Oracle® Fusion Middleware

Administering WebLogic Server for Oracle Exalogic Elastic Cloud

12*c* (12.2.1.3.0)
E80389-01
August 2017

Oracle Fusion Middleware Administering WebLogic Server for Oracle Exalogic Elastic Cloud, 12*c* (12.2.1.3.0)

E80389-01

Contents

Preface

Documentation Accessibility vii

Conventions vii

1 Introduction and Roadmap

1.1 Document Scope and Audience 1-1

1.2 Guide to this Document 1-1

1.3 Related Documentation 1-2

1.4 Samples and Tutorials 1-2

1.4.1 Avitek Medical Records Application (MedRec) and Tutorials 1-2

1.5 New and Changed Features in This Release 1-3

2 WebLogic Server Domain Optimizations for Exalogic Elastic Cloud Software

2.1 Input/ Output and Work Manager Optimizations for Exalogic Elastic Cloud Software 2-1

2.1.1 Tuning I/O Performance for a Server Instance Using the Administration Console 2-2

2.2 Enabling JDBC Store Exalogic Optimizations 2-3

2.3 Enabling TopLink Exalogic Optimizations 2-3

2.4 Enabling Exalogic Optimizations 2-3

3 Using the WebLogic Replicated Store for WebLogic Server Messaging Services

3.1 Overview of the Replicated Store 3-1

3.1.1 Features of the Replicated Store 3-3

3.1.1.1 Scalability and High Availability of Replicated Stores 3-3

3.1.1.2 Server and Service Migration for Replicated Stores 3-4

3.2 Administering a Replicated Store 3-5

3.2.1 Quick Start Guide to Use a Replicated Store 3-5

3.2.1.1 Starting a Replicated Store 3-5
3.2.1.2 Shutting Down a Replicated Store 3-6
3.2.2 Administering a Global Directory 3-6
3.2.2.1 Configuration and Tuning Requirements for a Global Directory 3-7
3.2.3 Administering a WebLogic Replicated Store 3-7
3.2.3.1 How to Create and Configure a WebLogic Replicated Store 3-8
3.2.3.2 Monitoring a Replicated Store 3-8
3.2.3.3 Best Practices and Considerations for WebLogic Replicated Stores 3-9
3.2.4 Administering a Daemon Cluster 3-10
3.2.4.1 Configuring a Daemon 3-10
3.2.4.2 Starting and Stopping a Daemon Cluster 3-11
3.2.4.3 Logging Daemon Information 3-13
3.2.4.4 Administering a Daemon Cluster using the Administration Utility 3-14
3.2.4.5 Understanding Daemon Clusters 3-19
3.2.5 Managing Memory Utilization for Replicated Stores 3-22
3.2.5.1 Replicated Store Memory Usage 3-22
3.2.5.2 Monitoring Memory Utilization 3-22
3.2.5.3 Controlling Memory Utilization 3-23
3.2.5.4 Replicated Store Behavior when Exceeding Available Memory 3-25
3.3 Interoperability Considerations for a Replicated Store 3-25
3.4 Security Considerations for a Replicated Store 3-25
3.4.1 Security Considerations for Administrators 3-25
3.4.2 General Security Considerations for Replicated Stores 3-26
3.5 Limitations of a Replicated Store 3-26

4 WebLogic Server Cooperative Memory Management in Oracle Exalogic Elastic Cloud Environments

4.1 What is Cooperative Memory Management? 4-1
4.2 Using CMM in Your Environment 4-2
4.3 CMM for WebLogic Data Sources 4-2
4.3.1 Memory Management for Statement Cache Size 4-2
4.3.1.1 Example 4-3
4.3.2 Memory Management for Shrinking Data Sources 4-4

5 Message Compression In Oracle Exalogic Elastic Cloud Environments

5.1 Selecting a Message Compression Option 5-1
5.2 Message Compression for JMS Servers in Exalogic Environments 5-2

5.3 Message Compression for Store-and-Forward Sending Agents in Exalogic Environments 5-2

Preface

This preface describes the document accessibility features and conventions used in this guide—*Administering WebLogic Server for Oracle Exalogic Elastic Cloud*.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Accessible Access to Oracle Support

Oracle customers who have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction and Roadmap

This chapter describes the contents and organization of this guide—*Administering WebLogic Server for Oracle Exalogic Elastic Cloud*.
This chapter includes the following sections:

- Document Scope and Audience
- Guide to this Document
- Related Documentation
- Samples and Tutorials
- New and Changed Features in This Release

Note:

The WebLogic Replicated Store, which is intended only for use in Oracle Exalogic Elastic Cloud environments, is deprecated as of Oracle WebLogic Server version 12.2.1.3.0 and will be removed from WebLogic Server in a future release. WebLogic Server components that enable the configuration and management of the WebLogic Replicated Store, such as the `DomainMBean.ReplicatedStores` attribute, the `weblogic.management.configuration.ReplicatedStoreMBean` API, and this document, are also deprecated and will be removed. As of Oracle WebLogic Server 12.2.1.3.0, Oracle recommends that you use either a JDBC store or a custom file store for JMS message storage.

1.1 Document Scope and Audience

This document is written for application developers and system administrators who develop and administer applications for WebLogic Server in Exalogic Elastic Cloud Software environments. It is assumed that readers are familiar with the WebLogic Server platform, Java Platform, Enterprise Edition (Java EE) programming, and Exalogic Elastic Cloud Software.

1.2 Guide to this Document

- This chapter, Introduction and Roadmap, introduces the organization of this guide.
- The chapter WebLogic Server Domain Optimizations for Exalogic Elastic Cloud Software provides information on how to optimize WebLogic Server domains for Exalogic Elastic Cloud.
- The chapter Using the WebLogic Replicated Store for WebLogic Server Messaging Services explains how to use a WebLogic Replicated Store to provide a scalable and high-performance storage option for WebLogic Server Messaging services that require persistence in Oracle Exalogic Elastic Cloud environments.

- The chapter WebLogic Server Cooperative Memory Management in Oracle Exalogic Elastic Cloud Environments provides information about how to use Cooperative Memory Management to promote effective memory utilization by WebLogic Server resources.
- The chapter Message Compression In Oracle Exalogic Elastic Cloud Environments provides information about how to configure message compression for JMS Store I/O operations which may provide significant performance improvements in Oracle Exalogic environments.

1.3 Related Documentation

This document contains specific development and administration information for WebLogic Server in Oracle Exalogic Elastic Cloud environments, see Exalogic Elastic Cloud Software in *Licensing Information*.

For comprehensive guidelines for developing and managing WebLogic Server applications, see the following documents:

- Introduction to Oracle WebLogic Server in *Understanding Oracle WebLogic Server* provides an overview of Oracle WebLogic Server features and describes how you can use them to create enterprise ready-solutions.
- Performance Tuning Roadmap in *Tuning Performance of Oracle WebLogic Server* provides information on how to improve system performance and tune the components in a WebLogic Server environment.
- Understanding JMS Resource Configuration in *Administering JMS Resources for Oracle WebLogic Server* provides information on how to configure, manage, and monitor WebLogic JMS resources.
- Using the WebLogic Persistent Store in *Administering the WebLogic Persistent Store*.
- Understanding WebLogic Server Deployment in *Deploying Applications to Oracle WebLogic Server* is the primary source of information about deploying WebLogic Server applications.

1.4 Samples and Tutorials

In addition to this document, Oracle optionally provides a variety of code samples and tutorials for developing applications. The examples and tutorials illustrate WebLogic Server in action, and provide practical instructions on how to perform key application development tasks. You can start the Examples server from the `ORACLE_HOME\user_projects\domains\wl_server` directory, where `ORACLE_HOME` is the directory you specified as Oracle Home when you installed Oracle WebLogic Server. For more information, see Sample Applications and Code Examples in *Understanding Oracle WebLogic Server*.

1.4.1 Avitek Medical Records Application (MedRec) and Tutorials

MedRec is an end-to-end sample Java EE application shipped with WebLogic Server that simulates an independent, centralized medical record management system. The MedRec application provides a framework for patients, doctors, and administrators to manage patient data using a variety of different clients.

MedRec demonstrates WebLogic Server and Java EE features, and highlights Oracle-recommended best practices. MedRec is optionally installed in the WebLogic Server installation. You can start MedRec from the `ORACLE_HOME\user_projects\domains\medrec` directory, where `ORACLE_HOME` is the directory you specified as Oracle Home when you installed Oracle WebLogic Server. For more information, see Sample Applications and Code Examples in *Understanding Oracle WebLogic Server*.

1.5 New and Changed Features in This Release

WebLogic Server includes the following new features for Oracle Exalogic Elastic Cloud environments:

- A WebLogic Replicated Store to provide a scalable and high-performance storage option for WebLogic Server Messaging services, see Using the WebLogic Replicated Store for WebLogic Server Messaging Services.
- Cooperative Memory Management to promote effective memory utilization by WebLogic Server resources, see WebLogic Server Cooperative Memory Management in Oracle Exalogic Elastic Cloud Environments.
- An optimization for JDBC stores to commit a batch of `INSERT` or `DELETE` operations instead of issuing a separate commit call to the database server, see Enabling JDBC Store Exalogic Optimizations.
- TopLink optimizations that include: batch writing, increasing the default cache size, and enable other optimizations such as weaving eager relationships. See Enabling TopLink Exalogic Optimizations.

