BEA Liquid Data for WebLogic™

Deploying Liquid Data

Release: 1.0.1
Document Date: October 2002
Revised: December 2002

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Deploying Liquid Data

Part Number	Date	Software Version
N/A	October 2002	1.0
N/A	December 2002	1.0.1

Contents

Index

About This Document

This document describes the overall process and the types of tasks necessary to deploy a BEA Liquid Data for WebLogic™ data integration solution in a production environment.

This document covers the following topics:

- Chapter 1, "Introduction," identifies resources associated with Liquid Data deployments and provides an overview of the deployment process.
- Chapter 2, "Deployment Tasks," provides detailed instructions for the tasks involved in deploying Liquid Data in a production environment, including: defining deployment requirements, designing a deployment, and deploying Liquid Data in various configurations (standalone, multi-node, and clustered deployments) in different types of domains (BEA WebLogic Platform™, BEA WebLogic Integration™, BEA WebLogic Server™, BEA WebLogic Workshop™, and BEA WebLogic Portal™).
- Chapter 3, "Tuning Performance," describes Liquid Data performance factors, identifies ways to monitor Liquid Data performance, and provides instructions for tuning performance in a Liquid Data deployment.
- Chapter 4, "Troubleshooting a Deployment," identifies resources to assist with troubleshooting and provides instructions on how to troubleshoot out of memory problems.

What You Need to Know

This document is intended mainly for administrators who are responsible for deploying and maintaining Liquid Data in a production environment. Administrators must know how to navigate the WebLogic Server Administration Console, be able to complete the installation tasks in Installing Liquid Data and the administrative tasks in the Liquid Data Administration Guide, and have platform knowledge.

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/.

Related Information

For more information in general about Java and XQuery, refer to the following sources.

- The Sun Microsystems, Inc. Java site at:

  `http://java.sun.com/`

- The World Wide Web Consortium XML Query section at:

  `http://www.w3.org/XML/Query`

For more information about BEA products, refer to the BEA documentation site at:

`http://edocs.bea.com/`

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 1.0 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`*

Convention	Item
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
`...`	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Introduction

Deployment is the process of setting up BEA Liquid Data for WebLogic™ in a production environment. This topic introduces Liquid Data deployment. It contains the following sections:

- WebLogic Platform Deployment Documentation
- Resources to Deploy
- Resources to Configure
- Overview of Deployment Steps

WebLogic Platform Deployment Documentation

In addition to this document, the BEA WebLogic Platform™ provides a comprehensive deployment documentation page at the following URL: http://edocs.bea.com/platform/docs70/interm/deploy.html

For specific WebLogic Platform components, see the following URLs:

Table 1-1 Deployment Documentation

Resource	Documentation
BEA WebLogic Server™	http://edocs.bea.com/wls/docs70/deployment.html
BEA WebLogic Integration™	http://edocs.bea.com/wli/docs70/deploy/index.htm
BEA WebLogic Portal™	http://edocs.bea.com/wlp/docs70/dev/deploy.htm
BEA WebLogic Workshop™	http://e-docs.bea.com/workshop/docs70/help/index.html#guide/howdoi/navHowDoITopics.html

Resources to Deploy

The following table summarizes the primary resources involved in a Liquid Data deployment:

Table 1-2 Resources to Deploy in a Liquid Data Deployment

Resource	Description
WebLogic Platform	The WebLogic Platform includes the following components: ■ WebLogic Server ■ WebLogic Workshop ■ WebLogic Integration ■ WebLogic Portal **Note:** At a minimum, WebLogic Server must be deployed. Other components are deployed as needed.
Liquid Data software	Liquid Data is deployed in an enterprise archive (`LDS.ear`) file on a single WebLogic Server instance or, in a clustered deployment, across all servers in the cluster.
Liquid Data configuration	The following settings must be configured: ■ Liquid Data server configuration ■ data source descriptions ■ server repository ■ results cache configuration ■ security, if used ■ custom function descriptions, if used For more information, see "Resources to Configure" on page 1-4.
Data sources	Data sources used in Liquid Data queries must be deployed and accessible to the Liquid Data Server.

Resources to Configure

You need to configure the following resources as applicable for your deployment:

Table 1-3 Liquid Data Resources to Configure

Resource	Description
Liquid Data server settings	You need to configure server settings on the General tab in the Liquid Data node in the Administration Console, especially the repository location. For instructions, see "Configuring Liquid Data Server Settings" in the Liquid Data *Administration Guide*.
Data source descriptions	For each data source accessed in a Liquid Data query, you need to add a data source description using the Data Sources tab in the Liquid Data node in the Administration Console. Additional configuration tasks are required for certain data source types. For instructions, see the following topics in the Liquid Data *Administration Guide*: ■ "Configuring Access to Relational Databases" ■ "Configuring Access to XML Files" ■ "Configuring Access to Web Services" ■ "Configuring Access to Application Views" ■ "Configuring Access to Data Views"
Server repository	You need to configure the location of the server repository on the General tab in the Liquid Data node in the Administration Console. You also need to populate and configure the repository on the Repository tab in the Liquid Data node in the Administration Console. For instructions, see "Managing the Liquid Data Server Repository" in the Liquid Data *Administration Guide*.
Results caching	If you want to cache results for stored queries in this deployment, you must explicitly enable results caching on the General tab in the Liquid Data node in the Administration Console. In addition, for each stored query that you want cached, you need to explicitly configure the caching policy. For instructions, see "Configuring the Query Results Cache" in the Liquid Data *Administration Guide*.

Table 1-3 Liquid Data Resources to Configure (Continued)

Resource	Description
Security	If you want to use security in this deployment, you must enable secure mode on the General tab in the Liquid Data node in the Administration Console. In addition, you need to explicitly configure secure access to the Data View Builder, data source descriptions, custom function descriptions, the server repository, and stored queries. For instructions, see "Implementing Security" in the Liquid Data *Administration Guide*.
Custom functions	If custom functions are used in this deployment, you need to create a custom function description for each custom function and add certain files to the server repository. For instructions, see "Configuring Access to Custom Functions" in the Liquid Data *Administration Guide*.

Note: Alternatively, you can copy configuration information from another, fully configured Liquid Data Server according to the instructions in "Copying a Server Configuration to Another Server" on page 2-48.

Overview of Deployment Steps

The process of deploying Liquid Data involves the following basic steps:

1. Plan the deployment according to the instructions in "Defining Deployment Requirements" on page 2-2 and "Designing a Deployment" on page 2-3. Completing these tasks is necessary to ensure that your deployment architecture best suits your business requirements.
2. Install the WebLogic Platform and Liquid Data software on the target server(s) according to the instructions in Installing Liquid Data.
3. On the target server(s), create a domain and deploy Liquid Data to that domain, as described in Chapter 2, "Deployment Tasks."
4. If it is not already up and running, start WebLogic Server in the domain according to the instructions in "Starting and Stopping the Server" in the Liquid Data *Administration Guide*.
5. Start the Administration Console and configure the resources described in "Resources to Configure" on page 1-4 as applicable to your deployment.
6. Monitor the ongoing operation of the Liquid Data Server according to the instructions in "Monitoring the Server" in the Liquid Data *Administration Guide*.
7. Tune performance on the Liquid Data Server, as necessary, according to the instructions in Chapter 3, "Tuning Performance."

These steps provide only a summary of the steps required to deploy Liquid Data in a production environment. For example, multi-node and clustered deployments involve additional steps. For detailed deployment instructions based on the deployment architecture you are using, see Chapter 2, "Deployment Tasks."

CHAPTER

2 Deployment Tasks

This topic describes the tasks involved in deploying BEA Liquid Data for WebLogic™ in a production environment. It includes the following sections:

Defining Deployment Requirements

Defining deployment requirements is a critical first step in the process of deploying Liquid Data in a production environment.

When planning a deployment, consider the following issues:

- Business requirements—Identify and prioritize business requirements in a production environment, such as:
  - system performance required at anticipated peak loads
  - high availability and failover requirements
  - security and data access requirements
- Resources to be deployed—Identify the components of the BEA WebLogic Platform™ to use.

  At a minimum, BEA WebLogic Server™ and Liquid Data need to be deployed on at least one server.
- Production environment—Identify the hardware and software components of the production environment, including the network infrastructure, server nodes, and data sources.
- Usage requirements—Characterize how users are likely to utilize the system.

  Which types of queries (stored or ad hoc queries) will they run? How frequently will users be running different types of queries? Which queries will be run more frequently or less frequently than others? What are the anticipated peak loads for processing concurrent query requests?
- Data source requirements—Determine how to access any data sources used in Liquid Data queries.
- Security requirements—Determine which users need access to Liquid Data resources and the level of access they need.
- Licensing requirements—Determine the BEA software licenses needed to implement your deployment solution.

Clarifying the specific requirements for your deployment provides the knowledge you will need to design your deployment.

Designing a Deployment

This topic describes how to design a Liquid Data deployment. It describes:

- Single Domain Deployments
  - Single Server Deployments
  - Multi-Node Deployments
  - Clustered Deployments
- Multi-Domain Deployments

The deployment architecture that you use depends on the requirements for your particular production environment, as described in "Defining Deployment Requirements" on page 2-2.

You choose one of these types of deployments when you create the domain using the WebLogic Platform Configuration Wizard. For more information, see "Creating a New WebLogic Domain" in *Using the Configuration Wizard* in the WebLogic Platform documentation.

Single Domain Deployments

The following sections describe single domain deployments:

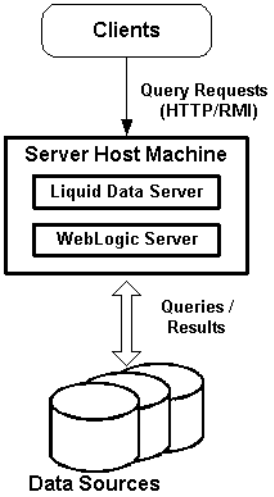
- Single Server Deployments
- Multi-Node Deployments
- Clustered Deployments

Single Server Deployments

In a single server (standalone) deployment, the Liquid Data software is installed on a standalone WebLogic Server. This design is the simplest to set up and it is the one suggested for use in a development environment or in a production environment in which Liquid Data is the primary application and failover protection is not the top priority.

The following illustration shows Liquid Data and WebLogic Server deployed on a single server:

Figure 2-1 Liquid Data and WebLogic Platform Deployed on a Single Server



For instructions on how to deploy Liquid Data and other WebLogic Platform components on a single server (standalone) deployment, see the following sections:

- "Deploying Liquid Data on a Standalone WebLogic Platform Domain" on page 2-8
- "Deploying Liquid Data On a Standalone WebLogic Integration Domain" on page 2-20
- "Deploying Liquid Data in a Standalone WebLogic Server Domain" on page 2-30
- "Deploying Liquid Data in a Standalone WebLogic Portal Domain" on page 2-40
- "Deploying Liquid Data in a Standalone WebLogic Workshop Domain" on page 2-43

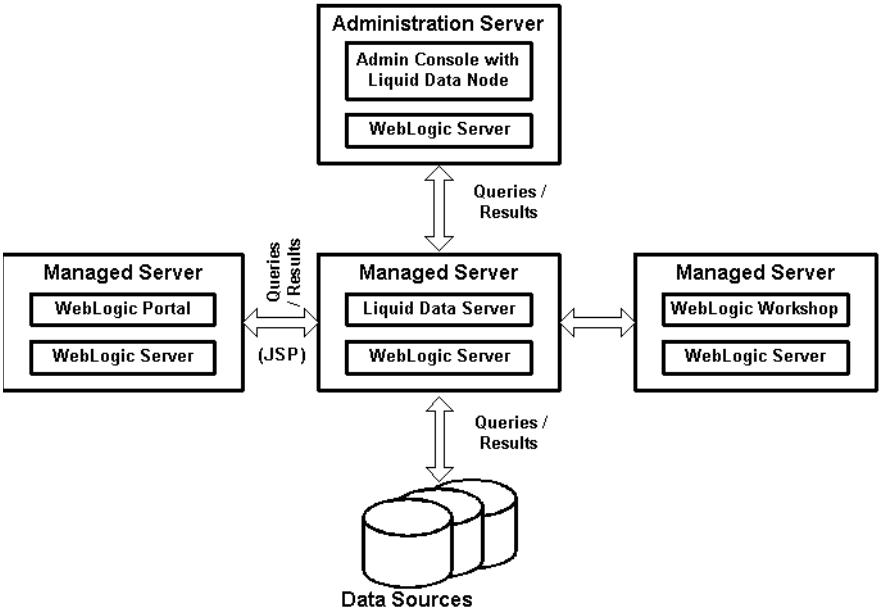
Multi-Node Deployments

A *multi-node deployment* distributes Liquid Data and WebLogic Platform software components across multiple server machines. A multi-node deployment allows you to run the various software components on dedicated servers, distributing resource contention across machines and optimizing system performance.

In each multi-node WebLogic Server domain, one WebLogic Server instance acts as the Administration Server—the server instance that configures, manages, and monitors all other server instances and resources in the domain. In a multi-node deployment, the Administration Server has administrative control over the domain and hosts the server repository. All other servers in the domain are configured as Managed Servers—servers that are managed by the Administration Server.

The following illustration shows Liquid Data deployed in a sample multi-node domain with three dedicated servers, one each for Liquid Data, WebLogic Portal, and WebLogic Workshop:

Figure 2-2 Liquid Data Deployed in a Multi-Node Domain



To deploy this design, follow the steps described in "Deploying Liquid Data on a Multi-Node WebLogic Domain" on page 2-11.

Clustered Deployments

A *clustered deployment* distributes Liquid Data and WebLogic Platform software components across multiple server machines. A *cluster* is a group of servers that work together to provide an application platform that is more powerful and reliable than a single server. A cluster appears to its clients as a single server but it is, in fact, a group of servers acting as one. In contrast to a multi-node deployment (in which Liquid Data and WebLogic Platform software components are installed separately on *dedicated* machines), in a clustered deployment, all Liquid Data and WebLogic Platform software components are installed on *every* machine.

