Oracle® Fusion Middleware

Administering Oracle Coherence

14c (14.1.1.0.0)
F23523-16
October 2023

Oracle Fusion Middleware Administering Oracle Coherence, 14c (14.1.1.0.0)

F23523-16

Primary Author: Oracle Corporation

Contents

5 Production Checklist

7 Federating Caches Across Clusters

A Platform-Specific Deployment Considerations

B Log Message Glossary

Preface

Welcome to *Administering Oracle Coherence*. This document provides key administration concepts and detailed instructions for administering Coherence clusters and caches.

This preface includes the following sections:

Audience

This guide is intended for the following audiences:

- Primary Audience – Administrators and Operators who want to administer Coherence clusters in their network environment.
- Secondary Audience – System Architects and developers who want to understand the options for administering Coherence.

The audience should be familiar with Java and JavaEE. In addition, the examples in this guide require the installation and use of the Oracle Coherence product. Users should be familiar with running command line scripts.

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.

Access to Oracle Support

Oracle customers that 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.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our

initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documents

For more information, see the following documents in the Oracle Coherence documentation set:

- *Administering HTTP Session Management with Oracle Coherence*Web*
- *Developing Applications with Oracle Coherence*
- *Developing Remote Clients for Oracle Coherence*
- *Installing Oracle Coherence*
- *Integrating Oracle Coherence*
- *Managing Oracle Coherence*
- *Securing Oracle Coherence*
- *Java API Reference for Oracle Coherence*
- *C++ API Reference for Oracle Coherence*
- *.NET API Reference for Oracle Coherence*
- *Release Notes for Oracle Coherence*

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.

Part I
Basic Administration

There are many administrative tasks to consider when moving Coherence applications to a production environment. Tasks such as testing the network environment and tuning production systems are essential for a successful deployment.

Part I contains the following chapters:

1

Deploying Coherence Applications

Coherence can be deployed as a standalone application or as a Java EE application to an application server. Specific instructions are provided for WebLogic Server deployment in addition to instructions for generic application server deployment.
This chapter includes the following topics:

- Deploying Standalone Coherence Applications
- Deploying Coherence Applications on Docker and Kubernetes
- Deploying Coherence Applications to WebLogic Server
- Deploying Coherence Applications to an Application Server (Generic)
- Running Multiple Applications in a Single Cluster

Deploying Standalone Coherence Applications

Standalone Coherence applications are deployed as a set of distributed processes. For deployment, it is often beneficial to logically group these processes into tiers based on their role; however, it is not a requirement for deployment. The most common tiers are a data tier, application tier, proxy tier, and extend client tier. Tiers facilitate deployment by allowing common artifacts, packaging, and scripts to be defined and targeted specifically for each tier.
This section includes the following topics:

- Deploying a Data Tier
- Deploying an Application Tier
- Deploying a Proxy Tier for Extend Clients
- Deploying Extend Clients

Deploying a Data Tier

A data tier is comprised of cache servers that are responsible for storing cached objects. A Coherence application may require any number of cache servers in the data tier. The number of cache servers depends on the amount of data that is expected in the cache and whether the data must be backed up and survive a server failure. Each cache server is a Coherence cluster member and runs in its own JVM process and multiple cache server processes can be collocated on a physical server. See Cache Size Calculation Recommendations and Hardware Recommendations.

Cache servers are typically started using the `com.tangosol.net.DefaultCacheServer` class. The class contains a `main` method and is started from the command line. See Starting Cache Servers in *Developing Applications with Oracle Coherence*.

The following application artifacts are often deployed with a cache server:

- Configuration files such as the operational override configuration file, the cache configuration file and the POF user type configuration file
- POF serializers and domain objects

- Data grid processing implementations such as queries, entry processor, entry aggregators, and so on
- Event processing implementations
- Cache store and loader implementations when caching objects from data sources

There are no restrictions on how the application artifacts must be packaged on a data tier. However, the artifacts must be found on the server classpath and all configuration files must be found before the `coherence.jar` library if the default names are used; otherwise, the default configuration files that are located in the `coherence.jar` library are loaded. The following example starts a single cache server using the configuration files in the *APPLICATION_HOME*`\config` directory and uses the implementations classes in the *APPLICATION_HOME*`\lib\myClasses` library:

```
java -server -Xms4g -Xmx4g -cp
APPLICATION_HOME\config;APPLICATION_HOME\lib\myClasses.jar;COHERENCE_HOME\lib\c
oherence.jar com.tangosol.net.DefaultCacheServer
```

If you choose to include any configuration overrides as system properties (rather than modifying an operational override file), then they can be included as `-D` arguments to the `java` command. As a convenience, you can reuse the *COHERENCE_HOME*`\bin\cache-server` script and modify it as required.

GAR Deployment

Coherence application artifacts can be packaged as a Grid ARchive (GAR) and deployed with the `DefaultCacheServer` class. A GAR adheres to a specific directory structure and includes an application descriptor. See Building a Coherence GAR Module. The instructions are included as part of WebLogic server deployment, but are also applicable to a GAR being deployed with the `DefaultCacheServer` class.

The following example starts a cache server and uses the application artifacts that are packaged in the `MyGar.gar` file. The default name (`MyGAR`) is used as the application name, which provides a scope for the application on the cluster.

```
java -server -Xms4g -Xmx4g -cp
APPLICATION_HOME\config;COHERENCE_HOME\lib\coherence.jar
com.tangosol.net.DefaultCacheServer D:\example\MyGAR.gar
```

You can override the default name by providing a different name as an argument. See Overview of the DefaultCacheServer Class in *Developing Applications with Oracle Coherence*. For details about application scope, see Running Multiple Applications in a Single Cluster.

Deploying an Application Tier

An application tier is comprised of any number of clients that perform cache operations. Cache operations include loading objects in the cache, using cached objects, processing cached data, and performing cache maintenance. The clients are Coherence cluster members, but are not responsible for storing data.