For a comprehensive listing of the new WebLogic Server features introduced in this release, see What’s New in Oracle WebLogic Server 12.2.1.3.0.

2

WebLogic Server Domain Optimizations for Exalogic Elastic Cloud Software

This chapter describes WebLogic Server Domain Optimizations for Exalogic Elastic Cloud Software.

This chapter includes the following sections:

- Input/ Output and Work Manager Optimizations for Exalogic Elastic Cloud Software
- Enabling JDBC Store Exalogic Optimizations
- Enabling TopLink Exalogic Optimizations
- Enabling Exalogic Optimizations

2.1 Input/ Output and Work Manager Optimizations for Exalogic Elastic Cloud Software

Exalogic Elastic Cloud Software can be configured with input/output and Work Manager optimizations for WebLogic Server. Table 2-1 lists and describes the MBean attributes for configuring WebLogic Server with these options that are restricted to Exalogic Elastic Cloud Software licenses.

Table 2-1 Domain-Level

MBean Attribute	-D Option	Description
`KernelMBean.ScatteredReadsEnabled =true`	`-Dweblogic.ScatteredReadsEnabled`	Increases efficiency during I/O in environments with high network throughput.
`KernelMBean.GatheredWritesEnabled =true`	`-Dweblogic.GatheredWritesEnabled`	Increases efficiency during I/O in environments with high network throughput.
`KernelMBean.AddWorkManagerThreads ByCpuCount=true`	`-Dweblogic.AddWorkManagerThreadsByCpuCount`	Increases efficiency of the self-tuning thread pool by aligning it with the Exalogic processor architecture threading capabilities
`ServerMBean.useConcurrentQueueFor RequestManager=true`	`-Dweblogic.useConcurrentQueueForRequestManager`	Lockless request manager enables higher concurrency and efficiency in processing on Exalogic systems, especially for JMS use cases.

Table 2-1 (Cont.) Domain-Level

MBean Attribute	-D Option	Description
`KernelMBean.MuxerClass=weblogic.socket.NIOSharedWorkSocketMuxer`		Reduces queue wait time for requests with high impact from muxing, such as JMS messages, replication, RMI.

2.1.1 Tuning I/O Performance for a Server Instance Using the Administration Console

When enabled, these attributes increases efficiency during I/O in environments with high network throughput. The attribute is automatically enabled when `Enable Exalogic Optimizations` is enabled at the Domain level. See Enabling Exalogic Optimizations.

Note:

`AddWorkManagerThreadsByCpuCount` is not tunable using the WebLogic Server Administration Console.

To manage this behavior at the server level or using a server template:

1. If you have not already done so, in the Change Center of the WebLogic Server Administration Console, click **Lock & Edit**. See Use the Change Center in *Oracle WebLogic Server Administration Console Online Help*.
2. In the left pane of the WebLogic Server Administration Console, under **Domain Structure**, select **Environment > Servers** or **Environment > Clusters > Server Templates**.
3. Select a server or server template.
4. Optionally, select **Configuration > Tuning** and select the appropriate check box:
   - **Enable Gathered Writes**
   - **Enable Scattered Reads**
5. Select **Advanced** and optionally:
   - Optionally, select **Use Concurrent Queue For Request Manager**
   - Optionally, set **Muxer Class** to `weblogic.socket.NIOSharedWorkSocketMuxer` for Exalogic Environments. The default **Muxer Class** is `weblogic.socket.NIOSocketMuxer`.
6. Click **Save**, and then, to activate these changes, in the Change Center, click **Activate Changes**.
7. Shut down and restart the server. See Start and stop servers in *Oracle WebLogic Server Administration Console Online Help*.

2.2 Enabling JDBC Store Exalogic Optimizations

This release of WebLogic Server provides an Exalogic optimization for JDBC stores to commit a batch of `INSERT` or `DELETE` operations with the last operation of the transaction instead of issuing a separate commit call to the database server. This feature saves a server round trip per transaction and benefits applications that have many transactions of a small number of operations or small messages.

To manage this behavior for an individual store:

1. If you have not already done so, in the Change Center of the WebLogic Server Administration Console, click **Lock & Edit**. See Use the Change Center in *Oracle WebLogic Server Administration Console Online Help*.
2. In the left pane of the WebLogic Server Administration Console, under **Domain Structure**, select **Services > Persistent Stores**.
3. Select a JDBC store.
4. Select **Advanced** on the **Configuration** tab, optionally select **Oracle Piggyback Commit Enabled**.
5. Click **Save**, and then, to activate these changes, in the Change Center, click **Activate Changes**.
6. Shut down and restart the server. See Start and stop servers in *Oracle WebLogic Server Administration Console Online Help*.

2.3 Enabling TopLink Exalogic Optimizations

This release of WebLogic Server provides TopLink Exalogic optimizations that include: batch writing, increasing the default cache size, and enable other optimizations such as weaving eager relationships. A runtime tuning agent optimizes the use of query batch fetching and dynamic query processing.

For additional information on how to enable and configure features that can optimize how persistence applications perform in Oracle Exalogic environments, see Optimizing Persistence Applications for Oracle Exalogic in *Solutions Guide for Oracle TopLink* (note that this is an Oracle TopLink 12.1.3 publication, but it applies to the current version of WebLogic Server).

2.4 Enabling Exalogic Optimizations

Optimizations include improved thread management, request processing, and reduced lock contention. To enable all servers in a domain to use Exalogic optimizations:

1. If you have not already done so, in the Change Center of the WebLogic Server Administration Console, click **Lock & Edit**. See Use the Change Center in *Oracle WebLogic Server Administration Console Online Help*.
2. In the left pane of the WebLogic Server Administration Console, under **Domain Structure**, select the domain name.
3. Select **Configuration > General** and select the **Enable Exalogic Optimizations** check box.

4. Click **Save**, and then, to activate these changes, in the Change Center, click **Activate Changes**.
5. Shut down and restart all servers that are currently running in the domain. See Start and stop servers in *Oracle WebLogic Server Administration Console Online Help*.

3 Using the WebLogic Replicated Store for WebLogic Server Messaging Services

This chapter explains how to use a WebLogic Replicated Store to provide a scalable and high-performance storage option for WebLogic Server Messaging services that require persistence in Oracle Exalogic Elastic Cloud environments.

Note:

The WebLogic Replicated Store, which is intended only for use in Oracle Exalogic Elastic Cloud environments, is deprecated as of Oracle WebLogic Server version 12.2.1.3.0 and will be removed in a future release. WebLogic Server components that enable the configuration and management of the WebLogic Replicated Store, such as the `DomainMBean.ReplicatedStores` attribute and the `weblogic.management.configuration.ReplicatedStoreMBean` API, are also deprecated and will be removed. As of Oracle WebLogic Server 12.2.1.3.0, Oracle recommends that you use either a JDBC store or a custom file store for JMS message storage.

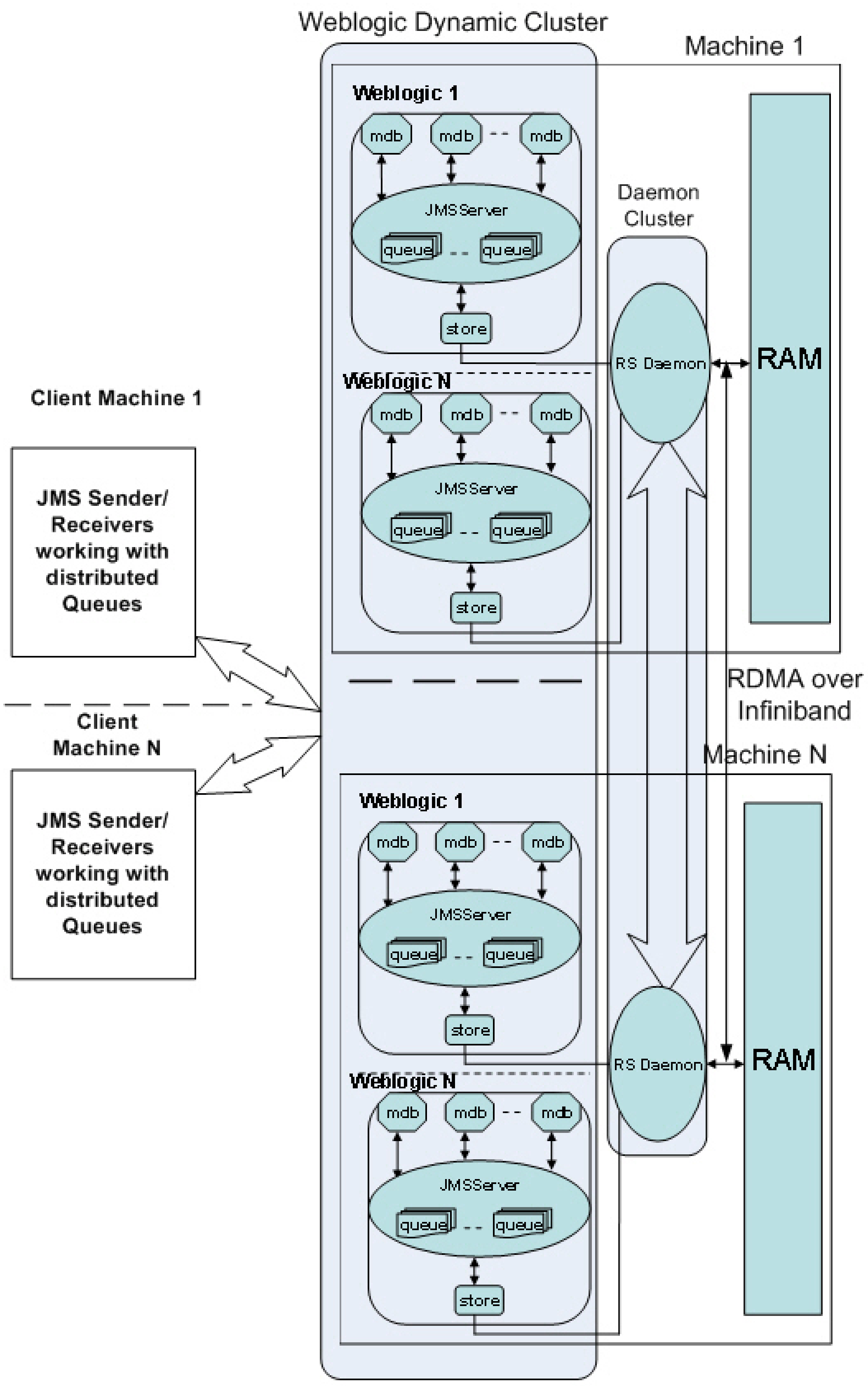
Note also that the WebLogic Replicated Store is supported only on Oracle Exalogic Elastic Cloud environments hosted on Oracle Enterprise Linux systems. The WebLogic Replicated Store is not supported in virtual Exalogic environments. See Oracle Fusion Middleware Supported System Configurations for details about Oracle Enterprise Linux versions supported on Exalogic systems.

