cache, as required.

The TTL setting is set individually, for each data service function. In general, the more dynamic the underlying data, the more frequently the cache should be set to expire. In some cases, caching should not be used at all. Here are two examples:

- If the data changes frequently and real-time access to it is critical cache should not be enabled. On the other hand, for functions that return static data, you can configure the results cache so that it never expires. If the cache policy expires for a particular function, AquaLogic Data Services Platform flushes the cache result automatically on the next invocation.

- Cache should never be set for functions without parameters. Every physical data service function based around a relational table, for example, falls into this category. Caching such a function can have a very negative impact of performance unless the table itself has very few records.

In the event of a Liquid Data Server shutdown, the contents of the results cache are retained. Upon server restart, the Liquid Data Server resumes caching as before. On first invocation of a cache-enabled function, the Liquid Data Server checks the results cache to determine whether the cached results for this function are valid or have expired, and then proceeds accordingly.

Caching API

AquaLogic Data Services Platform provides an API allowing client applications to bypass any existing cached results in favor of the physical data source. This API provides automatic client-side cache

refresh of the affected function. For details see the following discussions related to bypassing cached data in the *Application Developer's Guide*:

- "Bypassing a Data Cache When Using the Mediator API" in the Accessing Data Services from Java Clients chapter.

- "Bypassing a Function Results Cache When Using a Data Service Controls" in the Accessing Data Services from Workshop Applications chapter.

Note: Caching is particularly effective in cases when significant processing has been applied against large data sets, producing filtered results. For optimal performance, it is recommended that you not enable caching on functions that simply return large data sets directly from a relational database data source.
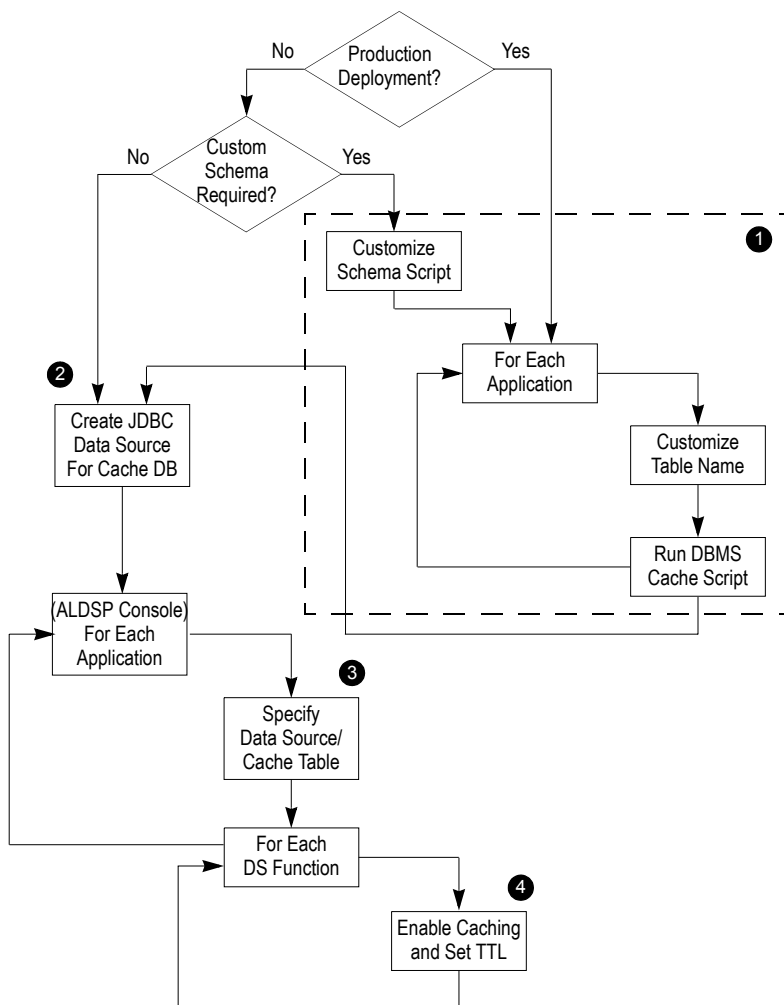
AquaLogic Data Services Platform can set up the cache table in the data source for you (if the server is in development mode), or you can create it yourself as described in the following section. Note that it is recommended that AquaLogic Data Services Platform application not share cache tables. There should be separate tables for each application.

Note: Since the AquaLogic Data Services Platform cache may contain sensitive data, it is important to maintain access control over the cache database so that only authorized users can access it. Also, it is recommended that the JDBC data source used for cache not be used for other purposes.

Setting Up Caching

The steps for setting up cache depend on several factors, including whether you are in development or production mode and whether you need to customize the cache table schema. Figure 7-1 shows the steps for setting up caching.

Figure 7-1 Cache Setup Steps

The steps illustrated in Figure 7-1 are described in the following sections:

- Step 1: (Optional) Run the SQL Script to Create the Cache Tables

- Step 2: Create the JDBC Data Source for the Cache Database

- Step 3: Specify the Cache Data Source and Table

- Step 4: Enabling Caching by Function

Step 1: (Optional) Run the SQL Script to Create the Cache Tables

For a WebLogic server that is in development mode, you can have AquaLogic Data Services Platform set up the cache table automatically from the AquaLogic Data Services Console using whichever data source you choose. For production environments, or if you want to customize the cache schema, you will need to run the SQL scripts manually.

You can create the cache table using SQL scripts in the subdirectory corresponding to a particular DBMS at the following location:

<WebLogicHome>/liquiddata/dbscripts/

For example:

<WebLogicHome>/liquiddata/dbscripts/oracle/ld_cache.sql

To create the cache table:

1. Open the script from the subdirectory that corresponds to your DBMS and modify the name of the created table so that it is unique for the application.

   It is recommended that each application keep its cached data in its own cache table. For example, you can name the table *<appname>*_CACHE.

2. Make any other schema changes, as required.

   You should not change the column names or otherwise modify the structure of the schema tables (except in specific cases, as noted in "Modifying the Cache Table Structure" on page 7-6). See Table 7-1 for information about the cache table schema.

3. Run the script.

4. Index the table based on the CHASH column (for retrieval) and the CUID column (for record updates).

   When the table is created automatically by AquaLogic Data Services Platform (as described in"Step 3: Specify the Cache Data Source and Table" on page 7-7), an index for CHASH is created. The automatically created name is the table name with "_INDEX" appended to it.

Note: On DB2, the name is truncated to a maximum of 18 characters.

Modifying the Cache Table Structure

AquaLogic Data Services Platform requires that its cache tables have a specific schema. Therefore, you should generally not modify the structure of the cache table. In some cases, however, the default column sizes may need to be adjusted based on the deployment. This may be a requirement in cases when you have data services that frequently serve result sets that are larger than the content columns in the default database tables and you are using either DB2 or Pointbase as your DBMS.

For DB2 and Pointbase, the scripts create the CINVKEY and CCONTENT columns (which store the results data) with a specific size, as shown in Table 7-1. If any serialized keys or content need to be larger than that size, the table schema should be adjusted accordingly before running the script.

Before attempting to implement customizations to the cache table, you should be familiar with the schema as shown in Table 7-1.

Table 7-1 Cache Table Schema

Column	Description
CUID	Unique numeric identifier for the cache entry.
CHASH	Hash value of the key (CINVKEY) as a 64-bit integer. This field enables fast searches, since searching by the key itself is inefficient as the key is stored as a binary object. (In fact, searching by the key itself is impossible for any DBMS for which the scripts create the CINVKEY as a BLOB type).
CEXPIRE	Timestamp value indicating when the record expires. This value is computed during record insertion as current time plus the TTL value defined for the function.
CFID	Serialized name of the function. When the table is created automatically, VARCHAR(512) type is used. The value should be adjusted to a lower or higher size if names of all functions in an application are smaller or if some names are larger then 512 characters.
CFARITY	The number of arguments the function accepts. This is used to differentiate functions in case of function overloading (not currently used).
CINVKEY	The serialized invocation identifier consisting of the function and its arguments (created with a size of 50 kilobytes on a Pointbase DBMS).
CCONTENT	Binary data constituting the cached results. (Created with size of 1 gigabyte for DB2 and 200K for a Pointbase DBMS.)

Step 2: Create the JDBC Data Source for the Cache Database

After creating the cache table, you can use the WebLogic Administration Console to create a JDBC data source on the WebLogic Server that points to the database that you have set up for the AquaLogic Data Services Platform cache.

Note: If using Oracle as your cache database, you must set the Honor Global Transactions setting to FALSE (it is set to TRUE by default). When you create the Oracle JDBC data source in the WebLogic Administration Console, you must uncheck the Honor Global Transactions box.

Once created, you can enable the result cache as described in the following section.

Step 3: Specify the Cache Data Source and Table

After configuring the table that you want to use for caching as a JDBC data source in the WebLogic Administration Console, you can set up the cache tables using the AquaLogic Data Services Console.

To specify the cache database and enable caching:

1. Select the application node in the Navigation pane.

   The General tab appears, as illustrated in Figure 7-2.