A Liquid Data cluster is a deployment in which multiple copies of Liquid Data, referred to as *instances*, run simultaneously and work together to provide increased scalability and reliability. A Liquid Data cluster appears to clients to be a single Liquid Data instance. The server instances that constitute a cluster can run on the same machine, or can be located on different machines. You can increase a cluster's capacity by adding additional server instances to the cluster on an existing machine, or you can add machines to the cluster to host the incremental server instances. Each server instance in a cluster must run the same version of WebLogic Server. For extensive information about clustering, see "Using WebLogic Server Clusters" in the WebLogic Server documentation.

In each WebLogic Server domain, one WebLogic Server instance acts as the Administration Server—the server instance that configures, manages, and monitors all other server instances and resources in the domain. In a clustered deployment, the Administration Server has administrative control over the domain and hosts the server repository. All other servers in the domain are configured as Managed Servers—servers that are managed by the Administration Server. In a single instance deployment, the sole WebLogic Server instance is the Administration Server by default.

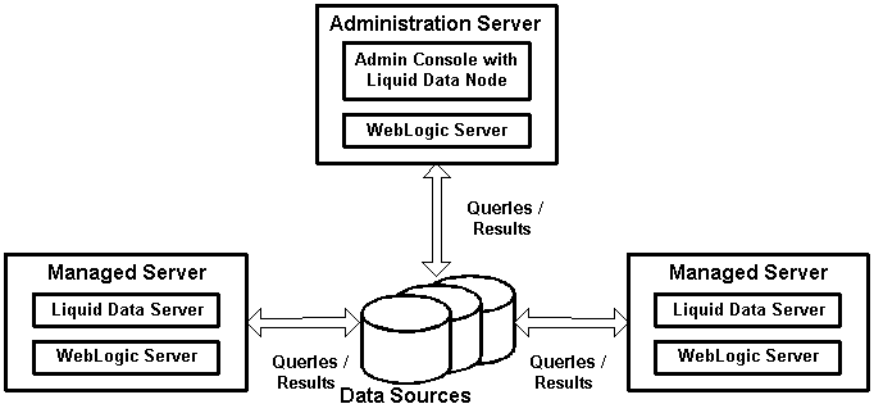
Note: In general, avoid configuring the Administration Server as part of the cluster.

In a clustered Liquid Data deployment, you select the algorithm to use for load balancing (round robin, weight-based, or random). Once configured, the load balancing mechanism directs incoming XQuery query requests to available Liquid Data Server instances. For more information, see "Load Balancing in a Cluster" in *Using WebLogic Server Clusters* in the WebLogic Server documentation.

In a clustered Liquid Data deployment, if a server fails, the WebLogic cluster can redirect request processing to another Liquid Data Server instance in the cluster. For more information, see "Replication and Failover for EJBs and RMIs" in "Failover and Replication in a Cluster" in *Using WebLogic Server Clusters* in the WebLogic Server documentation.

The following illustration shows Liquid Data deployed in a clustered domain.

Figure 2-3 Liquid Data Deployed in a Clustered Domain



This design is suggested for a deployment environment in which:

- The high volume of concurrent query requests necessitates additional machines in order to achieve system performance requirements.
- High availability through failover protection in the event of a server failure is a deployment requirement.

Before proceeding to deploy Liquid Data in a cluster, you should determine whether running Liquid Data on a dedicated server machine, as configured in "Multi-Node Deployments" on page 2-5, meets the performance and availability requirements for your deployment. Using the single server as a baseline, you can monitor run-time performance, conduct capacity planning, and test to determine whether a single machine provides sufficient performance at high query request loads.

To deploy this design, follow the steps described in:

- "Deploying a Liquid Data Cluster on a WebLogic Platform Domain" on page 2-15

- "Deploying Liquid Data in a Clustered WebLogic Integration Domain" on page 2-23
- "Deploying Liquid Data in a Clustered WebLogic Server Domain" on page 2-33

Multi-Domain Deployments

In a multi-domain deployment, Liquid Data and the WebLogic Platform component are installed on separate WebLogic domains. To deploy this design for WebLogic Portal, follow the steps described in "Deploying Liquid Data and WebLogic Portal in Separate Domains" on page 2-43.

Deploying Liquid Data in a WebLogic Platform Domain

The following topics describe how to deploy Liquid Data with WebLogic Platform:

- Deploying Liquid Data on a Standalone WebLogic Platform Domain
- Deploying Liquid Data on a Multi-Node WebLogic Domain
- Deploying a Liquid Data Cluster on a WebLogic Platform Domain

Deploying Liquid Data on a Standalone WebLogic Platform Domain

This section describes how to deploy Liquid Data on a standalone (single server) WebLogic Platform domain. For more information about single server deployments, see "Single Server Deployments" on page 2-3.

To deploy Liquid Data on a standalone WebLogic Platform domain:

1. Install the WebLogic Platform on the target server according to the instructions in "Installing WebLogic Platform" in the WebLogic Platform documentation.

   Note: Do not start up WebLogic Server until you are instructed to do so later in this section.

2. Install the Liquid Data software on the target server according to the instructions in Installing Liquid Data.

   Note: Check the Liquid Data Release Notes for any last-minute deployment instructions.

3. Unless you are deploying onto an existing WebLogic Platform domain, start the WebLogic Platform Configuration Wizard on the target node and create a WebLogic Platform domain (referred to in this section as *`platform_domain_name`*):

   - For the server type, select Single Server (Standalone).
   - Later in this procedure, you use the Administrative user created in this step to configure the domain.

   For instructions, see "Creating a New WebLogic Domain" in *Using the Configuration Wizard* in the WebLogic Platform documentation.

   Note: If you want to deploy Liquid Data on an *existing* WebLogic Platform domain instead of creating a new one, then the existing domain *must* use a WebLogic Server compatibility security realm. For more information, see "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.

4. Edit the server startup file in *`BEA_HOME`*`/user_projects/`*`platform_domain_name`*. You need to specify the Liquid Data home directory (*`ld_home_dir`*), such as `WL_HOME\liquiddata` (on Windows) or `$WL_HOME/liquiddata` (Unix).

   Note: If your deployment will use application views as data sources, you need to add the path to the Application Integration `wlai-client.jar` file (such as *`%WLI_HOME%`*`\lib\wlai-client.jar`) to the *end* of the Liquid Data classpath (`LDCLASSPATH`).

   - **Windows**: Edit the `startweblogic.cmd` file and add the following command lines *before* `startWLS.cmd` executes:

     ```
     set LD_HOME=ld_home_dir
     set LDI_LIB=%LD_HOME%\server\lib
     ```

```
set
LDCLASSPATH=%LDI_LIB%\wlldi.jar;%LDI_LIB%\LDS-descriptions.j
ar;%LDI_LIB%\castor-0.9.3.9.jar;%LDI_LIB%\xercesImpl.jar

set CLASSPATH=%CLASSPATH%;%LDCLASSPATH%
```

- **Unix**: Edit the `startweblogic.sh` file and add the following command lines *before* `startWLS.sh` executes:

```
LD_HOME=ld_home_dir

LDI_LIB=$LD_HOME/server/lib

LDCLASSPATH=$LDI_LIB/wlldi.jar:$LDI_LIB/LDS-descriptions.jar
:$LDI_LIB/castor-0.9.3.9.jar:$LDI_LIB/xercesImpl.jar

CLASSPATH=$CLASSPATH:$LDCLASSPATH

export CLASSPATH
```

  Note: The `$LDCLASSPATH` must *follow* the current `CLASSPATH` setting.

5. Copy the following files from *`LD_HOME`*`/server/lib` to *`BEA_HOME`*`/user_projects/`*`platform_domain_name`*.
   - `ConfigMap.xml`
   - `webapp.template`
6. Start WebLogic Server and wait until it is in running mode.
7. Start the Administration Console on the WebLogic Platform domain, logging in with the username/password you had created when you configured your domain, and configure Liquid Data security according to the following steps:
   a. In the left pane, click on *`platform_domain_name`*->Compatibility Security->Users and add a new user: `ldsystem`.
   b. Click on *`platform_domain_name`*->Compatibility Security->Groups and add a new group: `LDAdmin`.
   c. Add the `ldsystem` user to the `LDAdmin` group.
   d. Add the `ldsystem` user to the `Administrators` group.
   e. Click on *`platform_domain_name`*->Compatibility Security->ACLs and add a new ACL: `LD`.
   f. Add the following new permission attributes to the `LD` ACL: `execute`, `read`, and `modify`, and then add the `Administrators` group to the grantee list of permissions.

8. Deploy the `LDS.ear` file according to the following steps:

   a. In the left pane, click on *`platform_domain_name`*`->Deployments->Applications`, click on Configure a New Application, click on "Upload it through your browser," select *`LD_HOME`*`/server/lib/LDS.ear`, and upload it.

   b. Click on the Select link next to the `LDS.ear` file, and then click Upload.

   c. Select the target server on which to deploy Liquid Data, and then click Configure and Deploy.

9. Exit the Administration Console.

10. Start the Administration Console, logging in as `ldsystem`.

11. Configure data sources, security, the repository, and other Liquid Data server settings, as needed, according to the instructions in the Liquid Data Administration Guide.

Deploying Liquid Data on a Multi-Node WebLogic Domain

This section describes how to deploy Liquid Data on a WebLogic domain with an Administration Server and Managed Servers in a multi-node, non-clustered environment. For more information about multi-node domains, see "Multi-Node Deployments" on page 2-5.

Notes: Due to limitations with WebLogic Integration, do *not* configure WebLogic Integration on a WebLogic domain in a multi-node environment.

Before you begin, verify that your BEA software license is appropriate for your deployment.

To deploy Liquid Data on a multi-node WebLogic Platform domain:

1. Install the WebLogic Platform on the machines that will host the Administration Server and the target Managed Servers, according to the instructions in "Installing WebLogic Platform" in the WebLogic Platform documentation.

Note: Do not start up WebLogic Server until you are instructed to do so later in this section.

2. Install the Liquid Data software on the machines that will host the Administration Server and the target server, according to the instructions in Installing Liquid Data.

   Note: Check the Liquid Data Release Notes for any last-minute deployment instructions.

3. Unless you are deploying onto an existing WebLogic domain, start the WebLogic Platform Configuration Wizard on the Administration Server and create a WebLogic domain (referred to in this section as *wls_domain_name*):

   - For the server type, select Admin Server with Managed Servers.
   - Later in this procedure, you use the Administrative user created in this step to configure the domain.

   For instructions, see "Creating a New WebLogic Domain" in *Using the Configuration Wizard* in the WebLogic Platform documentation.

   Note: If you want to deploy Liquid Data on an *existing* WebLogic Platform domain instead of creating a new one, then the existing domain *must* use a WebLogic Server compatibility security realm. For more information, see "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.

4. Unless you are deploying onto an existing WebLogic Platform domain, start the WebLogic Platform Configuration Wizard on *each* Managed Server in the domain, create a WebLogic Platform domain with same name as the one created on the Administration Server, and, for the server type, select Managed Server (with Owning Admin Server Configuration).

5. Edit the server startup files in *BEA_HOME*`/user_projects/`*wls_domain_name*. You need to specify the Liquid Data home directory (*ld_home_dir*), such as `WL_HOME\liquiddata` (on Windows) or `$WL_HOME/liquiddata` (Unix).

   Note: If your deployment will use application views as data sources, you need to add the path to the Application Integration `wlai-client.jar` file (such as *%WLI_HOME%*`\lib\wlai-client.jar`) to the *end* of the Liquid Data classpath (`LDCLASSPATH`).

- **Windows**: Edit the `startweblogic.cmd` file (on the Administration Server and `startManagedWeblogic.cmd` file (on Managed Servers) and add the following command lines *before* `startWLS.cmd` executes:

  ```
  set LD_HOME=ld_home_dir

  set LDI_LIB=%LD_HOME%\server\lib

  set
  LDCLASSPATH=%LDI_LIB%\wlldi.jar;%LDI_LIB%\LDS-descriptions.j
  ar;%LDI_LIB%\castor-0.9.3.9.jar;%LDI_LIB%\xercesImpl.jar

  set CLASSPATH=%CLASSPATH%;%LDCLASSPATH%
  ```

- **Unix**: Edit the `startweblogic.sh` file (on the Administration Server and `startManagedWeblogic.sh` file (on Managed Servers) and add the following command lines *before* `startWLS.sh` executes:

  ```
  LD_HOME=ld_home_dir

  LDI_LIB=$LD_HOME/server/lib

  LDCLASSPATH=$LDI_LIB/wlldi.jar:$LDI_LIB/LDS-descriptions.jar
  :$LDI_LIB/castor-0.9.3.9.jar:$LDI_LIB/xercesImpl.jar

  CLASSPATH=$CLASSPATH:$LDCLASSPATH

  export CLASSPATH
  ```

  Note: The `$LDCLASSPATH` must *follow* the current `CLASSPATH` setting.

6. On the Administration Server, copy the following files from *`LD_HOME`*`/server/lib` to *`BEA_HOME`*`/user_projects/`*`platform_domain_name`*.
   - `ConfigMap.xml`
   - `webapp.template`
7. Start the Administration Server and wait until it is in running mode.
8. On the Administration Server, start the Administration Console on the WebLogic domain, logging in as an administrator, and configure Liquid Data security according to the following steps:
   a. In the left pane, click on *`wls_domain_name`*->Compatibility Security->Users and add a new user: `ldsystem`.
   b. Click on *`wls_domain_name`*->Compatibility Security->Groups and add a new group: `LDAdmin`.

c. Add the `ldsystem` user to the `LDAdmin` group.

d. Add the `ldsystem` user to the `Administrators` group.

e. Click on *`wls_domain_name`*->Compatibility Security->ACLs and add a new ACL: `LD`.

f. Add the following new permission attributes to the `LD` ACL: `execute`, `read`, and `modify`, and then add the `Administrators` group to the grantee list of permissions.

9. Start all Managed Servers in the domain and wait until they are all in running mode.

10. Deploy the `LDS.ear` file according to the following steps:

    a. In the left pane, click on *`wls_domain_name`*`->Deployments->Applications`, click on Configure a New Application, click on "Upload it through your browser," select *`LD_HOME`*`/server/lib/LDS.ear`, and upload it.

    b. Click on the Select link next to the `LDS.ear` file, and then click Upload.

    c. Select one or more target Managed Servers on which to deploy Liquid Data, and then click Configure and Deploy.