The following application artifacts are often deployed with a client:

- Configuration files such as the operational override configuration file, the cache configuration file and the POF user type configuration file
- POF serializers and domain objects

- Data grid processing implementations such as queries, entry processor, entry aggregators, and so on
- Event processing implementations
- Cache store and loader implementations when caching objects from data sources

There are no restrictions on how the application artifacts must be packaged on an application tier. Clients must include the *COHERENCE_HOME*/lib/coherence.jar library on the application classpath. Coherence configuration files must be included in the classpath and must be found before the coherence.jar library if the default names are used; otherwise, the default configuration files that are located in the coherence.jar library are loaded. The following example starts a client using the configuration files in the *APPLICATION_HOME*\config directory and uses the implementations classes in the *APPLICATION_HOME*\lib\myClasses.jar library.

```
java -cp
APPLICATION_HOME\config;APPLICATION_HOME\lib\myClasses.jar;COHERENCE_HOME\lib\coherence
.jar com.MyApp
```

If you choose to include any system property configuration overrides (rather than modifying an operational override file), then they can be included as -D arguments to the java command. For example, to disable storage on the client, the tangosol.coherence.distributed.localstorage system property can be used as follows:

```
java -Dcoherence.distributed.localstorage=false -cp
APPLICATION_HOME\config;APPLICATION_HOME\lib\myClasses.jar;COHERENCE_HOME\lib\coherence
.jar com.MyApp
```

> **Note:**
>
> If a GAR is used for deployment on a cache server, then cache services are restricted by an application scope name. Clients must use the same application scope name; otherwise, the clients can not access the cache services. See Running Multiple Applications in a Single Cluster.

Deploying a Proxy Tier for Extend Clients

A proxy tier is comprised of proxy servers that are responsible for handling extend client requests. Any number of proxy servers may be required in the proxy tier. The number of proxy servers depends on the expected number of extend clients and the expected request load of the clients. Each proxy server is a cluster member and runs in its own JVM process and multiple proxy server processes can be collocated on a physical server. See Defining Extend Proxy Services in *Developing Remote Clients for Oracle Coherence*.

A proxy server is typically started using the com.tangosol.net.DefaultCacheServer class. The class contains a main method and is started from the command line. See Starting Cache Servers in *Developing Applications with Oracle Coherence*. There is no difference between a proxy server and a cache server.

The following application artifacts are often deployed with a proxy:

- Configuration files such as the operational override configuration file, the cache configuration file and the POF user type configuration file
- POF serializers and domain objects. If an extend client is implemented using C++ or .NET, then a Java version of the objects must also be deployed for certain use cases.

- Data grid processing implementations such as queries, entry processor, entry aggregators, and so on
- Event processing implementations
- Cache store and loader implementations when caching objects from data sources

There are no restrictions on how the application artifacts must be packaged on a proxy tier. However, the artifacts must be found on the server classpath and all configuration files must be found before the `coherence.jar` library; otherwise, the default configuration files that are located in the `coherence.jar` library are loaded. The following example starts a single proxy server using the configuration files in the *`APPLICATION_HOME`*`\config` directory and uses the implementations classes in the *`APPLICATION_HOME`*`\lib\myClasses` library:

```
java -server -Xms512m -Xmx512m -Dcoherence.distributed.localstorage=false -cp
APPLICATION_HOME\config;APPLICATION_HOME\lib\myClasses.jar;COHERENCE_HOME\lib\c
oherence.jar com.tangosol.net.DefaultCacheServer
```

GAR Deployment

Coherence application artifacts can be packaged as a Grid ARchive (GAR) and deployed with the `DefaultCacheServer` class. A GAR adheres to a specific directory structure and includes an application descriptor. See Building a Coherence GAR Module. The instructions are included as part of WebLogic server deployment, but are also applicable to a GAR being deployed with the `DefaultCacheServer` class.

The following example starts a proxy server and uses the application artifacts that are packaged in the `MyGar.gar` file. The default name (`MyGAR`) is used as the application name, which provides a scope for the application on the cluster.

```
java -server -Xms512m -Xmx512m -Dcoherence.distributed.localstorage=false -cp
APPLICATION_HOME\config;APPLICATION_HOME\lib\myClasses.jar;COHERENCE_HOME\lib\c
oherence.jar com.tangosol.net.DefaultCacheServer D:\example\MyGAR.gar
```

You can override the default name by providing a different name as an argument. See Starting Cache Servers in *Developing Applications with Oracle Coherence*. For details about application scope, see Running Multiple Applications in a Single Cluster.

Deploying Extend Clients

Extend clients are implemented as Java, C++, or .NET applications. In addition, any client technology that provides a REST client API can use the caching services in a Coherence cluster. Extend clients are applications that use Coherence caches, but are not members of a Coherence cluster. See Configuring Extend Clients in *Developing Remote Clients for Oracle Coherence*.

The following Coherence artifacts are often deployed with an extend client:

- Configuration files such as the operational override configuration file, the cache configuration file and the POF user type configuration file
- POF serializers and domain objects
- Data grid processing implementations such as queries, entry processor, entry aggregators, and so on
- Event processing implementations

Deploying Coherence Applications on Docker and Kubernetes

Oracle Coherence applications can be deployed as Docker containers and orchestrated using Kubernetes. These industry-leading standards offer a modern, cloud-neutral deployment solution that is highly scalable and easy to manage.
Oracle has released Docker files for Coherence and supporting scripts to GitHub. The posted files are examples to help you get started and include documentation and samples. The Coherence images are available for Java 8 and support both the Coherence full installation and the Coherence quick installation. See docker-images/OracleCoherence on GitHub.

In addition, Oracle has released the Coherence Kubernetes Operator to GitHub. The operator is a Coherence-specific Kubernetes controller that facilitates running and managing containerized Coherence applications on Kubernetes. The operator is installed using Helm and provides integrations with ELK (Elasticsearch, Logstash, and Kibana) for logging, and Prometheus and Grafana for monitoring. The operator includes documentation and samples to help you get started. See the coherence-kubernetes-operator project on GitHub.

To build custom Coherence Docker images, see the *coherence-docker readme* document on the Coherence CE GitHub repository - coherence-docker.

Deploying Coherence Applications to WebLogic Server

WebLogic Server includes a Coherence integration that standardizes the way Coherence applications can be deployed and managed within a WebLogic Server domain. The integration allows administrators to set up distributed Coherence environments using familiar WebLogic Server components and infrastructure, such as Java EE-styled packaging and deployment, remote server management, server clusters, WebLogic Scripting Tool (WLST) automation, and configuration through the Administration Console.
This section includes the following topics:

- Overview of the WebLogic Server Coherence Integration
- Packaging Coherence Applications for WebLogic Server
- Setting Up a WebLogic Server Domain Topology for Coherence
- Deploying Coherence Applications To a WebLogic Server Domain
- Performing Basic Coherence Administration Tasks

Overview of the WebLogic Server Coherence Integration

Coherence is integrated with WebLogic Server. The integration aligns the lifecycle of a Coherence cluster member with the lifecycle of a managed server: starting or stopping a server JVM starts and stops a Coherence cluster member. The first member of the cluster starts the cluster service and is the senior member. The integration is detailed in Configuring and Managing Coherence Clusters in *Administering Clusters for Oracle WebLogic Server*.