Figure 7-2 Enabling Results Caching for an Application
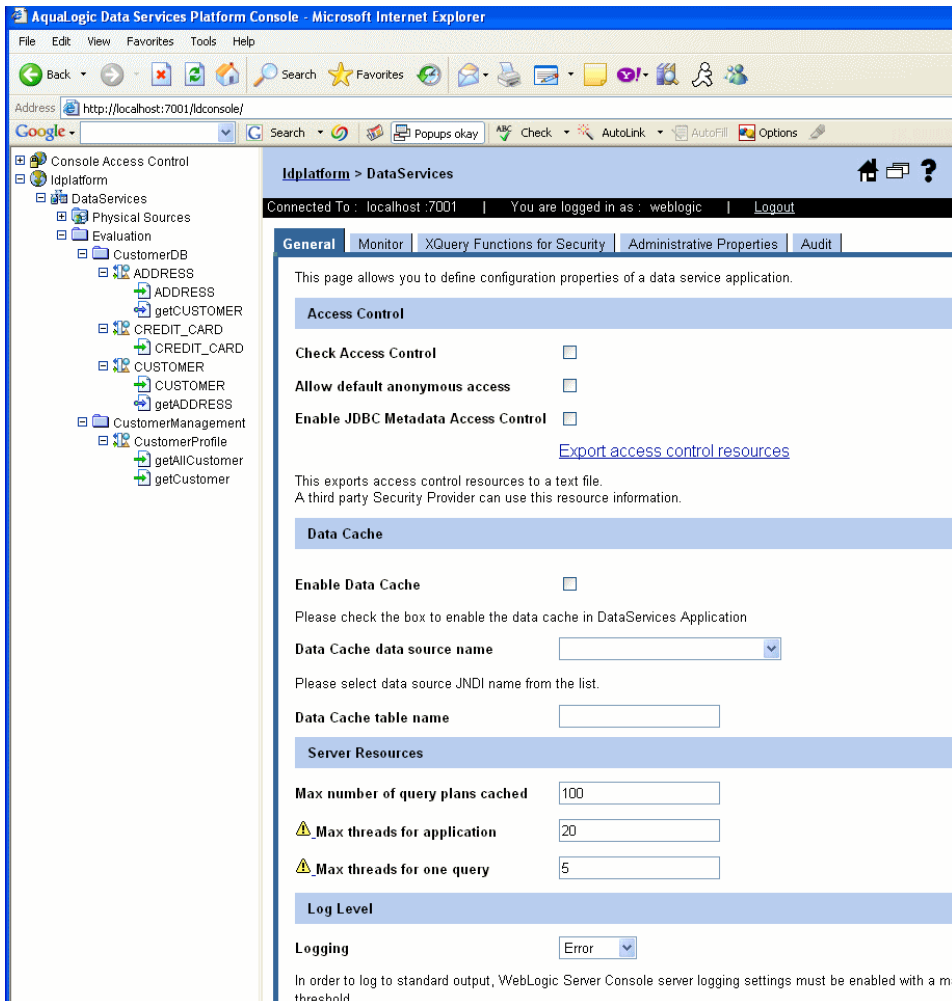
2. In the Cache section of the General tab, click Enable Cache.

3. Using the Cache Data source name drop-down list, choose the JNDI name of the data source you configured for the cache table.

   If you did not create a cache table, choose the data source in which you want AquaLogic Data Services Platform to create the cache table.

4. If you created a custom cache table for the application, enter its name in the Cache table name field.

   Otherwise, either enter another name for AquaLogic Data Services Platform to use when creating the table or leave the field blank, in which case the default name, *<appName>*_CACHE, will be used.

5. Click Apply.

Once caching is enabled, you need to configure results caching for each function.

Step 4: Enabling Caching by Function

After enabling Cache settings for the application, you can configure data service function caching. For each function, you can specify whether caching should be enabled, and set the time-to-live (in seconds) for cache entries.

To enable caching by function:

1. Click the data service name in the Navigation pane.

   The Cache page appears, as illustrated in Figure 7-3.

Figure 7-3 Enabling Caching by Function
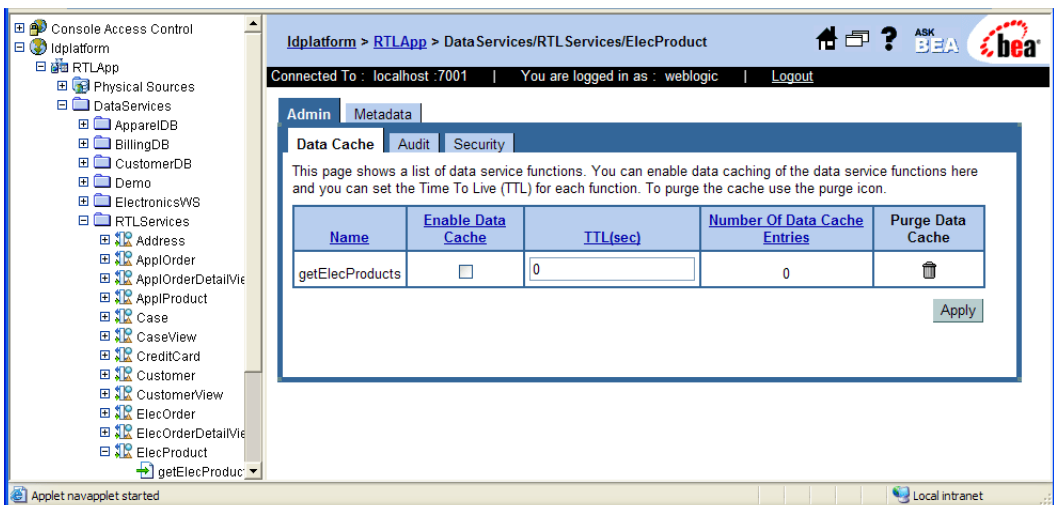
2. Check the Enable Cache checkbox for each function for which you want to enable caching.

3. Enter a time-to-live value, in seconds, for each cache-enabled function.

   The more dynamic the underlying data, the more frequently the cache should be set to expire.

4. Click Apply to save your changes. Notice that you can also purge the cache by function on this page and view the current cached entries.

Purging Cache Entries

Purging the cache removes cached entries from the cache database. When the cache is purged, each function will execute against its data sources until it is cached again. AquaLogic Data Services Platform flushes the cached query result for a given stored query whenever any of the following events occur:

- The data service function is modified or deleted
- Caching is disabled on the Liquid Data Server

AquaLogic Data Services Platform flushes the cached function result on the next invocation whenever any of the following events occur:

- The function results have expired per the cache policy
- The cache policy for a function is updated or deleted

You can also purge the cache manually, either for the entire application at once, or for individual functions. This section describes the following:

- "Purging the Cache for an Application" on page 7-10
- "Purging the Cache for a Function" on page 7-11

Purging the Cache for an Application

You can purge the cache for an application using the General Application Settings page. To purge the cache for an application:

1. Select the application node in the Navigation pane of the AquaLogic Data Services Console.

   The General Application Settings page appears, as illustrated in Figure 7-4.

Figure 7-4 Purging the Cache for an Application

Purge
Cache
for the
Application

2. Click the Purge Data Cache link in the Data Cache section of the General tab.

    The console asks for confirmation before purging the cache.
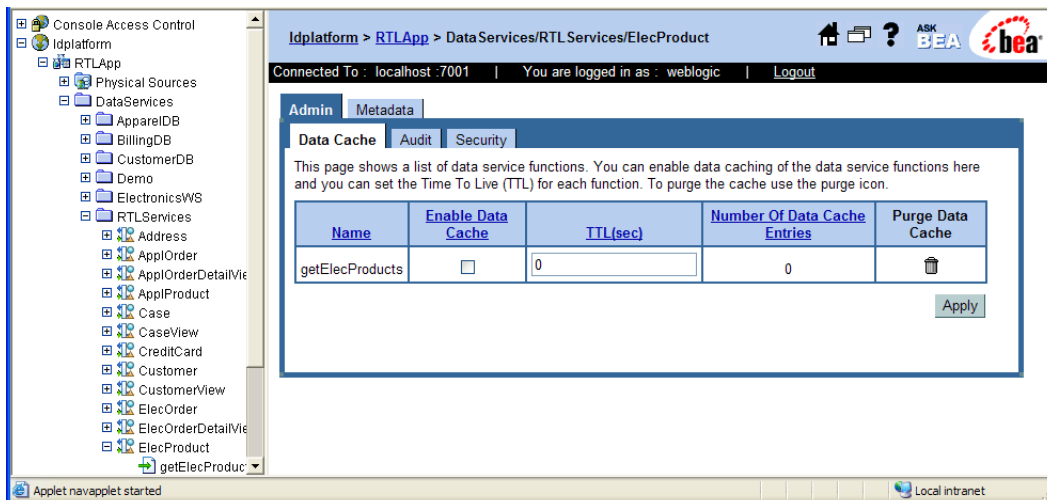
3. Click Yes.

    The purge occurs immediately, without having to apply changes.

Purging the Cache for a Function

You can purge the cache for individual functions using the Cache page, as illustrated in Figure 7-5.

Figure 7-5 Purging the Cache for a Function

To purge cache by function:

1. Click the data service for which you want to purge cache by function in the Navigation pane.

2. Click the Trash can next the function for which you want to purge cache.

Viewing Metadata

The Data Services Metadata Browser, a component of the AquaLogic Data Services Platform Console, enables you to view information on data services, their functions, and their dependencies in the currently active WebLogic Server.

This chapter describes how to use the MetaData Browser, and includes the following sections:

- Introducing the MetaData Browser

- Using the Metadata Browser

- Searching Metadata

Introducing the MetaData Browser

The MetaData Browser enables you to view metadata related to a AquaLogic Data Services Platform deployment. The information includes the data services that are deployed, their functions and return types, dependencies between data services, and more. Essentially, metadata documents the data model represented by the AquaLogic Data Services Platform deployment.

The MetaData Browser is particularly useful for:

- AquaLogic Data Services Platform administrators needing to gauge effects of changes to underlying data sources.

- Developers of AquaLogic Data Services Platform client applications wanting to determine what data services are available and their calling conventions.

You can use the MetaData Browser to access metadata in the following ways:

- Browse metadata by data service. You can display metadata associated with a specific data service. For more information, see "Metadata Browser Interface for Data Services" on page 8-3.

- Browse metadata associated with data service functions. You can display function metadata. For more information, see "Metadata Browser Interface for Data Service Functions" on page 8-9.

