BEALiquid Data for WebLogic®

Administration Guide

Contents

About This Document

1. Overview of Liquid Data Administration

2. Using the WebLogic Server Console

3. Deploying Liquid Data Applications

4. Using the Liquid Data Console

5. Configuring Liquid Data Applications

6. Securing Liquid Data Resources

7. Configuring the Query Results Cache

8. Viewing Metadata

9. Using Logging Information

Index

About This Document

This document explains how to configure, manage, and monitor BEA Liquid Data for WebLogic®.

This document includes the following chapters:

- Chapter 1, "Overview of Liquid Data Administration," introduces BEA Liquid Data for WebLogic administration, and describes how to create new WebLogic domains for BEA Liquid Data for WebLogic or add Liquid Data to an existing WebLogic domain.

- Chapter 2, "Using the WebLogic Server Console," introduces the WebLogic Server Administration Console, and explains how to start and stop the WebLogic Server.

- Chapter 3, "Deploying Liquid Data Applications," describes how to deploy BEA Liquid Data for WebLogic applications to an Administration Server, Managed Server, or to a cluster. The chapter also describes how to deploy Liquid Data applications from development to production mode.

- Chapter 4, "Using the Liquid Data Console," describes how to use the Liquid Data Console to configure BEA Liquid Data for WebLogic.

- Chapter 5, "Configuring Liquid Data Applications," explains how to configure application-level settings for BEA Liquid Data for WebLogic.

- Chapter 6, "Securing Liquid Data Resources," describes how to secure BEA Liquid Data for WebLogic resources, in particular, how to control access to those resources.

- Chapter 7, "Configuring the Query Results Cache," describes how to set up and manage caching for data services in BEA Liquid Data for WebLogic.

- Chapter 8, "Viewing Metadata," explains how to view information on data services, their functions, and their dependencies.

- Chapter 9, "Using Logging Information," describes how to monitor a running WebLogic Server that has at least one Liquid Data project.

What You Need to Know

This document is intended mainly for system administrators who will be configuring and managing the Liquid Data data integration platform. Since several configuration and management tasks are accomplished through the BEA WebLogic Server Administration Console, a working knowledge of standard BEA WebLogic Server system administration is helpful in understanding the concepts in this guide.

e-docs Web Site

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

How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File →Print option on your Web browser.

A PDF version of this document is available on the Liquid Data documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the Liquid Data documentation Home page, click the PDF files button and select the document you want to print.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at http://www.adobe.com/.

The following documents contain information that is useful to refer to after installing BEA Liquid Data:

- Liquid Data *Concepts Guide*
  (`http://e-docs.bea.com/liquiddata/docs85/concepts/index.html`)

- Liquid Data *Installation Guide*
  (`http://e-docs.bea.com/liquiddata/docs85/install/index.html`)

Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for the BEA Liquid Data for WebLogic 8.1 release.

If you have any questions about this version of Liquid Data, or if you have problems installing and running Liquid Data, contact BEA Customer Support through BEA WebSupport at **www.bea.com**. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

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

- Your name, e-mail address, phone number, and fax number

- Your company name and company address

- Your machine type and authorization codes

- The name and version of the product you are using

- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.
`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`
`monospace boldface text`	Identifies significant words in code.
	Example:
	`void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code.
	Example:
	`String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators.
	*Example*s:
	LPT1
	SIGNON
	OR

Convention	Item
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

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

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

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

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

About This Document

Overview of Liquid Data Administration

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

The chapter contains the following sections:

- Administration Tasks in Liquid Data

- Understanding WebLogic Domains and Administration

- License Key Updates

Administration Tasks in Liquid Data

BEA Liquid Data for WebLogic is integration software that unifies data programming through the use of data services. Since it is deployed to a WebLogic Server, you can administer Liquid Data 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 for Liquid Data. Generally these arise from Liquid Data's role as data integration software and include managing applications that use Liquid Data 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 Liquid Data-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

Liquid Data 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.

Liquid Data 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 Liquid Data Resources."

Caching Query Results

Liquid Data can cache query results for data service functions to enhance overall Liquid Data 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 Liquid Data Console.

For more information, see Chapter 5, "Configuring Liquid Data 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. Liquid Data 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 Liquid Data by revealing what data is available and how to use it.

- It helps administrators maintain Liquid Data 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 Liquid Data application to your domain.

Liquid Data supplements the WebLogic Administration Console with the Liquid Data Console. The Liquid Data Console gives you access to configuration settings specific for Liquid Data, such as caching and data resource security controls as well as metadata information.

Understanding the Relationship of Liquid Data to WebLogic Domains

Liquid Data is an application and a set of associated resources that are deployed in a WebLogic domain. Starting, stopping, and managing Liquid Data is accomplished by starting the WebLogic Server in the domain in which Liquid Data is deployed, and using the Administration Console for that server to configure and manage Liquid Data resources for that domain.

Creating a New Domain

Liquid Data applications work with WebLogic domains that have been provisioned for Liquid Data. You can use the BEA WebLogic Configuration Wizard to create these domains.

To create a new domain provisioned with Liquid Data:

1. On Windows systems, choose Programs →BEA WebLogic Platform 8.1 →Configuration Wizard.

2. In the wizard, choose Liquid Data 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.

Adding Liquid Data to an Existing Domain

In cases when you have WebLogic Server domain in which you want to use Liquid Data, the next step is to provision the domain for Liquid Data. Once a domain is provisioned with Liquid Data, you can deploy applications that contain Liquid Data projects. For more information see Chapter 3, "Deploying Liquid Data Applications."

Understanding Console Users

The Liquid Data Console is targeted for two types of users:

- Client developers
- Liquid Data 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 Liquid Data Resources."

License Key Updates

Liquid Data requires a valid product license to run. The Liquid Data license is included as a component in the WebLogic Server license file, `license.bea`. If you need to apply or update 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 Liquid Data

- Starting the WebLogic Server

- Launching the Administration Console

- Exploring the Administration Console

- Stopping the WebLogic Server

Using the Administration Console to Manage Liquid Data

When deployed on a Liquid Data provisioned domain, Liquid Data-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 Liquid Data application using the WebLogic Administration Console.

Before you can configure or manage a Liquid Data 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 Liquid Data 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 Liquid Data. 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 Liquid Data 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
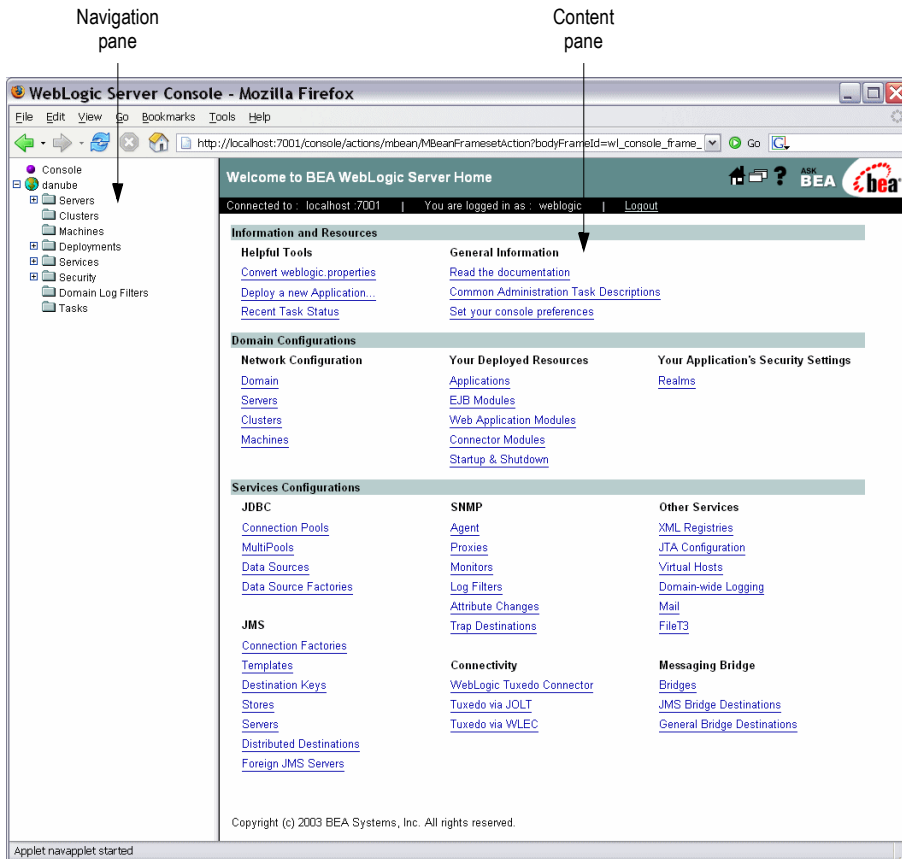
When you start WebLogic Administration Console, the general administration page is shown in the Content pane, as illustrated in Figure 2-1. You can use the topic links on the home page initially to navigate to top level resource nodes, or use the Navigation pane which contains a hierarchical tree—a domain 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 Liquid Data, you will need to access and use console pages for standard WebLogic Server resources as well as console pages specific to Liquid Data 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_console) in the WebLogic Server documentation.

Finding the Liquid Data Application Node

Liquid Data 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 Liquid Data Resources in the WebLogic Administration Console
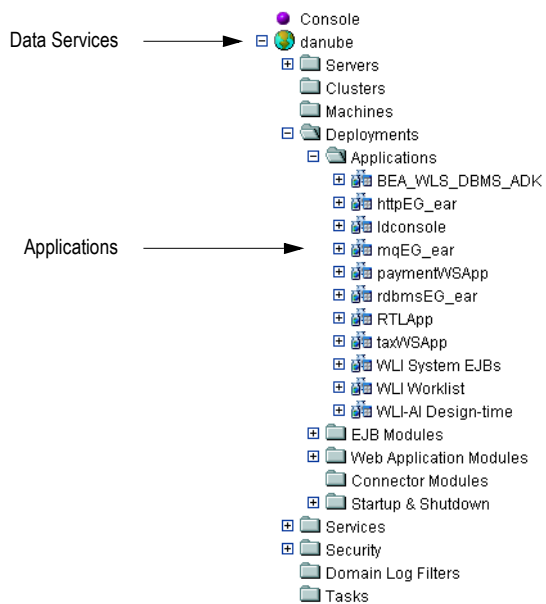
Stopping the WebLogic Server

You can stop a WebLogic Server running a Liquid Data application from the WLS 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 Liquid Data 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 Liquid Data Applications

This chapter describes how to deploy BEA Liquid Data for WebLogic applications to an Administration Server, Managed Server, or to a cluster. The chapter also describes how to deploy Liquid Data applications from development to production mode.