    Note: Do *not* deploy the `LDS.ear` file on the Administration Server.

11. Verify that the following files are deployed successfully:
    - `ldconsole.war`
    - `ldcacheListener.war`
    - `cacheEjb.jar`
    - `ejb_query.jar`
    - `XMediator.war`
    - `ejb_qbc.jar`

12. In the left pane, click on *`wls_domain_name`*`->Deployments->Applications->LDS->ldconsole.war`, click the Targets tab, select the Administration Server from the list of available servers, and then click Apply.

13. Click on the Deploy tab and then click the Deploy button to deploy `ldconsole.war` to the Administration Server.

14. Exit the Administration Console.
15. Start the Administration Console, logging in as `ldsystem`.
16. Configure data sources, security, the repository, and other Liquid Data server settings, as needed, according to the instructions in the Liquid Data Administration Guide.

Deploying a Liquid Data Cluster on a WebLogic Platform Domain

This section describes how to deploy a Liquid Data cluster on a WebLogic domain with an Administration Server and Managed Servers. Clustering is typically used to provide scalability in deployments with high volumes of query requests and to provide fail-over protection in deployments with high availability requirements. For more information about clustered deployments, see "Clustered Deployments" on page 2-6.

Note: Before you begin, verify that your BEA software license is appropriate for your deployment.

To deploy a Liquid Data cluster on a WebLogic Platform domain:

1. Install the WebLogic Platform on the machines that will host the Administration Server and all target Managed Servers, according to the instructions in "Installing WebLogic Platform" in the WebLogic Platform documentation.

   Note: Do not start up WebLogic Server until you are instructed to do so later in this section.

2. Install the Liquid Data software on the machines that will host the Administration Server and all target Managed Servers, according to the instructions in Installing Liquid Data.

   Note: Check the Liquid Data Release Notes for any last-minute deployment instructions.

3. Unless you are deploying onto an existing WebLogic domain, start the WebLogic Platform Configuration Wizard on the Administration Server and create a clustered WebLogic domain (referred to in this section as *`platform_domain_name`*):

- For the server type, select Admin Server with Clustered Managed Server(s).
- Later in this procedure, you use the Administrative user created in this step to configure the domain.

For instructions, see "Creating a New WebLogic Domain" in *Using the Configuration Wizard* in the WebLogic Platform documentation.

Note: If you want to deploy Liquid Data on an *existing* WebLogic Platform domain instead of creating a new one, then the existing domain *must* use a WebLogic Server compatibility security realm. For more information, see "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.

4. Unless you are deploying onto an existing WebLogic Platform domain, start the WebLogic Platform Configuration Wizard on *each* Managed Server in the cluster, create a WebLogic Platform domain with same name as the one created on the Administration Server, and, for the server type, select Managed Server (with Owning Admin Server Configuration).

5. Edit the server startup files in *BEA_HOME*/user_projects/*platform_domain_name*. You need to specify the Liquid Data home directory (*ld_home_dir*), such as `WL_HOME\liquiddata` (on Windows) or `$WL_HOME/liquiddata` (Unix).

   Note: If your deployment will use application views as data sources, you need to add the path to the Application Integration `wlai-client.jar` file (such as *%WLI_HOME%*`\lib\wlai-client.jar`) to the *end* of the Liquid Data classpath (`LDCLASSPATH`).

   - **Windows**: Edit the `startweblogic.cmd` file (on the Administration Server and `startManagedWeblogic.cmd` file (on Managed Servers) and add the following command lines *before* `startWLS.cmd` executes:

     ```
     set LD_HOME=ld_home_dir

     set LDI_LIB=%LD_HOME%\server\lib

     set
     LDCLASSPATH=%LDI_LIB%\wlldi.jar;%LDI_LIB%\LDS-descriptions.j
     ar;%LDI_LIB%\castor-0.9.3.9.jar;%LDI_LIB%\xercesImpl.jar

     set CLASSPATH=%CLASSPATH%;%LDCLASSPATH%
     ```

   - **Unix**: Edit the `startweblogic.sh` file (on the Administration Server and `startManagedWeblogic.sh` file (on Managed Servers) and add the following command lines *before* `startWLS.sh` executes:

```
LD_HOME=ld_home_dir

LDI_LIB=$LD_HOME/server/lib

LDCLASSPATH=$LDI_LIB/wlldi.jar:$LDI_LIB/LDS-descriptions.jar
:$LDI_LIB/castor-0.9.3.9.jar:$LDI_LIB/xercesImpl.jar

CLASSPATH=$CLASSPATH:$LDCLASSPATH

export CLASSPATH
```

Note: The `$LDCLASSPATH` must *follow* the current `CLASSPATH` setting.

6. On the Administration Server, copy the following files from *`LD_HOME`*`/server/lib` to *`BEA_HOME`*`/user_projects/`*`platform_domain_name`*.
   - `ConfigMap.xml`
   - `webapp.template`
7. Start the Administration Server and wait until it is in running mode.
8. On the Administration Server, start the Administration Console on the WebLogic Platform domain, logging in with the username/password you had created when you configured your domain, and configure Liquid Data security according to the following steps:
   a. In the left pane, click on *`platform_domain_name`*->Compatibility Security->Users and add a new user: `ldsystem`.
   b. Click on *`platform_domain_name`*->Compatibility Security->Groups and add a new group: `LDAdmin`.
   c. Add the `ldsystem` user to the `LDAdmin` group.
   d. Add the `ldsystem` user to the `Administrators` group.
   e. Click on *`platform_domain_name`*->Compatibility Security->ACLs and add a new ACL: `LD`.
   f. Add the following new permission attributes to the `LD` ACL: `execute`, `read`, and `modify`, and then add the `Administrators` group to the grantee list of permissions.
9. By default, WebLogic Server uses the round-robin algorithm as the default load balancing strategy for clustered object stubs. If you want to use weight-based or random load balancing for EJBs and RMI objects instead, complete the following steps:

a. In the left pane of the Administration Console, select the Clusters node.

b. Select your cluster.

c. Click the drop-down arrow next to the Default Load Algorithm to display load balancing algorithms.

d. In the item list, select the load balancing algorithm you want to use in the cluster.

e. Enter the desired value in the Service Age Threshold field (see the Administration Console Online Help for more information).

f. Click Apply to save your changes.

For more information about load balancing algorithms, see "Load Balancing for EJBs and RMI Objects" in "Load Balancing in a Cluster" in *Using WebLogic Server Clusters* in the WebLogic Server documentation.

10. Start all Managed Servers in the domain and wait until they are all in running mode.

11. Deploy the `LDS.ear` file according to the following steps:

    a. In the left pane, click on *`platform_domain_name`*`->Deployments->Applications`, click on Configure a New Application, click on "Upload it through your browser," select *`LD_HOME`*`/server/lib/LDS.ear`, and upload it.

    b. Click on the Select link next to the `LDS.ear` file, and then click Upload.

    c. Select the cluster on which to deploy Liquid Data, and then click Configure and Deploy.

    Note: Do *not* deploy the `LDS.ear` file on the Administration Server.

12. Verify that the following files are deployed successfully:

    - `ldconsole.war`
    - `ldcacheListener.war`
    - `cacheEjb.jar`
    - `ejb_query.jar`
    - `XMediator.war`
    - `ejb_qbc.jar`

13. In the left pane, click on *platform_domain_name*->Deployments->Applications->LDS->ldconsole.war, click the Targets tab, select the Administration Server from the list of available servers, and then click Apply.
14. Click on the Deploy tab, click the Deploy button to deploy `ldconsole.war` to the Administration Server.
15. Exit the Administration Console.
16. On the Administration Server, start the Administration Console, logging in as `ldsystem`.
17. In the left pane, click on the Liquid Data node, then click the General tab. The default repository directory is `ldrepository` in the *BEA_HOME*/user_projects/*platform_domain_name* directory.

    You need to change the repository directory path so that it points to a shared volume to which all Managed Servers have access. On Windows, for example, you would point to a shared network drive mapped to a specific drive letter (such as `R:\ldrepos`).
18. On each Managed Server, you need configure the repository location by mounting (Unix NFS) or mapping (Windows) logical drives to point to the configured repository root on the Administration Server. The directory must be shared and mapped to a virtual disk name. The path must be identical to the value specified in the Repository Root Directory field on the General tab in the Liquid Data node.
    - On Unix, use NFS to mount the exported directory in the respective *BEA_HOME*/user_projects/*platform_domain_name*. Make a short cut of the mapped drive in the managed domain and name it the same as in the admin domain.
    - On Windows, map to the drive using the exact same drive letter and path (such as `R:\ldrepos`).
19. Configure data sources, security, the repository, and other Liquid Data server settings, as needed, according to the instructions in the Liquid Data Administration Guide.
20. Test the clustered deployment by verifying the operation of each machine—individually and in combination with other machines. Under a given query request load, you should see performance increase as you bring additional Managed Servers online.

Deploying Liquid Data in a WebLogic Integration Domain

The following sections describe how to deploy Liquid Data with WebLogic Integration.

- Deploying Liquid Data On a Standalone WebLogic Integration Domain
- Deploying Liquid Data in a Clustered WebLogic Integration Domain

Note: Due to limitations with WebLogic Integration, do *not* configure WebLogic Integration on a WebLogic domain in a multi-node environment.

Deploying Liquid Data On a Standalone WebLogic Integration Domain

This section describes how to deploy Liquid Data on a standalone (single server) WebLogic Integration domain. For more information about single server deployments, see "Single Server Deployments" on page 2-3.

To deploy Liquid Data on a standalone WebLogic Integration domain:

1. Install the WebLogic Platform, which includes WebLogic Integration, on the target server according to the instructions in "Installing WebLogic Platform" in the WebLogic Platform documentation.

   Note: Do not start up WebLogic Server until you are instructed to do so later in this section.

2. Install the Liquid Data software on the target server according to the instructions in Installing Liquid Data.

   Note: Check the Liquid Data Release Notes for any last-minute deployment instructions.

3. Unless you are deploying onto an existing WebLogic Integration domain, start the WebLogic Platform Configuration Wizard on the target server and create a WebLogic Integration domain (referred to in this section as *wli_domain_name*):

   - For the server type, select Single Server (Standalone Server).
   - Later in this procedure, you use the Administrative user created in this step to configure the domain.

   For instructions, see "Creating a New WebLogic Domain" in *Using the Configuration Wizard* in the WebLogic Platform documentation.

   Note: If you want to deploy Liquid Data on an *existing* WebLogic Integration domain instead of creating a new one, then the existing domain *must* use a WebLogic Server compatibility security realm. For more information, see "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.

4. Edit the server startup file in *BEA_HOME*/user_projects/*wli_domain_name*. You need to specify the Liquid Data home directory (*ld_home_dir*), such as WL_HOME\liquiddata (on Windows) or $WL_HOME/liquiddata (Unix).

   Note: If your deployment will use application views as data sources, you need to add the path to the Application Integration wlai-client.jar file (such as *%WLI_HOME%*\lib\wlai-client.jar) to the *end* of the Liquid Data classpath (LDCLASSPATH).

   - **Windows**: Edit the startweblogic.cmd file. Immediately *before* the line in which WebLogic Server gets started, add the following command lines:

     ```
     set LD_HOME=%BEA_HOME%\weblogic700\liquiddata

     set LDI_LIB=%LD_HOME%\server\lib

     set
     LDCLASSPATH=%LDI_LIB%\wlldi.jar;%LDI_LIB%\LDS-descriptions.j
     ar;%LDI_LIB%\castor-0.9.3.9.jar;%LDI_LIB%\xercesImple.jar

     set SVRCP=%SVRCP%;%LDCLASSPATH%
     ```

   - **Unix**: Edit the startweblogic.sh file. Immediately *before* the line in which WebLogic Server gets started, add the following command lines:

     ```
     LD_HOME=$BEA_HOME/weblogic700/liquiddata

     LDI_LIB=$LD_HOME/server/lib

     LDCLASSPATH=$LDI_LIB/wlldi.jar:$LDI_LIB/LDS-descriptions.jar
     :$LDI_LIB/castor-0.9.3.9.jar:$LDI_LIB/xercesImpl.jar
     ```

```
SVRCP=$SVRCP:$LDCLASSPATH
```

5. Copy the following files from *LD_HOME*/server/lib to *BEA_HOME*/user_projects/*wli_domain_name*.
   - ConfigMap.xml
   - webapp.template
6. Use the WebLogic Integration Database Wizard (wliconfig) to switch to—and populate—the database you want to use for your deployment. For instructions, see "Using the Database Wizard" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing* in the WebLogic Integration documentation.
7. Start WebLogic Server and wait until it is in running mode.
8. Start the Administration Console on the WebLogic Platform domain, logging in with administrator privileges, and configure Liquid Data security according to the following steps:
   a. In the left pane, click on *wli_domain_name*->Compatibility Security->Users and add a new user: ldsystem.
   b. Click on *wli_domain_name*->Compatibility Security->Groups and add a new group: LDAdmin.
   c. Add the ldsystem user to the LDAdmin group.
   d. Add the ldsystem user to the Administrators group.
   e. Click on *wli_domain_name*->Compatibility Security->ACLs and add a new ACL: LD.
   f. Add the following new permission attributes to the LD ACL: execute, read, and modify, and then add the Administrators group to the grantee list of permissions.
9. Verify that the ACL for the weblogic.jdbc.connectionPool resource has the following permissions: admin, reserve, shrink, reset, and modify. The grantee needs to be the Administrators group.
10. Deploy the LDS.ear file according to the following steps:

a. In the left pane, click on *wli_domain_name*->Deployments->Applications, click on Configure a New Application, click on "Upload it through your browser," select *LD_HOME*/server/lib/LDS.ear, and upload it.

b. Click on the Select link next to the LDS.ear file, and then click Upload.

c. Select the target server on which to deploy Liquid Data, and then click Configure and Deploy.

11. Verify that the following files are deployed successfully:
    - ldconsole.war
    - ldcacheListener.war
    - cacheEjb.jar
    - ejb_query.jar
    - XMediator.war
    - ejb_qbc.jar
12. Exit the Administration Console.
13. Start the Administration Console, logging in as ldsystem.
14. Configure data sources, security, the repository, and other Liquid Data server settings, as needed, according to the instructions in the Liquid Data Administration Guide.