- Overview of the Replicated Store
- Administering a Replicated Store
- Interoperability Considerations for a Replicated Store
- Security Considerations for a Replicated Store
- Limitations of a Replicated Store

3.1 Overview of the Replicated Store

WebLogic Messaging Services (JMS) can use Replicated Stores as a high performance alternative to existing File and JDBC storage options. A Replicated Store stores data in local Exalogic node memory and replicates it to memory on a second node providing high availability with no single point of failure while yielding linearly scalable performance.

Figure 3-1 Clustered Replicated Store



WebLogic Server Messaging services use a WebLogic Replicated Store to persist data to a Daemon Cluster using configuration information stored in a shared Global Directory. The message state is stored locally but can be recovered from any node that hosts the same Daemon Cluster by running the store instance on that node. A Replicated Store is analogous to a File or JDBC Store where a Region in a Daemon Cluster corresponds to a File Store file or JDBC Store table.

Note:

For more information about File and JDBC stores, see The WebLogic Persistent Store in *Administering the WebLogic Persistent Store*.

Table 3-1 defines the WebLogic services that can create connections to a Replicated Store. Each subsystem that uses the Replicated Store specifies a unique connection ID that identifies that subsystem.

Table 3-1 Replicated Store Users

Subsystem/Service	What It Stores	More Information
JMS Servers and SAF Agents	Persistent messages and durable subscribers.	Understanding the Messaging Models in *Developing JMS Applications for Oracle WebLogic Server*
Path Service	The mapping of a group of messages to a messaging resource.	Using the WebLogic Path Service in *Administering JMS Resources for Oracle WebLogic Server*
Store-and-Forward (SAF) Service Agents	Messages for a sending SAF agent for retransmission to a receiving SAF agent	Understanding the Store-and-Forward Service in *Administering the Store-and-Forward Service for Oracle WebLogic Server*.

For more information about the store connection IDs, see Monitoring Store Connections.

3.1.1 Features of the Replicated Store

The Replicated Store leverages Oracle Exalogic Elastic Cloud hardware and software which provides a unique combination of redundant hardware, large physical memory, high bandwidth InfiniBand networking, and efficient Remote Direct Memory Access (RDMA).

- Scalability and High Availability of Replicated Stores
- Whole Server Migration for Replicated Stores
- Service Level Migration for Replicated Stores

3.1.1.1 Scalability and High Availability of Replicated Stores

A Replicated Store provides superior linear scalability while providing failure resilience that is much higher than non-persistent messaging.

Table 3-2 summarizes the relative performance and high availability provided by available data storage types in Oracle Exalogic Elastic Cloud environments.

Table 3-2 Comparison of Replicated Store Performance and High-Availability Level for Data Storage

Store Type	Storage	Performance	High Availability Level
None	non-persistent	Fastest	Single point of failure.
Replicated Store	Data stored in replicated memory Regions of a Daemon Cluster.	Second Fastest	No single point of failure. Simultaneous failure of two nodes or processes required to cause data loss
File Store	Data stored in a file on the file system.	Third Fastest	No single point of failure when configured to use the Oracle ZFS Storage Appliance.
JDBC Store	Data stored in a JDBC table.	Slowest	No single point of failure when configured with an Oracle Exadata Database Machine. Optional ability to configure multi-site disaster recovery with Oracle Data Guard.

3.1.1.2 Server and Service Migration for Replicated Stores

Replicated Store configurations support server and service level migration as described in the following sections.

3.1.1.2.1 Whole Server Migration for Replicated Stores

For higher availability, the WebLogic Replicated Store instance can be migrated along with its parent server as part of the Whole Server Migration (WSM) feature, which provides both automatic and manual migration at the server level, rather than on the service level. WSM automatically restarts or migrates failed WebLogic Server Replicated Store instances. When a JMS Server instance or Replicated Store instance fails, a Replicated Store instance can recover its particular Regions by restarting on any machine that hosts a running RS Daemon in its Daemon Cluster. For more information, see Whole Server Migration in *Administering Clusters for Oracle WebLogic Server*.

3.1.1.2.2 Service Level Migration for Replicated Stores

A WebLogic Replicated Store instance can also be migrated as part of Automatic Service Migration (ASM) for JMS-related services, such JMS servers, SAF agents, and the path service, which rely on stores to maintain data. Service-level migration is controlled by a *migratable target*, which serves as a grouping of JMS-related services, and which is hosted on only one physical server in a cluster. Such hosted services can be automatically migrated from the current unhealthy hosting server to a healthy active server with the help of the Health Monitoring subsystem. JMS services hosted by a migratable target can also be manually migrated, either in response to a server failure or as part of regularly scheduled server maintenance. When the migratable target is migrated, all pinned services hosted by that target are also migrated. ASM automatically restarts or migrates failed WebLogic Server Replicated Store instances. When a JMS Server instance or Replicated Store instance fails, a Replicated Store instance can recover its particular Regions by restarting on any machine that hosts a running RS Daemon in its Daemon Cluster. For more information on service-level migration, see Service Migration in *Administering Clusters for Oracle WebLogic Server*.

> **Note:**
>
> As a best practice, a path service should use its own Replicated Store and migratable target.

3.2 Administering a Replicated Store

The following sections provide information on how to administer the a Replicated Store:

- Quick Start Guide to Use a Replicated Store
- Administering a Global Directory
- Administering a WebLogic Replicated Store
- Administering a Daemon Cluster
- Managing Memory Utilization for Replicated Stores

3.2.1 Quick Start Guide to Use a Replicated Store

Use the following procedures start and stop a Replicated Store:

3.2.1.1 Starting a Replicated Store

Use the following steps to configure and start a Replicated Store:

1. Create a common Global Directory on the ZFS Storage Appliance in an NFSv4 file system. This directory maintains configuration and state shared by a cluster of Replicated Store Daemons. See Administering a Global Directory.
2. Create a `rs_daemons.cfg` file in the Global Directory to configure each Daemon in a Daemon Cluster. A Daemon needs to be configured and started on each node that hosts a WebLogic Server that depends on a Replicated Store. See Creating a rs_daemons.cfg File.
3. Start each Daemon in the Daemon Cluster using the `startRSDaemon.sh` script.
   - Ensure that the Global Directory used by the script exactly matches the directory specified in the `config.xml` file in the WebLogic Replicated Store configuration created in the next step, and the directory created in step 1.
   - Configure the logging settings required for your environment.

See Starting Daemons using the startRSDaemon.sh Script.

4. Create a WebLogic Replicated Store using the WebLogic Server Administration Console. The value of the `Directory` attribute must be the same location specified in Step 1 and Step 3. See Create Replicated Stores in *Oracle WebLogic Server Administration Console Online Help*.

> **Note:**
>
> If a WebLogic Replicated Store instance references the Global Directory for a Daemon Cluster that has no Daemon started on the local machine, then the instance will not start.

3.2.1.2 Shutting Down a Replicated Store

Use the following steps to stop a Replicated Store:

1. Shutdown the associated WebLogic cluster. To temporarily stop using a store, you can disconnect from a Daemon Cluster without shutting down to WebLogic Server by untargeting the store from its server instance, dynamic cluster, or migratable target.
2. Optionally, shutdown each Daemon in the Daemon Cluster by running the `stopRSDaemon.sh` script. If you choose not to stop the Daemons, the store data will remain available for later recovery by restarted stores. See Shutting Down Daemons using the stopRSDaemon.sh Script.

3.2.2 Administering a Global Directory

A Global Directory is shared directory requiring a custom tuned NFS mount on the host Exalogic machine's ZFS Storage Appliance for a Daemon Cluster. A WebLogic Replicated Store instance and the administration utility reference a Global Directory to access its associated Daemon Cluster.