Like other Java EE modules, Coherence supports its own application module, which is called a Grid ARchive (GAR). The GAR contains the artifacts of a Coherence application and includes a deployment descriptor. A GAR is deployed and undeployed in the same way as other Java EE modules and is decoupled from the cluster service lifetime. Coherence applications are isolated by a service namespace and by class loader.

Coherence is typically setup in tiers that provide functional isolation within a WebLogic Server domain. The most common tiers are: a data tier for caching data and an application tier for

consuming cached data. A proxy server tier and an extend client tier should be setup when using Coherence*Extend. An HTTP session tier should be setup when using Coherence*Web. See Using Coherence*Web with WebLogic Server in *Administering HTTP Session Management with Oracle Coherence*Web*.

WebLogic managed servers that are associated with a Coherence cluster are referred to as managed Coherence servers. Managed Coherence servers in each tier can be individually managed but are typically associated with respective WebLogic Server clusters. A GAR must be deployed to each data and proxy tier server. The same GAR is then packaged within an EAR and deployed to each application and extend client tier server. The use of dedicated storage tiers that are separate from client tiers is a best practice that ensures optimal performance.

Packaging Coherence Applications for WebLogic Server

Coherence applications must be packaged as a GAR module for deployment. A GAR module includes the artifacts that comprise a Coherence application and adheres to a specific directory structure. A GAR can be left as an unarchived directory or can be archived with a `.gar` extension. A GAR is deployed as both a standalone module and within an EAR. An EAR cannot contain multiple GAR modules.

This section includes the following topics:

- Building a Coherence GAR Module
- Packaging a GAR Module in an EAR Module

Building a Coherence GAR Module

To build a Coherence GAR module:

1. Create the following GAR directory structure:

   ```
   /
   /lib/
   /META-INF/
   ```

2. Add the Coherence cache configuration file and the POF configuration file (if required) to a directory within the GAR. For example:

   ```
   /
   /lib/
   /META-INF/coherence-cache-config.xml
   /META-INF/pof-config.xml
   ```

   > **Note:**
   >
   > The configuration files should not be placed in the root directory of the GAR. If the configuration files are placed in the root, do not use the default names as shown; otherwise, the configuration files are loaded from the `coherence.jar` file which is located in the system classpath.

3. Create a `coherence-application.xml` deployment descriptor file and save it to the `/META-INF` directory. A Coherence GAR must contain a `coherence-application.xml` deployment descriptor that is located within the `META-INF` directory. The presence of the deployment descriptor indicates a valid GAR.

```
/
/lib/
/META-INF/coherence-application.xml
/META-INF/coherence-cache-config.xml
/META-INF/pof-config.xml
```

4. Edit the `coherence-application.xml` file and specify the location of the configuration files from step 2. For example:

```
<?xml version="1.0"?>
<coherence-application>
   xmlns="http://xmnls.oracle.com/coherence/coherence-application">
   <cache-configuration-ref>META-INF/coherence-cache-config.xml
   </cache-configuration-ref>
   <pof-configuration-ref>META-INF/pof-config.xml</pof-configuration-ref>
</coherence-application>
```

> **Note:**
>
> - Configuration files can be placed on a network and referenced using a URL instead of copying the files locally to the GAR.
> - The cache configuration file can be overridden at runtime with a cluster cache configuration file. See Overriding a Cache Configuration File in *Administering Clusters for Oracle WebLogic Server*.
> - The cache configuration file can also be overridden at runtime using a JNDI property. See Using JNDI to Override Configuration in *Developing Oracle Coherence Applications for Oracle WebLogic Server*.

5. Place all Coherence application Java classes (entry processors, aggregators, filters, and so on) in the root directory within the appropriate package structure.
6. Place any library dependencies in the `/lib` directory.
7. Use the Java `jar` command from the root directory to compress the archive with a `.gar` extension. For example:

```
jar cvf MyApp.gar *
```

Packaging a GAR Module in an EAR Module

A GAR module must be packaged in an EAR module to be referenced by other modules. See Enterprise Applications in *Developing Applications for Oracle WebLogic Server*.

To include a GAR module within an EAR module:

1. Copy a GAR to the root directory of an EAR together with any application modules (WAR, EJB, and so on) that use Coherence.
2. Edit the `META-INF/weblogic-application.xml` descriptor and include a reference to the GAR using the `<module>` element. The reference is required so that the GAR is deployed when the EAR is deployed. For example:

```
<?xml version = '1.0'>
<weblogic-application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.bea.com/ns/weblogic/weblogic-application
   http://www.bea.com/ns/weblogic/weblogic-application/1.0/
   weblogic-application.xsd"
```

```
    xmlns="http://www.bea.com/ns/weblogic/weblogic-application">
    <module>
       <name>MyAppGAR</name>
       <type>GAR</type>
       <path>MyApp.gar</path>
    </module>
</weblogic-application>
```

Setting Up a WebLogic Server Domain Topology for Coherence

This section includes the following topics:

- Guidelines for Setting Up a Domain Topology
- Create a Coherence Cluster
- Create Coherence Deployment Tiers
- Create Managed Coherence Servers For a Coherence Deployment Tier

Guidelines for Setting Up a Domain Topology

Coherence supports different domain topologies within a WebLogic Server domain to provide varying levels of performance, scalability, and ease of use. For example, during development, a single managed Coherence server instance may be used as both a cache server and a cache client. The single-server topology is easy to setup and use, but does not provide optimal performance or scalability. For production, Coherence is typically setup using WebLogic Server clusters. A WebLogic Server cluster is used as a Coherence data tier and hosts one or more cache servers; a different WebLogic Server cluster is used as a Coherence application tier and hosts one or more cache clients; and (if required) different WebLogic Server clusters are used for the Coherence proxy tier that hosts one or more managed Coherence proxy servers and the Coherence extend client tier that hosts extend clients. The tiered topology approach provides optimal scalability and performance. A domain topology should always be based on the requirements of an application.

Use the following guidelines when creating a domain topology for Coherence:

- A domain typically contains a single Coherence cluster.
- Multiple WebLogic Server clusters can be associated with a Coherence cluster.
- A managed server that is associated with a Coherence cluster is referred to as a managed Coherence server and is the same as a Coherence cluster member.
- Use different managed Coherence server instances (and preferably different WebLogic Server clusters) to separate Coherence cache servers and clients.
- Coherence members managed within a WebLogic Server domain should not join an external Coherence cluster comprised of standalone JVM cluster members. Standalone JVM cluster members cannot be managed within a WebLogic Server domain.