The chapter contains the following sections:

- Introduction

- Deploying Liquid Data Applications to an Administration Server

- Deploying Liquid Data Applications to a Managed Server

- Deploying Liquid Data Applications to a Cluster

- Deploying Liquid Data Applications from Development to Production Mode

- Checking the Liquid Data Version

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 Liquid Data 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

Liquid Data Components to Deploy

The WebLogic deployment wizard automatically transfers the required items to the target server. These include the Liquid Data 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 Liquid Data project.

Table 3-1 Contents of Liquid Data Application EAR file

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

Deploying Liquid Data Applications to an Administration Server

An Administration Server is the central configuration repository for the set of WebLogic Servers in a domain. Once the Liquid Data 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
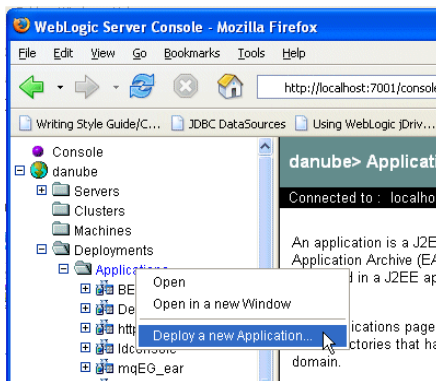
3. Using the Location links, navigate to the directory where the EAR file, JAR, or EJB is located.

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

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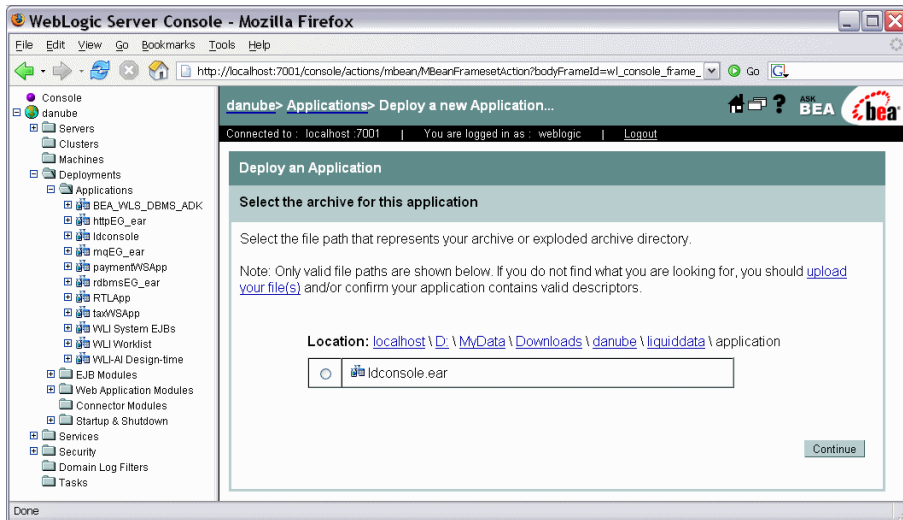
5. After reviewing the deployment information, click Deploy.

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 Liquid Data 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 Liquid Data application in the Navigation pane.

3. Click the Deploy tab in the Contents pane.

   The Administration Console displays the Liquid Data Deploy tab.

Figure 3-3 Deploy Tab for a Liquid Data 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 Liquid Data 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 Liquid Data 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
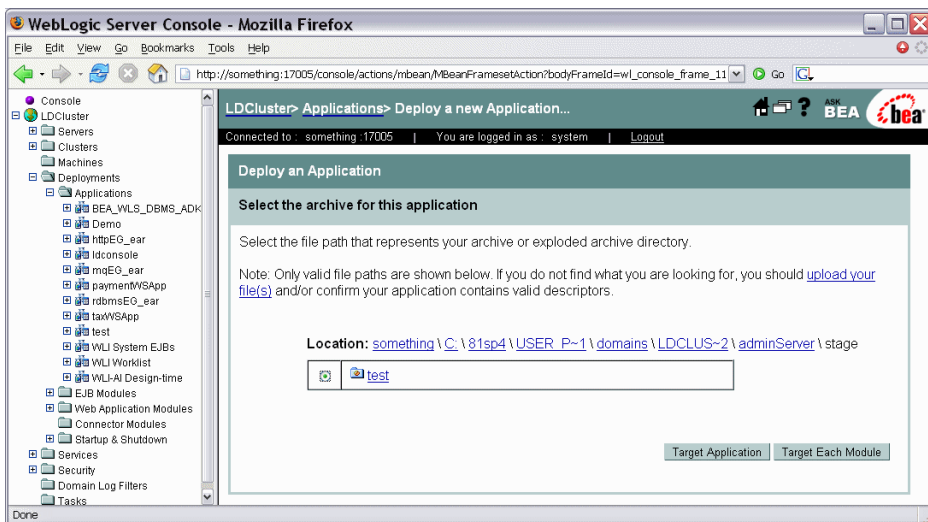
4. Click the radio button for the application you want to deploy, and click Target Application.

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

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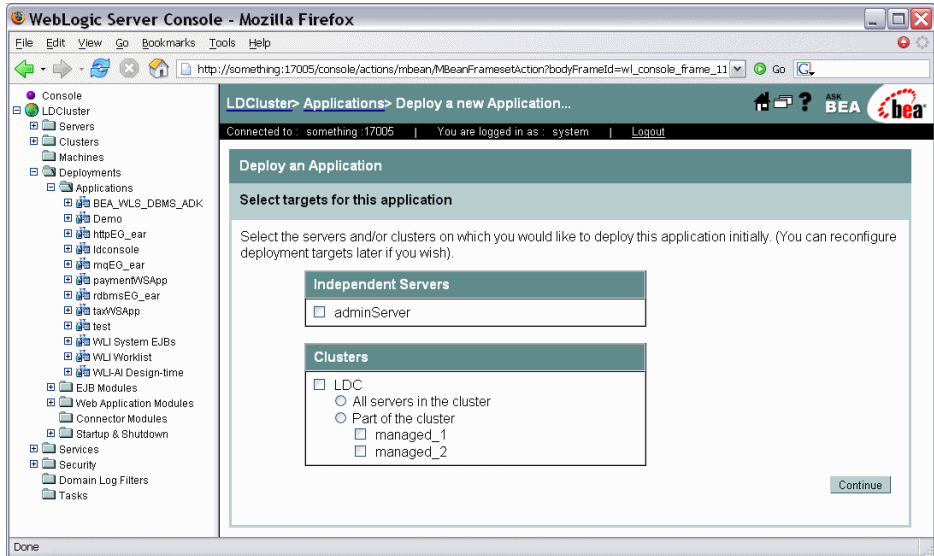
5. Click the radio button corresponding to the cluster or part of cluster to which you want to deploy the Liquid Data application, and click Continue.

6. After reviewing the deployment information, click Deploy.

Deploying Liquid Data Applications from Development to Production Mode

Liquid Data 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 Liquid Data applications from development to production mode:

- Migrating applications using configuration templates
- Manually migrating applications

Migrating Liquid Data Applications Using Configuration Templates

You can migrate Liquid Data 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 Liquid Data 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 Liquid Data 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 Liquid Data 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 Liquid Data applications from development to production mode, if required.

To manually deploy an application from development to production mode:

1. Create a Liquid Data 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 Liquid Data 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 Liquid Data application and Liquid Data 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 Liquid Data security policies from the development domain to the production domain.

   Export the Liquid Data security policies in the development domain and import them into the production domain. For more information about exporting Liquid Data security policies, see "Exporting Access Control Resources" on page 6-20.

7. Migrate the `ldcontrol.properties` file from development to the production domain.

   Each domain that runs Liquid Data Control applications has a single `ldcontrol.properties` file, which stores the connection information for all Liquid Data Control applications running in the domain. The `ldcontrol.properties` file is located at the root directory of your domain where the Liquid Data Control application EAR file is deployed. For more information about migrating the `ldcontrol.properties` file, see the BEA WebLogic Workshop Online Help at:

   http://e-docs.bea.com/workshop/docs81/doc/en/core/index.html

8. Update the WebLogic Workshop configuration settings by adding `-Djavax.xml.rpc.ServiceFactory="weblogic.webservice.core.rpc. ServiceFactoryImpl"` to the `WL_HOME\workshop\workshop.cfg` file.

9. Start the WebLogic Server and verify that the Liquid Data application is working properly.

Checking the Liquid Data Version

You can determine which version of Liquid Data you are through the WebLogic Administration Console.

To determine the Liquid Data version number:

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.

Deploying Liquid Data Applications

Using the Liquid Data Console

This chapter describes how to use the Liquid Data Console to configure BEA Liquid Data for WebLogic.

The chapter contains the following sections:

- Introducing the Liquid Data Console

- Launching the Liquid Data Console

- Navigating the Liquid Data Console

- Controlling Access to the Liquid Data Console

Introducing the Liquid Data Console

The Liquid Data Console is a web-based administration interface specifically designed for managing and using Liquid Data applications. You can use the console to set security and caching policies for data services, and configure Liquid Data runtime settings such as thread usage and logging levels.

The Liquid Data Console also provides access to the Liquid Data Metadata Browser. The Metadata Browser provides information useful to both Liquid Data 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 Liquid Data Console.

Figure 4-1 Liquid Data Console
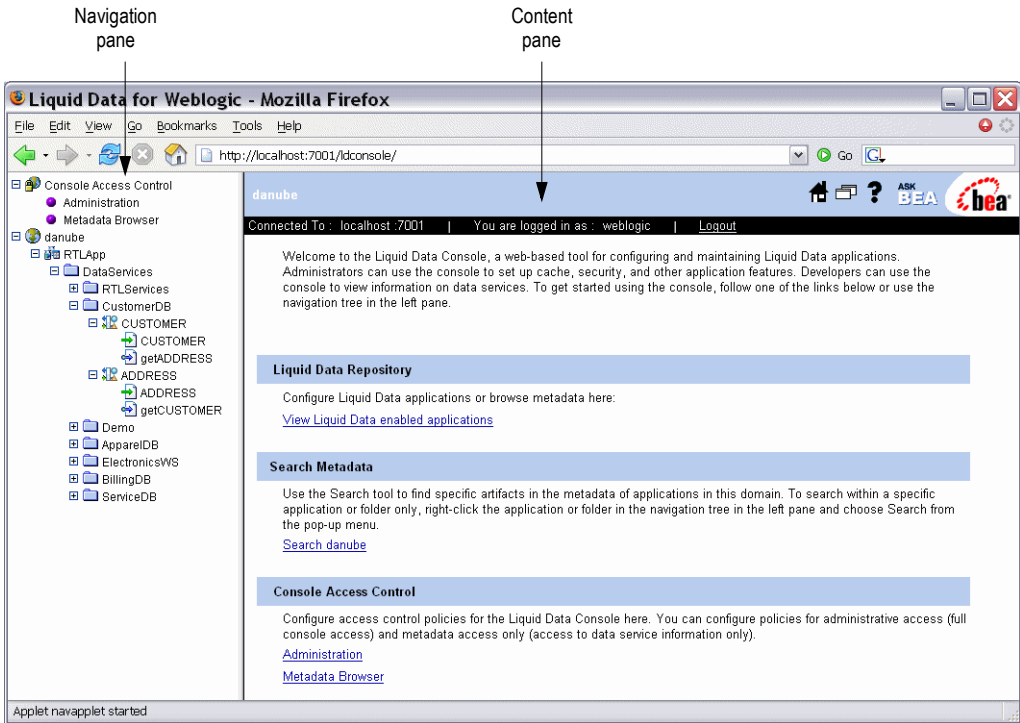
Launching the Liquid Data Console

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

To launch the Liquid Data Console:

1. Start the WebLogic Server in the WebLogic domain in which Liquid Data 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 Liquid Data 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.