- Search for metadata in an application or project. You can perform basic or advanced searches on metadata in an application or in a project folder. For more information, see "Searching Metadata" on page 8-13.

Using the Metadata Browser

You can use the MetaData Browser to introspect both data service metadata and function metadata.

Metadata Browser Requirements for Data Lineage Graph

The Table 8-1 outlines the browser requirements to ensure visibility of the data lineage graph. If your system does not meet the requirements stated in the table, revert to the tabular view of the Metadata Browser.

The Adobe® SVG Viewer plugin required for Internet Explorer and Netscape can be downloaded from:

    http://www.adobe.com/svg/viewer/install/main.html

Table 8-1 Browser Support Information for Viewing Data Lineage Graph

Browser (Version)	SVG Viewer Information	Additional Information
Internet Explorer (6.0 and above)	Can auto-detect SVG viewer. If SVG viewer is not installed, a message is displayed with the URL to download the viewer. Install the viewer and the data lineage graph will be visible instantly.	• On Windows platform only.
Netscape (8.0)	Can auto-detect SVG viewer. If SVG viewer is not installed, a message is displayed with the URL to download the viewer. Install the viewer and the data lineage graph will be visible instantly.	• On Windows platform only. • You need to add the URL to the list of trusted sites to view the data lineage graph. Perform the following steps: a. Click the **Open Site Controls** icon on the browser tab when you log in to the Administration Console. b. In the pop-up dialog box, select the **I trust this site** radio button. c. Click **Done** to save your preference. This will enable you to view the data lineage graph.
Mozilla Firefox (1.5)	Has native SVG viewer support.	• On Windows and Linux platforms. • The data lineage graph is visible without the zoom in or zoom out operations. However, you can scroll up and down using the scroll bar.

Metadata Browser Interface for Data Services
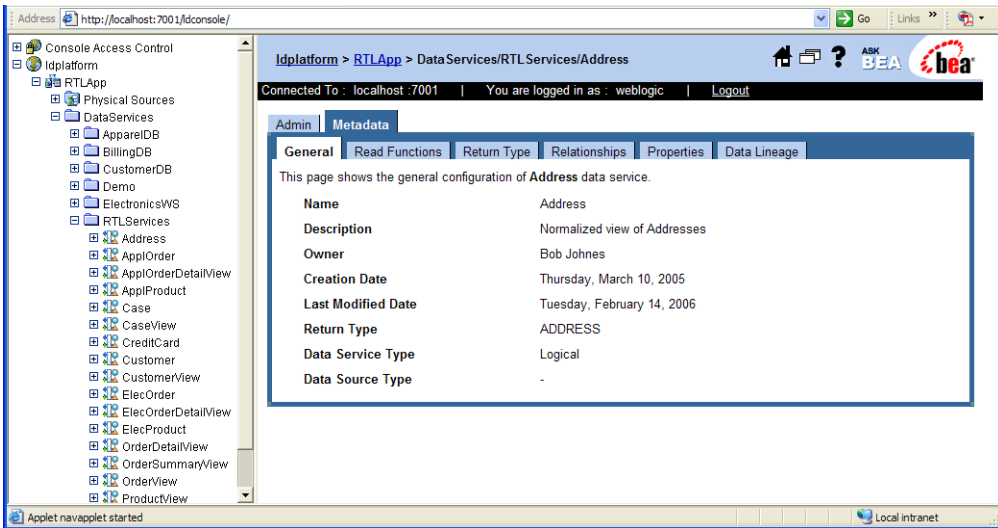
You can browse data service metadata including general information about a specific data service, its data lineage, its read functions and return types, relationships, dependencies, and more using the Metadata tab in the AquaLogic Data Services Console.

To browse data service metadata:

1. Select a data service in the Navigation pane. The Admin/Metadata screen appears (Figure 8-1). The Metadata tab in the console displays general information about the metadata associated with the data service.

Figure 8-1 Data Service Metadata

2. Click the corresponding tab to display general information, data service read functions, return type, relationships, properties, and data service lineage information. Table 8-2 describes the metadata information accessible through the tabs.

Table 8-2 Metadata Information

Tab	Description
General	Provides general configuration information about the data service, including the following: • **Name.** The name of the data service. • **Description.** A user-supplied description. • **Owner.** The owner of the service. • **Creation Date.** The date when the data service was created. • **Last Modified Date.** The date on which the data service was last changed. • **Return Type.** The type returned by the data service. • **Data Service Type.** Either physical or logical. For more information about data service types, see "Introspecting Data Service Metadata" on page 8-7. • **Data Source Type.** The type of the data source.
Read Functions	Displays a table of read functions. The table also lists the parameter names, if any, and return type (schema file name) for each function.
Return Type	Displays the content of the schema associated with the return type of the data service.
Relationships	Displays a table of related read functions. The table also lists the parameter names, if any, and return type (schema file name) for each function.
Properties	Lists any user-defined properties assigned to the data service.
Lineage	Provides a visual representation of the lineage between the currently selected data service. Relationships can be displayed in one of the two possible directions: • Dependencies • Where used Each entry includes name and path information.

Data Service Lineages

Data service lineages can be viewed in graphical or tabular format. The graphical view is ideal for getting a visual understanding of the lineage associated with a particular data service.

To start with, select a data service from the Navigation pane.

There are two ways to view a data service lineage:

- **Where used view.** The currently selected data service and the data services which make use of it appear. This is the *downstream view*.

- **Dependency view.** The currently selected data service and the data services it is dependent upon appear. This is the *upstream view*.

  Data service dependencies associated with navigation functions are shown based on the following rules:

  – If a data service contains a read function which calls a navigation function, the data service containing the navigation function appears as a dependency.

  – If the data service contains a navigation function that calls a read function (such as the constructor function in the related data service), that relationship is insufficient for the data service to be identified as a dependent.

    The reason for this is that navigation functions are often created automatically during the import metadata process. For details see "Obtaining Enterprise Metadata," in the *Data Services Developer's Guide*.

Figure 8-2 OrderView Data Service and Its Dependents
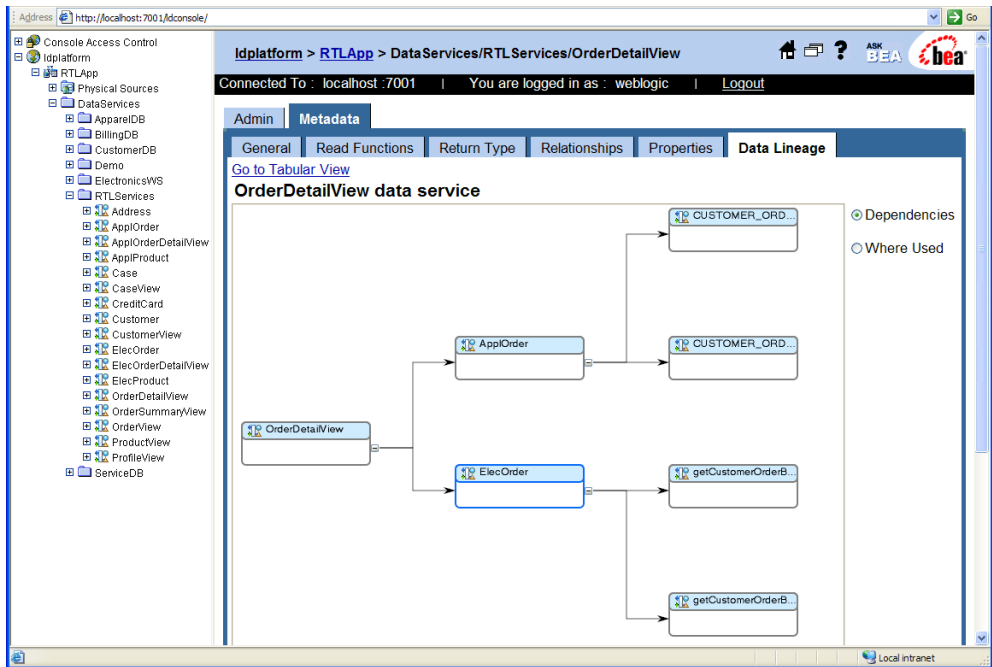
Once visual rendering appears, several options become available:

- **Panning (Alt + Click, then drag).** Allows you to move through the lineage representation in any direction.

- **Zoom out (Ctrl + Shift + Click).** Allows you to zoom out, providing information on data services that are further removed from your current selection.

- **Zoom in (Ctrl + Click).** Allows you to zoom in on a set of data services.

- **Expanding/Contracting.** You can use the +/- sign adjacent to the object to expand or collapse that node.

You can navigate to a new data service simply by double-clicking on it in the lineage diagram.

Note: Panning and Zoom operations work only with the Adobe SVG Viewer.

Introspecting Data Service Metadata

There are two types of data services:

- **Physical** data service**s.** These represent a single data source, typically a relational database table, stored procedure, or a web service.

- **Logical** data service**s.** These can be composed from multiple data sources and represent a view of data which typically is not available from any single data source.

The metadata that is available through the MetaData Browser varies depending on whether a data service is physical or logical. Logical data sources always have dependencies while the physical data services always have dependents.

Figure 8-3 illustrates a tabular view of dependencies and where used information of a logical data service.

Figure 8-3 Logical Data Service Dependencies and Where Used

As you would expect of a logical data service, the return type displays the schema of the data from multiple data sources, according to the design of the data service, as illustrated in Figure 8-4.

Figure 8-4 Return Type for a Logical Data Service

Metadata Browser Interface for Data Service Functions

You can browse metadata associated with a function.

To display function metadata:

1. Select a function in the Navigation pane.

   The console displays the General metadata associated with the function.

2. Click the corresponding tab to display general information, function dependencies, where used information, properties, and the return type.