Table 3-3 describes the structure of a Global Directory.

Table 3-3 Components of a Global Directory

File Name	Description
`./rs_daemons.cfg`	An administrator created configuration file. All Daemons in the same Daemon Cluster and all the clients for these Daemons share the same `rs_daemons.cfg` file and look for the file at the root of their shared Global Directory. • '.' denotes the Global Directory root • Do not edit this file while a running Daemon Cluster is using the Global Directory.
`./logs/`*`rs_daemon_ddd_xxx.xxx.xxx.xxx_ppppp_nnn`*`.log`	Generated Daemon log files. See Logging Daemon Information.
`./daemons/...`	Internal runtime files.

Table 3-3 (Cont.) Components of a Global Directory

File Name	Description
`./regions/`*`region-name_admin`*`|`*`client`*`)_meta.f`	Lock files that the Daemons use to protect a region. Lock files normally exist only when a region is open.
`./regions/`*`region-name`*`.RGN`	Lock file that WebLogic Replicated Store configuration uses to protect a region. This lock file continues to exist even after the region is closed. It can only be deleted after a WebLogic Replicated Store configuration is deleted.

3.2.2.1 Configuration and Tuning Requirements for a Global Directory

The following section outlines configuration and tuning requirements for administrators creating Global Directories for associated Daemon Clusters:

- Create a Global Directory using the `startRSDaemon.sh` script. See Starting and Stopping a Daemon Cluster .
- There is a one-to-one correspondence between Daemon Clusters and Global Directories. Different Daemon Clusters cannot share the same Global Directory.
- A Global Directory is a tuned NFS mount that must be located on an Exalogic machine's ZFS Storage Appliance and must be centrally accessible by all Exalogic nodes that host components of an associated Daemon Cluster.
- To ensure stability, the NFS mount requires the following tuning:
  - Set `/etc/fstab "actimeo=0"` on each Exalogic node. This significantly impacts file operation performance so Oracle recommends restricting the use a Global Directory NFS mount to a Replicated Store.
  - Always use NFS v4 for all NFS clients and servers. See Configuring NFS Version 4 (NFSv4) on Exalogic in *Oracle Exalogic Elastic Cloud Machine Owner's Guide*.
  - A best practice is to minimize file activity in the ZFSA NFS volume hosting the Global Directory. For instance, if Daemon verbose tracing is required, use a volume other than the ZFSA NFS volume that hosts the Global Directory. See the `-logdir` parameter at Starting and Stopping a Daemon Cluster .

3.2.3 Administering a WebLogic Replicated Store

A WebLogic Server Replicated Store instance runs on a WebLogic Server, cluster, or migratable target and acts as a client to persist data to Regions in a Daemon Cluster. A particular instance can only attach to a Daemon that is running on the same node and shares the same Global Directory. Once attached, an instance creates and /or opens a set of Regions in a Daemon Cluster that are owned by (and uniquely named for) that instance. Lock files in the Global Directory ensure instances do not share Regions.

3.2.3.1 How to Create and Configure a WebLogic Replicated Store

For more information on how to create and configure a WebLogic Replicated Store, see Create replicated stores in *Oracle WebLogic Server Administration Console Online Help*.

3.2.3.2 Monitoring a Replicated Store

You can monitor statistics for each existing Replicated Store and for each open store connection.

3.2.3.2.1 Monitoring Stores

Each Replicated Store is represented at run time by an instance of the PersistentStoreRuntimeMBean, which provides the following options.

Table 3-4 Replicated Store Run-time Options

Option	What It Does
CreateCount	Number of create requests issued to this store.
ReadCount	Number of read requests issued to this store.
UpdateCount	Number of update requests issued by this store.
DeleteCount	Number of delete requests issued by this store.
ObjectCount	Number of objects contained in the store.
Connections	Number of active connections in the store.
PhysicalWriteCount	Number of times the store flushes its data to durable storage.

3.2.3.2.2 Monitoring Store Connections

For each open Replicated Store connection, a PersistentStoreConnectionRuntimemMBean is registered, which provides the following options.

Table 3-5 Replicated Store Connection Runtime Options

Option	What It Does
CreateCount	Number of create requests issued to this connection.
ReadCount	Number of read requests issued to this connection.
UpdateCount	Number of update requests issued by this connection.
DeleteCount	Number of delete requests issued by this connection.
ObjectCount	Number of objects contained in the connection.

Table 3-6 defines most of the run-time prefix names of the WebLogic services and subsystems that can create a connection to the Replicated Store.

Table 3-6 Replicated Store Run-Time Prefix Names

Subsystem/Service	Run-Time Prefix Name
JMS Service	JMS server: `weblogic.messaging.jmsServer.`*`internal`* where *`internal`* is the name of the JMS server connection JMS durable subscriber: `weblogic.messaging.jmsServer.durablesubs.`*`internal`* where *`internal`* is the name of the durable subscriber connection
Path Service	`weblogic.messaging.PathService.`*`internal`* where *`internal`* is the name of the path service connection
SAF Service	SAF agent `weblogic.messaging.SAFAgent@server1.`*`internal`* where *`internal`* is the name of the SAF agent's connection SAF durable subscriber: `weblogic.messaging.SAFAgent@server1.durablesubs.`*`internal`* where *`internal`* is the name of the durable subscriber connection

3.2.3.3 Best Practices and Considerations for WebLogic Replicated Stores

The following section provides information on best practices and other considerations when configuring WebLogic Replicated Store instances:

- If no Daemon is available on the same node as the instance, the Replicated Store fails to start and logs an error. For configuration options that cause a failing Replicated Store to automatically restart or migrate, see Server and Service Migration for Replicated Stores.
- If there are multiple Exalogic instances in a single node, they should connect to a single Daemon on each node. This promotes high availability by ensuring that each Daemon replicates its state to a Daemon that is running on a different node.
- If a non-production environment configures multiple Daemons on a single node, the `Local Index` attribute is used to determine the attachment. See LocalIndex in *MBean Reference for Oracle WebLogic Server*.
- The size of a region is configurable using the `RegionSize` parameter, see Replicated Store: Configuration in *Oracle WebLogic Server Administration Console Online Help*. The size of a Region cannot be changed once it's created; changing the `RegionSize` affects any new regions that an instance may create after the change is implemented.
- The region size must be larger than the largest single message, batch of messages, or SAF Window size processed when sending and receiving large

messages. Additionally, you may need to consider setting the `MaxMessageSize` as described in Setting Maximum Message Size for Network Protocols in *Tuning Performance of Oracle WebLogic Server*.

- Choose a region size that allows several regions to be created for each store instance that might share the same Daemon memory. For example: If a store instance will host up to 512 MB of data, choose a region size smaller than 50MB.
- Replicated Store instances recycle the space that they use within their regions. When data in a given memory location of a Region is no longer needed, the memory location is made available to store new data. A new region is created only when existing regions are too full to accommodate new data (for example, a growing backlog of unprocessed JMS messages).
- A region is not deleted and its backing machine memory is not freed until either:
  - The entire Daemon Cluster is shut down or
  - The WebLogic Replicated Store instance that references the region is shutdown and the region is administratively deleted. See Administering a Daemon Cluster using the Administration Utility.

3.2.4 Administering a Daemon Cluster

The following sections provide information on how to administer a Daemon Cluster:

- Configuring a Daemon
- Starting and Stopping a Daemon Cluster
- Logging Daemon Information
- Administering a Daemon Cluster using the Administration Utility
- Understanding Daemon Clusters

3.2.4.1 Configuring a Daemon

All Daemons in the same Daemon Cluster and all of the clients for these Daemons share the same `rs_daemons.cfg` file which is located at the root of their Global Directory.

3.2.4.1.1 Creating a rs_daemons.cfg File

The `rs_daemons.cfg` is a simple text file created by an administrator that contains a single entry for each Daemon in a Daemon Cluster. It may include optional blank lines and optional comments that are prefixed with a "`#`". Each line entry specifies an address, port, shared memory key, and optional memory limit using the following format where each value is separated by one or more blank spaces:

```
address port starting-shared-memory-key shared-memory-limit
```

- Address and Port—Each entry must specify a unique combination of address and port. No two entries can have both the same address and the same port. The address can be a name or a numeric IP but must correspond to an InfiniBand address on the Daemon's node. The port should be in the range 1024-49151.

Note:

It may be possible to configure port numbers higher than 49151 with additional OS tuning.

- Shared Memory Key—A dynamically generated key used to address the unique location of each Region's primary or secondary shared memory. Daemons that run on the same node must have different shared memory keys. If there are conflicts with the Shared Memory Key, the Daemon will not start. See Daemon Shared Memory Keys.
- Shared Memory Limit—Daemon's use shared memory to store Region data. A Daemon's shared memory limit is specified as an integer qualified by an "M" or "MB" for megabytes or a "G" or "GB" for gigabytes. Choose a memory limit that allows a Daemon to easily accommodate the predicted number of store instance primary and secondary Regions after a node fails. For example, in a three node cluster, each node should be able to accommodate a 50 percent surge in memory usage in case one node fails. See Managing Memory Utilization for Replicated Stores.

Example 3-1 provides an example contents for a `rs_daemon.cfg` file.