Create a Coherence Cluster

To create a Coherence cluster using the WebLogic Server console:

1. From the console home page's Environment section, click **Coherence Clusters**.
2. From the Summary of Coherence Clusters page, click **New**.

3. From the Create a Coherence Cluster Configuration page, enter a name for the cluster using the Name field.
4. Click **Next** and skip to step 6.

   Or,

   Click to select the Use a Custom Cluster Configuration File check-box. WebLogic Server MBeans expose a subset of the operational settings that are sufficient for most use cases. However, for advanced use cases that require full control over operational settings, a cluster configuration file (such as the `tangosol-coherence-override.xml` file) can be used. Click **Next**. See Setting Up a Cluster in *Developing Applications with Oracle Coherence*.

   > **Note:**
   >
   > The use of an external cluster configuration file is only recommended for operational settings that are not available through the provided MBeans. That is, avoid configuring the same operational settings in both an external cluster configuration file and through the MBeans.

5. From the Create a Coherence Cluster Configuration File screen, use the File Path field to enter the path and name of a cluster configuration file that is located on the administration server. Click Next and skip to step 7.
6. From the Coherence Cluster Addressing section, leave the default clustering mode (Unicast) and change the port if required. To use multicast, use the drop-down list and select **Multicast** and provide a unique multicast address and port for the cluster. Click **Next**.

   If Unicast is used, the cluster automatically creates a Well Known Addresses (WKA) list based on the managed Coherence server instances in the Coherence cluster (one per machine). You can edit the cluster definition using the Administration Console and define your own WKA list if you wish to change the number of members. Addresses must be entered using the actual IP address on the host and not `localhost`; otherwise, the managed Coherence servers will not be able to join with other cluster members. See Using Well Known Addresses in *Developing Applications with Oracle Coherence*.
7. From the Coherence Cluster Members section, click to select the managed Coherence servers or WebLogic Server clusters to be part of the Coherence cluster or skip this section if managed Coherence servers and WebLogic Clusters are yet to be defined.
8. Click **Finish**. The Summary of Coherence Clusters screen displays and the Coherence Clusters table lists the cluster.

Create Coherence Deployment Tiers

The preferred approach for setting up Coherence in a WLS domain is to separate Coherence cache servers, clients, and proxies into different tiers that are associated with the same Coherence cluster. Typically, each tier is associated with its own WebLogic Server cluster of managed Coherence servers. However, a tier may also be comprised of standalone managed Coherence servers. The former approach provides the easiest way to manage and scale Coherence because the managed Coherence servers can inherit the WebLogic Server cluster's Coherence settings and deployments. Use the instructions in this section to create different WebLogic Server clusters for the data, application, and proxy tiers. See Configuring and Managing Coherence Clusters in *Administering Clusters for Oracle WebLogic Server*.

To create Coherence deployment tiers:

1. From the console home page's Environment section, click **Clusters**.
2. From the Summary of Clusters page, click **New** and select **Cluster**.
3. From the Create a New Cluster page, use the name field to enter a name for the WebLogic Server cluster.
4. Leave the default messaging mode (Unicast) and change the broadcast channel as required, or use the drop-down list and select **Multicast** and provide a different multicast address and port if required.
5. Click **OK**. The Summary of Clusters page displays and the Cluster table lists the cluster.
6. From the Clusters table, click the cluster to configure it.
7. From the Coherence tab, use the Coherence Cluster drop-down list and select a Coherence cluster to associate it with this WebLogic Server cluster. Click **Save.** By default, the managed Coherence servers assigned to this WebLogic Server cluster will be storage-enabled Coherence members (cache servers) as indicated by the Local Storage Enabled field.
8. Repeat steps 1 to 6 to create another WebLogic Server cluster to be used for the application tier. From the Coherence tab, use the Coherence Cluster drop-down list and select the Coherence cluster to associate it with this WebLogic Server cluster.
9. Click the Local Storage Enabled check box to remove the check mark and disable storage on the application tier. The managed Coherence servers assigned to this WebLogic Server cluster will be storage-disabled Coherence members (cache factory clients). Click **Save**.
10. (If applicable) Repeat steps 1 to 6 to create another WebLogic Server cluster to be used for the proxy tier. From the Coherence tab, use the Coherence Cluster drop-down list and select the Coherence cluster to associate it with this WebLogic Server cluster.
11. Click the Local Storage Enabled check box to remove the check mark and disable storage on the proxy tier. The managed Coherence servers assigned to this WebLogic Server cluster are storage-disabled Coherence members. Click **Save**.
12. (If applicable) Repeat steps 1 to 6 to create another WebLogic Server cluster to be used for the extend client tier. From the Coherence tab, use the Coherence Cluster drop-down list and select the Coherence cluster to associate it with this WebLogic Server cluster.
13. Click the Local Storage Enabled check box to remove the check mark and disable storage on the proxy tier. The managed Coherence servers assigned to this WebLogic Server cluster are storage-disabled Coherence members. Click **Save**.

Create Managed Coherence Servers For a Coherence Deployment Tier

Managed servers that are associated with a Coherence cluster are Coherence cluster members and are referred to as managed Coherence servers. Use the instructions in this section to create managed servers and associate them with a WebLogic Server cluster that is configured as a Coherence deployment tier. Managed servers automatically inherit Coherence settings from the WebLogic Server cluster. Existing managed Coherence servers can be associated with a WebLogic Server cluster as well.

To create managed servers for a Coherence deployment tier:

1. From the console home page's Environment section, click **Servers**.
2. Click **New** to create a new managed server.
3. From the Create a New Server page, enter the server's properties as required.
4. Click the Yes option to add the server to an existing cluster and use the drop-down list to select a WebLogic Server cluster that has been configured as a Coherence tier. The managed server inherits the Coherence settings from the WebLogic Server cluster.
5. Click **Finish**. The Summary of Servers page displays and the new server is listed.
6. Repeat these steps to create additional managed servers as required.
7. Click the Control tab and select the servers and click **Start**.

Deploying Coherence Applications To a WebLogic Server Domain

This section includes the following topics:

Overview of WebLogic Server Domain Deployment

Each Coherence deployment tier must include a Coherence application module. Deploying the application module starts the services that are defined in the GAR's cache configuration file. See Packaging Coherence Applications for WebLogic Server.