   Figure 8-5 illustrates the function metadata displayed.
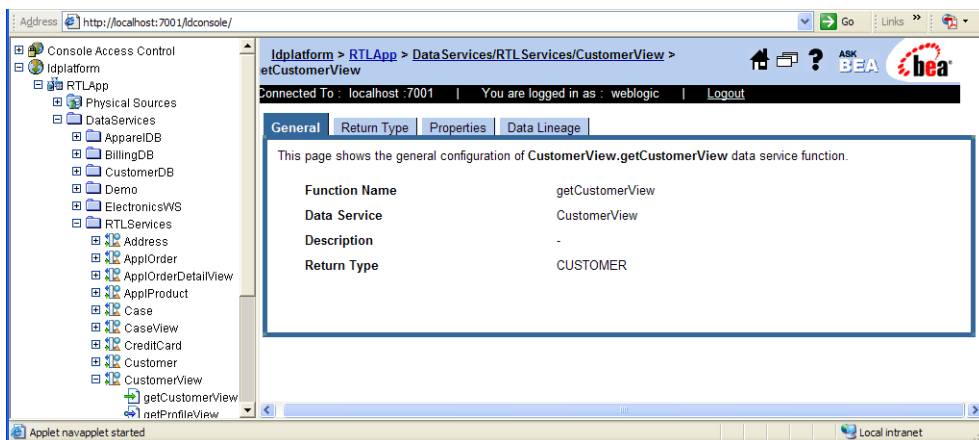
Figure 8-5 Function Metadata

Table 8-3 describes the function metadata available.

Table 8-3 Function Metadata

Function Metadata	Description
General	General metadata information for the function, including the following: • Function name. The name of the function. • data service. The containing data service. • Description. A user-supplied description of the function. • Return Type. The type returned by the function.
Lineage	Provides a visual representation of the relationships between the currently selected data service read, navigation, or private function. Lineage can be displayed in one of the two possible directions: • Dependencies • Where used • Each entry includes name, path, and type information.
Properties	Displays any user-defined properties associated with the function.
Return Type	Displays details about the return type of the function.

Data Service Function Lineages

Data service function lineages can be viewed in graphical or tabular format. The graphical view is ideal for getting a visual understanding of the lineage associated with a particular function. The view includes all functions that directly or indirectly call your selected function, or are called by your selected function.

To start with, select a data service from the Navigation pane. Click on the data service and then select from the list of available read or relationship functions.

There are two ways to view a data service function lineage:

- **Dependency view.** The currently selected data service function and any functions that it calls (said another way, it depends upon).

- **Where used view.** The currently selected data service function and any functions that make use of it (said another way, depend on it).

Figure 8-6 OrderView Data Service and Its Dependents
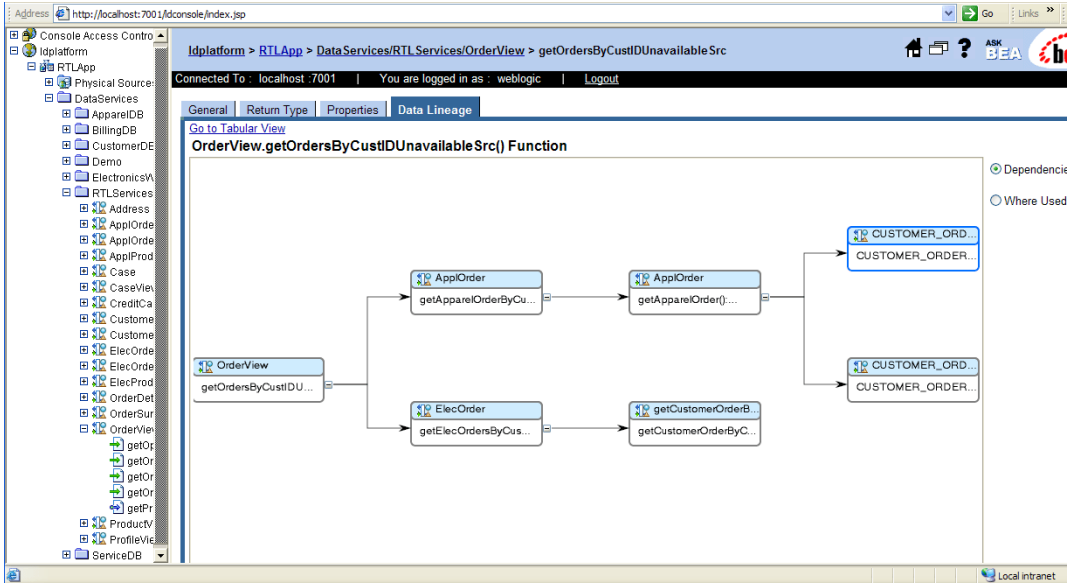
Once visual rendering appears, several options become available:

- **Panning (Alt + Click, then drag).** Allows you to move through the lineage representation in any direction.

- **Zoom out (Ctrl + Shift + Click).** Allows you to zoom out, providing information on data services that are further removed from your current selection.

- **Zoom in (Ctrl + Click).** Allows you to zoom in on a set of data services.

- **Expanding/Contracting.** You can use the +/- sign adjacent to the object to expand or collapse that node.

You can navigate between functions simply by double-clicking on the particular function of interest that appears in your graph. Lineage for both read and relationships functions can be traced.

Cyclic Dependency

Cyclic dependency can be observed in a graphical view of both data service lineages and data service function lineages. If a data service is used more than once, each instance of the data service in the graphical view is indicated in a dark blue color. Similarly, if a data service function is used more than once, each instance of the data service function in the graphical view is indicated in a dark blue color. Cyclic redundancy is applicable only when the duplicating nodes are part of the same branch.

Figure 8-7 shows the cyclic dependency of a data service. The text **`<<Recursive`** is specific to a data service and is displayed only in the case of a data service dependency.

Figure 8-7 Illustrating Cyclic Dependency of Data Services in a Graphic View
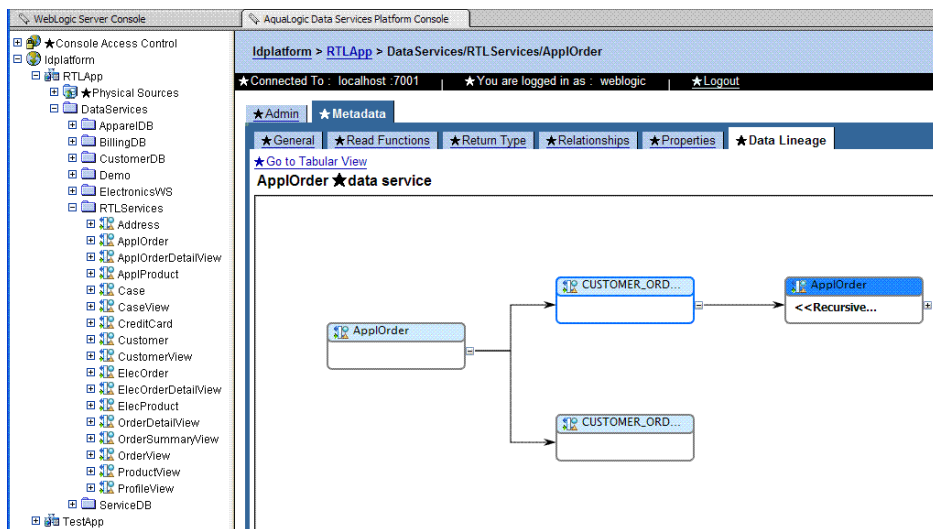
Searching Metadata

The MetaData Browser provides both a basic and an advanced search facility. You can use the search capabilities to locate data services based on metadata associated with the services. You can then generate a report using the results from either of the search modes.

Search algorithms that include wildcards are based on standards governing regular expression syntax. For detailed information on regular expression syntax see one of the following currently available Web sites:

- http://en.wikipedia.org/wiki/Regular_expression
- http://www.english.uga.edu/humcomp/perl/regex2a.html

Alternatively, any other standardized regular expression reference can be consulted.

The following topics are covered in this section:

- Performing a Basic Metadata Search
- Performing an Advanced Metadata Search
- Exploring Metadata Search Results
- Generating Reports

Performing a Basic Metadata Search

You can search for data services based on the data service name, description, function name, or return type.

To perform a basic search:

1. Right-click on an application or project node in the Navigation pane, and choose Search in the context-sensitive menu.

   The basic search screen appears, as illustrated in Figure 8-8.

Figure 8-8 Basic MetaData Browser Search Facility
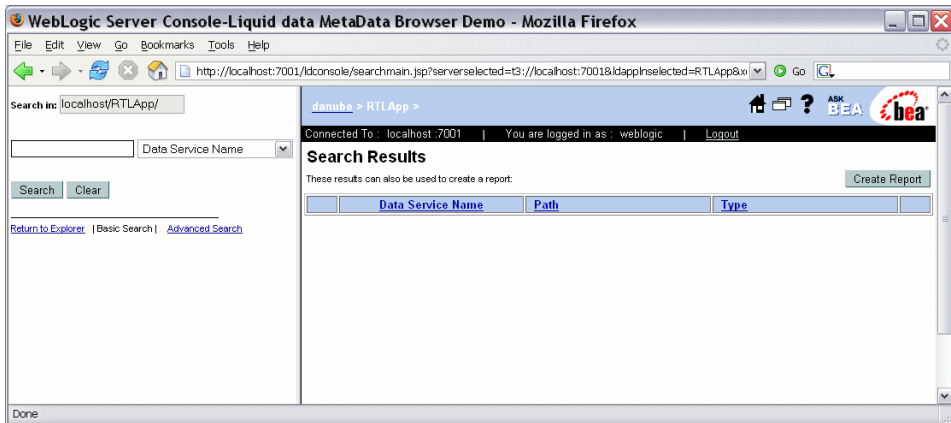
2. Choose the search criteria in the drop-down list.

   You can choose to search based on the data service name, description, function name, and return type.