Example 3-1 Example Daemon Configuration in a rs_daemon.cfg File

```
#ipaddress   portno shmkey memlmt
 123.456.78.9 4545   4545   2G
 123.456.78.6 4546   4545   2G
```

A Daemon's *number* is automatically determined by its relative file location in the `rs_daemons.cfg` file (comments and blank lines are excluded). Daemon numbering starts with zero so the first entry in the file corresponds with Daemon number 0 and every additional line increases the daemon number by 1. This number is used to identify a daemon for logging and runtime administration purposes.

3.2.4.1.2 Editing a rs_daemons.cfg File

Oracle recommends that a `rs_daemons.cfg` file should not be modified if any Daemon in the Daemon Cluster is running. New clients and Daemons will fail to start if they attempt to connect to an existing Daemon Cluster after changes have been implemented. Existing running Daemons and clients continue to run using the old configuration.

3.2.4.2 Starting and Stopping a Daemon Cluster

Oracle provides a `startRSDaemon.sh` and `stopRSDaemon.sh` scripts to manage a Daemon Cluster. These scripts are located in the *WL_HOME*`/server/bin` directory of your Weblogic Server installation.

3.2.4.2.1 Starting Daemons using the startRSDaemon.sh Script

The `startRSDaemon.sh` script is used to start a Daemon using the information in the `rs_daemons.cfg` file located in the specified Global Directory and set the associated logging configuration.

Table 3-7 Configuration Parameters for the startRSDaemon.sh Script

Parameter	Description
`-dir` *`path`*	*path* specifies the location of the Replicated Store Global Directory that exactly matches the shared directory specified for a WebLogic Replicated Store in the WebLogic Server `config.xml` file and the shared directory for a Daemon Cluster. • If `"."` is specified as the value of *path*, the current directory is used. • This directory must contain an `rs_daemons.cfg` file. • Default value is `"."` This directory must be located on a specially tuned NFS mount, see Configuration and Tuning Requirements for a Global Directory. Oracle recommends using absolute paths when specifying the location of the Replicated Store Global Directory.
`-localindex` *`idx`*	*`idx`* specifies which particular local Daemon to start when more than one Daemon is configured to run on the current node. For use in development environments only, see Using a Local Index in Development Environments.
`-loglevel` *`num`*	*`num`* specifies the logging level for this Daemon. Defined log levels are: 0=None, 1=Error, 3 or higher enables debugging/ tracing. Default value is 2.
`-logdir` *`path`*	*`path`* specifies the path location for log files for this Daemon. When specified as a relative path, the directory is created relative to the *GlobalDir*`/log` directory, where *GlobalDir* specifies the path of the Global Directory. For example, if *GlobalDir* is `/scratch/rs_global_dir` then `-logdir ./logs` specifies `/scratch/rs_global_dir/log/log`. Specify an absolute directory path to save the log files to a location other than the *GlobalDir*`/log` directory. Default value is "."
`-logfilesize` *`num`*	*`num`* specifies the maximum size of an individual log file in MB. Default value is 500.
`-logfilemax` *`num`*	Maximum number of log files. Default value is 10.
`-?|-h|-help`	The help for this script.

3.2.4.2.2 Shutting Down Daemons using the stopRSDaemon.sh Script

The `stopRSDaemon.sh` script is used to stop a Daemon using the information in the `rs_daemons.cfg` file located in the specified Global Directory.

Table 3-8 Configuration Parameters for the stopRSDaemon.sh Script

Parameter	Description
`-dir` *`path`*	*path* specifies the location of the Replicated Store Global Directory that exactly matches the shared directory specified for a WebLogic Replicated Store in the WebLogic Server `config.xml` file and the shared directory for a Daemon Cluster. • If `"."` is specified as the value of *path*, the current directory is used. • This directory must contain an `rs_daemons.cfg` file. • Default value is `"."` This directory must be located on a specially tuned NFS mount, see Configuration and Tuning Requirements for a Global Directory. Oracle recommends using absolute paths when specifying the location of the Replicated Store Global Directory.
`-localindex` *`idx`*	*`idx`* specifies which particular local Daemon to stop when more than one Daemon is configured to run on the current node. For use in development environments only, see Using a Local Index in Development Environments. Default value is 0.
`-force` | `-safe`	Where: • `force` allows the command to shutdown a daemon with risk of data loss, including regions that are currently opened by a Replicated Store. Affected Replicated Stores will fail and report an Error. Daemon regions are not resilvered (see Daemon High Availability). • `safe` protects against data loss by using the following protocol during daemon shutdown: 1. Regions associated with the specified Daemons are checked to see if they are still open. If open regions are found, no Daemons are shutdown, an error message is printed, and the command exits. 2. Memory is checked to determine if there is enough available memory to resilver. If not enough memory is available, an error message is printed, and the command exits. 3. Any attempt to open a store is blocked until all regions are resilvered. If the Daemon runs out of memory, the command blocks until more memory is made available. 4. The specified Daemons are shutdown. Default value is `-safe`
`-?|-h|-help`	The help for this script.

3.2.4.3 Logging Daemon Information

Daemons report initial bootstrapping information to the WebLogic Server Administration Console (`stdout`) and generate one or more unique log/trace files in the root of the RS Global Directory. The log file contains any information that has been reported to the WebLogic Server Administration Console as well as runtime log messages. A Daemon's log file is named using the following pattern:

`rs_daemon_`*dnum_xxx.xxx.xxx.xxx_ppppp_nnn*`.log`

where:

- *dnum* specifies the three digit, zero filled value of the Daemon number in the `rs_daemons.cfg` file. See Creating a rs_daemons.cfg File.
- *xxx.xxx.xxx.xxx* specifies the IP address.
- *ppppp* specifies the port number.
- *nnn* specifies the log file number.

The log configuration (log file location, maximum file size, and maximum number of log files) is controlled by parameters in the `startRSDaemon.sh` script, see Starting Daemons using the startRSDaemon.sh Script. Log files are rotated so that old messages are moved to another file when the current log file reaches a specific size determined by the `logfilesize` parameter. The newest log file has the number 000 and the oldest possible log file would have the value of `logfilemax - 1`. If *nnn* for a log file is equal to `logfilemax`, it is deleted.

3.2.4.4 Administering a Daemon Cluster using the Administration Utility

The administration utility enables administrators to troubleshoot a Daemon cluster, including commands to manage running Daemons and associated Regions. This utility can be run from a Java command line as shown in Examples Demonstrating the Administration Utility.

Note:

Replicated Store commands are not supported using WLST.

To administer a Daemon Cluster, you must first attach to a local Daemon of the Daemon Cluster using the `rsattach` command.The most common use cases for the utility are to monitor memory utilization across the Daemon Cluster, delete unused Regions, and to shutdown Daemons.When finished, use the `rsdetach` command to disconnect from the Daemon Cluster.

Table 3-9 defines the available Replicated Store administration commands.

Table 3-9 Replicated Store Administration Options

Java Command	Parameters	What It Does
`rshelp`		If no parameter is specified, displays all available Replicated Store commands, usage, and examples. Where: *parameter* is the name of a command. If the name of a command is specified, additional help specific to the named command is presented. The administration utility does not need to be attached to a Daemon to use this command.
`rsattach`	`-dir` *`direcoryname`*	Attaches administration utility to a Daemon. Prompt changes back to `storeadmin:[RS]->` if successful. Success or failure to attach is logged to `stdout`. Where: *`directoryname`* is a non-null string specifying the relative or absolute location of the Global Resources Directory where the `rs_daemons.cfg` file exists for this Daemon. The default value is "." (the root).
	`-localindex` *`num`*	Where: *`num`* is a number used to determine which local Daemon to attach to when there are multiple Daemon entries that match the current node.The default value is 0 with a range from 0 to `MAX_INT`.
`rsdetach`		Disconnect from a Daemon Cluster. Prompt changes back to `storeadmin->` if successful. Success or failure to detach is logged to `stdout`.
`lsd`	`[-daemon all|local|n[,n]*|n-n]`	Monitors Daemons. Where: • `all` indicates all Daemons. • `local` indicates the currently attached Daemon. • `n` indicates a specific Daemon number. • `n[,n]*` indicates a comma separated list of Daemons • `n-n` indicates a range of Daemon numbers. • Default value is `all`.
	`[-sort name|time|size]`	Sorts list or table output. Where: • `name` sorts by Daemon number. • `time` sorts by newest to oldest. • `size` sorts by smallest to largest. • Default is `name`.

Table 3-9 (Cont.) Replicated Store Administration Options

Java Command	Parameters	What It Does
	[-format table | list]	Format of the output. Where: • `table` generates a table report, see Example Table Output from the lsd Command. • `list` generates a record report, see Example List Output from the lsd Command. • Default is `table`.
`shutdown`	`-daemon all|local|n[,n]*|n-n`	Shuts down Daemons. The administration utility automatically detaches from a shutdown Daemon. Where: • `all` indicates all Daemons. • `local` indicates the currently attached Daemon. • `n` indicates a specific Daemon number. • `n[,n]*` indicates a comma separated list of Daemons • `n-n` indicates a range of Daemon numbers.
	`-force|-safe`	Where: • `-force` allows the command to shutdown a daemon with risk of data loss, including regions that are currently opened by a Replicated Store. Affected Replicated Stores will fail and report an Error. Daemon regions are not resilvered. • `-safe` protects against data loss by using the following protocol during daemon shutdown: 1. Regions associated with the specified Daemons are checked to see if they are still open. If open regions are found, no Daemons are shutdown, an error message is printed, and the command exits. 2. Memory is checked to determine if there is enough available memory to resilver. If not enough memory is available, an error message is printed, and the command exits. 3. Activity to open store is blocked until all regions are resilvered. If the Daemon runs out of memory, the command blocks until more memory is made available. 4. The specified Daemons are shutdown.