Deploy Coherence modules as follows:

- Data Tier (cache servers) – Deploy a standalone GAR to each managed Coherence server of the data tier. If the data tier is setup as a WebLogic Server cluster, deploy the GAR to the cluster and the WebLogic deployment infrastructure copies the module to each managed Coherence server.
- Application Tier (cache clients) – Deploy the EAR that contains GAR and the client implementation (Web Application, EJB, and so on) to each managed Coherence server in the cluster. If the application tier is setup as a WebLogic Server cluster, deploy the EAR to the cluster and the WebLogic deployment infrastructure copies the module to each managed Coherence server.
- Proxy Tier (proxy servers) – Deploy the standalone GAR to each managed Coherence server of the proxy tier. If the proxy tier is setup as a WebLogic Server cluster, deploy the GAR to the cluster and the WebLogic deployment infrastructure copies the module to each managed Coherence server.

> **Note:**
>
> Proxy tier managed Coherence servers must include a proxy service definition in the cache configuration file. You can deploy the same GAR to each tier, and then override the cache configuration file of just the proxy tier servers by using a cluster-level cache configuration file. See Overriding a Cache Configuration File in *Administering Clusters for Oracle WebLogic Server*.

- Extend Client Tier (extend clients) – Deploy the EAR that contains the GAR and the extend client implementation to each managed server that hosts the extend client. If the extend client tier is setup as a WebLogic Server cluster, deploy the EAR to the cluster and the WebLogic deployment infrastructure copies the module to each managed server.

> **Note:**
>
> Extend tier managed servers must include a remote cache service definition in the cache configuration file. You can deploy the same GAR to each tier, and then override the cache configuration file of just the extend tier servers by using a cluster-level cache configuration file. See Overriding a Cache Configuration File in *Administering Clusters for Oracle WebLogic Server*.

Deploy the Data Tier GAR

To deploy a GAR on the data tier:

1. From the console home page's Your Deployed Resources section, click **Deployments**.
2. Click Install.
3. From the Install Application Assistant page, locate and select the GAR to be deployed. Click **Next**.
4. Select the data tier (WebLogic Server cluster or standalone managed Coherence servers) to which the GAR should be deployed. Click **Next**.
5. Edit the Source accessibility settings and select the option to have the module copied to each target. Click **Finish**. The Summary of Deployments page displays and the GAR is listed in the Deployments table.
6. From the list of deployments, select the check box for the GAR and click **Start**.

Deploy the Application Tier EAR

To deploy an EAR on the application tier:

1. From the console home page's Your Deployed Resources section, click **Deployments**.
2. Click Install.

3. From the Install Application Assistant page, locate and select the EAR to be deployed. Click **Next**.
4. Keep the default target style and click **Next**.
5. Select the application tier (WebLogic Server cluster or standalone managed Coherence servers) to which the EAR should be deployed. Click **Next**.
6. Edit the Source accessibility settings and select the option to have the module copied to each target. Click **Finish**. The Summary of Deployments page displays and the EAR is listed in the Deployments table.
7. From the list of deployments, select the check box for the EAR and click **Start**.

Deploy the Proxy Tier GAR

To deploy a GAR on the proxy tier

1. From the console home page's Your Deployed Resources section, click **Deployments**.
2. Click Install.
3. From the Install Application Assistant page, locate and select the GAR to be deployed. Click **Next**.
4. Select the proxy tier (WebLogic Server cluster or standalone managed Coherence servers) to which the GAR should be deployed. Click **Next**.
5. Edit the Source accessibility settings and select the option to have the module copied to each target. Click **Finish**. The Summary of Deployments page displays and the GAR is listed in the Deployments table.
6. From the list of deployments, select the check box for the GAR and click **Start**.

Performing Basic Coherence Administration Tasks

Administrators use WebLogic Server tools to manage a Coherence environment within a WebLogic domain. These tools simplify the tasks of administering a cluster and cluster members. This section provides an overview of using the Administration Console tool to perform basic administrative task. See *Oracle WebLogic Server Administration Console Online Help*. Many of the tasks can also be performed using the WebLogic Scripting Tool (WLST). See Using the WebLogic Scripting Tool in *Understanding the WebLogic Scripting Tool*.

Table 1-1 Basic Administration Task in the Administration Console

To...	Use the...
Create a Coherence cluster	Coherence Clusters page
Add or remove cluster members or WebLogic Server clusters from a Coherence Cluster	Members Tab located on a Coherence cluster's Settings page.
Configure unicast or multicast settings for a Coherence cluster	General Tab located on a Coherence cluster's Settings page. If unicast is selected, the default well known addresses configuration can be overridden using the Well Known Addresses tab.
Use a custom cluster configuration file to configure a Coherence cluster	General Tab located on a Coherence cluster's Settings page

Table 1-1 (Cont.) Basic Administration Task in the Administration Console

To...	Use the...
Import a cache configuration file to a cluster member and override the cache configuration file deployed in a GAR	Cache Configurations Tab located on a Coherence cluster's Settings page
Configuring Logging	Logging Tab located on a Coherence cluster's Settings page
Assign a managed server to a Coherence Cluster	Coherence Tab located on a managed server's Settings page
Configure Coherence cluster member properties	Coherence Tab located on a managed server's Settings page
Associate a WebLogic Server cluster with a Coherence cluster and enable or disable storage for the managed Coherence servers of the cluster	Coherence Tab located on a WebLogic Server cluster's Settings page
Assign a managed server to WebLogic Server cluster that is associated with a Coherence cluster	General Tab located on a managed server's Settings page

Deploying Coherence Applications to an Application Server (Generic)

Java EE applications that are deployed to an application server, other than WebLogic Server, have two options for deploying Coherence: as an application server library or as part of a Java EE module.

Coherence cluster members are class loader scoped. Therefore, the option selected results in a different deployment scenario. All modules share a single cluster member if Coherence is deployed as an application server library. Whereas, a Java EE module is its own cluster member if Coherence is deployed as part of the module. Each option has its own benefits and assumptions and generally balances resource utilization with how isolated the cluster member is from other modules.

> **Note:**
>
> For Coherence*Web deployment, see Using Coherence*Web on Other Application Servers in *Administering HTTP Session Management with Oracle Coherence*Web*.

This section includes the following topics:

- Deploying Coherence as an Application Server Library
- Deploying Coherence in a Java EE Module

Deploying Coherence as an Application Server Library

Coherence can be deployed as an application server library. In this deployment scenario, an application server's startup classpath is modified to include the

COHERENCE_HOME/lib/coherence.jar library. In addition, any objects that are being placed into the cache must also be available in the server's classpath. Consult your application server vendor's documentation for instructions on adding libraries to the server's classpath.