Navigating the Liquid Data Console

You can navigate to the various pages in the Liquid Data Console using the tree in the Navigation pane. The administration pages are organized by application and data service, as shown in Figure 4-2.

Figure 4-2 Console Tree Panel

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

1 **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 6-6.

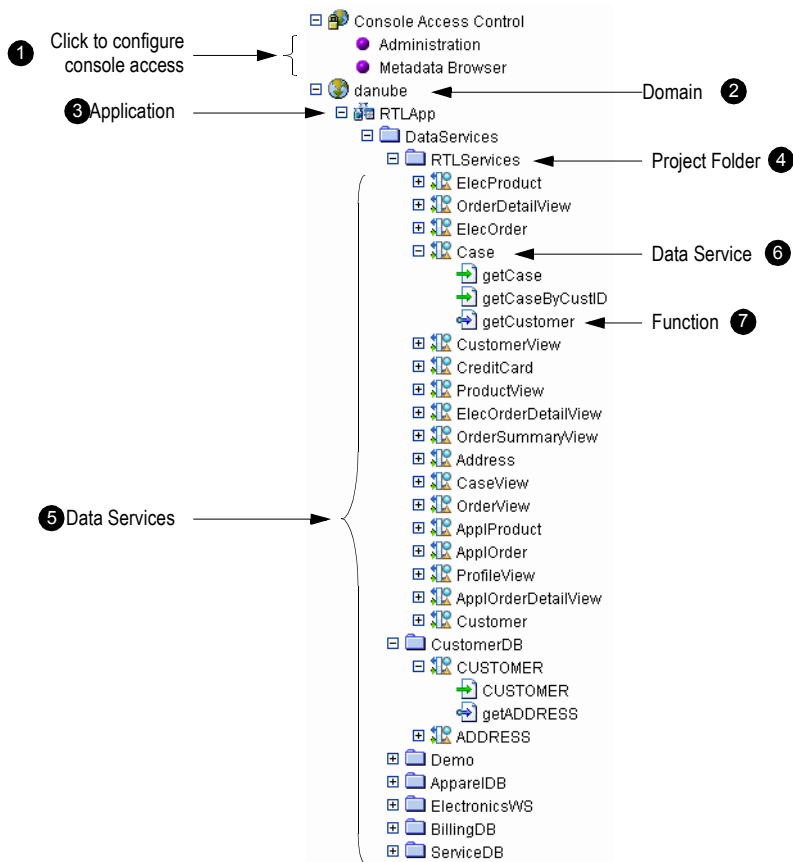**2** **Domains.** Expand to display the Liquid Data applications in the domain. Alternatively, you can click a domain name to display the list of applications in the Content pane. Right-click and choose Search in the context-sensitive menu to search metadata in the domain (see "Searching Metadata" on page 8-8).

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 5-2. Right-click and choose Search in the context-sensitive menu to search metadata in the application (see "Searching Metadata" on page 8-8).

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 8-8).

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 Application 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 8-8).

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 4-7.

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 4-9. 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 6-5).

Displaying Application Data Services

You can display the data services available to an application, along with information about each data service.

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 Data Services project folder in the Navigation pane.

    The list of data services contained in the project folder appears in the Content pane, as illustrated in Figure 4-3.

Figure 4-3 Data Services Available to a Project
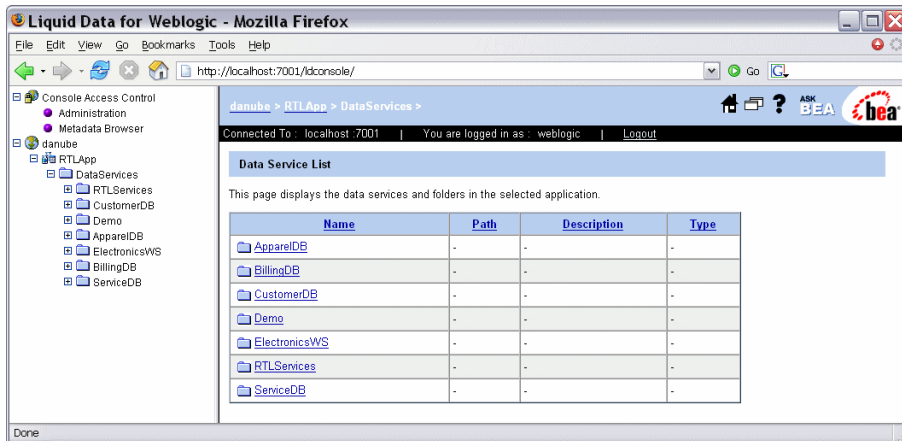
Table 4-1 describes the information presented for each data service.

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 4-4. For more information about administering data service functions, see "Setting Up Caching" on page 7-4, "Securing Data Service Functions" on page 6-11, and "Understanding Data Service Metadata" on page 8-5.

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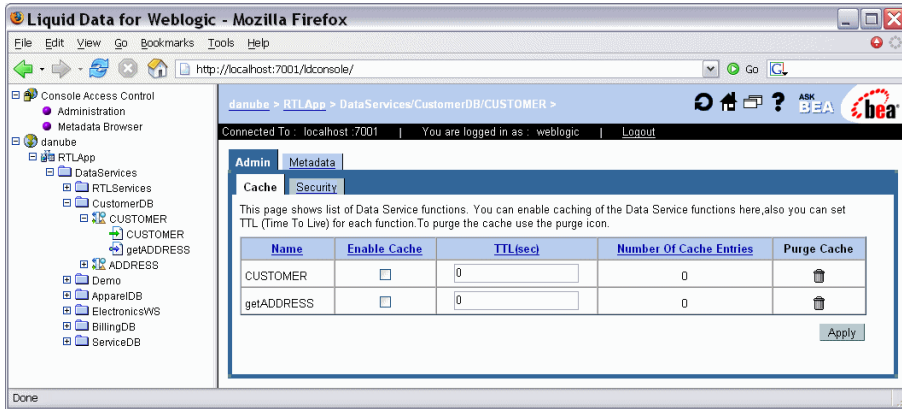
There are two types of functions identified in the Navigation tree, as described in Table 4-2.

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 4-5. For more information, see "Displaying Function Metadata" on page 8-6.

Figure 4-5 Function Details

Controlling Access to the Liquid Data Console

The Liquid Data 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 access to resources in the console, see Chapter 6, "Securing Liquid Data Resources."

Configuring Liquid Data Applications

This chapter describes how to configure application-level settings for BEA Liquid Data for WebLogic. The chapter contains the following sections:

- General Application Settings

- Guidelines for Setting the Thread Count

- Monitoring Applications

- Terminating Query Execution

- Using Administrative Properties

- Setting the Transaction Isolation Level

General Application Settings

You can view and configure run-time application-wide settings, including access control, cache settings, server resources (including thread usage), and log levels.

To specify general application settings:

1. Click the name of the Liquid Data application node in the Navigation pane of the Liquid Data 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
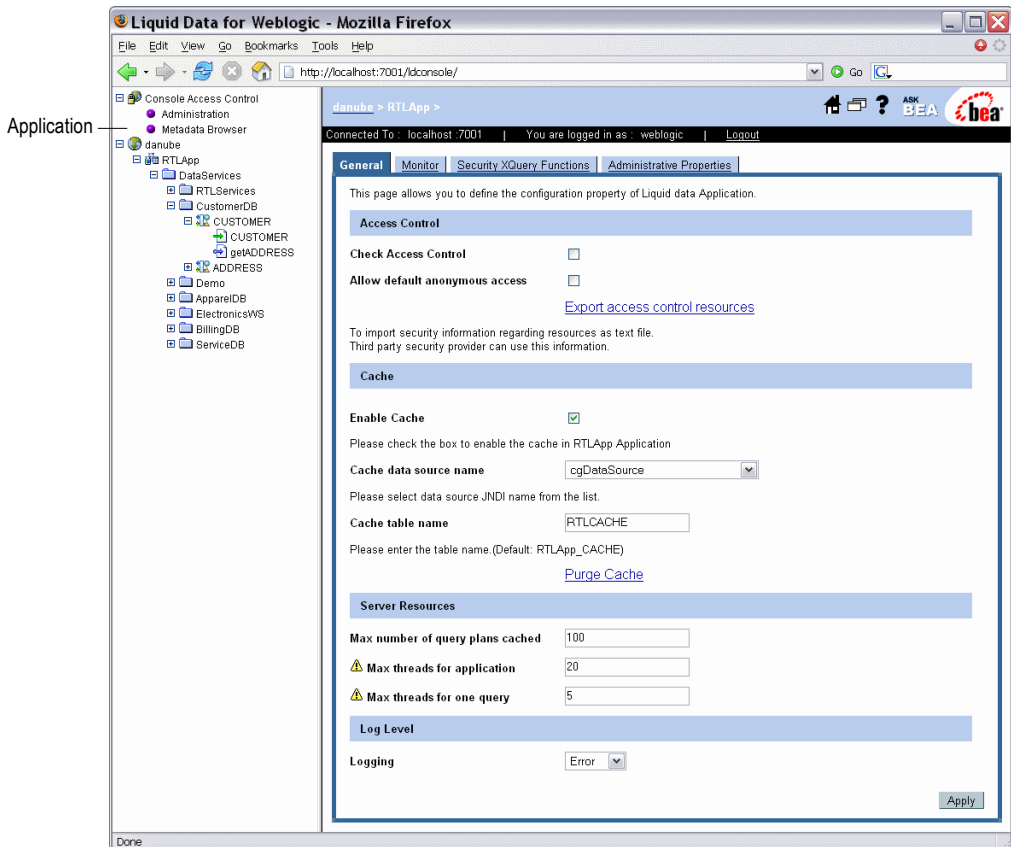
2. Specify settings, as appropriate.