Table 3-9 (Cont.) Replicated Store Administration Options

Java Command	Parameters	What It Does
`lsr`	`[-daemon all|local|n[,n]*|n-n]`	Lists the Regions of a Daemon Cluster. Where: • `all` indicates all Daemons. • `local` indicates the currently attached Daemon. • `n` indicates a specific Daemon number. • `n[,n]*` indicates a comma separated list of Daemons • `n-n` indicates a range of Daemon numbers. • Default value is `all`.
	`[-sort name|time|size]`	Sorts list or table output. Where: • `name` sorts by Daemon number. • `time` sorts by newest to oldest. • `size` sorts by smallest to largest. • Default is `name`.
	`[-format [table|list]]`	Format of the output. Where: • `table` generates a table report. • `list` generates a record report. • Default is `table`.
	`[exp]`	Where `exp` is a limited regular expression with support for the * wildcard, where * indicates zero or more of any character. For example, *A*B*C* matches region names that start with *A*, have a *B* in the middle, and end in *C*.
`rmr`	`-force exp`	Deletes all copies of specified Regions, including primary and secondary copies, unless a Region is currently opened by a Replicated Store. If required, you can use the `-force` parameter to delete open regions. Where: • [-force] allows a region to be deleted even if it is currently opened by a Replicated Store. Any WebLogic Replicated Store that was associated with open regions will fail and log an error. • `exp` is a limited regular expression with support for the * wildcard, where * indicates zero or more of any character. For example, *A*B*C* matches region names that start with *A*, have a *B* in the middle, and end in *C*.
`quit`		Ends the administration session.

3.2.4.4.1 Examples Demonstrating the Administration Utility

Before you start, set your shell environment by running a script such as `$WL_HOME/server/bin/setWLEnv.sh`.

To open the administration utility from a Java command line, type: **java weblogic.store.Admin**. For example:

```
> java weblogic.store.Admin
> storeadmin->
```

3.2.4.4.1.1 Accessing rshelp for Replicated Stores

Type `rshelp` for detailed descriptions on available Replicated Store administration commands, as well as examples of typical command usage. For example, the following comprehensive help is provided for the `rsdetach` command, which is used to release the utility from the Daemon.

```
storeadmin->rshelp rsdetach
Command:
   rsdetach
Summary:
    detach from a RS Daemon Cluster
Usage:
   rsdetach
Description:
Use "rsdetach" to detach from a RS Daemon Cluster.

When an rsdetach succeeds, the command prompt will
change so that it no longer includes '[RS]'.
. . .
```

3.2.4.4.1.2 Attaching to a Daemon

This example attaches the administration utility to a Daemon.

```
storeadmin:-> rsattach
INFO:  Attached to Daemon 3 with global dir [.].
storeadmin:[RS]->
. . .
```

3.2.4.4.1.3 Example Table Output from the lsd Command

The following section provides example of table output from an `lsd` command.

```
Current Time: 2014-02-18 11:50:29 AM EST
Idx IP              Port   Up          Rgns   Rgns   Rgns   Rgns   Mem     Mem
    Address                Time        Closed Open   Open   Total  Used    Used
                           HHHHH:MM:SS        Prima  Rplca         MB      %
--- --------------- ------ ----------- ------ ------ ------ ------ ------- -----
000   192.168.0.127   8000     0:27:28      0      0      2      2     257  3.14
001   192.168.0.128   8000     0:27:27      0      2      0      2     257  3.14
```

3.2.4.4.1.4 Example List Output from the lsd Command

The following section provides example of table output from an `lsd` command.

```
                              Index : 001
                             Status : UP
                  Reachable over IB : TRUE
                         IP Address : 192.168.0.128
                               Port : 8000
             Shared Memory Key (hex) : 0x1f40
         Shared Memory Key (decimal) : 8000
  Startup Time (YYYY-MM-DD HH:MM:ss) : 2014-02-18 11:23:02 AM EST
  Current Time (YYYY-MM-DD HH:MM:ss) : 2014-02-18 11:55:40 AM EST
              Up Time (HHHH:MM:ss) : 00019:32:37
   Startup Time (micros since epoch) : 1392740582473595
   Current Time (micros since epoch) : 1392742540074899
                  Up Time (micros) : 1957601304
        Total Configured Memory (MB) : 8192
           Current Memory Usage (MB) : 257
            Number of Regions Closed : 0
 Number of Regions Opened as Primary : 2
 Number of Regions Opened as Replica : 0
     Total Number of Regions Managed : 2
     Number of Resilvers in Progress : 0
        Number of Daemons in Cluster : 2
```

3.2.4.5 Understanding Daemon Clusters

A Daemon Cluster is replicated memory storage that spans multiple Exalogic nodes. This storage is organized as a set of uniquely named Regions and managed by one or more Daemons.

A WebLogic Replicated Store persists data to a Daemon Cluster, analogous to a File or JDBC Store where a Region in a Daemon Cluster corresponds to a File Store file or JDBC Store table. Once a WebLogic Replicated Store attaches to a Daemon, subsequent communication uses shared memory and InfiniBand RDMA.

- Region Uniqueness
- Daemon High Availability
- Daemon Shared Memory Keys
- Administering a Global Directory
- Using a Local Index in Development Environments

3.2.4.5.1 Region Uniqueness

A WebLogic Server Replicated Store instance can open one or more Regions in a Daemon Cluster. Similar to File Store files and JDBC Store tables, a Region can only be safely accessed by a single client at a time. To ensure Region integrity, Oracle uses lock files and configuration checks where ever possible to prevent more than one client from accessing a Region at a given time.

Note:

If more than one WebLogic domain is configured to use the same Daemon Cluster, you must ensure that the WebLogic domains are unique. Otherwise, same-named WebLogic Replicates Stores between the domains will clash attempting to open the same Region resulting in lock errors in the WebLogic Server and Daemon logs.

3.2.4.5.2 Daemon High Availability

A Daemon Cluster uses *resilvering*, the process of creating a new synchronized copy of a Region from an existing Region, to ensure a Region has two copies of each Region on separate Daemons.

When a WebLogic Replicated Store opens a new Region, its local Daemon is responsible for maintaining the Region's primary copy and the next available Daemon maintains the Region's secondary copy. The next available Daemon is determined by:

- Finding the primary's nearest subsequently defined Daemon in the `rs_daemons.cfg` file that is up and running, or
- If the primary happens to be the last Daemon in the file, then start at the top of the `rs_daemons.cfg` file and look for the next subsequently defined Daemon.

When a Replicated Store opens an existing Region that has no pre-existing copy on the store's local Daemon but already has copies elsewhere in the cluster, then the open succeeds and one of the existing copies is transparently resilvered to the local Daemon as part of the Region open. In addition, the Region's secondary is resilvered to the next available Daemon. The resilvering of the secondary helps ensure that Regions stay evenly distributed throughout a Daemon Cluster.

Note:

A Region can only be opened by a single client at a time. WebLogic Replicated Stores are expected to restart and/or migrate to a new node in order to open and recover their Regions after failing.

3.2.4.5.2.1 What Happens When a Daemon Fails?

If a Daemon fails, any attached client (for example, a Weblogic Replicated Store) also fails. Clients can recover their particular Region data by periodically trying to reattach to the failed Daemon or by trying to attach to a different Daemon in the same Daemon Cluster. For information on automating periodic retries, see Server and Service Migration for Replicated Stores.

The other Daemons in the cluster detect the failure, and each of the failed Daemon's primary and secondary Region copies are automatically resilvered to another Daemon (provided there's a Daemon with enough memory available). Resilvering occurs even if the WebLogic Replicated Store does not restart and reattach somewhere else in the cluster.

Note:

If resilvering is in progress and if a Daemon Cluster is down to a single Daemon, or if a Daemon Cluster doesn't have enough memory to host two copies of each region, then Regions are vulnerable to a single point of failure.

3.2.4.5.2.2 Additional Considerations

- An administrator can restart a Daemon in place and it will then rejoin the Daemon Cluster.
- A Daemon can be administratively shutdown safely using administration utility (see Administering a Daemon Cluster using the Administration Utility) or Daemon shutdown script (see Shutting Down a Replicated Store). By default, a shutdown resilvers all of a Daemon's Regions (primaries and secondaries) and blocks until the resilvering completes prior to shutting down the Daemon.
- If a Region is resilvering a primary to a new secondary and the secondary's host Daemon is killed, then resilvering transparently and asynchronously starts over with a new secondary on another Daemon. There must be at least two Daemons still running in the cluster and there is another Daemon with sufficient free memory to host the Region copy.

3.2.4.5.3 Daemon Shared Memory Keys

Daemons maintain shared memory to store Region data. Each new Region (primary or secondary) has its own unique location that is internally addressed using a dynamically generated private Shared Memory Key. This memory is pinned to prevent the O/S from paging the information to disk. This shared memory is only freed when:

- A Daemon crashes, is killed, or is administratively shutdown.
- A Region is administratively deleted.