Deploying Liquid Data in a Clustered WebLogic Integration Domain

This section describes how to deploy Liquid Data in a clustered WebLogic domain. Clustering is typically used to provide scalability in deployments with high volumes of query requests and to provide fail-over protection in deployments with high availability requirements. For more information about clustered deployments, see "Clustered Deployments" on page 2-6.

Note: Before you begin, verify that your BEA software license is appropriate for your deployment.

To deploy Liquid Data in a clustered WebLogic Integration domain:

1. Install the WebLogic Platform, which includes WebLogic Integration, on the machines that will host the Administration Server and all target Managed Servers, according to the instructions in "Installing WebLogic Platform" in the WebLogic Platform documentation.

   Note: Do not start up WebLogic Server until you are instructed to do so later in this section.

2. Install the Liquid Data software on the machines that will host the Administration Server and all target Managed Servers, according to the instructions in Installing Liquid Data.

   Note: Check the Liquid Data Release Notes for any last-minute deployment instructions.

3. Unless you are deploying onto an existing WebLogic Integration domain, start the WebLogic Platform Configuration Wizard on the target Administration Server and create a WebLogic Integration domain (referred to in this section as `wli_domain_name`):

   - For the server type, select Admin Server with Clustered Managed Server(s).
   - Later in this procedure, you use the Administrative user created in this step to configure the domain.

   For instructions, see "Creating a New WebLogic Domain" in *Using the Configuration Wizard* in the WebLogic Platform documentation.

   Note: If you want to deploy Liquid Data on an *existing* WebLogic Integration domain instead of creating a new one, then the existing domain *must* use a WebLogic Server compatibility security realm. For more information, see "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.

4. Unless you are deploying onto an existing WebLogic Integration domain, start the WebLogic Platform Configuration Wizard on *each* Managed Server in the cluster, create a WebLogic Integration domain with same name as the one created on the Administration Server, and, for the server type, select Managed Server (with Owning Admin Server Configuration).

5. From inside the domain in which the Administration Server was configured, use the WebLogic Integration Database Wizard (`wliconfig`) to switch to the database you want to use to configure the `wlaiPool`. For instructions, see "Using the Database Wizard" in "Customizing WebLogic Integration" in *Starting, Stopping, and Customizing* in the WebLogic Integration documentation.

   Note: If you use any database other than PointBase, then after you exit `wliconfig`, you need to create and load the database. Copy any applicable scripts from the respective database directory to the database server machine, and then run the scripts on the database.

6. On the Administration Server, copy the following files from *`LD_HOME`*`/server/lib` to *`BEA_HOME`*`/user_projects/`*`wli_domain_name`*.

   - `ConfigMap.xml`
   - `webapp.template`

7. Edit the server startup file in *`BEA_HOME`*`/user_projects/`*`wli_domain_name`*. You need to specify the Liquid Data home directory (*`ld_home_dir`*), such as `WL_HOME\liquiddata` (on Windows) or `$WL_HOME/liquiddata` (Unix).

   Note: If your deployment will use application views as data sources, you need to add the path to the Application Integration `wlai-client.jar` file (such as *`%WLI_HOME%`*`\lib\wlai-client.jar`) to the *end* of the Liquid Data classpath (`LDCLASSPATH`).

   - **Windows**: Edit the `startweblogic.cmd` file (on the Administration Server and `startManagedWeblogic.cmd` file (on Managed Servers). Immediately *before* the line in which WebLogic Server gets started, add the following command lines:

     ```
     set LD_HOME=%BEA_HOME%\weblogic700\liquiddata
     set LDI_LIB=%LD_HOME%\server\lib
     set
     LDCLASSPATH=%LDI_LIB%\wlldi.jar;%LDI_LIB%\LDS-descriptions.j
     ar;%LDI_LIB%\castor-0.9.3.9.jar;%LDI_LIB%\xercesImple.jar
     ```

     On the Administration Server, specify the following command:

     ```
     set SVRCP=%SVRCP%;%LDCLASSPATH%
     ```

     On any Managed Servers, specify the following command:

     ```
     set CLASSPATH=%CLASSPATH%;%LDCLASSPATH%
     ```

In the startup script, verify that the following information about the Administration Server is correct:

```
set ADMIN_URL=admin_server_name:admin_server_listening_port
```

- **Unix**: Edit the `startweblogic.sh` file (on the Administration Server and `startManagedWeblogic.sh` file (on Managed Servers). Immediately *before* the line in which WebLogic Server gets started, add the following command lines:

  ```
  LD_HOME=$BEA_HOME/weblogic700/liquiddata

  LDI_LIB=$LD_HOME/server/lib

  LDCLASSPATH=$LDI_LIB/wlldi.jar:$LDI_LIB/LDS-descriptions.jar
  :$LDI_LIB/castor-0.9.3.9.jar:$LDI_LIB/xercesImpl.jar
  ```

  On the Administration Server (only), specify the following command:

  ```
  SVRCP=$SVRCP:$LDCLASSPATH
  ```

  On any Managed Servers (only), specify the following command:

  ```
  CLASSPATH=$CLASSPATH:$LDCLASSPATH
  ```

  Note: The `$LDCLASSPATH` must *follow* the current `CLASSPATH` setting.

  In the startup script, verify that the `JAVA_OPTIONS` environment variable has the following entry with the correct information about the Administration Server: "
  `-Dweblogic.management.server=admin_server_hostname:admin_server_listening_port`"

8. Start the Administration Server and wait until it is in running mode.
9. On the Administration Server, start the Administration Console on the WebLogic Platform domain, logging in with the username/password you had created when you configured the domain, and configure Liquid Data security according to the following steps:
   a. In the left pane, click on *`platform_domain_name`*->Compatibility Security->Users add a new user: `ldsystem`.
   b. Click on *`platform_domain_name`*->Compatibility Security->Groups and add a new group: `LDAdmin`.
   c. Add the `ldsystem` user to the `LDAdmin` group.
   d. Add the `ldsystem` user to the `Administrators` group.

e. Click on *platform_domain_name*->Compatibility Security->ACLs and add a new ACL: `LD`.

f. Add the following new permission attributes to the `LD` ACL: `execute`, `read`, and `modify`, and then add the `Administrators` group to the grantee list of permissions.

10. By default, WebLogic Server uses the round-robin algorithm as the default load balancing strategy for clustered object stubs. If you want to use weight-based or random load balancing for EJBs and RMI objects instead, complete the following steps:

    a. In the left pane of the Administration Console, select the Clusters node.

    b. Select your cluster.

    c. Click the drop-down arrow next to the Default Load Algorithm to display load balancing algorithms.

    d. In the item list, select the load balancing algorithm you want to use in the cluster.

    e. Enter the desired value in the Service Age Threshold field (see the Administration Console Online Help for more information).

    f. Click Apply to save your changes.

    For more information about load balancing algorithms, see "Load Balancing for EJBs and RMI Objects" in "Load Balancing in a Cluster" in *Using WebLogic Server Clusters* in the WebLogic Server documentation.

11. Verify that the ACL for the `weblogic.jdbc.connectionPool` resource has the following permissions: `admin`, `reserve`, `shrink`, `reset`, and `modify`. The grantee needs to be the `Administrators` group.

12. Shut down the Administration Server, start it up again, and wait until it is in running mode.

13. Start all Managed Servers and wait until they are all in running mode.

14. Deploy the `LDS.ear` file according to the following steps:

a. In the left pane, click on *wli_domain_name*->Deployments->Applications, click on Configure a New Application, click on "Upload it through your browser," select *LD_HOME*/server/lib/LDS.ear, and upload it.

b. Click on the Select link next to the LDS.ear file, and then click Upload.

c. Select the <*LD_cluster_name*> from the list of Available Clusters to the Target Clusters, and then click Configure and Deploy.

15. Verify that the following files are deployed successfully:
    - ldconsole.war
    - ldcacheListener.war
    - cacheEjb.jar
    - ejb_query.jar
    - XMediator.war
    - ejb_qbc.jar
16. In the left pane, click on *wli_domain_name*->Deployments->Applications->LDS->ldconsole.war, click the Deploy tab, click the Targets tab, select the Administration Server from the list of available servers, and then click Apply.
17. Exit the Administration Console.
18. Start the Administration Console, logging in as ldsystem.
19. In the left pane, click on the Liquid Data node, then click the General tab. The default repository directory is ldrepository in the *BEA_HOME*/user_projects/*wli_domain_name* directory.

    You need to change the repository directory path so that it points to a shared volume to which all Managed Servers have access. On Windows, for example, you would point to a shared network drive mapped to a specific drive letter (such as R:\ldrepos).
20. On each Managed Server, you need configure the repository location by mounting (Unix NFS) or mapping (Windows) logical drives to point to the configured repository root on the Administration Server. The directory must be

shared and mapped to a virtual disk name. The path must be identical to the value specified in the Repository Root Directory field on the General tab in the Liquid Data node.

- On Unix, use NFS to mount the exported directory in the respective *BEA_HOME*/user_projects/*wli_domain_name*. Make a short cut of the mapped drive in the managed domain and name it the same as in the admin domain.
- On Windows, map to the drive using the exact same drive letter and path (such as R:\ldrepos).

21. Configure data sources, security, and other Liquid Data server settings, as needed, according to the instructions in the Liquid Data Administration Guide.
22. Test the clustered deployment by verifying the operation of each machine—individually and in combination with other machines. Under a given query request load, you should see performance increase as you bring additional Managed Servers online.

Deploying Liquid Data in a WebLogic Server Domain

The following sections describe how to deploy Liquid Data in a WebLogic Server domain:

Deploying Liquid Data in a Standalone WebLogic Server Domain

This section describes how to deploy Liquid Data in a standalone (single server) WebLogic Server domain. For more information about single server deployments, see "Single Server Deployments" on page 2-3.

To deploy Liquid Data in a standalone WebLogic Server domain:

1. Install the WebLogic Platform, which includes WebLogic Server, on the target server according to the instructions in "Installing WebLogic Platform" in the WebLogic Platform documentation.

   Note: Do not start up WebLogic Server until you are instructed to do so later in this section.

2. Install the Liquid Data software on the target server according to the instructions in Installing Liquid Data.

   Note: Check the Liquid Data Release Notes for any last-minute deployment instructions.

3. Unless you are deploying onto an existing WebLogic Server domain, start the WebLogic Platform Configuration Wizard on the target server and create a WebLogic Server domain (referred to in this section as *`wls_domain_name`*):

   - For the server type, select Single Server (Standalone Server).
   - Later in this procedure, you use the Administrative user created in this step to configure the domain.

   For instructions, see "Creating a New WebLogic Domain" in *Using the Configuration Wizard* in the WebLogic Platform documentation.

   Note: If you want to deploy Liquid Data on an *existing* WebLogic Server domain instead of creating a new one, then the existing domain *must* use a WebLogic Server compatibility security realm. For more information, see "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.

4. Edit the `config.xml` file in *`BEA_HOME`*`/user_projects/`*`wls_domain_name`* and make the following changes:

- Modify the `<Security>` element so that it includes all of the following attribute settings:

  ```
  <Security CompatibilityMode="true" GuestDisabled="false"
  Name="wls_domain_name" Realm="myRealm" RealmSetup="true"/>
  ```

  Be sure to set `CompatibilityMode` to `true` and to specify the domain name in *`wls_domain_name`*.

- Add the following elements and attributes:

  ```
  <Realm FileRealm="myFileRealm" Name="myRealm"/>
  ```

  ```
  <FileRealm Name="myFileRealm"/>
  ```

  This example assumes that you are using a file realm for the compatibility security realm. For other types of security realms, see "Specifying a Security Realm" in "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.

- Modify the `<SSL>` element and replace or add the following attributes and settings:

  ```
  ServerCertificateFileName="democert.pem"
  ```

  ```
  ServerKeyFileName="demokey.pem" TrustedCAFileName="ca.pem"
  ```

- Remove the following attributes from the `<SSL>` section:

  ```
  ServerPrivateKeyAlias="demokey"
  ```

  ```
  ServerPrivateKeyPassPhrase="WhatEverHere"
  ```

5. Copy the following files from *`LD_HOME`*`/server/lib` to *`BEA_HOME`*`/user_projects/`*`wls_domain_name`*.

   - `ConfigMap.xml`
   - `webapp.template`

6. Edit the server startup file in *`BEA_HOME`*`/user_projects/`*`wls_domain_name`*. You need to specify the Liquid Data home directory (*`ld_home_dir`*), such as `WL_HOME\liquiddata` (on Windows) or `$WL_HOME/liquiddata` (Unix).

   Note: If your deployment will use application views as data sources, you need to add the path to the Application Integration `wlai-client.jar` file (such as *`%WLI_HOME%`*`\lib\wlai-client.jar`) to the *end* of the Liquid Data classpath (`LDCLASSPATH`).

   - **Windows**: Edit the `startweblogic.cmd` file and add the following command lines *before* `startWLS.cmd` executes:

```
set LD_HOME=ld_home_dir

set LDI_LIB=%LD_HOME%\server\lib

set
LDCLASSPATH=%LDI_LIB%\wlldi.jar;%LDI_LIB%\LDS-descriptions.j
ar;%LDI_LIB%\castor-0.9.3.9.jar;%LDI_LIB%\xercesImpl.jar;

set CLASSPATH=%CLASSPATH%;%LDCLASSPATH%
```

- **Unix**: Edit the `startweblogic.sh` file and add the following command lines *before* `startWLS.sh` executes:

```
LD_HOME=ld_home_dir

LDI_LIB=$LD_HOME/server/lib

LDCLASSPATH=$LDI_LIB/wlldi.jar:$LDI_LIB/LDS-descriptions.jar
:$LDI_LIB/castor-0.9.3.9.jar:$LDI_LIB/xercesImpl.jar

CLASSPATH=$CLASSPATH:$LDCLASSPATH

export CLASSPATH
```

  Note: The `$LDCLASSPATH` must *follow* the current `CLASSPATH` setting.