3. Click Apply to save the settings.

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

Table 5-1 Liquid Data 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 Liquid Data 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 Liquid Data 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 Liquid Data 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 the 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 Liquid Data query engine for processing requests on application view and web service data sources.
		For more information on configuring thread counts, see "Guidelines for Setting the Thread Count."

Table 5-1 Liquid Data 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

Guidelines for Setting the Thread Count

The optimal thread count settings you configure depends on the physical resources of the machine on which you deploy Liquid Data, 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.

Liquid Data 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 Liquid Data 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 Liquid Data application node in the Navigation pane of the Liquid Data 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 5-2.

Figure 5-2 Monitor Tab
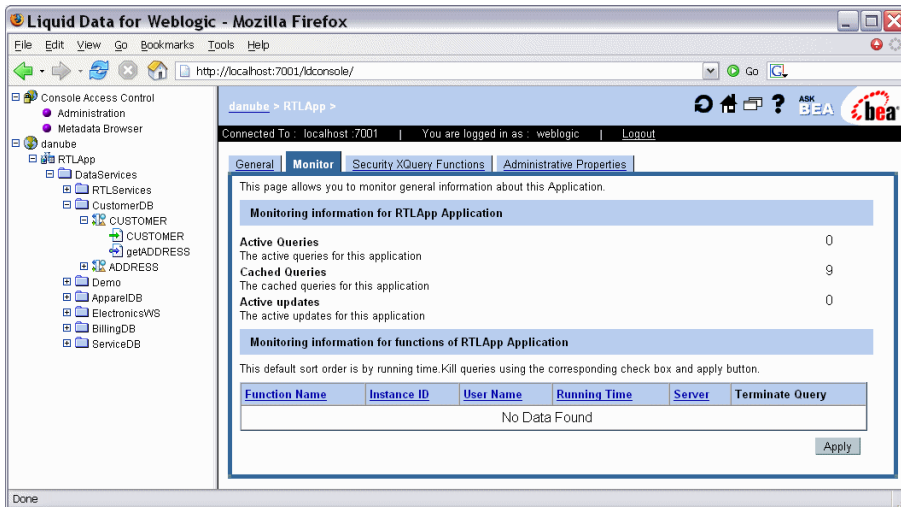
Table 5-2 describes the information displayed in the Monitor tab.

Table 5-2 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 Liquid Data 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.
	Number of Cached Entries	The number of instances of the query in cache. A cache entry is made for each distinct invocation of the named function with different input parameters.

Terminating Query Execution

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 Liquid Data 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 Liquid Data 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 5-3.
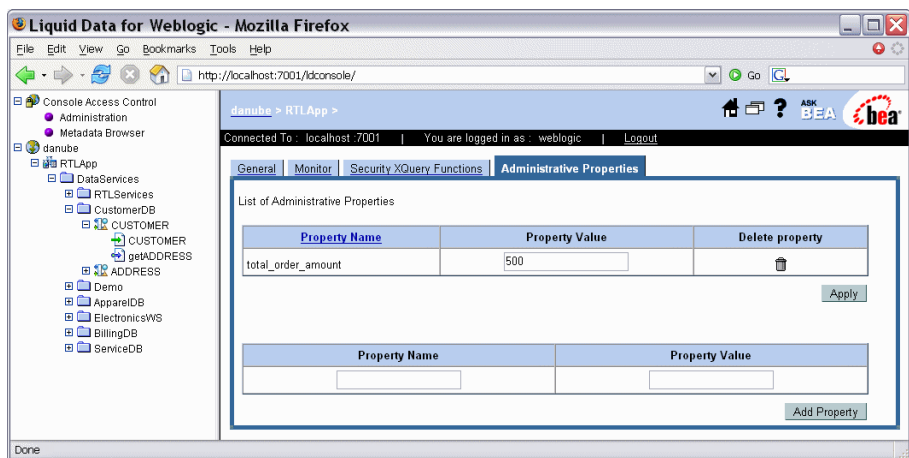
Figure 5-3 Administrative Properties Tab

Table 5-3 describes the information displayed in the Administrative Properties tab:

Table 5-3 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, Liquid Data may not be able to read data from a database table because another application has locked the table, causing queries issued by Liquid Data 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 Liquid Data application runs, and click the name of the connection pool you want to configure.

   The Connections tab appears, as illustrated in Figure 5-4.

Figure 5-4 Connections Tab
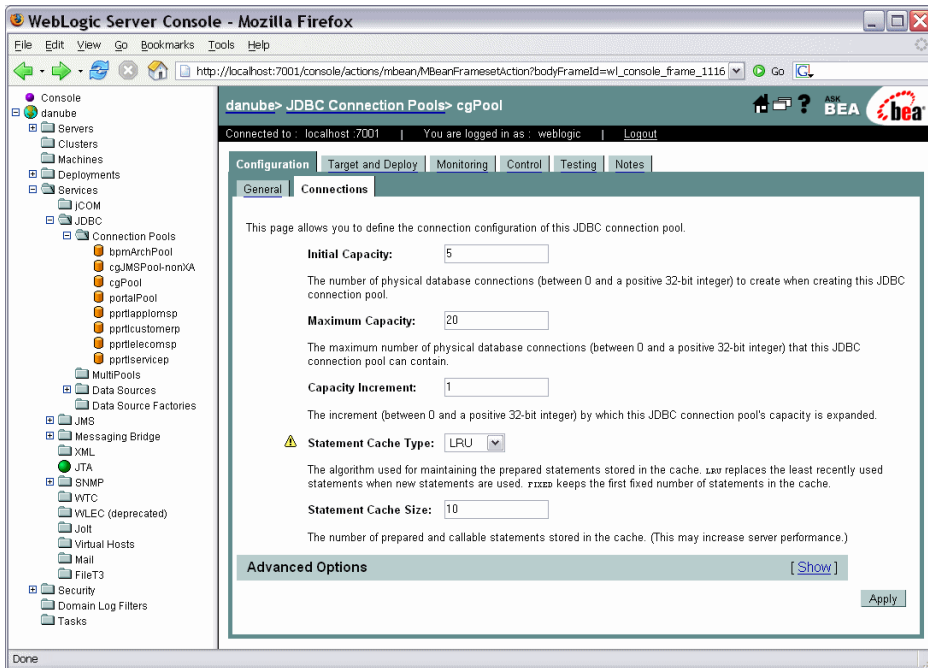
3. Click Show in the Advanced Options section of the page.

   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 Liquid Data Resources

This chapter describes how to secure BEA Liquid Data for WebLogic resources, in particular, how to control access to those resources.

The chapter contains the following sections:

- Introducing Liquid Data Security

- What is a Securable Resource?

- Understanding Security Policies

- Securing Liquid Data Resources

- Securing Access to the Liquid Data Console

- Exporting Access Control Resources

Introducing Liquid Data Security

Liquid Data uses the security features of the underlying WebLogic platform to ensure the security of the information it provides. Specifically, Liquid Data 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.

Liquid Data exposes its deployed artifacts as resources that can be secured through WebLogic role-based security policy control. With Liquid Data, 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 Liquid Data deployment or just to a credit card number element in an order.

When a request comes to Liquid Data for a secured resource, Liquid Data 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
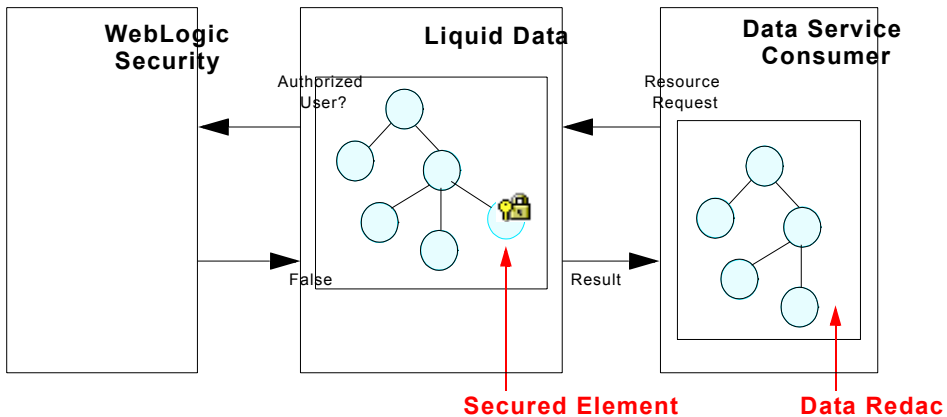
Note: By default, WebLogic security uses the ATZ authorization provider module. ATZ keeps policies in an LDAP system. Other authenticators can use any external resource necessary to implement the policy evaluation.

Setting up Liquid Data security in the Liquid Data 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 Liquid Data 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 Liquid Data 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 Liquid Data 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 6-20 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 Liquid Data artifacts is as follows:

- Application

- Data Service

- Function

- Data element

Figure 6-2 illustrates the securable resources in a Liquid Data application.

Figure 6-2 Securable Resources
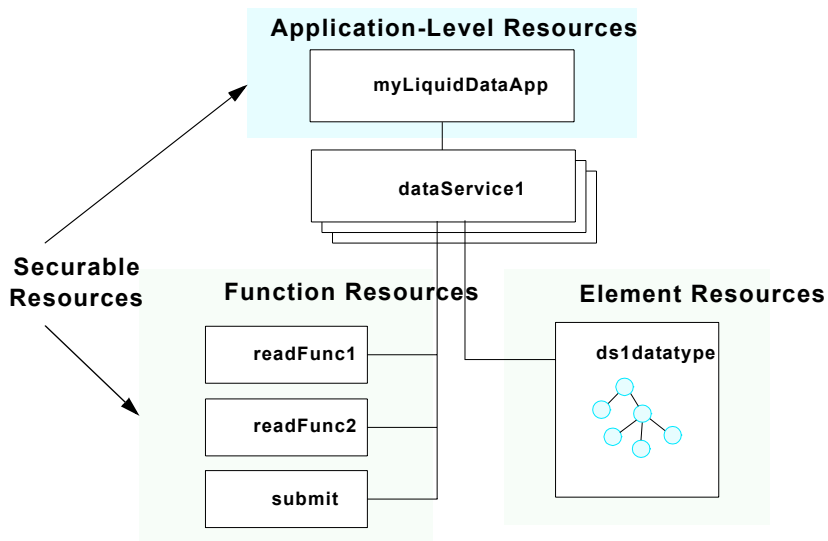
Enabling anonymous access is a special type of application-level setting. It enables you to either disable access to the application by default (unless a more specific policy permits it) or enable access (unless a more specific policy blocks it). If enabled, all users can access resources by default, even unauthenticated users. The anonymous access option works only with the WebLogic Authorization provider.