An administrator configures a candidate public Shared Memory Key for each Daemon in the `rs_daemons.cfg` file (one public key shared memory location per Daemon). For example, if a Daemon hosts 5 Region primaries and 4 Region secondaries, then it reserves a total of 9 dynamically generated private shared memory keys and one public key.

3.2.4.5.4 Using a Local Index in Development Environments

A local index determines which particular local Daemon to start when more than one Daemon is configured to run on the current node. The following formula determines which Daemon is picked:

```
((idx) modulo (number-of-local-daemons))
```

Where *idx* is an entry in the `rs_daemons.cfg` file. By default (*idx*=0), this resolves to the first entry in the `rs_daemons.cfg` file with the same network address as the current node.

Note:

Using a local index is not recommended for production environments. To ensure high availability in a production environment, configure a Daemon Cluster on multiple nodes and assign one Daemon on each node.

3.2.5 Managing Memory Utilization for Replicated Stores

Each Daemon in a Daemon Cluster has a shared memory limit defined in the associated `rs_daemons.cfg` file to guard against over aggressive memory usage. Oracle provides defined and tunable system properties to manage memory pressure events and prevent catastrophic failure of a Replicated Store.

- Replicated Store Memory Usage
- Monitoring Memory Utilization
- Controlling Memory Utilization
- Replicated Store Behavior when Exceeding Available Memory

For more information on setting the shared memory limit of a Daemon, see Creating a rs_daemons.cfg File.

3.2.5.1 Replicated Store Memory Usage

The following section describes actions that change Daemon's memory utilization:

- A Daemon's memory usage only decreases when:
  - An attached Replicated Store is migrated to another Daemon.
  - A Replicated Store's regions are administratively deleted.
  - A Region's hosted secondary copy moves to another Daemon.
- A Daemon memory usage only increases when an attached Replicated Store instance's current data set increases to the point where it needs to create new Regions.
- When a Replicated Store instance deletes a message record, it's Region memory is unchanged and remains allocated for storing subsequent new records.

Note:

Similar to all other types of persistence, Replicated Store persisted messages are cached in WebLogic Server JVM memory. You can tune JMS Server paging to reduce this memory usage, however it may result in a severe performance impact.

3.2.5.2 Monitoring Memory Utilization

The following section provides information on how administrators can monitor Daemon shared memory utilization:

- The Administration Utility provides commands that report memory usage for Regions and Daemons. See Administering a Daemon Cluster using the Administration Utility.
- The UNIX `ipcs` command provides monitoring information on shared memory keys.
- Daemon logging when the trace level is set to 3 or higher. As the memory usage goes above or below each 10 percent increment of the `shared-memory-limit`, the Daemon logs the memory usage. Memory utilization messages are written at the `INFO` logging level until memory utilization rises to the 80 percent warning threshold, where all memory utilization logging information is written at the `WARNING` logging level. See Configuring a Daemon.
- WebLogic Server logs the shared memory limits of a Daemon:
  – Logs a memory usage `INFO` message as defined by the `SpaceLoggingStartPercent` of the `shared-memory-limit`.
  – Logs a memory usage `WARNING` message as defined by the `SpaceOverloadYellowPercent` of the `shared-memory-limit`.
  – Logs a memory usage ERROR message as defined by the `SpaceOverloadRedPercent` of the `shared-memory-limit`.

  See Table 3-10 for information on the tuning memory utilization behavior of WebLogic Server.
- Monitoring Byte and Message current and pending counts on JMS servers and destinations.

3.2.5.3 Controlling Memory Utilization

The following section provides information on how administrators can monitor Daemon shared memory utilization:

- The `shared-memory-limit`. Daemons check that the `shared-memory-limit` is less than or equal to the operating system shmlimit and memlock limits. See Creating a rs_daemons.cfg File.
- WebLogic Server provides tunable system properties to manage the memory usage of a Replicated Store and associated Daemons and to protect against overload situations. These control the way that the system logs a Replicated Store's memory usage, the maximum size of a message that can be put into a destination that uses a replicated store, and the thresholds that trigger overload protection actions.The properties in Table 3-10 to all replicated stores on a WebLogic server instance, or a particular store named *`store-name`*.

Table 3-10 Tunable System Properties for Memory Pressure Management

Property	Type	Default Value	Range	Description

`weblogic.store.replicated.MaximumMessageSizePercent`
`weblogic.store.replicated.`*`store-name`*`.MaximumMessageSizePercent`
	int	1% of Replicated Store's region size	1 to 100	The maximum message size for a JMS destination that is backed by a replicated store, specified as a percentage of the store region size. New messages that exceed this size get a `ResourceAllocationException`.
				The total size of all concurrently written replicated store messages must be less than the Region size or failures can result. See related settings for the “JMS Server or Destination Maximum Message Size".

`weblogic.store.replicated.SpaceLoggingStartPercent`
`weblogic.store.replicated.`*`store-name`*`.SpaceLoggingStartPercent`
	int	70	1 to 100	The percentage of the Daemon Shared Memory Limit when a Replicated Store starts logging Daemon shared memory usage.

`weblogic.store.replicated.SpaceLoggingDeltaPercent`
`weblogic.store.replicated.`*`store-name`*`.SpaceLoggingDeltaPercent`
	int	10	1 to 100	Replicated store daemon space usage logging delta.
				For example: The default is 10 percent, which means the store will log every 10 percent of space usage change.

`weblogic.store.replicated.SpaceOverloadYellowPercent`
`weblogic.store.replicated.`*`store-name`*`.SpaceOverloadYellowPercent`
	int	80	1 to 100	If the memory used by all Replicated Stores on a Daemon exceeds this percentage of the Daemon Shared Memory Limit, and also the data stored in the store exceeds this percentage of the total region memory allocated for the particular store, then a JMS server will reject new messages with a `ResourceAllocationException`.

`weblogic.store.replicated.SpaceOverloadRedPercent`
`weblogic.store.replicated.`*`store-name`*`.SpaceOverloadRedPercent`
	int	90	1 to 100	If the memory used by all Replicated Stores on a Daemon exceeds this percentage of the Daemon Shared Memory Limit, new or migrated stores will fail to open and log an `ERROR` message.

- Tune messaging quota attributes `Bytes Maximum`, `Messages Maximum`, and `Maximum Message Size` on destinations and JMS Servers. Messages that exceed these quota limits cause sending applications to receive `ResourceAllocationException`s.

3.2.5.4 Replicated Store Behavior when Exceeding Available Memory

In the rare situation where there's not enough memory for an already running WebLogic Replicated Store to allocate a new Region, the Store will fail, close, and log an `ERROR` message. As with any Store instance failure, stores that fail to start or close due to exceeding available memory can be automatically restarted or migrated using Automatic Service Migration or Whole Server Migration. See Server and Service Migration for Replicated Stores.

3.3 Interoperability Considerations for a Replicated Store

This section provides information on the interoperability of the Replicated Store:

- Future releases of the replicated store may be incompatible with this release. If so, the Replicated Store fails to open, a version incompatibility message is logged, and the associated Region is left unmodified.
- A WebLogic Replicated Store or Daemon Cluster will not start on an unsupported platform and will log an error message. For information on supported platforms, see Exalogic Elastic Cloud Software in *Licensing Information*.

3.4 Security Considerations for a Replicated Store

This section provides security considerations when using Replicated Stores:

- Security Considerations for Administrators
- General Security Considerations for Replicated Stores

3.4.1 Security Considerations for Administrators

The following section provides security requirements Administrators must implement to secure Replicated Stores:

- All Daemons must be started using the same UID and be able to write to the Global Directory. A Daemon can not communicate with other Daemons in a cluster unless it has read/write permission on the Global Directory and specifies the same Global Directory as the other Daemons.
- The administration utility must have the same UID as the Daemons. The administration utility can not attach unless it has read/write permission on the Global Directory and this directory is the same as the Daemon's Global Directory.
- WebLogic Server must have the same GID as the Daemon it's attached to in order to access to shared memory.
- The administration utility must have the same UID as the Daemon it's attached to.
- Oracle recommends that Daemons should be run as regularly privileged user, but a Daemon executable binary still needs to be assigned UNIX `root`, `set uid`, and `set group` privileges so that a Daemon has permission to raise its own process priority and to change its runtime UID and group to the UID and group of the user that starts the Daemon. See Starting and Stopping a Daemon Cluster .

Note:

If a Daemon executable doesn't have sufficient privileges, the Daemon start script will print the instructions necessary to assign the privileges to a Daemon.

3.4.2 General Security Considerations for Replicated Stores

The following section provides information on how secure access is enforced between Replicated Store components:

- Global Directory Access Permissions—The Global Directory must be read/write accessible to the O/S user that launches clients and Daemons.If a user doesn't have permission to access this directory, the user can not launch Daemons or clients that use this directory.
- Same Directory Verification—A Daemon verifies that connection requests from other Daemons, Replicated Stores, or the administration utility refer to the same Global Directory by:
  - Ensuring the directory path matches and its `rs_daemons.cfg` checksum matches.
  - Verifying that the requestor can send the value of a UUID that it has written to that directory (a random shared secret). The UUID changes once per Daemon restart.
- Shared Memory Access Permissions—A Daemon creates its shared memory with group only permissions, and only clients with a user that has matching group permissions can use this shared memory.