3. Enter the search item in the text box, and click Search.

   The search results appear in the Contents pane. For more information about the search results, see "Exploring Metadata Search Results" on page 8-16.

4. Click Create Report in the Content pane to generate a report from the search results.

   For more information about generating reports, see "Generating Reports" on page 8-17.

5. Click Return to Explorer to exit the search facility and return to the main interface.

   Clicking Advanced Search enables you to specify additional criteria when performing a search. For more information, see "Performing an Advanced Metadata Search" on page 8-14.

Performing an Advanced Metadata Search

You can use the advanced search facility to narrow your search criteria in cases when a basic search produces a large number of results. Using the advanced search option, you can specify criteria such as creation date, last modified data, owner, comments, and user-defined properties.

To perform an advanced search:

1. Right-click on a AquaLogic Data Services Platform application or project node in the Navigation pane, and choose Search in the context-sensitive menu.

   The basic search screen appears. The advanced search tool is available as a link below the basic search interface. For more information about the AquaLogic Data Services Console user interface, see "Introducing the AquaLogic Data Services Platform Console" on page 4-1.

2. Click Advanced Search.

   The advanced search pane appears, as illustrated in Figure 8-9.

Figure 8-9 MetaData Browser Advanced Search

3. Enter the search criteria, as appropriate, and click Search.

   Table 8-4 describes the criteria you can specify using the advanced search facility.

Table 8-4 Advanced Search Criteria

Search Criteria	Description
Data Service Name	The name of the data service.
Data Service Description	The user-supplied description of the data service.
Function Name	The name of the function appearing as part of the data service.

Table 8-4 Advanced Search Criteria (Continued)

Search Criteria	Description
Return Type	The return type of the data service.
Creation Date	The date the data service was created. You can select a relational operator when specifying the date from among the following: • **= (On this date).** Matches the date specified. • **< (Earlier than).** Matches dates earlier than the specified date. • **<= (On this date or earlier).** Matches the specified date or earlier dates. • **>= (On this date or later).** Matches the specified date or later dates. • **> (Later than).** Matches dates later than the specified date.
Last Modified Date	The date the data service was last modified. You can select a relational operator when specifying the date.
Owner	The owner of the data service.
Comment	The comment associated with the data service.
Name	The name of a user-defined property.
Value	The value associated with a user-defined property.

The search results appear in the Contents pane. For more information about the search results, see "Exploring Metadata Search Results" on page 8-16.

4. Click Create Report in the Content pane to generate a report from the search results.

   For more information about generating reports, see "Generating Reports" on page 8-17.

5. Click Return to Explorer to exit the search facility and return to the main interface.

Exploring Metadata Search Results

The MetaData Browser displays basic and advanced search results in the Contents pane. The information displayed is the same for both types of searches. Figure 8-10 illustrates the search results page.

Figure 8-10 Metadata Search Results
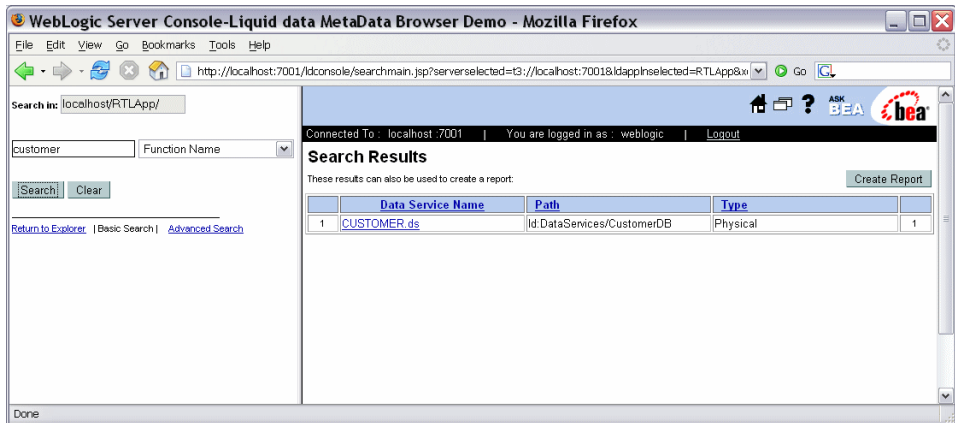
Table 8-5 describes the information displayed as search results.

Table 8-5 Search Results Information

Search Result	Description
Name	The name of the data service.
Path	The path identifying the data service.
Type	Either physical or logical. For more information about data service types, see "Introspecting Data Service Metadata" on page 8-7.

Generating Reports

You can generate an HTML report based on the results of a basic or advanced search. In preparing the report, you specify the information to include such as read functions, return type, relationships, and more.

To generate a report:

1. Right-click on a AquaLogic Data Services Platform application or project node in the Navigation pane, and choose Search in the context-sensitive menu.

The basic search screen appears. The advanced search tool is available as a link below the basic search interface.
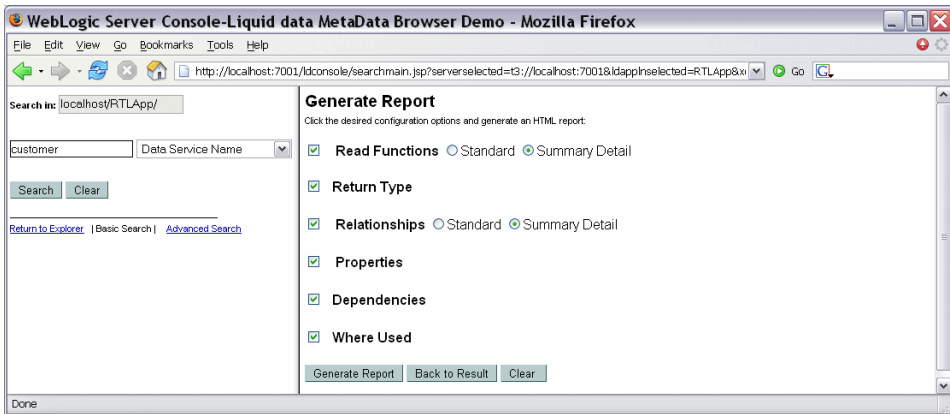
2. Specify the criteria for either a basic or advanced search, and click Search.

    The search results appear in the Contents pane.

3. Click Create Report in the Content pane to generate a report from the search results.

    The Generate Report page appears, as illustrated in Figure 8-11, enabling you to specify the information to include in the generated report.

Figure 8-11 Generating Reports

4. Select the information you want to include in the report, and click Generate Report.

    The generated report appears in the Contents pane. Alternative, you can click Clear to reset the Generate Report page, or click Back to Result to return to the search results.

5. Click Return to Explorer to exit the search facility and return to the main interface.

Table 8-6 describes the options you can select to defined the information included in the generated report.

Table 8-6 Report Information

Information	Description
Read Function	Includes read functions in the report. You can choose to include standard or summary information for each function.
Return Type	Includes the return type of the data service in the report.
Relationships	Includes related data services in the report.
Properties	Includes user-defined properties associated with the data service as part of the report.
Dependencies	Includes the data services on which the resulting data service depends. The data services listed in this table contribute content to the current function's return value.
Where Used	Includes the data services where the resulting data service is used.

Viewing Metadata

Working With Audit and Log Information

This chapter describes the auditing framework, performance profiling, and logging capabilities provided with the AquaLogic Data Services Platform. It contains the following sections:

- Auditing

- Monitoring the Server Log

- Monitoring a WebLogic Domain

- Using Other Monitoring Tools

For information on data service monitoring, see "Monitoring Applications" on page 5-18.

Auditing

The auditing framework system is used to collect auxiliary runtime data using a normal XQuery operation in an AquaLogic Data Services Platform application. This information may be used for security auditing, performance profiling, and other purposes.

Audit Data Structure

The data structure comprises a sequence of audit records containing an unordered collection of audit properties. Each audit record contains properties of a specific type, usually identified using a hierarchal name. Each audit record corresponds to an operation performed by AquaLogic Data Services Platform. For example, access to a relational data source may generate a record of "evaluation/wrappers/relational" type that includes the following audit properties: sql, datasource, returnedRows, evaluationTime, parameters, message, and exception.

Any individual property may be configured to be collected. Each property has an individual intrinsic severity level that can be used to configure an overall threshold of what properties to collect. In certain cases, like when an exception occurs, some properties may be added to the record even if they are not configured to be collected. Typically, this information would be identifiers for a failed data source or update operation.

On the other hand, a property configured for collection need not necessarily be collected. This might be attributed to any one of the following reasons:

- Data might be unavailable due to internal implementation logic.

- A property is collected by an audit based on the need to record internal conditions, for external analysis.

- If an exception is encountered. This will result in an alternate execution path and impact the information being collected.

Elements of the data structure collected can be individually configured to be:

- Submitted to the WebLogic Server auditing framework and processed by an auditing provider.

- Written to an application server or system logging stream.

- Transferred to a client application.

Note: Auditing occurs whenever the engine is invoked and the Auditing option is enabled. Timestamps and other collected data enable you to match auditing information with particular query operations.

Use the AquaLogic Data Services Console to configure application audits such as setting the global audit severity level and overriding audit settings for particular properties of interest.

The Auditing Tab (Figure 9-1) opens a page where you can select properties to be included in the AquaLogic Data Services Platform XQuery engine analysis, update, deployment, and re-configuration event audits. Auditing can be enabled or disabled for individual aspects of a query such as parameters or exceptions. There are also some global auditing options that inherently apply to every aspect of the auditing process.

Note: By default, the audit report generation utility is turned off. Before you begin generating reports for the first time, you need to specify the audit settings described in the subsequent sections. With auditing enabled, performance may be affected, depending on the audit levels and the number of properties being audited.