Note: Note that the Liquid Data 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 Liquid Data 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 Liquid Data Console incorporates the WebLogic Policy Editor interface for creating Liquid Data security policies. You can use the policy editor for both Liquid Data application resources, such as data elements and functions, and administrative resources (Liquid Data Console).

To create a policy using the WebLogic Policy Editor:

1. Choose a condition from the Policy Condition list box.

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

Figure 6-3 Policy Editor
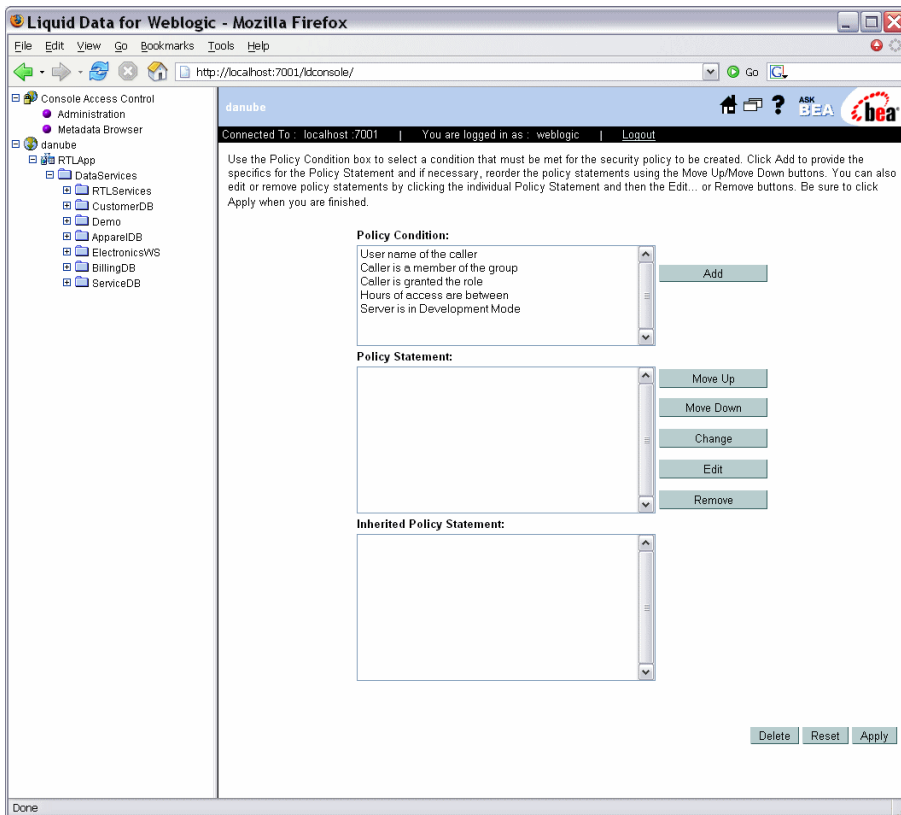
2. Click Add.

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
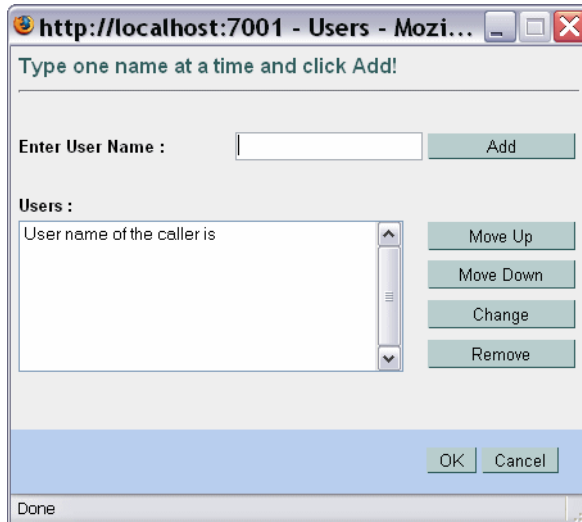
3. If needed, repeat steps 1 and 2 to add expressions based on different policy conditions.

4. 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.

5. 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

User Role Considerations

In a WebLogic domain, a user group is a logical collection of users. A role is similar to a group, except that while membership in a group is statically defined, membership in a security role is dynamically allocated based on factors such as user name, group membership, or time of day.

In WebLogic there are two types of roles, global and scoped. Scoped roles prevent naming conflicts with roles configured for securing other WebLogic resources. Liquid Data, however, only supports global roles. Therefore, when creating roles for use with Liquid Data security, you may want to name the roles with a distinguishing prefix, such as "ld_" (for example, ld_admin).

For more information on WebLogic security roles, see the following WebLogic documentation:
http://e-docs.bea.com/wls/docs81/secwlres/secroles.html

Securing Liquid Data Resources

You can secure Liquid Data 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 6-9

- "Securing Data Service Functions" on page 6-11

- "Securing Data Elements" on page 6-12

- "Using Data-Driven Security Policies" on page 6-14

Securing Applications

Enabling access control checking activates the security policies in the application. Once access control checking is turned on, access to any resource in the application is blocked unless a more specific policy (one at the data service, function, or element level) permits it for a user.

You can invert this rule by enabling default anonymous access. If this option is selected, access to application resources is enabled by default, unless a more specific policy blocks access.

To set the application access policy:

1. Select the application node in the Navigation pane.

Figure 6-5 Securing an Application
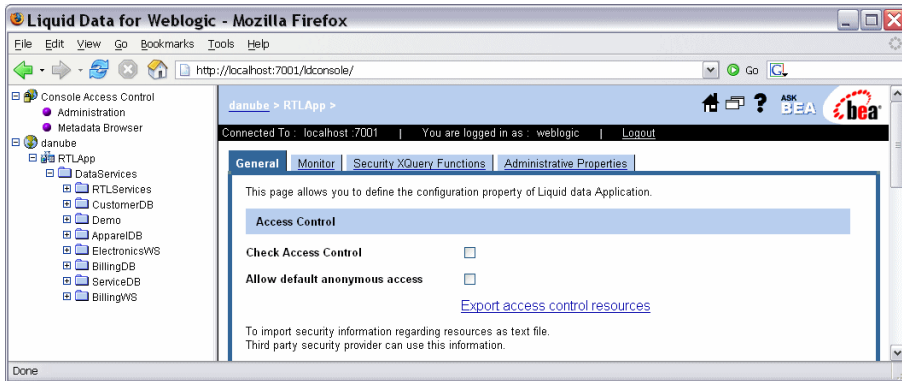
2. Click the Check Access Control box in the Access Control area of the General tab.

   When this option is selected, access to all resources is blocked by default and security policies are applied. You can either keep this restrictive policy as the default and selectively configure security policies on individual resources, or choose to permit access by default.

3. If you want to permit user access to resources by default, enable the Allow default anonymous access option.

   This permits access to all resources, even to unauthenticated users, unless a more specific policy blocks it.

4. Click Apply.

You can now set function or element level security policies on Liquid Data resources.

Securing Data Service Functions

A data service typically has several functions, including one or more read functions, navigation functions, and a submit function. A submit function allows users to save data changes to back-end data sources. Function-level security policies enable you to control who can use data service functions and when. This enables you to set stricter controls on the ability to change data, for example, compared to the ability to read data.

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 an another data service, the policy is not evaluated against the caller.

Note: 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.

To create a function 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, as illustrated in Figure 6-6.

Figure 6-6 Security Policy Function List

3. Click the Action icon ( ).

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

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

Note: You must enable access control for the application to have function-level security policies applied to users. For more information, see "Securing Applications" on page 6-9.

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.

Warning: Any element for which you want to create a security policy must be defined as optional or repeating in the schema definition of the data service type. Mandatory elements in the schema definition are not securable to ensure conformance with the XSD.

An element-level security policy applies across all functions of the data service. However, note that it applies only in the context of that data service. 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
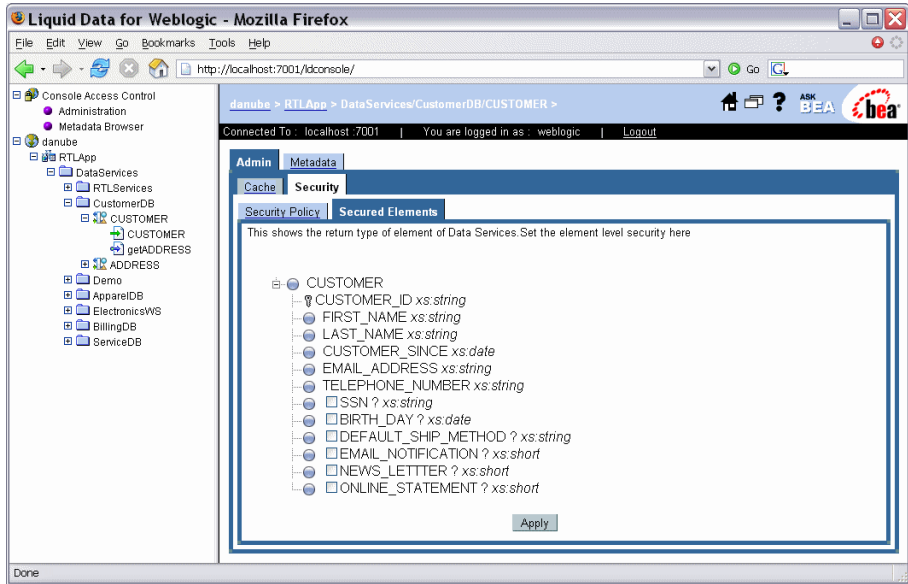
4. Select the check box for the data elements you want to secure.

Selecting a parent node includes all children of the parent.

5. Click Apply.

6. Click the Security Policy tab.

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 6-6 or "Using Data-Driven Security Policies" on page 6-14.

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 6-9.

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 6-15.)

3. Apply a security XQuery function to a data element. (For more information, see "Applying a Security XQuery Function" on page 6-17.)