This scenario results in a single cluster member that is shared by all applications that are deployed in the server's containers. This scenario minimizes resource utilization because only one copy of the Coherence classes are loaded into the JVM. See Running Multiple Applications in a Single Cluster.

Deploying Coherence in a Java EE Module

Coherence can be deployed within an EAR file or a WAR file. This style of deployment is generally preferred because modification to the application server run-time environment is not required and because cluster members are isolated to either the EAR or WAR.

This section includes the following topics:

- Deploying Coherence Within an EAR
- Deploying Coherence Within a WAR

Deploying Coherence Within an EAR

Coherence can be deployed as part of an EAR. This deployment scenario results in a single cluster member that is shared by all Web applications in the EAR. Resource utilization is moderate because only one copy of the Coherence classes are loaded per EAR. However, all Web applications may be affected by any one module's use of the cluster member. See Running Multiple Applications in a Single Cluster.

To deploy Coherence within an enterprise application:

1. Copy the coherence.jar library to a location within the enterprise application directory structure.
2. Using a text editor, open the META-INF/application.xml deployment descriptor.
3. Add a <java> element that contains the path (relative to the top level of the application directory) and name of the coherence library. For example:

```
<application>
   <display-name>MyApp</display-name>
   <module>
      <java>coherence.jar</java>
   </module>
   ...
</application>
```

4. Make sure any objects that are to be placed in the cache are added to the application in the same manner as described above.
5. Save and close the descriptor.
6. package and deploy the application.

Deploying Coherence Within a WAR

Coherence can be deployed as part of a Web application. This deployment scenario results in each Web application having its own cluster member, which is isolated from all other Web applications. This scenario uses the most amount of resources because there are as many copies of the Coherence classes loaded as there are deployed Web applications that include

Coherence. This scenario is ideal when deploying only a few Web applications to an application server.

To deploy Coherence within a Web application:

1. Copy the `coherence.jar` library to the Web Application's `WEB-INF/lib` directory.
2. Make sure any objects that are to be placed in the cache are located in either the `WEB-INF/lib` or `WEB-INF/classes` directory.
3. Package and deploy the application.

Running Multiple Applications in a Single Cluster

Coherence can be deployed in shared environments where multiple applications use the same cluster but define their own set of Coherence caches and services. For such scenarios, each application uses its own cache configuration file that includes a scope name that controls whether the caches and services are allowed to be shared among applications.
This section includes the following topics:

Specifying a Scope Name

The `<scope-name>` element is used to specify a service namespace that uniquely identifies the caches and services in a cache configuration file. If specified, all caches and services are isolated and cannot be used by other applications that run on the same cluster.

The following example configures a scope name called `accounts` and results in the use of `accounts` as a prefix to all services instantiated by the `ConfigurableCacheFactory` instance that is created based on the configuration. The scope name is an attribute of a cache factory instance and only affects that cache factory instance.

> **Note:**
>
> The prefix is only used for service names, not cache names.

```
<?xml version='1.0'?>

<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-cache-config
   coherence-cache-config.xsd">
      <defaults>
         <scope-name>accounts</scope-name>
```

```
   </defaults>
<caching-scheme-mapping>
...
```

Scoping Applications in WebLogic Server

Multiple deployed Coherence applications (GARs) are isolated by a service namespace and by ClassLoader by default in WebLogic Server and do not require scope name configuration. However, a scope name may still be configured to share caches between GARs. Directly configuring the scope in the cache configuration file is typically performed for advanced use cases.

> **Note:**
>
> If you want to deploy multiple GARs with the same scope name, then the configuration files in all GARs must be identical; otherwise, the deployment fails.

The deployment name is used as the default scope name when deploying a GAR. If a deployment name is not specified during deployment, the artifact name is used as the deployment name. For example, for the `MyApp.gar` module, the default deployment name is `MyApp`. In the case of a GAR packaged in an EAR, the deployment name is the module name specified for the GAR in the `weblogic-application.xml` file.

Scoping Applications in a Java EE Environment (Generic)

Deploying Coherence as an application server library, or as part of an EAR, allows multiple applications to use the same cluster as a single cluster member (one JVM). In such deployment scenarios, multiple applications may choose to use a single set of Coherence caches and services that are configured in a single `coherence-cache-config.xml` file. This type of deployment is only suggested (and only practical) in controlled environments where application deployment is coordinated. The likelihood of collisions between caches, services and other configuration settings is high and may lead to unexpected results. Moreover, all applications may be affected by any one application's use of the Coherence node.

The alternative is to have each application include its own cache configuration file that defines the caches and services that are unique to the application. The configurations are then isolated by specifying a scope name using the `<scope-name>` element in the cache configuration file. Likewise, applications can explicitly allow other applications to share their caches and services if required. This scenario assumes that a single JVM contains multiple `ConfigurableCacheFactory` instances that each pertains to an application.

This section includes the following topics:

- Isolating Applications in a JavaEE Environment
- Sharing Application Data in a JavaEE Environment

Isolating Applications in a JavaEE Environment

The following example demonstrates the steps that are required to isolate two Web applications (`trade.war` and `accounts.war`) from using each other's caches and services:

1. Create a cache configuration file for the trade application (for example, `trade-cache-config.xml`) that defines a scope name called `trade` and include any cache scheme definitions for the application:

```
<?xml version='1.0'?>

<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-cache-
config
   coherence-cache-config.xsd">
      <defaults>
         <scope-name>trade</scope-name>
      </defaults>
   ...
```

2. Create a cache configuration file for the accounts application (for example, `accounts-cache-config.xml`) that defines a scope name called `accounts` and include any cache scheme definitions for the application:

```
<?xml version='1.0'?>

<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-cache-
config
   coherence-cache-config.xsd">
      <defaults>
         <scope-name>accounts</scope-name>
      </defaults>
   ...
```

3. Ensure the cache configurations files are included in their respective WAR files (typically in the `WEB-INF/classes` directory) so that they can be loaded at run time and used by the application.