7. Start WebLogic Server and wait until it is in running mode.
8. Start the Administration Console on the WebLogic Server domain, logging in with the username / password you created when you configured the domain, and configure Liquid Data security according to the following steps:
   a. In the left pane, click on *wls_domain_name*->Compatibility Security->Users and add a new user: `ldsystem`.
   b. Click on *wls_domain_name*->Compatibility Security->Groups and add a new group: `LDAdmin`.
   c. Add the `ldsystem` user to the `LDAdmin` group.
   d. Add the `ldsystem` user to the `Administrators` group.
   e. Click on *wls_domain_name*->Compatibility Security->ACLs and add a new ACL: `LD`.
   f. Add the following new permission attributes to the `LD` ACL: `execute`, `read`, and `modify`, and then add the `Administrators` group to the grantee list of permissions.
9. Deploy the `LDS.ear` file according to the following steps:

a. In the left pane, click on *wls_domain_name*->Deployments->Applications, click on Configure a New Application, click on "Upload it through your browser," select *LD_HOME*/server/lib/LDS.ear, and upload it.

b. Click on the Select link next to the LDS.ear file, and then click Upload.

c. Select the server on which to deploy Liquid Data, and then click Configure and Deploy.

10. Verify that the following files are deployed successfully:
    - ldconsole.war
    - ldcacheListener.war
    - cacheEjb.jar
    - ejb_query.jar
    - XMediator.war
    - ejb_qbc.jar
11. Exit the Administration Console.
12. Start the Administration Console, logging in as ldsystem.
13. Configure data sources, security, the repository, and other Liquid Data server settings, as needed, according to the instructions in the Liquid Data Administration Guide.

Deploying Liquid Data in a Clustered WebLogic Server Domain

This section describes how to deploy Liquid Data in a clustered WebLogic Server domain. Clustering is typically used to provide scalability in deployments with high volumes of query requests and to provide fail-over protection in deployments with high availability requirements. For more information about clustered deployments, see "Clustered Deployments" on page 2-6.

Note: Before you begin, verify that your BEA software license is appropriate for your deployment.

To deploy Liquid Data in a clustered WebLogic Server domain:

1. Install the WebLogic Platform, which includes WebLogic Server, on the machines that will host the Administration Server and all target Managed Servers, according to the instructions in "Installing WebLogic Platform" in the WebLogic Platform documentation.

   Note: Do not start up WebLogic Server until you are instructed to do so later in this section.

2. Install the Liquid Data software on the machines that will host the Administration Server and all target Managed Servers, according to the instructions in Installing Liquid Data.

   Note: Check the Liquid Data Release Notes for any last-minute deployment instructions.

3. Unless you are deploying onto an existing WebLogic Server domain, start the WebLogic Platform Configuration Wizard on the target server and create a WebLogic Server domain (referred to in this section as *`wls_domain_name`*):

   - For the server type, select Admin Server with Clustered Managed Server(s).
   - Later in this procedure, you use the Administrative user created in this step to configure the domain.

   For instructions, see "Creating a New WebLogic Domain" in *Using the Configuration Wizard* in the WebLogic Platform documentation.

   Note: If you want to deploy Liquid Data on an *existing* WebLogic Server domain instead of creating a new one, then the existing domain *must* use a WebLogic Server compatibility security realm. For more information, see "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.

4. Unless you are deploying onto an existing WebLogic Server domain, start the WebLogic Platform Configuration Wizard on *each* Managed Server in the cluster, create a WebLogic Server domain with same name as the one created on the Administration Server, and, for the server type, select Managed Server (with Owning Admin Server Configuration).

5. Edit the `config.xml` file in *`BEA_HOME`*`/user_projects/`*`wls_domain_name`* and make the following changes:

   - Modify the `<Security>` element so that it includes all of the following attribute settings:

```
<Security CompatibilityMode="true" GuestDisabled="false"
Name="wls_domain_name" Realm="myRealm" RealmSetup="true"/>
```

Be sure to set `CompatibilityMode` to `true` and to specify the domain name in *`wls_domain_name`*.

- Add the following elements and attributes:

  ```
  <Realm FileRealm="myFileRealm" Name="myRealm"/>
  ```

  ```
  <FileRealm Name="myFileRealm"/>
  ```

  This example assumes that you are using a file realm for the compatibility security realm. For other types of security realms, see "Specifying a Security Realm" in "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.

- Modify the `<SSL>` element and replace or add the following attributes and settings:

  ```
  ServerCertificateFileName="democert.pem"
  ```

  ```
  ServerKeyFileName="demokey.pem" TrustedCAFileName="ca.pem"
  ```

- Remove the following attributes from the `<SSL>` section:

  ```
  ServerPrivateKeyAlias="demokey"
  ```

  ```
  ServerPrivateKeyPassPhrase="WhatEverHere"
  ```

6. Edit the server startup files in *`BEA_HOME`*`/user_projects/`*`wls_domain_name`*. You need to specify the Liquid Data home directory (*`ld_home_dir`*), such as `WL_HOME\liquiddata` (on Windows) or `$WL_HOME/liquiddata` (Unix).

   Note: If your deployment will use application views as data sources, you need to add the path to the Application Integration `wlai-client.jar` file (such as *`%WLI_HOME%`*`\lib\wlai-client.jar`) to the *end* of the Liquid Data classpath (`LDCLASSPATH`).

   - **Windows**: Edit the `startweblogic.cmd` file (on the Administration Server and `startManagedWeblogic.cmd` file (on Managed Servers) and add the following command lines *before* `startWLS.cmd` executes:

     ```
     set LD_HOME=ld_home_dir
     ```

     ```
     set LD_LIB=%LD_HOME%\server\lib
     ```

     ```
     set
     LDCLASSPATH=%LD_LIB%\wlldi.jar;%LD_LIB%\LDS-descriptions,jar
     ;%LD_LIB%\castor-0.9.3.9.jar;%LD_LIB%\xercesImpl.jar
     ```

     ```
     set CLASSPATH=%CLASSPATH%;%LDCLASSPATH%
     ```

- **Unix**: Edit the `startweblogic.sh` file (on the Administration Server and `startManagedWeblogic.sh` file (on Managed Servers) and add the following command lines *before* `startWLS.sh` executes:

  ```
  LD_HOME=ld_home_dir

  LDI_LIB=$LD_HOME/server/lib

  LDCLASSPATH=$LDI_LIB/wlldi.jar:$LDI_LIB/LDS-descriptions.jar
  :$LDI_LIB/castor-0.9.3.9.jar:$LDI_LIB/xercesImpl.jar

  CLASSPATH=$CLASSPATH:$LDCLASSPATH

  export CLASSPATH
  ```

  Note: The `$LDCLASSPATH` must *follow* the current `CLASSPATH` setting.

7. On the Administration Server, copy the following files from *`LD_HOME`*`/server/lib` to *`BEA_HOME`*`/user_projects/`*`platform_domain_name`*.
   - `ConfigMap.xml`
   - `webapp.template`
8. On the Administration Server, copy the following files from *`LD_HOME`*`/server/lib` to *`BEA_HOME`*`/user_projects/`*`wls_domain_name`*.
   - `ConfigMap.xml`
   - `webapp.template`
9. Start the Administration Server and wait until it is in running mode.
10. On the Administration Server, start the Administration Console on the WebLogic domain, logging in as an administrator, and configure Liquid Data security according to the following steps:
    a. In the left pane, click on *`wls_domain_name`*->Compatibility Security->Users and add a new user: `ldsystem`.
    b. Click on *`wls_domain_name`*->Compatibility Security->Groups and add a new group: `LDAdmin`.
    c. Add the `ldsystem` user to the `LDAdmin` group.
    d. Add the `ldsystem` user to the `Administrators` group.
    e. Click on *`wls_domain_name`*->Compatibility Security->ACLs and add a new ACL: `LD`.

f. Add the following new permission attributes to the `LD` ACL: `execute`, `read`, and `modify`, and then add the `Administrators` group to the grantee list of permissions.

11. By default, WebLogic Server uses the round-robin algorithm as the default load balancing strategy for clustered object stubs. If you want to use weight-based or random load balancing for EJBs and RMI objects instead, complete the following steps:

    a. In the left pane of the Administration Console, select the Clusters node.

    b. Select your cluster.

    c. Click the drop-down arrow next to the Default Load Algorithm to display load balancing algorithms.

    d. In the item list, select the load balancing algorithm you want to use in the cluster.

    e. Enter the desired value in the Service Age Threshold field (see the Administration Console Online Help for more information).

    f. Click Apply to save your changes.

    For more information about load balancing algorithms, see "Load Balancing for EJBs and RMI Objects" in "Load Balancing in a Cluster" in *Using WebLogic Server Clusters* in the WebLogic Server documentation.

12. Verify that the ACL for the `weblogic.jdbc.connectionPool` resource has the following permissions: `admin`, `reserve`, `shrink`, `reset`, and `modify`. The grantee needs to be the `Administrators` group.

13. Shut down the Administration Server, start it up again, and wait until it is in running mode.

14. Start all Managed Servers and wait until they are all in running mode.

15. Deploy the `LDS.ear` file according to the following steps:

    a. In the left pane, click on *`wls_domain_name`*`->Deployments->Applications`, click on Configure a New Application, click on "Upload it through your browser," select *`LD_HOME`*`/server/lib/LDS.ear`, and upload it.

    b. Click on the Select link next to the `LDS.ear` file, and then click Upload.

c. Select the <*LD_cluster_name*> from the list of Available Clusters to the Target Clusters, and then click Configure and Deploy.

16. Verify that the following files are deployed successfully:
    - `ldconsole.war`
    - `ldcacheListener.war`
    - `cacheEjb.jar`
    - `ejb_query.jar`
    - `XMediator.war`
    - `ejb_qbc.jar`
17. In the left pane, click on *`wls_domain_name`*`->Deployments->Applications->LDS->ldconsole.war`, click the Targets tab, click servers, select the Administration Server from the list of available servers, and then click Apply.
18. Click on the Cluster tab, click on the cluster to move it to the left, and then click Apply.
19. Click on the Deploy tab, click the Undeploy button to undeploy `ldconsole.war` from the cluster, and then click Deploy to deploy it to the Administration Server.

    The Liquid Data node appears at the bottom of the left pane.
20. Exit the Administration Console.
21. Start the Administration Console, logging in as `ldsystem`.
22. In the left pane, click on the Liquid Data node, then click the General tab. The default repository directory is `ldrepository` in the *`BEA_HOME`*`/user_projects/`*`wls_domain_name`* directory.

    You need to change the repository directory path so that it points to a shared volume to which all Managed Servers have access. On Windows, for example, you would point to a shared network drive mapped to a specific drive letter (such as `R:\ldrepos`).
23. On each Managed Server, you need configure the repository location by mounting (Unix NFS) or mapping (Windows) logical drives to point to the configured repository root on the Administration Server. The directory must be

shared and mapped to a virtual disk name. The path must be identical to the value specified in the Repository Root Directory field on the General tab in the Liquid Data node.

- On Unix, use NFS to mount the exported directory in the respective *BEA_HOME*/user_projects/*wls_domain_name*. Make a short cut of the mapped drive in the managed domain and name it the same as in the admin domain.
- On Windows, map to the drive using the exact same drive letter and path (such as R:\ldrepos).

24. Configure data sources, security, and other Liquid Data server settings, as needed, according to the instructions in the Liquid Data Administration Guide.
25. Test the clustered deployment by verifying the operation of each machine—individually and in combination with other machines. Under a given query request load, you should see performance increase as you bring additional Managed Servers online.

Deploying Liquid Data in a WebLogic Portal Domain

The following topics describe how to deploy Liquid Data with WebLogic Portal:

- Deploying Liquid Data in a Standalone WebLogic Portal Domain
- Deploying Liquid Data and WebLogic Portal in Separate Domains

For general information about managing WebLogic Portal in a production environment, see "System Administration" in the WebLogic Portal *Administration Guide*.

Deploying Liquid Data in a Standalone WebLogic Portal Domain

This section describes how to deploy Liquid Data in a standalone (single server) WebLogic Portal domain. For more information about single server deployments, see "Single Server Deployments" on page 2-3.

To deploy Liquid Data in a standalone WebLogic Portal domain:

1. Install the WebLogic Platform, which includes WebLogic Server, on the target server according to the instructions in "Installing WebLogic Platform" in the WebLogic Platform documentation.

   Note: Do not start up WebLogic Server until you are instructed to do so later in this section.

2. Install the Liquid Data software on the target server according to the instructions in Installing Liquid Data.

   Note: Check the Liquid Data Release Notes for any last-minute deployment instructions.

3. Unless you are deploying onto an existing WebLogic Portal domain, start the WebLogic Platform Configuration Wizard on the target server and create a WebLogic Portal domain (referred to in this section as `portal_domain_name`):

   - For the server type, select Single Server (Standalone Server).
   - Later in this procedure, you use the Administrative user created in this step to configure the domain.

   For instructions, see "Developing Portals—Tutorials" in the *Development Guide* in the WebLogic Portal documentation.

   Note: If you want to deploy Liquid Data on an *existing* WebLogic Portal domain instead of creating a new one, then the existing domain *must* use a WebLogic Server compatibility security realm. For more information, see "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.

4. Edit the Portal server startup file in the `portal_domain_name` directory. You need to specify the Liquid Data home directory (`ld_home_dir`).

Note: If your deployment will use application views as data sources, you need to add the path to the Application Integration `wlai-client.jar` file (such as *`%WLI_HOME%`*`\lib\wlai-client.jar`) to the *end* of the Liquid Data classpath (`LDCLASSPATH`).

- **Windows**: Edit the `startPortal.bat` file and, at the end of the section in which environment variables are set, add the following commands:

```
set LD_HOME=ld_home_dir

set LDI_LIB=%LD_HOME%\server\lib

set
LDCLASSPATH=%LDI_LIB%\wlldi.jar;%LDI_LIB%\LDS-descriptions.j
ar;%LDI_LIB%\castor-0.9.3.9.jar;%LDI_LIB%\xercesImpl.jar;

set CLASSPATH=%CLASSPATH%;%LDCLASSPATH%
```

- **Unix**: Edit the `startPortal.sh` file and, at the end of the section in which environment variables are set, add the following commands:

```
LD_HOME=ld_home_dir

LDI_LIB=$LD_HOME/server/lib

LDCLASSPATH=$LDI_LIB/wlldi.jar:$LDI_LIB/LDS-descriptions.jar
:$LDI_LIB/castor-0.9.3.9.jar:$LDI_LIB/xercesImpl.jar

CLASSPATH=$CLASSPATH:$LDCLASSPATH

export CLASSPATH
```

  Note: The `$LDCLASSPATH` must *follow* the current `CLASSPATH` setting.