Figure 9-1 Auditing Options

Settings in this section apply to the entire application

Settings in this section enable you to control
at the individual property level
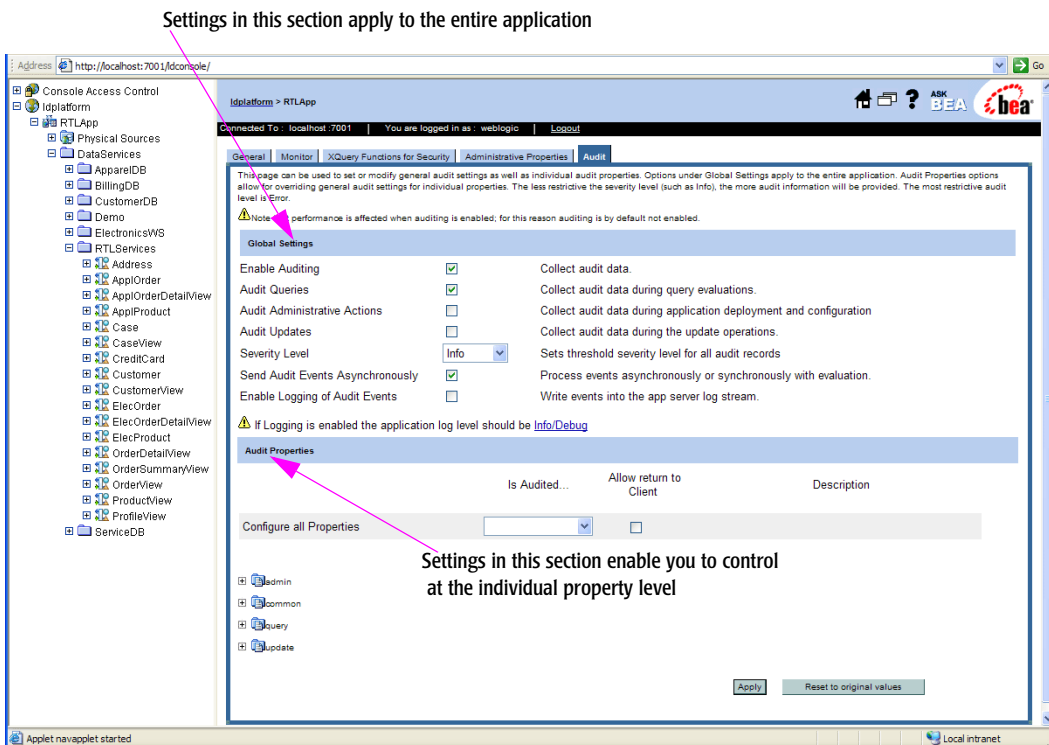
Setting Global Audit Properties

Table 9-1 describes available global auditing options. Click the respective check box in the AquaLogic
Data Services Platform Console to select and implement the desired audit options.

Table 9-1 AquaLogic Data Services Platform Global Auditing Options

Options	Description
Enable Auditing	Determines whether the auditing is activated or not.
	Note: When auditing is enabled, performance can be affected to a degree, depending on the audit level and the number of items being tracked.
Audit Queries	Determines whether the auditing is activated or not, during a query evaluation.

Table 9-1 AquaLogic Data Services Platform Global Auditing Options

Options	Description
Audit Administrative Actions	Determines whether the auditing is activated or not during administrative operations such as application deployment and configuration changes.
Audit Updates	Determines whether auditing is activated or not during update operations.
Severity Level	Determines the level of information to be captured by the auditing process. See Auditing Severity Levels section for more information.
Send Audit events Asynchronously	Determines whether the events are processed synchronously or asynchronously.
Enable Logging of Audit Events	Determines whether the auditing information is to be included in the application server log file.
	Note: If you enable this option (logging), ensure that the Log Level value in the General tab is set to either Info or Debug. Any other value will result in the log file not accepting any information.

Setting Individual Auditing Properties

This section helps you determine which properties you want to audit and to what level. You can propagate generic audit settings through the Configure all Properties row, details of which are listed in Table 9-2. Or, you can set the audit settings at the individual properties level, details of which are shown in Table 9-3.

Table 9-2 Configuring all Audit Properties

Option	Description
Allow return to Client	Click this checkbox to ensure the property-specific audit information is returned to the client API.
Is Audited	This drop-down list has three options, which are generic to all the properties. The options are At Default Level, Always, and Never. By selecting At Default Level, levels of all the individual properties will be set to pre-determined default levels.
	See Table 9-3 for more information on each level.

Note: After you set and apply individual auditing property settings, any changes you make on the individual properties will override the initial settings for that property only.

Table 9-3 lists the audit levels that you can set on each individual property. All levels listed in the table are not applicable to all the properties. Typically, each property has only three levels to choose from.

Table 9-3 Setting Individual Audit Properties

Level	Description
Always	In this setting, the audit information of the property is always collected.
Never	In this setting, the audit information of the property is always ignored.
At Info Level	In this setting, the audit information is collected if the global threshold level is **Information** or lower.
At Warning Level	In this setting, the audit information is collected if the global threshold level is **Warning** or lower.
At Failure Level	In this setting, the audit information is collected if the global threshold level is **Failure** or lower.
At Debug Level	In this setting, the audit information is collected if the global threshold level is **Debug**.

All the individual properties are categorized into four overall types (Admin, Common, Query and Update), depending on the corresponding operation that generates the audit data.

Admin Audit Properties

The audit information in this section pertains to the information exchanged while performing administration tasks such as configuration and application deployment. Only changes to the application made in the AquaLogic Data Services Console are collected during audit.