Sharing Application Data in a JavaEE Environment

Applications can share data by allowing access to their caches and services. The following example demonstrates allowing a Web application (`trade.war`) to access the caches and services of another Web application (`accounts.war`):

1. Create a cache configuration file for the trade application (for example, `trade-cache-config.xml`) that defines a scope name called `trade` and include any cache scheme definitions for the application:

```
<?xml version='1.0'?>

<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-cache-
config
   coherence-cache-config.xsd">
      <defaults>
         <scope-name>trade</scope-name>
      </defaults>
   ...
```

2. Create a cache configuration file (for example, `accounts-cache-config.xml`) for the accounts application that defines a scope name called `accounts` and include any cache scheme definitions for the application:

```
<?xml version='1.0'?>

<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-cache-config
   coherence-cache-config.xsd">
      <defaults>
         <scope-name>accounts</scope-name>
      </defaults>
   ...
```

3. Ensure the cache configurations files are included in their respective WAR files (typically in the `WEB-INF/classes` directory) so that they can be loaded at run time and used by the application.
4. The trade application must also include the `accounts-cache-config.xml` file to access the caches and services of the accounts application.
5. The trade application can then use the following pattern to create cache factories for the accounts application:

```
ClassLoader loader = ...
CacheFactoryBuilder builder = CacheFactory.getCacheFactoryBuilder();
ConfigurableCacheFactory tradesCcf =
   builder.getConfigurableCacheFactory(tradesUri, loader);
ConfigurableCacheFactory accountsCcf =
   builder.getConfigurableCacheFactory(accountsUri, loader);
```

Scoping Applications in a Standalone Environment

Standalone applications that use a single Coherence cluster can each include their own cache configuration files; however, these configurations are coalesced into a single `ConfigurableCacheFactory`. Since there is a 1 to 1 relationship between `ConfigurableCacheFactory` and `DefaultCacheServer`, application scoping is not feasible within a single cluster node. Instead, one or more instances of `DefaultCacheServer` must be started for each cache configuration, and each cache configuration must include a scope name.

The following example isolates two applications (trade and accounts) from using each other's caches and services:

1. Create a cache configuration file for the trade application (for example, `trade-cache-config.xml`) that defines a scope name called `trade` and include any cache scheme definitions for the application:

```
<?xml version='1.0'?>

<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-cache-config
   coherence-cache-config.xsd">
      <defaults>
         <scope-name>trade</scope-name>
      </defaults>
   ...
```

2. Start a `DefaultCacheServer` instance that loads the `trade-cache-config.xml` cache configuration file.
3. Create a cache configuration file for the accounts application (for example, `accounts-cache-config.xml`) that defines a scope name called `accounts` and include any cache scheme definitions for the application:

```
<?xml version='1.0'?>

<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-cache-
config
   coherence-cache-config.xsd">
      <defaults>
         <scope-name>accounts</scope-name>
      </defaults>
   ...
```

4. Start a `DefaultCacheServer` instance that loads the `accounts-cache-config.xml` cache configuration file.

> **Note:**
>
> To share data between applications, the applications must use the same cache configuration file. Coherence does not support using multiple cache configurations which specify the same scope name.

Providing a Custom Scope Resolver

The `com.tangosol.net.ScopeResolver` interface allows containers and applications to modify the scope name for a given `ConfigurableCacheFactory` at run time to enforce (or disable) isolation between applications. Implement the `ScopeResolver` interface and add any custom functionality as required.

To enable a custom scope resolver, the fully qualified name of the implementation class must be defined in the operational override file using the `<scope-resolver>` element within the `<cache-factory-builder-config>` node. For example:

```
<?xml version='1.0'?>

<coherence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-operational-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-operational-
config
   coherence-operational-config.xsd">
   <cache-factory-builder-config>
      <scope-resolver>
         <class-name>package.MyScopeResolver</class-name>
      </scope-resolver>
   </cache-factory-builder-config>
<coherence>
```

As an alternative, the `<instance>` element supports the use of a `<class-factory-name>` element to specify a factory class that is responsible for creating `ScopeResolver` instances, and a `<method-name>` element to specify the static factory method on the

factory class that performs object instantiation. The following example gets a custom scope resolver instance using the `getResolver` method on the `MyScopeResolverFactory` class.

```
<?xml version='1.0'?>

<coherence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-operational-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-operational-config
   coherence-operational-config.xsd">
   <cache-factory-builder-config>
      <scope-resolver>
         <class-factory-name>package.MyScopeReolverFactory</class-factory-name>
         <method-name>getResolver</method-name>
      </scope-resolver>
   </cache-factory-builder-config>
<coherence>
```

Any initialization parameters that are required for an implementation can be specified using the `<init-params>` element. The following example sets an `isDeployed` parameter to `true`.

```
<?xml version='1.0'?>

<coherence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-operational-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-operational-config
   coherence-operational-config.xsd">
   <cache-factory-builder-config>
      <scope-resolver>
         <class-name>package.MyScopeResolver</class-name>
         <init-params>
            <init-param>
               <param-name>isDeployed</param-name>
               <param-value>true</param-value>
            </init-param>
         </init-params>
      </scope-resolver>
   </cache-factory-builder-config>
<coherence>
```

2
Performing a Network Performance Test

Coherence provides a datagram utility and a message bus utility for testing network performance between two or more computers. Any production deployment should be preceded by a successful run of both tests.
This chapter includes the following sections:

- Using the Datagram Test Utility
- Using the Message Bus Test Utility

Using the Datagram Test Utility

The Coherence datagram test utility is used to test and tune network performance between two or more computers. The utility ensures that a network is optimally configured to support Coherence cluster management communication. There are two types of tests: a point-to-point test that tests the performance of a pair of servers to ensure they are properly configured, and a distributed datagram test to ensure the network itself is functioning properly. Both tests need to be run successfully.
The datagram test operates in one of three modes: either as a packet publisher, a packet listener, or both. When the utility is run, a publisher transmits packets to the listener who then measures the throughput, success rate, and other statistics. Tune an environment based on the results of these tests to achieve maximum performance. See Performance Tuning.

This section includes the following topics:

- Running the Datagram Test Utility
- How to Test Datagram Network Performance
- Understanding Datagram Report Statistics

Running the Datagram Test Utility

The datagram test utility is run from the command line using either the `com.tangosol.net.DatagramTest` class or by running the `datagram-test` script that is provided in the *`COHERENCE_HOME`*`/bin` directory. A script is provided for both Windows and UNIX-based platforms.

The following example demonstrates using the `DatagramTest` class:

```
java -server -cp coherence.jar com.tangosol.net.DatagramTest <command value> <command
value> ...
```

The following example demonstrates using the script:

```
datagram-test <command value> <command value> ...
```

Table 2-1 describes the available command line options for the datagram test utility.