5. Copy the following files from *`LD_HOME`*`/server/lib` to *`BEA_HOME`*`/user_projects/`*`platform_domain_name`*.
   - `ConfigMap.xml`
   - `webapp.template`
6. Start WebLogic Server and WebLogic Portal.
7. Start the Administration Console on the WebLogic Portal domain, logging in as Administrator, and configure Liquid Data security according to the following steps:
   a. In the left pane, click on *`portal_domain_name`*->Compatibility Security->Users and add a new user: `ldsystem`.
   b. Click on *`portal_domain_name`*->Compatibility Security->Groups and add a new group: `LDAdmin`.

c. Add the `ldsystem` user to the `LDAdmin` group.

d. Add the `ldsystem` user to the `Administrators` group.

e. Click on *`portal_domain_name`*->Compatibility Security->ACLs and add a new ACL: `LD`.

f. Add the following new permission attributes to the `LD` ACL: `execute`, `read`, and `modify`, and then add the `Administrators` group to the grantee list of permissions.

8. Verify that the ACL for the `weblogic.jdbc.connectionPool` resource has the following permissions: `admin`, `reserve`, `shrink`, `reset`, and `modify`. The grantee needs to be the `Administrators` group.

9. Deploy the `LDS.ear` file according to the following steps:

   a. In the left pane, click on *`portal_domain_name`*`->Deployments->Applications`, click on Configure a New Application, click on "Upload it through your browser," select *`LD_HOME`*`/server/lib/LDS.ear`, and upload it.

   b. Click on the Select link next to the `LDS.ear` file, and then click Upload.

   c. Select the WebLogic Portal server on which to deploy Liquid Data, and then click Configure and Deploy.

10. Verify that the following files are deployed successfully:
    - `ldconsole.war`
    - `ldcacheListener.war`
    - `cacheEjb.jar`
    - `ejb_query.jar`
    - `XMediator.war`
    - `ejb_qbc.jar`

11. Exit the Administration Console.

12. Start the Administration Console, logging in as `ldsystem`.

13. Configure data sources, security, the repository, and other Liquid Data server settings, as needed, according to the instructions in the Liquid Data Administration Guide.

Deploying Liquid Data and WebLogic Portal in Separate Domains

To deploy Liquid Data and WebLogic Portal in separate domains:

1. Create two separate domains: the WebLogic Portal domain and the WebLogic domain in which Liquid Data runs.
2. From the Liquid Data server, copy the `LDS-client.jar` and `LDS-taglib.jar` files from *`LD_HOME`*`/server/lib` to *`portal_domain`*`/beaApps/portalApp/`*`PortalWebApp`*`/WEB-INF/lib` on the WebLogic Portal server.
3. On the WebLogic Portal server, add the following code to the `<!-- Portal tag libraries -->` section in *`portal_domain`*`/beaApps/portalApp/`*`PortalWebApp`*`/WEB-INF/web.xml`:

```
<taglib>
   <taglib-uri>LDSTLD</taglib-uri>
   <taglib-location>/WEB-INF/lib/LDS-taglib.jar</taglib-locatio
n>
</taglib>
```

Note: When Liquid Data and the WebLogic Platform are deployed in separate domains, use the Liquid Data EJB API instead of the Liquid Data tag library. The Liquid Data tag library does not support multi-domain security. For more information, see "Invoking Queries in EJB Clients" in *Invoking Queries Programmatically*.

Deploying Liquid Data in a Standalone WebLogic Workshop Domain

This section describes how to deploy Liquid Data in a standalone (single server) WebLogic Workshop domain. For more information about single server deployments, see "Single Server Deployments" on page 2-3.

To deploy Liquid Data in a WebLogic Workshop domain:

1. Install the WebLogic Platform, which includes WebLogic Workshop, on the target server according to the instructions in "Installing WebLogic Platform" in the WebLogic Platform documentation.

   Note: Do not start up WebLogic Server until you are instructed to do so later in this section.

2. Install the Liquid Data software on the target server according to the instructions in Installing Liquid Data.

   Note: Check the Liquid Data Release Notes for any last-minute deployment instructions.

3. Unless you are deploying onto an existing WebLogic Workshop domain, start the WebLogic Platform Configuration Wizard on the target server and create a WebLogic Workshop domain (referred to in this section as *workshop_domain_name*):

   - For the server type, select Single Server (Standalone Server).
   - Later in this procedure, you use the Administrative user created in this step to configure the domain.

   For instructions, see "Creating a New WebLogic Domain" in *Using the Configuration Wizard* in the WebLogic Platform documentation.

   Note: If you want to deploy Liquid Data on an *existing* WebLogic Workshop domain instead of creating a new one, then the existing domain *must* use a WebLogic Server compatibility security realm. For more information, see "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.

4. Edit the `config.xml` file in *BEA_HOME*`/user_projects/`*workshop_domain_name* and add the following lines inside the `<domain>` tag:

```
<Realm FileRealm="myFileRealm" Name="myRealm"/>

<FileRealm Name="myFileRealm"/>

<Security CompatibilityMode="true" GuestDisabled="false"
Name="<workshop_domain_name>" Realm="myRealm"
RealmSetup="true"/>
```

5. Edit the server startup file in *BEA_HOME*/user_projects/*workshop_domain_name*. You need to specify the Liquid Data home directory (*ld_home_dir*), such as WL_HOME\liquiddata (on Windows) or $WL_HOME/liquiddata (Unix).

   Note: If your deployment will use application views as data sources, you need to add the path to the Application Integration wlai-client.jar file (such as *%WLI_HOME%*\lib\wlai-client.jar) to the *end* of the Liquid Data classpath (LDCLASSPATH).

   - **Windows**: Edit the startweblogic.cmd file and add the following command lines *before* startWLS.cmd executes:

     ```
     set LD_HOME=ld_home_dir

     set LDI_LIB=%LD_HOME%\server\lib

     set
     LDCLASSPATH=%LDI_LIB%\wlldi.jar;%LDI_LIB%\LDS-descriptions.j
     ar;%LDI_LIB%\castor-0.9.3.9.jar;%LDI_LIB%\xercesImpl.jar;

     set CLASSPATH=%CLASSPATH%;%LDCLASSPATH%
     ```

   - **Unix**: Edit the startweblogic.sh file and add the following command lines *before* startWLS.sh executes:

     ```
     LD_HOME=ld_home_dir

     LDI_LIB=$LD_HOME/server/lib

     LDCLASSPATH=$LDI_LIB/wlldi.jar:$LDI_LIB/LDS-descriptions.jar
     :$LDI_LIB/castor-0.9.3.9.jar:$LDI_LIB/xercesImpl.jar

     CLASSPATH=$CLASSPATH:$LDCLASSPATH

     export CLASSPATH
     ```

     Note: The $LDCLASSPATH must *follow* the current CLASSPATH setting.

6. Make a backup copy of the fileRealm.properties file. Edit the fileRealm.properties file and add the following commands at the end of the file, specifying the *domain_user_name* you created when you configured the domain:

   ```
   group.Administrators=system,domain_user_name

   acl.lookup.weblogic.jndi=everyone

   acl.modify.weblogic.jndi.weblogic.fileSystem=Administrators

   acl.admin.weblogic.jdbc=Administrators

   acl.lookup.weblogic.jndi.weblogic.fileSystem=Administrators
   ```

```
acl.lookup.weblogic.jndi.weblogic.ejb=Administrators
acl.list.weblogic.jndi.weblogic.rmi=Administrators
acl.shrink.weblogic.jdbc=Administrators
acl.create.weblogic.jms.ConnectionConsumer=Administrators
acl.lockServer.weblogic.admin=Administrators
acl.shutdown.weblogic.admin=Administrators
acl.boot.weblogic.server=Administrators
acl.list.weblogic.jndi.weblogic.fileSystem=Administrators
acl.lookup.weblogic.jndi.weblogic=Administrators
acl.modify.weblogic.jndi.weblogic.rmi=Administrators
acl.modify.weblogic.jndi.weblogic=Administrators
acl.list.weblogic.jndi.weblogic=Administrators
acl.send.weblogic.jms=Administrators
acl.lookup.weblogic.management=Administrators
acl.receive.weblogic.jms=Administrators
acl.reserve.weblogic.jdbc=Administrators
acl.modify.weblogic.management=Administrators
acl.lookup.weblogic.jndi.weblogic.rmi=Administrators
acl.list.weblogic.jndi=Administrators
acl.modify.weblogic.jdbc=Administrators
acl.modify.weblogic.admin.acl=Administrators
acl.list.weblogic.jndi.weblogic.ejb=Administrators
acl.modify.weblogic.jndi=Administrators
acl.write.managedObject=Administrators
acl.reset.weblogic.jdbc=Administrators
acl.read.managedObject=Administrators
acl.admin.weblogic.jdbc.connectionPoolcreate=Administrators
acl.unlockServer.weblogic.admin=Administrators
acl.execute.weblogic.servlet=Administrators
acl.modify.weblogic.jndi.weblogic.ejb=Administrators
```

7. Copy the following files from *LD_HOME*/server/lib to *BEA_HOME*/user_projects/*platform_domain_name*.
   - ConfigMap.xml
   - webapp.template
8. Start WebLogic Server and wait until it is in running mode.
9. Start the Administration Console on the WebLogic Workshop domain, logging in as platform/platform, and configure Liquid Data security according to the following steps:
   a. In the left pane, click on *workshop_domain_name*->Compatibility Security->Users and add a new user: ldsystem.
   b. Click on *workshop_domain_name*->Compatibility Security->Groups and add a new group: LDAdmin.
   c. Add the ldsystem user to the LDAdmin group.
   d. Add the ldsystem user to the Administrators group.
   e. Click on *workshop_domain_name*->Compatibility Security->ACLs and add a new ACL: LD.
   f. Add the following new permission attributes to the LD ACL: execute, read, and modify, and then add the Administrators group to the grantee list of permissions.
10. Deploy the LDS.ear file according to the following steps:
    a. In the left pane, click on *workshop_domain_name*->Deployments->Applications, click on Configure a New Application, click on "Upload it through your browser," select *LD_HOME*/server/lib/LDS.ear, and upload it.
    b. Click on the Select link next to the LDS.ear file, and then click Upload.
    c. Select the WebLogic Workshop server on which to deploy Liquid Data, and then click Configure and Deploy.
11. Exit the Administration Console.
12. Start the Administration Console, logging in as ldsystem.

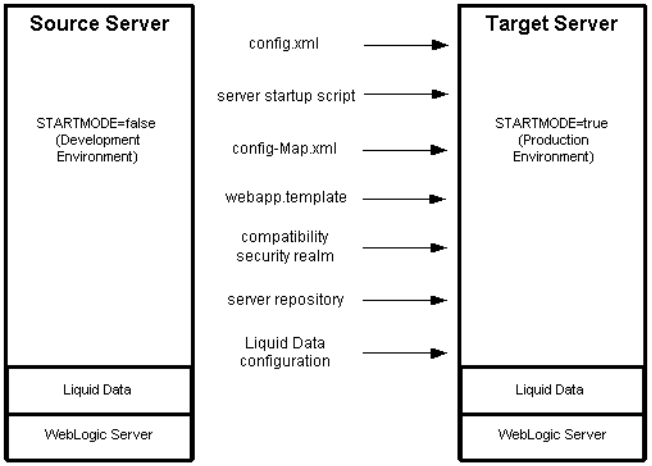
13. Configure data sources, security, the repository, and other Liquid Data server settings, as needed, according to the instructions in the Liquid Data Administration Guide.

Copying a Server Configuration to Another Server

This topic describes the steps involved in copying Liquid Data server configuration information from one server to another, such as from a development environment to a production environment. Rather than configuring the production server manually, you can transfer settings already configured in the development environment.

The following illustration shows the Liquid Data components that you need to transfer to the production server:

Figure 2-4 Migrating From a Development to a Production Server



To migrate from a development environment to a production environment, you need to migrate the following configuration information:

Table 2-1 Liquid Data Configuration Information to Migrate

Problem / Symptom	Description
`config.xml` file	Contains the WebLogic Server configuration, including the domain configuration and, if applicable, the JDBC database configuration. Open the `config.xml` file on the target server and edit any machine-specific parameters.
startup script (`startWebLogic.cmd` or `startWebLogic.sh`)	Contains the Liquid Data server startup script. Open the startup script on the target server and edit any machine-specific parameters. Also, for a production environment, change the `STARTMODE` variable to `true`.
`configMap.xml` and `webapp.template` files	Copy from the *`WL_HOME`*`/user_projects/`*`domain_name`* directory on the source server to the corresponding directory on the target server.
security realm	Replicate the security realm configuration on the production server. The Liquid Data security configuration includes the compatibility security realm, groups, users, and ACL configurations. How you copy the configuration depends on the type of compatibility security realm that you have set up according to the instructions in "Defining a Compatibility Security Realm" in "Implementing Security" in the Liquid Data *Administration Guide*. In general, copy the `filerealm.properties` file to the target machine and, if you have used any other type of compatibility security realm, use the appropriate import/export utility. For example, for an LDAP realm, use the LDAP import/export features. For more information, see "Specifying a Security Realm" in "Using Compatibility Security" in *Managing WebLogic Security* in the WebLogic Server documentation.
server repository	On the development server, create a compressed image of the complete server repository, including all sub-folders, using a TAR or ZIP file for the Unix and Windows platforms, respectively. Copy this compressed file to the production server and expand it. On the production server, start the Administration Console and configure the repository root directory on the General tab in the Liquid Data node.
Liquid Data configuration settings	On the development server, export the Liquid Data configuration information to an export file. On the production server, import the contents of the export file. For instructions, see "Importing and Exporting Liquid Data Configurations" in the Liquid Data *Administration Guide*.