Table 9-4 Administrator Properties

Property	Description
Configuration	
| notification | Records notification of deployed access control resource. For example:<br><br>```
notification: jmx.attribute.change
property: MAXNUMBEROFQUERYPLANCACHED
value: 101
``` |

Table 9-4 Administrator Properties

Property	Description
plancacheflushed	Notifies when the query plan was flushed. For example: `plancacheflushed: true`
property	Records any instance of the property that was changed in the AquaLogic Data Services Platform Console. For example: `notification: jmx.attribute.change`
value	Records a new value instance, for example: `value: 101`
Deployment	
application	Records the deployed application name. For example: `application: RTLApp`

Common Audit Properties

The audit information in this section pertains to the generic transaction related information. It includes generic information on the event, such as: event type, application name, user id, user access rights, date, and time.

Table 9-5 Common Properties

Property	Description
Application	
name	Records the deployed application name. For example: `name: RTLApp`
eventkind	Records the type of event or operation, it could be a query or an update and so on. For example: `eventkind: evaluation`

Table 9-5 Common Properties

Property	Description

principals	Records the groups to which the user belongs. For example:
	
	``principals:
	    weblogic
	    Administrators
	    IntegrationAdministrators
	    PortalSystemAdministrators```

user	Records the user id, for example: ```user: weblogic```
server	Records the application server's unique id. For example: ```server: cgServer```
exception	Records the exception message, if one occurred. For example: ```exception: ld:DataServices/ApparelDB/CUSTOMER_ORDER_LINE_ITEM.ds, line 77, column 7: {err}FORG0005: expected exactly one item, got 0 items```
transactionid	Records the unique transaction id for the event or operation.
Security **Access**	
resourcetype	Records the type of resource used, such as dataservice, application, submit and so on. For example: ```resourcetype: function```
resource	Records the request for resource identifier. For example: ```resource: <ld type="function"><app>RTLApp</app><ds>ld:DataServices/Cu stomerDB/ADDRESS.ds</ds><res>{ld:DataServices/CustomerD B/ADDRESS}ADDRESS:0</res></ld>```
decision	Records the security access settings for the application, for example: ```decision: PERMIT```

Table 9-5 Common Properties

Property	Description
SQL **Invocation**	
time	Records the date and time of the call to the next () method on the server side of the JDBC driver.
duration	Records the duration or the time required to compute the next block of the result, in milliseconds.
blocksize	Records the size of the returned serialized data block, in bytes.
Time	
duration	Records the time used to complete the audit event, in milliseconds. Calculates the time difference from initiation of the audit to its completion. For example: `duration: 2834`
timestamp	Records the time when the audit event was initiated, for example: `timestamp: Tue Feb 14 09:21:02 IST 2006`

Query Audit Properties

The audit information in this section pertains to all the information collected during query evaluation. The information includes the query itself, its result, the execution time, and details on the data source queried.

Note: When using the streaming APIs, or when using the `RequestConfig.OUTPUT_FILENAME` feature, the results of the query are not audited since they are presumed to be very large. This means the AuditEvent dispatched to the audit provider, as well as the DataServiceAudit returned to the client, will not contain a value for the audit property Query/Service/results.

Table 9-6 Query Properties

Property	Description
Adhoc	
query	Records the query that was executed.
result	Records the results obtained after execution of the query.

Table 9-6 Query Properties

Property	Description
variablenames	Records names of the variables passed to the query.
variables	Records the external parameters or variables passed to the query.
Cache **Data**	
forcedrefresh	Boolean value where TRUE indicates the data is from a current data source or FALSE if it is from a cache.
functionid	Records the name of the function.
remainttl	Indicates the time remaining, in seconds, before the query cache is refreshed.
retrieved	Indicates whether the data was obtained from the query cache or not.
Queryplan	**Note:** Queryplan audit properties are not collected when a function is executed from **Test View** in Workshop. This is because the function cache is not utilized for functions executed in **Test View**.
found	Indicates whether the query plan cache has been located or not.
inserted	Indicates whether the query plan cache has been inserted or not.
Failover	
exception	In the event of a failover, this records the exception that caused it.
function	Records the function name which can be either fn:bea:timeout or fn:bea:fail-over. For example: function: {http://www.bea.com/xquery/xquery-fncts}timeout-with-lbl
label	Records the user-defined label, if any. For example: label: lab
sourcecolumn	Records the source column of the function call. For example: sourcecolumn: 2
sourcefile	Records the source file of the function call. For example: sourcefile: [ad-hoc]

Table 9-6 Query Properties

Property	Description
sourceline	Records the source line of the function call. For example: `sourceline: 4`
timeout	Records the time-out that was exceeded, if applicable. For example: `timeout: 0`
Function	**Note:** Function audit properties are collected only when the individual functions of a data service are selected for auditing. See Auditing Functions for more information.
name	Records the name of the audited function. For example: `name: {ld:DataServices/CustomerDB/CUSTOMER}getCustomer`
parameters	Records the parameters passed through the audited function. For example: `parameters: CUSTOMER1`
result	Records the result after executing the audited function. For example: `result: <ns0:CUSTOMER`
Performance	
compiletime	Records the query compilation time, in milliseconds. For example: `compiletime: 19`
evaltime	Records the query evaluation time, in milliseconds. For example: `evaltime: 90`
Service	
dataservice	Records the name of the data service, for example: `dataservice: ld:DataServices/RTLServices/ApplOrder.ds`
function	Records the function name of the data service, for example: `function: getCustomer`

parameters	Records the parameters passed through the query, for example:
	`parameters:`
	`    1`
	`    foo`

Table 9-6 Query Properties

Property	Description
query	Records the complete text of the executed query on the data service, for example: ``` query: import schema namespace t1 = "urn:retailerType" at "ld:DataServices/RTLServices/schemas/ApplOrder.xsd"; declare namespace ns0="ld:DataServices/RTLServices/ApplOrder"; ```
result	Records the results of the executed query, for example: ``` ORDER_10_0 CUSTOMER0 2001-10-01 GROUND ```
Wrappers	
File	
exception	Records an exception, if any, when a function invoked belongs to a data service created over a File data source. For example: ``` exception: com.bea.ld.wrappers.df.exceptions.DFException: {bea-err}DF0004: [ld:DataServices/Demo/Valuation.csv]: Expected end of line at (row:2, column:3). ```
name	Records the unique function name. For example: ``` name: ld:DataServices/Demo/Valuation.csv ```
time	Records the time taken to query, in milliseconds. For example: ``` time: 20000 ```
Java	
exception	Records an exception, if any, when a function invoked belongs to a data service created over a Java class. For example: ``` exception: {ld:DataServices/Demo/Java/Physical/PRODUCTS}getFirstProd uct:0, line 4, column 5: {bea-err}JFW0401: Class or Method not found exception : {ld:DataServices/Demo/Java/Physical/PRODUCTS}getFirstProd uct ```

Table 9-6 Query Properties

Property	Description

name	Records the name of the service. It is always recorded if an exception property was added. For example:
	
	`name: public static int`
	`Demo.Java.JavaSource4West.echoInt(int)`

parameters	Records the external parameters passed to the service. For example: `parameters: 11`
result	Records the results of the executed query. For example: `result: 11`
time	Records the time taken to execute the query, in milliseconds. For example: `time: 20000`
Procedure	
datasource	Records the name of the data source, for example: `datasource: newDS`

exception	Records an exception, if any, when a function invoked belongs to a data service created over a stored procedure. For example:
	
	`exception:`
	`weblogic.xml.query.exceptions.XQueryDynException:`
	`{err}XP0021: "-ss": can not cast to`
	`{http://www.w3.org/2001/XMLSchema}decimal}`

name	Records the procedure identifier. It is always recorded if an exception property was added. For example: `name: WIRELESS.SIDEEFFECT_REG_PACKAGE.READ2`
parameters	Records the external parameters passed to the data service method. For example: `parameters: s 2.2 22.0 ss`
rows	Records the number of rows returned after execution of the procedure, for example: `rows: 0`
time	Records the time taken to execute the procedure, in milliseconds. For example: `time: 170`

Table 9-6 Query Properties

Property	Description
Relational	

exception	Records the relational database query exception, if any. For example:
	
	`exception:`
	`com.bea.ld.wrappers.rdb.exceptions.RDBWrapperException:…`

parameters	Records the external parameters passed through to the data service method, for example:
	
	`parameters:`
	`   ORDER_10_0`
	`   ORDER_10_1`

rows	Records the number of rows returned from the relational database, for example: `rows: 60`
source	Records the database source name. It is always recorded if an exception property was added. For example: `source: cgDataSource1`

sql	Records the SQL statement used for the query, for example:
	
	`sql:`
	`   SELECT '1' AS c15, t2."LINE_ID" AS c16, t2.`
	`   FROM "RTLAPPLOMS"."CUSTOMER_ORDER_LINE_ITEM" t2`
	`   WHERE ((? = t2."ORDER_ID") OR (? = t2."ORDER_ID")`

time	Records the time spent executing the query, in milliseconds. For example: `time: 5000`
WS	

exception	Records an exception, if any, when a function invoked belongs to a data service created over a web service. For example:
	
	`exception: {bea-err}WSW0101: Unable to create Call :`
	`{ld:DataServices/ElectronicsWS/getCustomerOrderResponse}g`
	`etCustomerOrder`

operation	Records the data service method that is executed. For example: `operation: getCustomerOrder`

Table 9-6 Query Properties

Property	Description

parameters	Records the parameters passed through to the data service method. For example:
	
	`parameters: <ns0:getCustomerOrder`
	`xmlns:ns0="http://www.openuri.org/">`

result	Records the result returned after the query is executed. For example:
	
	`result: <ns:getCustomerOrderResponse`
	`xmlns:ns="http://www.openuri.org/">`
	`<CustOrders`
	`xmlns="http://temp.openuri.org/SampleApp/CustOrder.xsd">`
	`<ORDER>`
	`<ORDER_ID>ORDER_1_0</ORDER_ID>`
	`<CUSTOMER_ID>CUSTOMER1</CUSTOMER_ID>`

time	Records the time spent executing the query, in milliseconds. For example: `time: 50000`
wsdl	Records the web service description. For example: `wsdl: http://localhost:7001/ElWS/cntrls/ElDBTest.jws?WSDL`

Auditing Functions

By default, auditing for all directly invoked functions can be enabled through the /query/service record in the application Audit tab. However, to limit auditing to specific functions, set all properties of the /query/service record to **NEVER** and then enable audit for individual functions by selecting the **Enable Audit** check box as shown below.

If auditing for a function is enabled, all external calls to this function are audited. If the **Enable Audit of Indirect Calls** check box is selected, all calls originating from other data services are also audited.

Note: Enabling audit of indirect calls may disable query optimization for that function, and decrease performance.

Update Audit Properties

The audit information in this section pertains to all the information related to performing an update function. It includes information on the time taken to update the source, when it was started, the unique transaction id and so on.

Table 9-7 Update Properties

Property	Description
Extension	
id	Records the id of the source being updated.
time	Records the time spent, in milliseconds, for the update.
Relational	
exception	Records the update exception, if any.
parameters	Records the parameters passed during the update of the relational database.
rowsModified	Records the number of rows updated in the relational database, on successful completion.
source	Records the data source name. It is always recorded if an exception property was added.
sql	Records the SQL statement used during the update of the relational database.
time	Records the time spend, in milliseconds, in updating the relational database.
Service	
dataservice	Records the data service used for the update.
sdoCount	Records the number of top level SDOs that were submitted for the update.
time	Records the total execution time, in milliseconds, for the update.

Table 9-8 DAquaLogic Data Services Platform Record and Property Auditing Options

Option	Description
Audit Level	Every audit property is set to one of the following audit levels: • **Always.** This setting over-rides the default audit level but not the global enabled/disabled setting. • **Default.** Follows the global default severity level setting. • **Never.** Ignores the global default severity level setting. Note that, in some error and failure cases, auditing of a properties behavior may be reported or ignored, despite its audit level setting.
May Be Returned	Determines if a particular property may be returned to the client application.
Description	Provides a brief description of the property.

Auditing Severity Levels

Severity levels are similar to those provided with WebLogic Server security. For WebLogic Server details, see "Message Severity" section in:

> http://e-docs.bea.com/wls/docs81/ConsoleHelp/logging.html#1037756

Table 9-9 AquaLogic Data Services Platform Audit Severity Levels

Level	Description
Debug	This setting is often referred to as "verbose". Any audit property that can be added to the audit report is collected.
Information	Properties with information or higher conditions are collected for the audit report.
Warning	Properties with warning or higher conditions are collected for the audit report.
Failure	Properties with error or more higher conditions are collected for the audit report.

Retrieving Audit Information

You can record the audit information collected in the following ways.

- **WebLogic Server Security Framework.** Each audit event is by default reported to the WebLogic Server Security Framework.

- **AquaLogic Data Services Platform Client API.** You can create a AquaLogic Data Services Platform client API to record the information collected during audit.

- **AquaLogic Data Services Platform Performance Profiling.** You can use the AquaLogic Data Services Platform audit provider for performance profiling by recording audit events generated by an application.

Values of the audit properties are represented as Java objects of types: String, Integer, java.util.Date, Boolean, or String [ ].

WebLogic Server Security Framework

Each audit event is sent to the WebLogic Server Security Framework as an instance of the `weblogic.security.spi.AuditEvent` interface. Table 9-10 describes each event.

Table 9-10 WebLogic Server Audit Events

`getEventType()`	Returns the event type, in this case `DSPaudit`.
`getFailureException()`	Returns the exception type, if one is encountered.
`getSeverity()`	Returns the event severity level.
`toString()`	Returns the audit event details in an XML formatted representation.

Depending on the configuration, each event can be sent to the WebLogic Server audit API asynchronously and buffered by the AquaLogic Data Services Platform application.

The weblogic.security.spi.AuditEvent interface is implemented in the ld.server.audit.DSPAuditEvent interface, which collects all the information in the form of a list, where each entry is an instance of com.bea.dsp.DSPAuditEvent.

DSPAuditEvent adds the interface described in Table 9-11.

Table 9-11 AquaLogic Data Services Platform AuditEvent API

`getAllRecords()`	Returns all records as a list of `com.bea.ld.DSPAuditRecord`.
`getRecords(String recordType)`	Returns all records of a particular type as a list of `com.bea.ld.DSPAuditRecord`.
`getProperty(String propertyId)`	Returns all values for a particular property, across multiple records.
`getApplication()`	Returns the AquaLogic Data Services Platform application identifier.

Table 9-11 AquaLogic Data Services Platform AuditEvent API

`getUser()`	Returns the user name of the application server user.
`getTimeStamp()`	Returns the time when the event was created.
`getEventKind()`	Returns the event type, which can be `EVALUATION_EVENT`, `CONFIGURATION_EVENT` or `UPDATE_EVENT`.
`getVersion()`	Returns the event version, for example `2.1` for the ALDSP 2.1 release.

com.bea.ld.DSPAuditRecord has the interface shown in Table 9-12.

Table 9-12 DSP AuditRecord API

getRecordType()	Returns the type of record, for example common/time/duration.
getAuditProperties()	Returns all properties in the record. Maps from String identifier to Object value.

A sample security services audit provider is included that demonstrates use of this API.

AquaLogic Data Services Platform Client API

You can use the com.bea.ld.DataServiceAudit client side instance as part of the com.bea.dsp.RequestConfig class, to collect the audit information from the client API. This class collects the audit information and returns it when the operation is successful. If the operation fails for any reason, the com.bea.ld.QueryException class can be used to collect the information as part of the exception thrown.

Note: When using Streaming APIs, auditing will not be complete until the returned XMLInputStream has its close( ) method called. This means that the AuditEvent will not be dispatched to the audit provider by the server, and the RequestConfig.getDataServiceAudit() method will return null, until close( ) is called.

Following are the four steps, with code examples, that need to be performed in order to retrieve audit information.

Initializing the RequestConfig Class

You need to initialize the RequestConfig class as shown in the code example below:

```
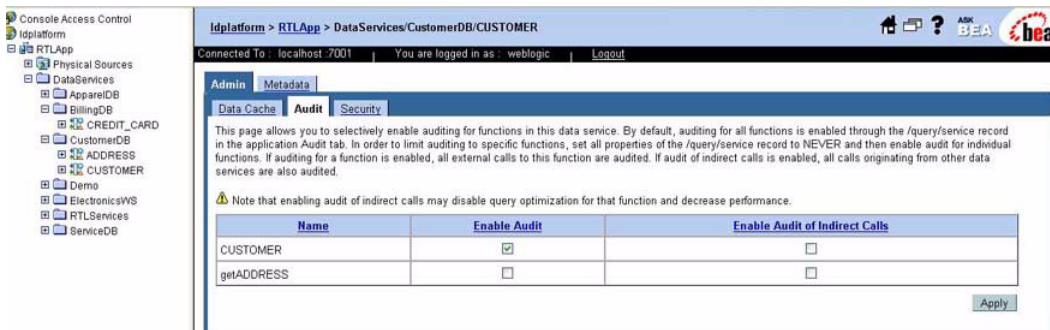RequestConfig requestCfg = new RequestConfig();
requestCfg.enableFeature(RequestConfig.RETURN_DATA_SERVICE_AUDIT);
```

```
requestCfg.enableFeature(RequestConfig.RETURN_AUDIT_PROPERTIES);
requestCfg.setStringArrayAttribute(RequestConfig.RETURN_AUDIT_PROPERTIES,
new String[]
{"query/service/dataservice"});
```

Passing the RequestConfig Object

You need to pass the RequestConfig object to the invoked operation. The code example below uses getCustomer as the invoked operation.

```
CUSTOMERDocument [] custDocRoot1 = (CUSTOMERDocument
[])custDS.invoke("getCustomer", params, requestCfg);
```

Filtering Audit Data

You need to filter the data and ensure there is no unsecured access to it. Only those audit properties that are configured in the Data Services Platform Console to be allowed to return to the client, will be returned to the client application.

Retrieving Data Service Audit

You need to retrieve the data service audit from the RequestConfig object, as shown in the code example below:

```
DataServiceAudit query = requestCfg.retrieveDataServiceAudit();
```

Retrieving Audit Properties

RequestConfig.RETURN_AUDIT_PROPERTIES is an array of string identifiers for audit properties. If you set this request attribute those specified properties will be collected for this particular evaluation even if they are not configured to be collected through the administration console. They will be returned only if it is allowed. If the RETURN_DATA_SERVICE_AUDIT request attribute is not enabled, only those properties will be returned.

RequestConfig.RETURN_DATA_SERVICE_AUDIT configures all collected audit information (that is allowed to be returned to the client application) to be returned.

AquaLogic Data Services Platform Performance Profiling

Performance profiling allows you to store select audit information in a relational database. Relational databases supported by the AquaLogic Data Services Platform audit provider are: Oracle, DB2, PointBase, Sybase, and MS SQL.

Information about audit events are stored as records in a table. A table can be used to record audit events for AquaLogic Data Services Platform applications running on a server, or for applications running on shared servers in a cluster.

You can deploy the AquaLogic Data Services Platform audit provider for performance profiling using the WebLogic Administration Console and configure it using the AquaLogic Data Services Platform Profiler MBean. Configuration parameters you need to set at the time of deployment are described in Table 9-13.

Table 9-13 Configuration Parameters for Performance Profiling

Parameter	Description
Data Source	Name of the JDBC data source.
Table	Name of the table in the JDBC data source that logs query execution information.
Source Table	Name of the table in the JDBC data source that logs source access information.
Summary Table	Name of the table in the JDBC data source that logs aggregated information (summary).
Event Buffer	Size of the internal event buffer. Determines the number of events a buffer stores before the profiler starts processing events.
Collect Execution Aggregate	Stores aggregates (by function) of individual query executions in memory; eventually writes the aggregate to the database.
Aggregate Group Size	Number of events processed by the profiler before the aggregates are written to the database. Default value is 10.
Collect Execution Detail	Writes a row to the database for every query execution, including aggregate of source access within the query. Useful in application development environment.
Collect Source Detail	Writes a row to the database for every source access in a query. Collect Execution Detail needs to be configured for this parameter to take effect.

Creating a Performance Profiler

This section lists the steps needed to create a performance profiler.

1. Create a table to store the following audit properties:

   – `common/time/timestamp`

   – `query/service/function`

   – `query/performance/evaltime`

   – `common/application/user`

   – `common/application/name`

   – `common/application/server`

   In addition to the above mentioned properties, you will also need to store:

   – information about the audit event exception, if any.

   – audit event severity level, which can be of types I (Information), W (Warning), S (Success), E (Error), F (Failure).

2. Modify the CLASSPATH to include a pointer to the JAR file.

3. Start WebLogic Server.

4. In the Audit page, configure the database tables as required.

5. In the Security Providers page of the WebLogic Administration Console, configure a AquaLogic Data Services Platform audit provider. See Table 9-13, "Configuration Parameters for Performance Profiling," on page 9-20 for details.

6. Restart your WebLogic Server.

7. Run the data service application and use the applicable database visualizer to view the results.

Using the Sample Performance Profiler

A AquaLogic Data Services Platform audit provider sample file `profiler.zip`, is available in the AquaLogic Data Services Platform root installation directory. The zip file contains the following files:

- `README.txt` lists steps to use the sample audit provider).

- `dsp_profile.sql` files – Contains table definitions.

- `build.xml` – Defines build configurations.

- `DSPProfilerMBean.xml` – MBean definition file for the AquaLogic Data Services Platform profiling auditor.

- `DSPProfilerImpl.java` – Sample java code that implements the weblogic.security.spi.AuditProvider and weblogic.security.spi.AuditChannel interfaces.

Monitoring the Server Log

Server log files contain information about the time spent to compile and execute a query. The log is in the following location:

`<BeaHome>\user_projects\domains\<domainName>\<serverName>\<server>.log`

For more information about WebLogic Server logs, see "Viewing the WebLogic Server Logs" at:

`http://e-docs.bea.com/wls/docs81/logging/viewing.html`

You can configure the log levels, by application, using the General application configuration page. For more information, see "General Application Settings" on page 5-1. The log levels include:

- **Error.** Runtime exceptions.

- **Notice.** Possible errors that do not affect runtime operation, as well as error level events.

- **Information.** Start/stop events, unsuccessful access attempts, query execute times, and so on, as well as error and notice level events.

Debug logging occurs by default for any server in development mode. Client applications can contribute to the server log through the WebLogic Logger facility. For more information, see "Using WebLogic Logging Services at:

`http://e-docs.bea.com/wls/docs81/logging/use_log.html`

Query strings are echoed in the server log as a debug-level log message when the log level is set to Information in the AquaLogic Data Services Platform Console and the WebLogic Administration Console is set to log debug messages to stdout.

Monitoring a WebLogic Domain

You can use the WebLogic Server Administration Console to monitor the health and performance of the domain in which WebLogic is deployed, including resources such as servers, JDBC connection pools, JCA, HTTP, the JTA subsystem, JNDI, and Enterprise Java Beans (EJB).

The domain log is located in the following directory:

`<BeaHome>\user_projects\domains\<domainName>\<domainName>.log`

For more information, see "Monitoring a WebLogic Server Domain" in *Configuring and Managing WebLogic Server*.

Using Other Monitoring Tools

You can use performance monitoring tools, such as the OptimizeIt and JProbe profilers, to identify AquaLogic Data Services Platform application "hot spots" that result in either high CPU utilization or high contention for shared resources.

For more information, see "Tuning WebLogic Server Applications." For a complete list of performance monitoring resources, see "Related Reading" in *WebLogic Server Performance and Tuning*.