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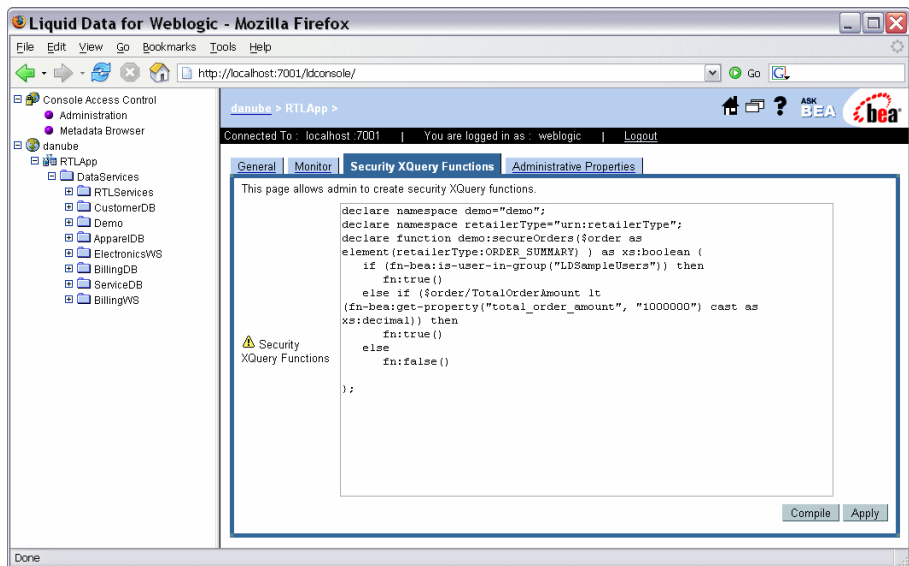
3. Add the XQuery function body in the text area of the tab.

   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 *Liquid Data for WebLogic 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="test:demo";
declare namespace

itemns="http://temp.openuri.org/DataServices/schemas/CustomerProf.xsd";

declare function demo:secureCustomer($ssn as xs:string) as xs:boolean {
if (fn-bea:is-access-allowed("ssn",
"ld:DataServices/CustomerProfile.ds"))
        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 6-17.

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.

Liquid Data 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.
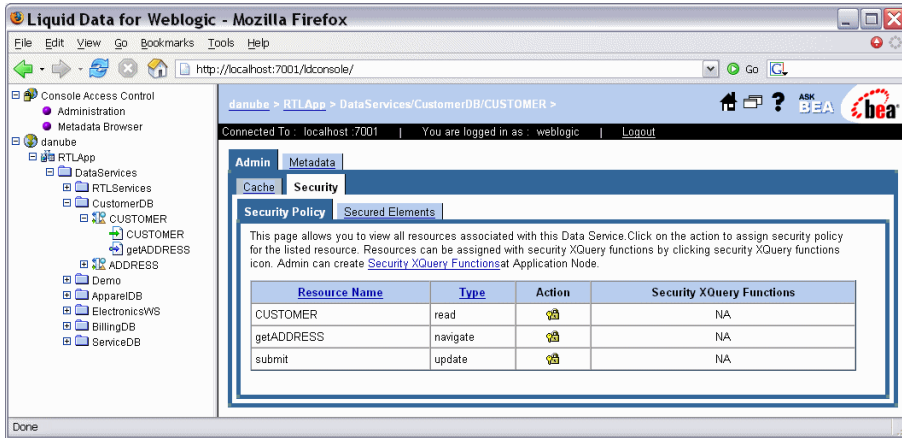
Applying a Security XQuery Function

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 6-15, 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.

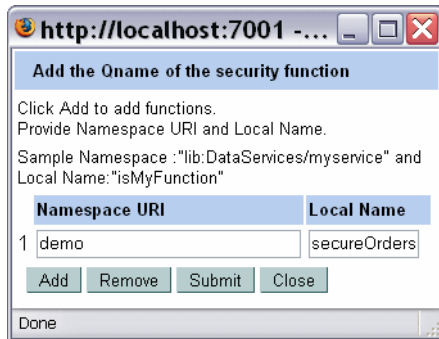**Figure 6-9 Applying Security XQuery Functions**

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 →Applications →*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 Liquid Data Console

Similar to the WebLogic Administration Console, the Liquid Data 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 Liquid Data 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 Liquid Data administrators and developers who want to use Liquid Data 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 Liquid Data 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 Liquid Data Console to configure your policies. In particular, resource identifiers already exist for Liquid Data 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 Liquid Data 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 Liquid Data Console, as shown in Figure 6-12.

Figure 6-12 Element Resources
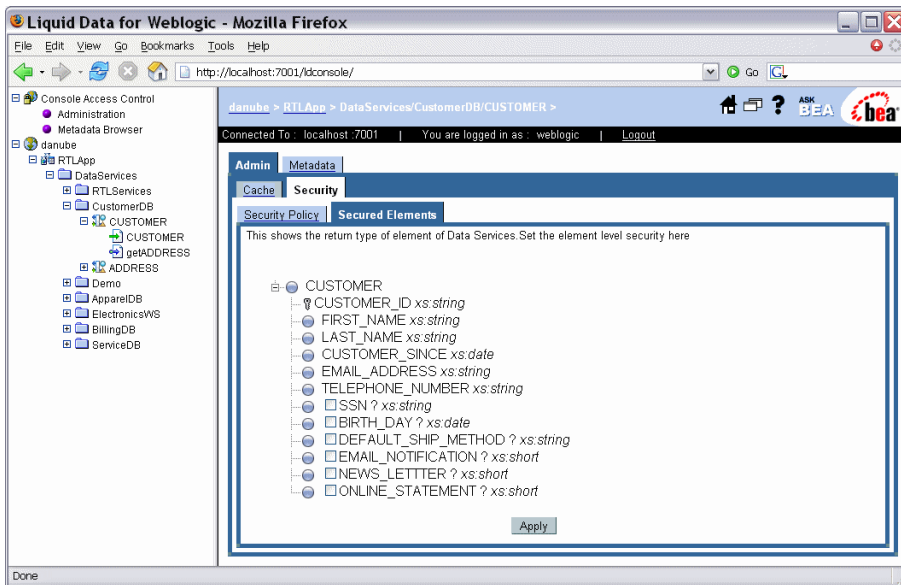
Configuring the Query Results Cache

This chapter describes how to set up and manage caching for data services in BEA Liquid Data for WebLogic.

The chapter contains the following sections:

- Understanding Results Caching
- Setting Up Caching
- Purging Cache Entries

Note: Caching is only available for data service function results. It does not apply to ad-hoc queries or to security XQuery functions, which are never cached.

Understanding Results Caching

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

When results caching is enabled, the first time a data service function is run, Liquid Data saves the results in the *query results cache*. The next time the function is run with the same parameters, Liquid Data checks the cache configuration and, if the results have not expired, retrieves the results from the cache rather than from the external source. Note that 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.

Caching is disabled by default. Once it is enabled, you can configure the cache for individual data service functions as needed. 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 time-to-live (TTL) setting can be set individually, that is, by 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.

For example, if the data changes frequently and real-time access to it is critical. 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, Liquid Data flushes the cache result automatically on the next invocation.

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.

Note that a data service developer can prevent caching for a data service function by setting the Cache Enabled attribute for the function to false. (By default, caching is enabled for data service functions.) When caching is disabled in the data service design, caching cannot be enabled in the Administration Console for any of that data service's functions. For more information, see the *Liquid Data for WebLogic Data Services Reference Guide*.

Liquid Data also provides a cache API that allows clients to bypass cached results, getting fresh results when needed. Also, the cache API allows for client-side cache flushing. Applications should manually refresh the cache after an update to ensure access to the most recent data.

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.

You can use any data source configured for WebLogic as the caching database. Liquid Data 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 Liquid Data application not share cache tables. There should be separate tables for each application.

To use results caching, you must have one of the following database servers installed and running:

- Oracle
- DB2
- Sybase
- Pointbase
- Microsoft SQL Server

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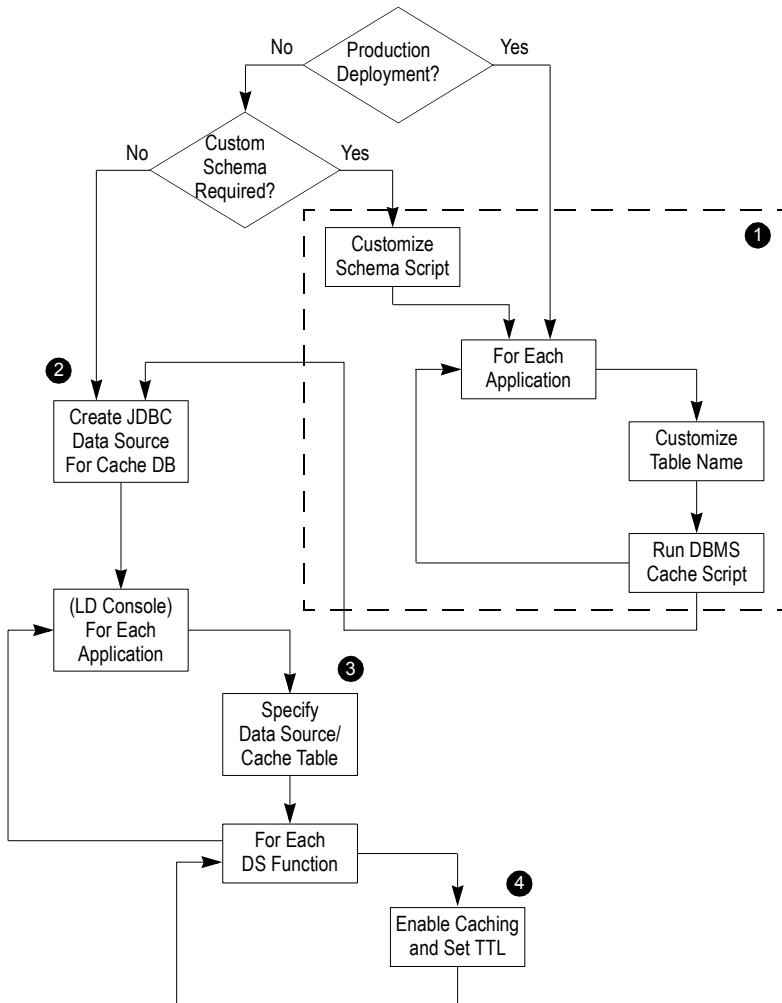
The steps illustrated in Figure 7-1 are described in the following sections:

- 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 Liquid Data set up the cache table automatically from the Liquid Data 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/dbscript.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 Liquid Data (as described in"Step 3: Specify the Cache Data Source and Table" on page 7-8), 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

Liquid Data 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.