CHAPTER

3 Tuning Performance

This topic describes how to tune performance in a BEA Liquid Data for WebLogic™ deployment. It includes the following sections:

- Where to Begin
- Liquid Data Performance Factors
- Monitoring Liquid Data Performance

In a production environment, Liquid Data performance is generally measured by the speed with which queries are processed and results are returned. This topic describes general guidelines for performance and, where possible, provides specific guidelines for tuning a Liquid Data deployment.

In addition to the material described in this topic, you can refer to the following documentation to tune WebLogic Server and WebLogic Integration performance:

- BEA WebLogic Server Performance and Tuning in the BEA WebLogic Server™ documentation
- "Tuning Performance" in *Deploying Solutions* in the BEA WebLogic Integration™ documentation
- "Performance Tuning" in "System Administration" in the BEA WebLogic Portal™ *Administration Guide*.

Where to Begin

This topic describes where to begin tuning a Liquid Data deployment. It contains the following sections:

- Checking the System Configuration
- Tuning Queries

Checking the System Configuration

Before you begin to investigate how to tune Liquid Data performance, make sure that the system on which Liquid Data runs is reasonably configured.

- System platform configuration—Do you have sufficient CPU, memory, and disk space resources? A properly tuned heap size? The right Java Virtual Machine (JVM)? Reasonable network speed? For more information, see "Platform Performance Factors" on page 3-10.
- Data sources configuration—Are data sources properly tuned? For example, if the deployment uses Relational Database data sources, do the tables in those data sources have adequate indexes? Is the JDBC connection pool size reasonably configured? If the deployment uses Web services data sources, is the Web service available and does it provide a reasonable response time? For more information, see "Data Source Performance Factors" on page 3-7.

Tuning Queries

In addition to the system configuration, Liquid Data performance is greatly affected by the query design, as described in "Query Performance Factors" on page 3-3. Therefore, make sure that the queries running on the Liquid Data server are reasonably designed:

- Does each query follow the appropriate query design pattern, as described in "Design Patterns" in "Designing Queries" in *Building Queries and Data Views*?

- Does each query provide the appropriate hints, as described in "Optimizing Queries" in *Building Queries and Data Views*?
- Does each query have the appropriate scope, as described in "Understanding Scope in Basic and Advanced Views" in "Designing Queries" in *Building Queries and Data Views*?
- Does any query produce a large intermediate result set or final result set? For more information, see "Query Performance Factors" on page 3-3.
- Have you debugged the query using a verbose logging mode (info) and reviewed the trace? For more information, see "Monitoring Liquid Data Performance" on page 3-15.

Liquid Data Performance Factors

Many factors influence overall Liquid Data performance. Certain factors, such as query design, are within the scope of Liquid Data's control, while other factors, such as data source processing speed, are outside the scope of Liquid Data control. This topic identifies the main factors that can affect Liquid Data performance. It includes the following sections:

Query Performance Factors

Liquid Data performance depends on the way queries are designed and configured for execution. The following query factors affect Liquid Data performance:

Table 3-1 Query Performance Factors

Factor	Description
Query complexity	Some query operations are more resource intensive than others. ■ Simple queries perform basic operations, such as retrieving a customer list or an employee's profile. ■ Analytical queries, such as complex joins or aggregates, perform interim processing steps that produce intermediate result sets. **Recommendations**: Because analytical queries generally consume more memory and CPU resources than simple queries, see the sections later in this table on caching and large result sets.
Query type	The type of query (stored or ad hoc) affects performance: ■ For stored queries, the query is compiled only once and query execution plan is cached. In addition, the query result can be cached for faster retrieval if the query is executed subsequently with the same parameters. For more information, see "Configuring the Query Results Cache" in the Liquid Data *Administration Guide*. ■ Ad hoc queries are always compiled and executed. No caching is involved. For certain queries, the time to compile the query might take much longer then the time to execute. **Recommendation**: Use stored queries in a production system.
Design pattern used (query hints)	When building queries, the appropriate design patterns must be used to ensure the fastest possible execution speed. For example, for joins, you should use a query hint to supply more information to the execution engine about the amount of data to search when processing a query. **Recommendation**: Use hints, if applicable. A query with an appropriate hint may perform substantially better than one without a hint. Hints are particularly significant with large data sources. If you are using the Data View Builder tool, design patterns for target schema are also very important. For more information, see "Understanding Query Design Patterns" in "Designing Queries" and "Optimizing Queries" in *Building Queries and Data Views*.

Table 3-1 Query Performance Factors (Continued)

Factor	Description
Caching	Caching improves performance for stored queries. Liquid Data supports two types of caching: ■ Query plan cache, which is always enabled, allows Liquid Data to retrieve the query plan for a repeated query from cache rather than regenerating the query plan. ■ Query results cache, which allows Liquid Data to retrieve the results for a repeated query from in-memory cache rather than running the query against the data source each time. Stored query results that are configured to expire sooner must be executed against the data source(s) more frequently. **Recommendations**: ■ Always use stored queries in a production system. ■ Using query results caching, if applicable. For a complex query, or for a query that retrieves data from slower data sources, caching the query result provides a substantial performance gain. Once the query is cached, the query execution time is the fixed cost of retrieving the results cache from the database, and returns it as an XML document. For more information, see "Configuring the Query Results Cache" in the Liquid Data *Administration Guide*.

Table 3-1 Query Performance Factors (Continued)

Factor	Description
Size of intermediate or final results returned and memory usage	The larger the result size (final or intermediate results), the longer it takes to retrieve and process the results. **Recommendations**: ■ Queries should be designed to retrieve only the required results. Restrict the number of elements in the target schema (project). Specify strict conditions that limit the number of rows returned in the result set. ■ Increase the heap size to as high as the system can allow. In general, increasing the heap size increases performance. If increasing the heap size solves the out of memory problem, then no further action is required. ■ For relational database data sources, use the `{--! merge !-- }` query hint, if applicable. The effect is similar to a database merge join. Using this technique has a space/performance trade-off: a merge hint provides better memory usage but it might also slow performance if extra sorting is required to perform the merge join. ■ For queries that perform joins on multiple data sources, specify the sequence of data sources in ascending order by increasing size: the smallest resource should appear in the first FOR loop, and the largest resource should appear in the last one. For more information, see "Source Order Optimization" in "Optimizing Queries" in Building Queries and Data Views. ■ Use the disk swapping option, which allows Liquid Data to store intermediate results on disk when the results exceed the `MEM_SORT_BUF` size specified in the server startup script (`startWebLogic.cmd` on Windows or `startWebLogic.sh` on Unix). The default setting for `MEM_SORT_BUF` size is 50MB. The recommended size is less than one third (1/3) of the maximum memory size defined in `MEM_ARGS` in the server startup script. To configure disk swapping, see "Configuring Liquid Data Server Settings" in the Liquid Data *Administration Guide*. **Note:** Liquid Data queries do not retrieve binary large object (BLOB) data from relational databases. For a list of supported data types, see "Supported Data Types" in *Building Queries and Data Views*.
Number of concurrent queries	The higher the number of concurrent queries, the slower the performance, particularly during peak loads. Performance improves through the use of additional CPUs and WebLogic Server clusters, as described in "Clustered Deployments" on page 2-6, and with tuning the thread pool, as described in "Using the Administration Console to Monitor Performance" on page 3-16.

Data Source Performance Factors

In general, Liquid Data performance depends on the speed at which the data source host system is able to process query requests and return results.

Performance Factors for All Data Sources

The following data source factors affect Liquid Data performance:

Table 3-2 Data Source Performance Factors

Factor	Description
Data source type	Some types of data sources offer higher performance (such as relational databases) than other types (such as application integrations or Web services). For more information, see Table 3-3, "Performance Factors for Data Source Types," on page 3-9.
Data source size	The size of the data source always affects performance. In general, the larger the data source, the longer it takes to retrieve the query results. For example, a large XML document takes longer to process than a small XML document. For relational databases, indexing substantially improves performance, particularly for large databases.
Number of data sources	For queries that access multiple data sources, data is retrieved from each data source in sequence, one data source at a time, for all data source types except application views, web services, and custom functions (which are processed asynchronously). For application views, web services, and custom functions, you can configure the maximum number of connections or the maximum number of concurrent threads to be used. If queries use web services, application views, or custom functions extensively, then consider tuning this setting.

Table 3-2 Data Source Performance Factors (Continued)

Factor	Description
Data source performance and availability	■ A query fails if a required data source is unavailable during query execution due to server failure, insufficient available connections, failed authentication, or other factor. ■ Performance delays result if the server hosting the data source is congested ■ Faster hardware (storage, memory, and CPU throughput) for the data source host machine generally provides higher performance, particularly for larger data sources. ■ Except for any XML files stored locally on the host system, all data sources are remote. Therefore, network connection availability and speed affects how quickly the query results are returned. In addition to network capacity and throughput speeds, the number of hops between nodes can greatly affect performance. Secure connections, such as SSL (Secure Sockets Layer) increase security but slow performance. Network speed is not a factor with local data sources, such as XML files stored locally on the host system. However, network speed is a factor with XML files stored remotely.
Transaction isolation level (relational databases only)	For relational databases, the transaction isolation level setting can affect query performance: ■ For databases containing static or read-only data, in general, use the default setting (`TRANSACTION_READ_COMMITTED`). ■ For databases containing dynamic or updateable data, use the transaction isolation level that supports the degree of concurrency required. However, increased concurrency can result in slower query performance. Using a transaction level of `TRANSACTION_SERIALIZABLE`, the highest level of concurrency, is more likely to reduce performance than using a lower level of concurrency, such as `TRANSACTION_REPEATABLE_READ` or `TRANSACTION_READ_COMMITTED`. For more information about configuring the transaction isolation level for a relational database, see "Creating a Relational Database Data Source Description" in "Configuring Access to Relational Databases" in the Liquid Data *Administration Guide*.

Performance Factors for Data Source Types

The following table describes the most important performance factors for each supported data source type:

Table 3-3 Performance Factors for Data Source Types

Type	Important Performance Factor(s)
Relational databases	■ A fast host machine is especially important for large databases. ■ An optimized database design and a highly tuned configuration substantially improves performance. For example, indexing improves performance, particularly for large databases. For more information, see your database vendor's documentation. ■ The JDBC driver configuration can affect query performance, and JDBC connection pool settings must be properly tuned. For more information, see "Managing JDBC Connectivity" in the BEA WebLogic Server *Administration Guide*. ■ The JDBC transaction isolation level can affect query performance. For more information, see the discussion of transaction isolation levels in Table 3-2, "Data Source Performance Factors," on page 3-7.
XML files	■ Smaller XML files are generally processed more quickly than larger ones.
Web services	■ Query performance depends primarily on the web service implementation and the speed at which the Web Service returns results, as well as the network connection speed. ■ Liquid Data runs query requests on web services asynchronously. If a query uses multiple data sources, the query can continue processing while waiting for a response from a web service. For web services, therefore, performance also depends on the Maximum Threads specified on the General tab in the Liquid Data node in the Administration Console, as described in "Configuring Liquid Data Server Settings" in the Liquid Data *Administration Guide*.
Application views	■ Query performance depends on the performance of the EIS and its associated adapter. ■ The Application View connections pool must be properly tuned. For more information, see "Monitoring and Tuning Application View Connections" in "Tuning Performance" in *Deploying Solutions* in the WebLogic Integration documentation. ■ Liquid Data runs query requests on application views asynchronously. If a query uses multiple data sources, the query can continue processing while waiting for a response from an application view. For application views, therefore, performance also depends on the Maximum Threads specified on the General tab in the Liquid Data node in the Administration Console, as described in "Configuring Liquid Data Server Settings" in the Liquid Data *Administration Guide*.
Data views	■ Complexity of the underlying query, as described in "Query Performance Factors" on page 3-3. ■ The data source performance factors described in "Performance Factors for All Data Sources" on page 3-7.

Platform Performance Factors

This section describes performance factors associated with the Liquid Data server, including the host server hardware, clustering Liquid Data servers, tuning threads, tuning WebLogic Server, and tuning WebLogic Integration. The most important factor is running Liquid Data on a very fast server machine with the maximum amount of available memory. For general information about platform performance, see "Tuning Hardware, Operating System, and Network Performance" in *BEA WebLogic Server Performance and Tuning*.

This section describes the following platform performance factors:

- General Platform Performance Factors
- WebLogic Integration Performance Factors
- Liquid Data Host Server Machine
- WebLogic Integration Performance Factors

General Platform Performance Factors

The following general performance factors are associated with a Liquid Data deployment:

Table 3-4 Server Hardware Performance Factors

Factor	Description
Network connection speed	For remote resources such as data sources, the speed and capacity of the network connection is an important factor. In addition to network capacity and throughput speeds, the number of hops between nodes can greatly affect performance. Secure connections, such as SSL (Secure Sockets Layer) increase security but slow performance.

Table 3-4 Server Hardware Performance Factors (Continued)

Factor	Description
Distribution of resources across servers	Performance is greatly affected by the way in which Liquid Data, WebLogic Server, and other WebLogic Platform resources are distributed across servers. For example: ■ If a Liquid Data deployment uses application views as data sources, it is generally optimal to run Liquid Data and Application Integration on separate server machines, as described in "Multi-Node Deployments" on page 2-5. ■ Liquid Data is built on the scalable WebLogic Server platform. In deployments that support a high volume of concurrent query requests, you can increase system performance by deploying on a WebLogic Server cluster—a group of WebLogic Servers that are managed as a single unit and distribute the load for processing query requests. For more information, see "Clustered Deployments" on page 2-6. For more information, see "Designing a Deployment" on page 2-3.

WebLogic Server Performance Factors

Liquid Data performance is affected by WebLogic Server performance. The WebLogic Server documentation provides a detailed suggestions for monitoring and tuning run-time performance. For detailed information, see BEA WebLogic Server Performance and Tuning in the WebLogic Server documentation.