3.5 Limitations of a Replicated Store

The following limitations apply to Replicated Stores:

- A Replicated Store should not be opened simultaneously by two server instances; otherwise, there is no guarantee that the data in a Region will not be corrupted. If possible, the Replicated Store will attempt to return an error in this case, but it will not be possible to detect this condition in every case. To help guard against this possibility, it is the responsibility of the administrator to ensure that no two same named domains, each with a same named Replicated Store, attempt to use the exact same Daemon Cluster. Two Replicated Stores will conflict if they have the same name, the same domain name, and the same Global Directory.
- Daemon Logging is not integrated with WebLogic Server diagnostics, Java Flight Recorder (JFR), logging, or DebugMBeans.

4 WebLogic Server Cooperative Memory Management in Oracle Exalogic Elastic Cloud Environments

WebLogic Server uses Cooperative Memory Management to promote effective memory utilization by WebLogic Server resources.
This chapter includes the following sections:

- What is Cooperative Memory Management?
- Using CMM in Your Environment
- CMM for WebLogic Data Sources

4.1 What is Cooperative Memory Management?

Cooperative Memory Management (CMM) is a WebLogic feature for Exalogic Elastic Cloud environments that promotes effective memory utilization by allowing WebLogic Server resources to adjust memory utilization based on the memory utilization of the whole machine.

When CMM is enabled, the memory usage of the operating system is monitored and a relative pressure level is published by a memory pressure agent. The published pressure level ranges from 0 (no pressure) through 10, with 10 being the highest pressure level (severe overload). WebLogic Server resources that have registered a memory pressure listener react to changes in the memory pressure levels by writing an INFO message to the server log and appropriately managing server resources to increase or decrease the utilization of available memory.

In high or rising memory pressure conditions, resources registered with the memory pressure agent implement strategies to reduce memory utilization to prevent server overload such as:

- Reducing the size of caches
- Compressing in-memory data that can be compressed
- Rejecting new work
- Implements graceful server recovery by temporarily degrading performance and reducing system thrashing.

Any actions taken by a server at a given pressure level remain in effect until the measured memory pressure by the agent changes. For example, if the data source memory pressure listener receives a pressure reading of 4, the available statement cache size is reduced to 60 percent. In order for the actual amount of statement cache used to improve to 75 percent, the memory pressure level reported by the memory pressure agent must change to 2, see Table 4-1.

4.2 Using CMM in Your Environment

CMM is only available for use in Exalogic Elastic Cloud environments. See Exalogic Elastic Cloud Software in *Licensing Information*.

To enable CMM in your environment:

- If not present, add the following to your `${DOMAIN_HOME}/bin/startWebLogic.sh` script used to start the WebLogic Servers in your environment. For example:

  ```
  . . .
  if [ -f ${WL_HOME}/server/bin/saveMemory.sh ] ; then
        . ${WL_HOME}/server/bin/saveMemory.sh
  fi
  . . .
  ```

  Create a `saveMemory.sh` file located at `${WL_HOME}/server/bin/` that contains the following:

  ```
  #!/bin/sh
  # The following flag is used to enable CMM
  export SAVE_MEMORY=true
  ```

- Ensure all servers in your environment use Exalogic optimizations, see Enabling Exalogic Optimizations.
- Restart all servers in your environment to enable the changes.

4.3 CMM for WebLogic Data Sources

WebLogic Server data sources can be configured to respond to memory pressure events by making configurable adjustments to statement cache sizes and the shrinking of connection pools:

- Memory Management for Statement Cache Size
- Memory Management for Shrinking Data Sources

4.3.1 Memory Management for Statement Cache Size

When a non-zero memory pressure state change is received, the prepared statement caches for all deployed datasources are adjusted according to the configured mapping of memory pressure level to percent cache size. By default, WebLogic datasources have a statement cache size of 10 for every pooled JDBC connection.

> **Note:**
>
> Datasource configurations that have disabled the statement cache (by setting the cache size to 0) or are using the Oracle JDBC driver statement cache are not affected by memory pressure events.

The default memory pressure level to cache size weights are graduated as described in the following table:

Table 4-1 Default Memory Pressure Cache Size Weights

Level	Cache Size, %
0	100
1	90
2	80
3	70
4	60
5	50
6	40
7	30
8	20
9	10
10	0

The default statement cache weights can be changed at server start time by setting the following Java system property:

```
-Dweblogic.jdbc.cmm.statementCacheWeights=weights
```

where *weights* is one or more name/value pair s of the form *level=percent reduction*, *level=percent reduction*,…

Using this notation the default weights are as follows:

```
1=10,2=10,3=10,4=10,5=10,6=10,7=10,8=10,9=10,10=10
```

The percentages are cumulative and the sum must be equal to or less than 100. At each level the percentage value indicates how much the configured cache size is reduced.

4.3.1.1 Example

The following example shows statement cache sizes corresponding to the various memory pressure levels when the server is started with the system property:

```
-Dweblogic.jdbc.cmm.statementCacheWeights=2=20,4=20,6=20,8=20,10=20
```

Table 4-2 Example Memory Pressure Cache Size Weights

Level	Cache Size, %
0	100
1	100

Table 4-2 (Cont.) Example Memory Pressure Cache Size Weights

Level	Cache Size, %
2	80
3	80
4	60
5	60
6	40
7	40
8	20
9	20
10	0

4.3.2 Memory Management for Shrinking Data Sources

Upon receiving a memory pressure level event where the pressure level is greater than the pool shrink threshold (8 by default), WebLogic Server issues a shrink operation to all deployed datasources. The shrink operations action is a one-time event initiated on every memory pressure level event where the memory pressure level exceeds the threshold value.

The default shrink threshold can be modified at server start time by setting the following Java system property:

```
-Dweblogic.jdbc.cmm.shrinkThreshold=value
```

Where *value* is an integer in the range 0 to 10.

5

Message Compression In Oracle Exalogic Elastic Cloud Environments

WebLogic Server provides the ability to configure message compression for JMS Store I/O operations which may provide significant performance improvements in Oracle Exalogic environments.
By selecting an appropriate message body compression option, JMS store I/O performance may improve for:

- Persistent messages that are read from or written to disk.
- Persistent and non-persistent messages are paged in or paged out when JMS paging is enabled.

The following sections provide information on how to configure message compression:

- Selecting a Message Compression Option
- Message Compression for JMS Servers in Exalogic Environments
- Message Compression for Store-and-Forward Sending Agents in Exalogic Environments

For general tuning information on JMS message compression, see Compressing Messages in *Tuning Performance of Oracle WebLogic Server*.

5.1 Selecting a Message Compression Option

This section provides information on the types of message compression available for use when message body compression is enabled.

Note:

The performance of each compression option is dependent on the operating environment, data type, and data size. Oracle recommends users test their environments to determine the most appropriate compression option.

Table 5-1 Message Body Compression Options

Compression Type	Description
GZIP DEFAULT_COMPRESSION	Use `GZIP_DEFAULT_COMPRESSION` to enable message compression using the JDK GZIP API with `DEFAULT_COMPRESSION` level. See java.util.zip package.

Table 5-1 (Cont.) Message Body Compression Options

Compression Type	Description
GZIP BEST_COMPRESSION	Use `GZIP_BEST_COMPRESSION` to enable message compression using the JDK GZIP API with `BEST_COMPRESSION` level. See java.util.zip package.
GZIP BEST_SPEED	Use `GZIP_BEST_SPEED` to enable message compression using the JDK GZIP API with `BEST_SPEED` level. See java.util.zip package.
LZF	Use LZF to enable message compression using Open Source LZF. See https://github.com/ning/compress.

5.2 Message Compression for JMS Servers in Exalogic Environments

To configure message body compression for JMS servers:

1. If you have not done so, create a JMS Server, see Create JMS servers in the *Oracle WebLogic Server Administration Console Online Help*
2. Use the instructions to Configure general JMS server properties in the *Oracle WebLogic Server Administration Console Online Help*. Update the following **Advanced** JMS server attributes for your environment:
   a. Optionally, select **Store Message Compression Enabled** to enable the JMS store to perform message body compression. See StoreMessageCompressionEnabled in *MBean Reference for Oracle WebLogic Server*.
   b. Optionally, select **Paging Message Compression Enabled** to enable the JMS paging store to perform message body compression on persistent and non-persistent messages. See PagingMessageCompressionEnabled in *MBean Reference for Oracle WebLogic Server*.
   c. In **Message Compression Options**, specify the type of message compression used. See MessageCompressionOptions in *MBean Reference for Oracle WebLogic Server*.

5.3 Message Compression for Store-and-Forward Sending Agents in Exalogic Environments

To configure message body compression for SAF Sending Agents:

1. If you have not done so, create a SAF Sending Agent, see Create Store-and-Forward agents in the *Oracle WebLogic Server Administration Console Online Help*
2. Use the instructions to Configure SAF agent general properties in the *Oracle WebLogic Server Administration Console Online Help*. Update the following **Advanced** Sending Agent attributes for your environment:

a. Optionally, select **Store Message Compression Enabled** to enable the JMS store to perform message body compression. See StoreMessageCompressionEnabled in *MBean Reference for Oracle WebLogic Server*.

b. Optionally, select **Paging Message Compression Enabled** to enable the JMS paging store to perform message body compression on persistent and non-persistent messages. See PagingMessageCompressionEnabled in *MBean Reference for Oracle WebLogic Server*.

c. In **Message Compression Options**, specify the type of message compression used. See MessageCompressionOptions in *MBean Reference for Oracle WebLogic Server*.