Table 7-1 Cache Table Schema (Continued)

Column	Description
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 Liquid Data 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 Liquid Data 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
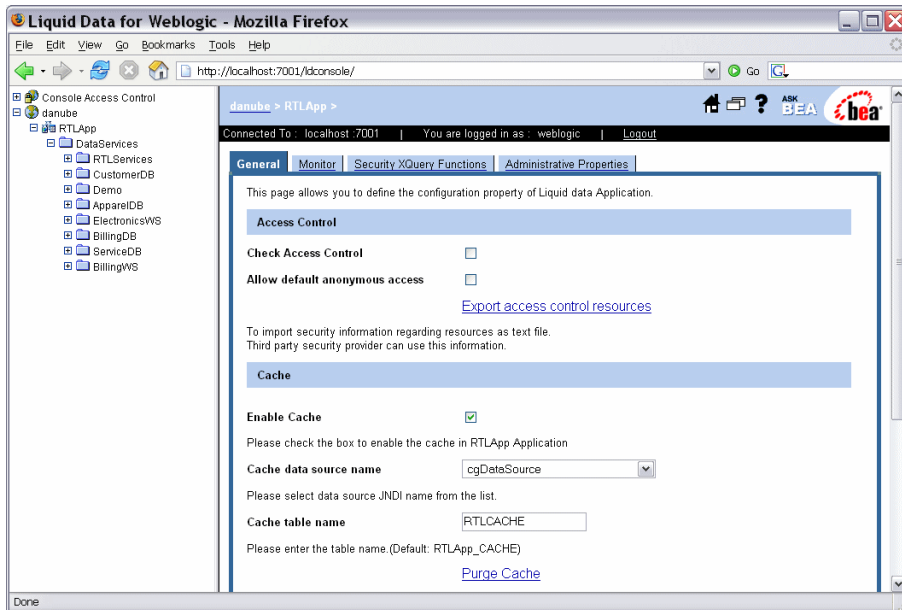
2. In the Cache section of the General tab, click Enable Cache.

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 Liquid Data 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 Liquid Data 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
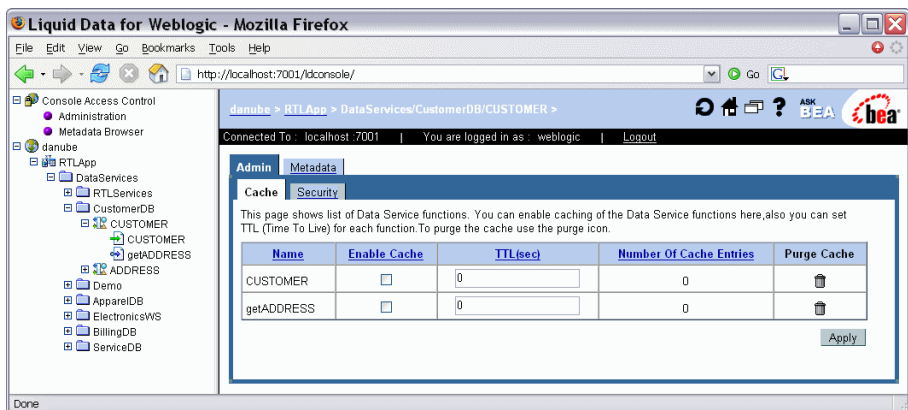
2. Check the Enable Cache checkbox for each function for which you want to enable caching.

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.

Note: All data service functions support caching by default, and can be enabled for caching as described in this procedure. Developers can, however, disable function caching using the WebLogic Workshop.

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. Liquid Data 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

Liquid Data 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 7-11
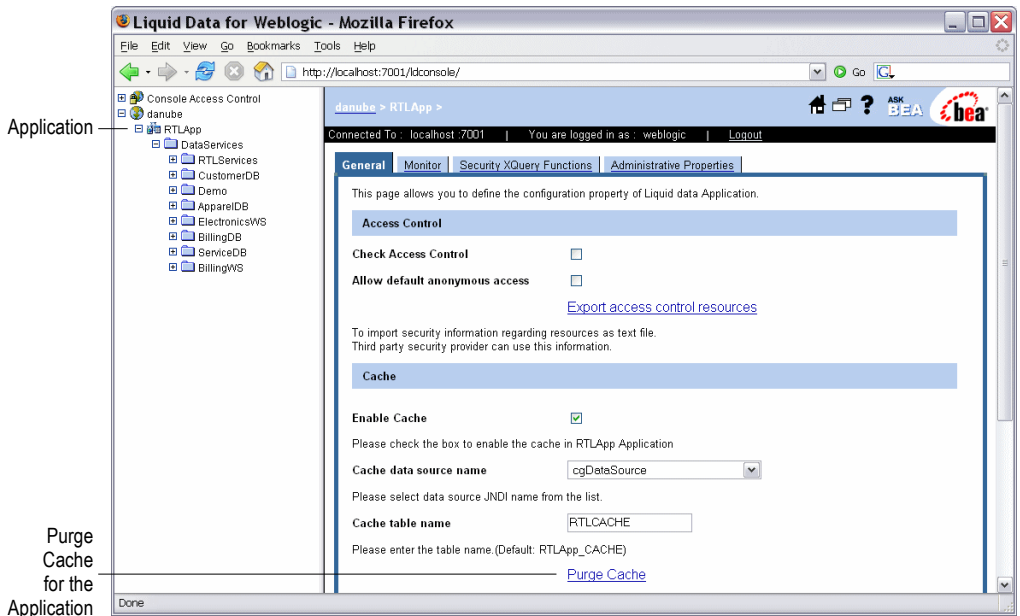- "Purging the Cache for a Function" on page 7-12

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 Liquid Data Console.

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

Figure 7-4 Purging the Cache for an Application

2. Click the Purge Cache link in the Cache section of the General tab.
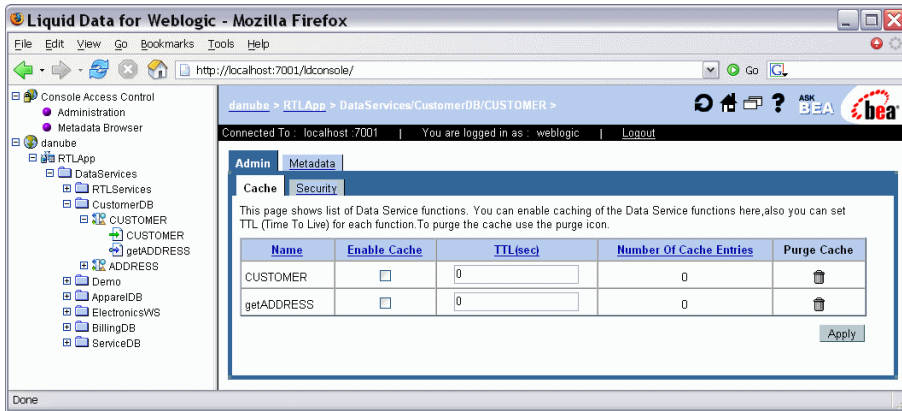
   The console asks for confirmation before purging the cache.

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 Metadata Browser, a component of the Liquid Data Console, enables you to view information on data services, their functions, and their dependencies. This chapter describes how to use the Metadata Browser, and includes the following sections:

- Introducing the Metadata Browser

- Browsing Data Service Metadata

- Displaying Function Metadata

- Searching Metadata

Introducing the Metadata Browser

The Metadata Browser enables you to view metadata related to a Liquid Data 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 Liquid Data deployment.

The Metadata Browser is particularly useful for:

- Liquid Data administrators who need to gauge effects of changes to underlying data sources.

- Developers of Liquid Data 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 "Browsing Data Service Metadata" on page 8-2.

- Browse metadata associated with data service functions. You can display function metadata. For more information, see "Displaying Function Metadata" on page 8-6.

- 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 8-8.

Browsing Data Service Metadata

You can browse data service metadata including general information, read functions and return types, relationships, dependencies, and more using the Metadata tab in the Liquid Data Console.

To browse data service metadata:

1. Select a data service in the Navigation pane.

   The Admin screen appears.

2. Click the Metadata tab.

   The console displays the General metadata associated with the data service, as ill.ustrated in Figure 8-1.

Figure 8-1 Data Service Metadata

3. Click the corresponding tab to display general information, data service read functions, return type, relationships, properties, dependencies, and where used information.

Table 8-1 describes the metadata information accessible through the tabs.

Table 8-1 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 "Understanding Data Service Metadata" on page 8-5.
	• **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.
Dependencies	Displays a table of data services on which the current data service depends. The data services listed in this table contribute content to the current data service's return value. The table shows the following information:
	• **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 "Understanding Data Service Metadata" on page 8-5.
Where Used	Displays a table showing the data services where the currently selected data service is used. Each entry includes name, path, and type information.

# Understanding Data Service Metadata

There are two types of data services, differentiated as follows:

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

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

The metadata that is available varies depending on whether a data service is physical or logical. Logical data sources always have dependencies. The Dependencies tab for the data service lists the composing data services. On the other hand, for a physical data service, the Where Used tab shows the logical data services that are at least partially based on it.

Figure 8-2 illustrates dependencies of a logical data service.

Figure 8-2 Dependencies of a Logical Data Service
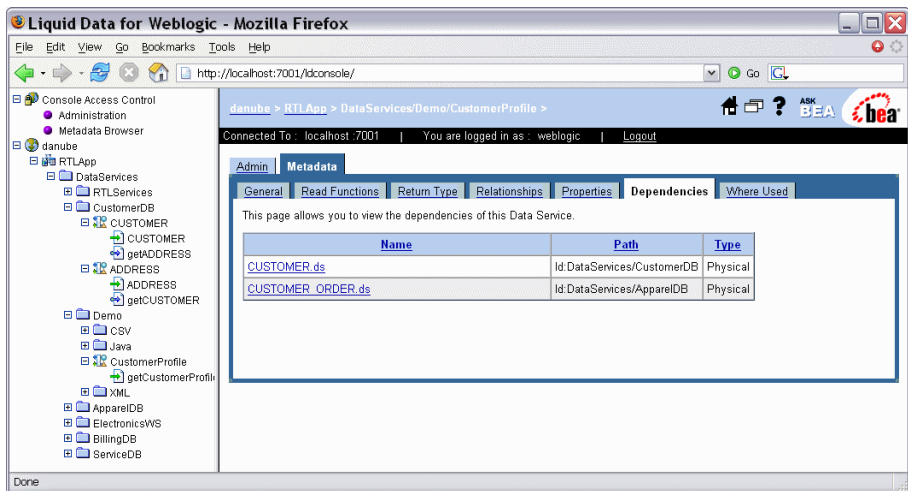
As you would expect of a logical data service, the return type displays the schema of the data from multiple data sources, according to the design of the data service, as illustrated in Figure 8-3.