The following table provides a summary of tuning factors, which are described at length in the WebLogic Server documentation:

Table 3-5 Summary of WebLogic Server Performance Factors

Component	Tunable Performance Factor(s)
Hardware Resources	■ WebLogic Server Platform Support Page (Certifications) at the following URL: `http://e-docs.bea.com/wls/certifications/certifications/index.html` ■ CPU—maximize speed and throughput ■ Memory—maximize capacity and speed ■ Network resources—maximum bandwidth and speed ■ Disk I/O and controllers—maximize disk speed and capacity ■ Number of machines—increase as needed

Table 3-5 Summary of WebLogic Server Performance Factors (Continued)

Component	Tunable Performance Factor(s)
Operating System	■ Configurable file descriptor limits ■ Memory allocation for user processes ■ Configurable TCP tuning parameters ■ Configurable settings for the threading model
Network Resources	■ Network hardware and software ■ Network bandwidth ■ LAN infrastructure
Java Virtual Machine (JVM)	■ JVM vendor and version ■ JVM heap size and garbage collection ■ Generational garbage collection ■ Mixed client/server JVMs ■ Unix threading models ■ Just-in-time (JIT) JVMs
WebLogic Server	■ config.xml file parameters ■ weblogic-ejb-jar.xml parameters ■ WebLogic Server startup parameters ■ Java compiler ■ WebLogic Server clusters ■ JDBC driver and JDBC connection pool
WebLogic Server Applications	■ Performance analysis tools ■ JDBC application tuning ■ Session persistence ■ Minimizing sessions ■ Execute queues and thread usage

Liquid Data Host Server Machine

Faster hardware (storage, memory, and CPU throughput), large capacity storage (for caching and disk swapping), and for the Liquid Data server host machine generally provides higher performance. The following performance factors are associated with the host server machine:

Table 3-6 Liquid Data Host Server Machine Performance Factors

Factor	Description
CPU utilization	Optimal utilization is up to 80%.
Storage utilization	Machine should have sufficient available workspace for disk swapping and other storage operations. For recommendations, see "Installation Prerequisites" in "Preparing to Install WebLogic Server" in the WebLogic Server *Installation Guide*.
Memory utilization	■ The default memory size is 256MB, but it is recommended that you add as much memory as the server machine can support. Memory size is one of the most significant factors for Liquid Data performance. ■ If Liquid Data uses application views or web services as data sources, the Maximum Threads specified on the General tab in the Liquid Data node in the Administration Console determines the number of threads available for asynchronous requests to application views and web services. For more information, see "Configuring Liquid Data Server Settings" in the Liquid Data *Administration Guide*.
Thread pools	■ The pool size for execute threads should be optimized. Liquid Data uses WebLogic Server's sophisticated multi-threading capabilities to process concurrent query requests more efficiently. By default, WebLogic Server uses 15 threads for the thread pool size. For more information, see "Tuning Threads" on page 3-14. ■ You need to tune the thread pool size according to the characteristics of the query request load on the Liquid Data server. Too many or too few threads can impede performance. In general, increase the number of threads if CPU utilization for the workload is low, and decrease the number of threads if CPU utilization exceeds 80%. You might need to experiment by repeatedly changing the maximum number of threads until you determine the optimum setting for your deployment.

WebLogic Integration Performance Factors

If Liquid Data is deployed with WebLogic Integration, then WebLogic Integration performance might affect Liquid Data performance, depending on how the two components interact. The WebLogic Integration documentation provides a detailed suggestions for monitoring and tuning run-time performance. For detailed information, see "Tuning Performance" in *Deploying Solutions* in the WebLogic Integration documentation.

The following table provides a summary of tuning factors, which are described at length in the WebLogic Integration documentation.

Table 3-7 Summary of WebLogic Integration Performance Factors

Component	Tunable Performance Factor(s)
Business Process Management	■ Event listener message-driven bean for event-driven workflows
Application Integration	■ application view bean for synchronous service invocations ■ thread pool size of the asynchronous request processor for asynchronous service invocations ■ J2EE-CA resource pool size for each adapter
B2B integration	There are no primary resources that can be tuned
WebLogic Server	■ EJB pool and cache sizes ■ JDBC connection pool sizes ■ Execution thread pool size ■ Resource connection pools for J2EE Connector Architecture adapters ■ Large message support for B2B
Java Virtual Machine (JVM)	■ JVM vendor and version ■ JVM heap size ■ Garbage Collection Control (Hotspot JVM)
Hardware Resources	■ CPU—maximize speed and throughput ■ Memory—maximize capacity and speed ■ Network resources—maximum bandwidth and speed ■ Disk I/O and controllers—maximize disk speed and capacity ■ Number of machines—increase as needed
Operating System	■ Configurable file descriptor limits ■ Memory allocation for user processes ■ Configurable TCP tuning parameters ■ Configurable settings for the threading model

Table 3-7 Summary of WebLogic Integration Performance Factors (Continued)

Component	Tunable Performance Factor(s)
JDBC Databases	■ Number of concurrently opened cursors ■ Disk I/O optimization ■ Database sizing and organization of table spaces ■ Checkpointing ■ Database compatibility ■ Database monitoring

Monitoring Liquid Data Performance

This section describes how to monitor Liquid Data performance. It includes the following sections:

- Monitoring Guidelines
- Using the Administration Console to Monitor Performance

For detailed information about monitoring resources for the Liquid Data Server, see "Monitoring the Server" in the WebLogic Server *Administration Guide*.

Monitoring Guidelines

When monitoring Liquid Data performance, consider the following guidelines:

- Try simulating workloads and monitor performance at different load levels—peaks, lulls, and mid-range volumes—to determine optimal overall settings.
- At a minimum, monitor server memory, threads, and CPU usage at these various load levels.
- Characterize the query request workload in the production environment, determining which queries are used and how often they are requested. For

example, you might learn that five queries represent 80% of the workload and ten queries represent the remaining 20%.

- Determine whether this workload involves the execution of small queries many times or large queries a few times and configure the thread pool accordingly.
- Determine whether clustering is appropriate for your deployment. For more information, see "Clustered Deployments" on page 2-6.
- Review your domain's server log for certain performance information. Liquid Data records the time to generate the query plan, compile the query plan, and execute the query. For relational databases, Liquid Data also records the SQL statement it generated for the query.

Using the Administration Console to Monitor Performance

You can use the Administration Console to monitor performance on a Liquid Data server, including the following areas:

- Active execute queues—As work enters a WebLogic Server, it is placed in an *execute queue* and then assigned to a thread in which the work is processed. The size of the execute pool determines the number of threads that can be running concurrently on an execute queue. Threads consume resources, so it is important that you tune the number of threads to optimize performance, taking care not to slow performance by increasing the value unnecessarily.
- JDBC connection pools—If queries in your deployment use relational databases for data sources, you should monitor any active JDBC connection pools for performance. A JDBC connection pool contains a group of JDBC connections. At run-time, when a query that uses an RDBMS data source is executed, the query borrows a connection from the connection pool, uses it, then returns it to the connection pool by closing it. The size of the JDBC connection pool determines the number of JDBC connections that can be used concurrently. Connections consume resources, so it is important that you tune the number of connections to optimize performance, taking care not to slow performance by increasing the value unnecessarily.

For detailed information about using the Administration Console to monitor server performance, see the following topics in the WebLogic Server documentation:

- "Monitoring a WebLogic Server Domain" in *Creating and Configuring WebLogic Server Domains*
- "Execute Queues" in the WebLogic Server Administration Console online help
- BEA WebLogic Server Performance and Tuning

CHAPTER

4 Troubleshooting a Deployment

This topic provides information to help you troubleshoot a BEA Liquid Data for WebLogic™ deployment. It includes the following sections:

- Troubleshooting Resources
- Troubleshooting Out of Memory Exceptions

This section provides general troubleshooting tips. For information about known limitations in specific Liquid Data releases, see the Liquid Data Release Notes.

Troubleshooting Resources

The following sections describe resources that you can use to troubleshoot a Liquid Data deployment:

- Liquid Data Resources
- WebLogic Server Resources
- Application Integration (AI) Resources
- Business Process Management (BPM) Resources
- WebLogic Portal Resources
- WebLogic Workshop Resources

- BEA Developer Center

Liquid Data Resources

Liquid Data provides log entries and release notes for troubleshooting resources.

Log Entries

Review the log entries in the domain log for errors. Liquid Data records status, problem, and performance information in the domain log. For more information, see "Monitoring the Server Log" in "Monitoring the Server" in the Liquid Data *Administration Guide*.

Release Notes

The Liquid Data Release Notes provide information about known limitations and workarounds for the version of Liquid Data that you are using.

Product Documentation

The Liquid Data product documentation provides detailed information about all aspects of the Liquid Data product. For more information, see the documentation CD that comes with the Liquid Data package or go to the Liquid Data Documentation page.

WebLogic Server Resources

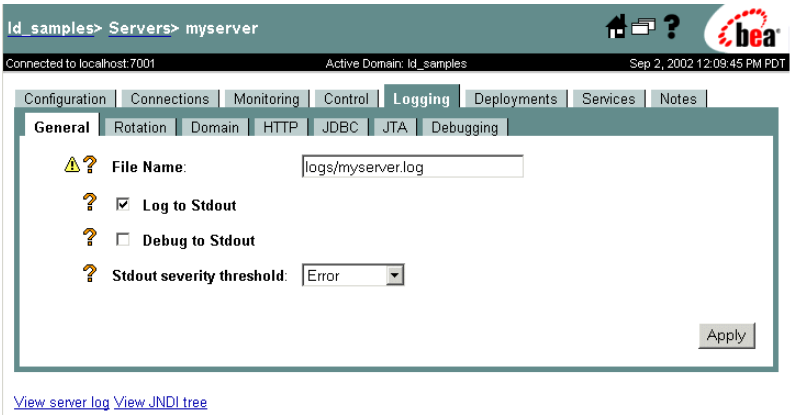
The BEA WebLogic Server™ server log contains valuable information about startup and run-time events that can help you troubleshooting a Liquid Data deployment. You can configure the amount of information that the WebLogic Server saves in the log file.

To configure the WebLogic Server server log:

1. On the Liquid Data server, start the Administration Console:
2. In the left pane of the Administration Console, click the name of the Liquid Data server for which you want to configure the log.

3. Click the Logging tab.

   The Administration Console displays the General tab showing the current log settings.



4. Change the log settings as needed.

   If you want the WebLogic Server to save the maximum amount of information to the log, select Log to Sdout, Debug to Stdout, and select a severity level of Info.

5. Click Apply.

Note: For debugging purposes, you can also click the Debugging tab, select Log Remote Exceptions and Instrument Stack Traces, and click Apply.

In a clustered environment, the domain log provides an aggregated view of log events for all servers in the cluster. For detailed information about WebLogic Server logs, see "Using Log Messages to Manage WebLogic Servers" in the WebLogic Server *Administration Guide*.

Application Integration (AI) Resources

BEA WebLogic Integration™ maintains separate logs for the Application View Management Console and for each adapter. These logs are located in the parent directory of the config/*`domain`* directory, where *`domain`* is the name of the domain that is currently running.

The Application View Management Console allows you to configure logging levels, as described in "Deploying the Application View" in "Steps for Defining an Application View" in "Defining an Application View" in *Using Application Integration*.

Business Process Management (BPM) Resources

You can enable debugging to route event data to the WebLogic Server Administration Console by setting the following debug flag on the server startup command line:

```
-Dwli.bpm.server.eventprocessor.debug=1
```

For more information, see the Debug Flags document in the WebLogic Integration documentation.

WebLogic Portal Resources

BEA WebLogic Portal™ uses the WebLogic Server logging mechanism to save messages to the WebLogic Server log. WebLogic Portal applications can control logging levels using classes and interfaces in the WebLogic Portal API, as described in the WebLogic Portal Javadoc.

WebLogic Workshop Resources

BEA WebLogic Workshop™ uses the log4j Java logging facility developed by the Jakarta Project of the Apache Foundation. You can learn more about log4j at The Log4j Project.

You configure logging in WebLogic Workshop by changing settings in the `workshopLogCfg.xml` file, which by default is located in *`BEA_HOME`*`/weblogic700/common/lib`. For more information, see "workshopLogCfg.xml Configuration File" in the WebLogic Workshop documentation.

By default, WebLogic Workshop uses two log files:

- `workshop.log`—configured to receive all internal logging messages emitted by the WebLogic Workshop runtime.
- `jws.log`—configured to receive all logging messages emitted by user code associated with web services. This includes code in JWS files, but also code in JSX and JAVA files that are used by a web service.

BEA Developer Center

The BEA Developer Center provides a range of technical resources, including newsgroups on technical topics, such as installation, clustering, JDBC, EJBs, servlets, and JSPs. For more information, visit the BEA Developer Center at the following URL:

http://www.bea.com/support/welcome.jsp

Troubleshooting Out of Memory Exceptions

This topic describes how to handle out of memory exceptions that can occur in a deployment. The following table provides a list of possible causes and associated fixes.

Table 4-1 Causes and Solutions for Out of Memory Exceptions

Possible Cause	Problem Description	Solution
User Error	Design flaw in the query. For example, a query attempts to perform a three-way join ($table_1$, $table_2$, and $table_3$), but a join predicate exists between $table_1$ and $table_2$ only.	Fix the query design.
Lack of Proper Hints	Lack of proper or correct hints.	Specify proper hints, as described in "Optimizing Queries" in *Building Queries and Data Views*. Alternatively, redesign the query.

Table 4-1 Causes and Solutions for Out of Memory Exceptions (Continued)

Possible Cause	Problem Description	Solution
Query Plan Not Optimized	■ Lack of proper or correct hints. ■ Possible limitations in Liquid Data Server implementation.	Specify proper hints. Redesign the query.
Large Result Set	Query returns a very large result set. For example, a query retrieves all the customers in the database.	■ Increase the heap size from the default (256MB) to as high as possible. ■ Specify hints. For example, for RDBMS data sources, use the `merge` hint, if applicable, although using the merge hint might slow query performance. ■ Use disk swapping, which improves system availability but might slow system performance.
Large Intermediate Result Set	Query returns a very large intermediate result set. For example, an analytical query, or a query that processes a large XML document, which Liquid Data loads entirely into memory before any query processing.	■ Increase the heap size from the default (256MB) to as high as possible. ■ Use disk swapping, which improves system availability but might slow system performance.

Index

W