Figure 8-3 Return Type for a Logical Data Service
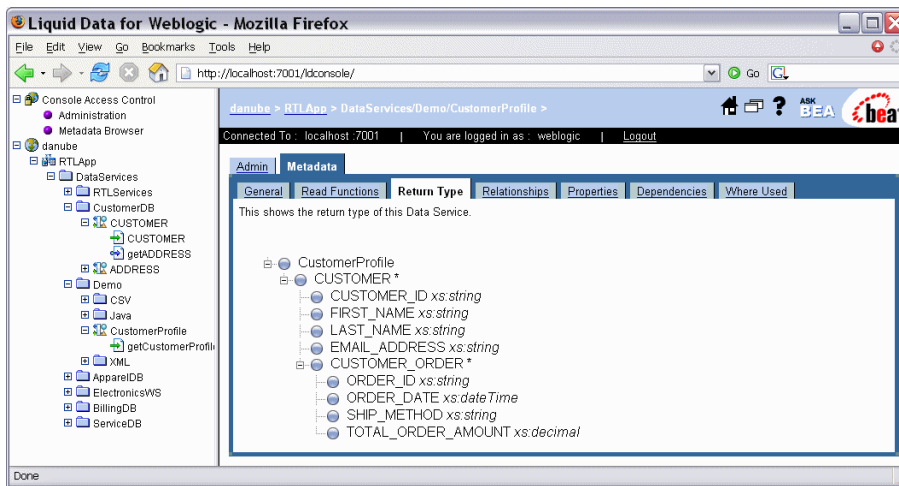
Displaying Function Metadata

You can display 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 8-4 illustrates the function metadata displayed.

Figure 8-4 Function Metadata
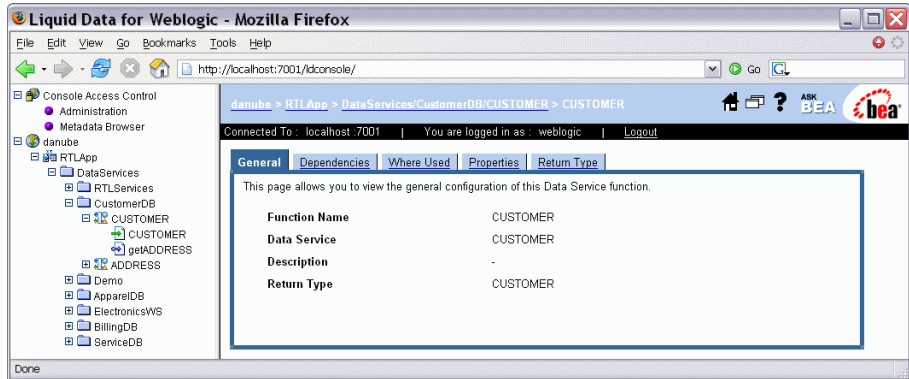
Table 8-2 describes the function metadata available.

Table 8-2 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
Dependencies	Displays a table of data services on which the current function depends. The data services listed in this table contribute content to the current function's return value. The table shows the following information: • **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 "Understanding Data Service Metadata" on page 8-5.
Where Used	Displays a table showing the data services where the currently selected function is 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.

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.

This section describes the following:

- "Performing a Basic Metadata Search" on page 8-8

- "Performing an Advanced Metadata Search" on page 8-9

- "Exploring Metadata Search Results" on page 8-11

- "Generating Reports" on page 8-13

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 a Liquid Data 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 8-5.

Figure 8-5 Basic Metadata Browser Search Facility
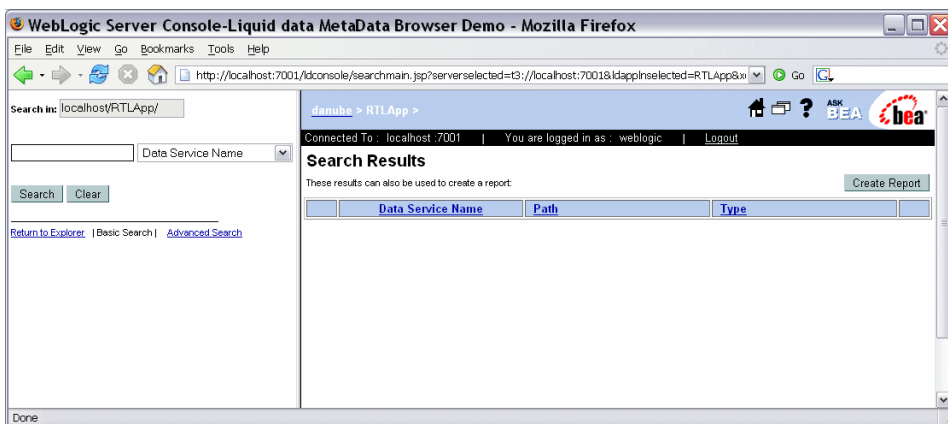
2. Choose the search criteria in the drop-down list.

   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 8-11.

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 8-13.

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 8-9.

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 Liquid Data 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 Liquid Data Console user interface, see "Introducing the Liquid Data Console" on page 4-1.

2. Click Advanced Search.

   The advanced search pane appears, as illustrated in Figure 8-6.

Figure 8-6 Metadata Browser Advanced Search
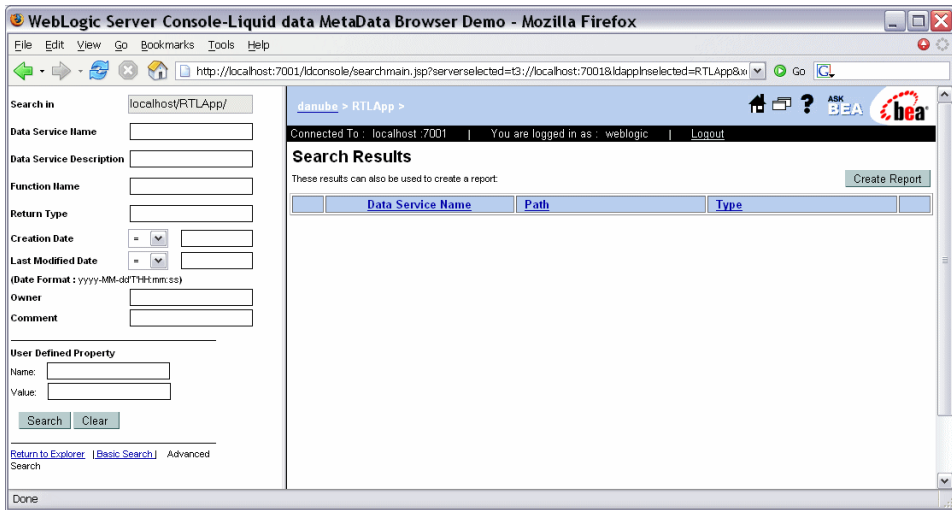
3. Enter the search criteria, as appropriate, and click Search.

    Table 8-3 describes the criteria you can specify using the advanced search facility.

Table 8-3 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.
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.

Table 8-3 Advanced Search Criteria (Continued)

Search Criteria	Description
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 8-11.

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 8-13.

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 8-7 illustrates the search results page.

Figure 8-7 Metadata Search Results
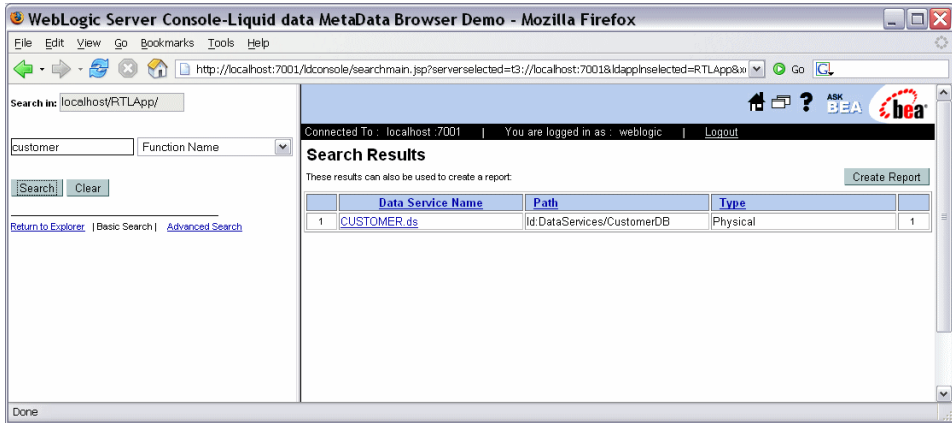
Table 8-4 describes the information displayed as search results.

Table 8-4 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 "Understanding Data Service Metadata" on page 8-5.

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 Liquid Data 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 8-8, enabling you to specify the information to include in the generated report.

Figure 8-8 Generating Reports
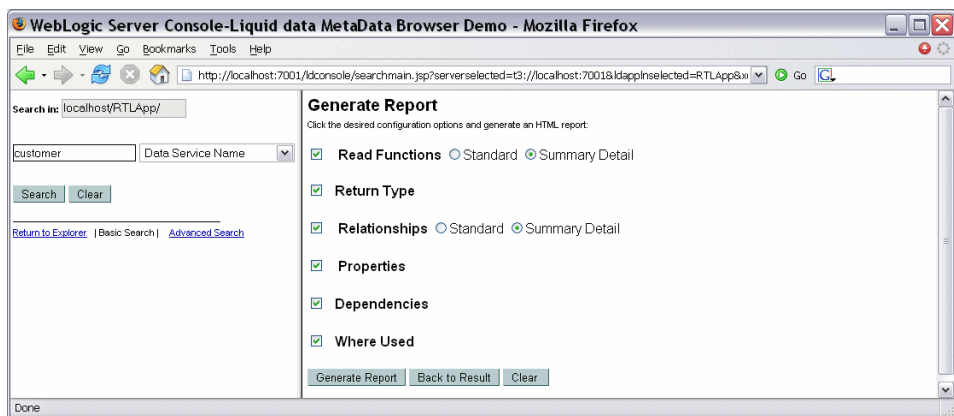
4. Select the information you want to include in the report, and click Generate Report.

   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 8-5 describes the options you can select to defined the information included in the generated report.

Table 8-5 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.

Using Logging Information

This chapter describes how to monitor a running WebLogic Server that has at least one Liquid Data project.

The chapter contains the following sections:

- Monitoring the Server Log

- Monitoring a WebLogic Domain

- Using Other Monitoring Tools

For information on data service monitoring, see "Monitoring Applications" on page 5-6.

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 5-2. 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 Liquid Data Console and the WebLogic Server 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 Liquid Data 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*.

Index