Table 2-1 Command Line Options for the Datagram Test Utility

Command	Required/ Optional	Applicability	Description	Default
-local	Optional	Both	The local address to bind to, specified as `addr:port`	localhost:9999
-packetSize	Optional	Both	The size of packet to work with, specified in bytes.	1468
-payload	Optional	Both	The amount of data to include in each packet. Use `0` to match packet size.	0
-processBytes	Optional	Both	The number of bytes (in multiples of 4) of each packet to process.	4
-rxBufferSize	Optional	Listener	The size of the receive buffer, specified in packets.	1428
-rxTimeoutMs	Optional	Listener	The duration of inactivity before a connection is closed.	1000
-txBufferSize	Optional	Publisher	The size of the transmit buffer, specified in packets.	16
-txRate	Optional	Publisher	The rate at which to transmit data, specified in megabytes.	*unlimited*
-txIterations	Optional	Publisher	Specifies the number of packets to publish before exiting.	*unlimited*
-txDurationMs	Optional	Publisher	Specifies how long to publish before exiting.	*unlimited*
-reportInterval	Optional	Both	The interval at which to output a report, specified in packets.	100000
-tickInterval	Optional	Both	The interval at which to output tick marks.	1000
-log	Optional	Listener	The name of a file to save a tabular report of measured performance.	*none*
-logInterval	Optional	Listener	The interval at which to output a measurement to the log.	100000
-polite	Optional	Publisher	Switch indicating if the publisher should wait for the listener to be contacted before publishing.	*off*
-provider	Optional	Both	The socket provider to use (`system`, `tcp`, `ssl`, `file:`*`xxx`*`.xml`)	system
arguments	Optional	Publisher	Space separated list of addresses to publish to, specified as `addr:port`.	*none*

How to Test Datagram Network Performance

This section includes instructions for running a point-to-point datagram test and a distributed datagram test. Both tests must be run successfully and show no significant performance issues or packet loss. See Understanding Datagram Report Statistics.

This section includes the following topics:

- Performing a Point-to-Point Datagram Test
- Performing a Bidirectional Datagram Test
- Performing a Distributed Datagram Test

Performing a Point-to-Point Datagram Test

The example in this section demonstrates how to test network performance between two servers— Server A with IP address `195.0.0.1` and Server B with IP address `195.0.0.2`. One server acts as a packet publisher and the other as a packet listener. The publisher transmits packets as fast as possible and the listener measures and reports performance statistics.

First, start the listener on Server A. For example:

```
datagram-test.sh
```

After pressing ENTER, the utility displays that it is ready to receive packets. Example 2-1 illustrates sample output.

Example 2-1 Output from Starting a Listener

```
starting listener: at /195.0.0.1:9999
packet size: 1468 bytes
buffer size: 1428 packets
  report on: 100000 packets, 139 MBs
    process: 4 bytes/packet
        log: null
     log on: 139 MBs
```

The test, by default, tries to allocate a network receive buffer large enough to hold 1428 packets, or about 2 MB. The utility reports an error and exits if it cannot allocate this buffer. Either decrease the requested buffer size using the `-rxBufferSize` parameter, or increase the operating system's network buffer settings. Increase the operating system buffers for the best performance. See Production Checklist .

Start the publisher on Server B and direct it to publish to Server A. For example:

```
datagram-test.sh servera
```

After pressing ENTER, the test instance on Server B starts both a listener and a publisher. However, the listener is not used in this configuration. Example 2-2 demonstrates the sample output that displays in the Server B command window.

Example 2-2 Datagram Test—Starting a Listener and a Publisher on a Server

```
starting listener: at /195.0.0.2:9999
packet size: 1468 bytes
buffer size: 1428 packets
  report on: 100000 packets, 139 MBs
    process: 4 bytes/packet
```

```
        log: null
     log on: 139 MBs

starting publisher: at /195.0.0.2:9999 sending to servera/195.0.0.1:9999
packet size: 1468 bytes
buffer size: 16 packets
  report on: 100000 packets, 139 MBs
    process: 4 bytes/packet
      peers: 1
       rate: no limit

no packet burst limit
oooooooooOooooooooOooooooooOooooooooOooooooooOooooooooOooooooooOooooooooO
```

The series of `o` and `O` marks appear as data is (O)utput on the network. Each `o` represents 1000 packets, with `O` indicators at every 10,000 packets.

On Server A, a corresponding set of `i` and `I` marks, representing network (I)nput. This indicates that the two test instances are communicating.

Performing a Bidirectional Datagram Test

The point-to-point test can also be run in bidirectional mode where servers act as publishers and listeners. Use the same test instances that were used in the point-to-point test and supply the instance on Server A with the address for Server B. For example on Server A run:

```
datagram-test.sh -polite serverb
```

The `-polite` parameter instructs this test instance to not start publishing until it starts to receive data. Run the same command as before on Server B.

```
datagram-test.sh servera
```

Performing a Distributed Datagram Test

A distributed test is used to test performance with more than two computers. For example, setup two publishers to target a single listener. This style of testing is far more realistic then simple one-to-one testing and may identify network bottlenecks that may not otherwise be apparent.

The following example runs the datagram test among 4 computers:

On Server A:

```
datagramtest.sh -txRate 100 -polite serverb serverc serverd
```

On Server B:

```
datagramtest.sh -txRate 100 -polite servera serverc serverd
```

On Server C:

```
datagramtest.sh -txRate 100 -polite servera serverb serverd
```

On Server D:

```
datagramtest.sh -txRate 100 servera serverb serverc
```

This test sequence causes all nodes to send a total of 100MB per second to all other nodes (that is, 33MB/node/second). On a fully switched 1GbE network this should be achievable without packet loss.

To simplify the execution of the test, all nodes can be started with an identical target list, they obviously transmit to themselves as well, but this loopback data can easily be factored out. It is important to start all but the last node using the `-polite` switch, as this causes all other nodes to delay testing until the final node is started.

Understanding Datagram Report Statistics

Each side of the test (publisher and listener) periodically report performance statistics. The publisher simply reports the rate at which it is publishing data on the network. For example:

```
Tx summary 1 peers:
   life: 97 MB/sec, 69642 packets/sec
    now: 98 MB/sec, 69735 packets/sec
```

The report includes both the current transmit rate (since last report) and the lifetime transmit rate.

Table 2-2 describes the statistics that can be reported by the listener.

Table 2-2 Listener Statistics

Element	Description
Elapsed	The time interval that the report covers.
Packet size	The received packet size.
Throughput	The rate at which packets are being received.
Received	The number of packets received.
Missing	The number of packets which were detected as lost.
Success rate	The percentage of received packets out of the total packets sent.
Out of order	The number of packets which arrived out of order.
Average offset	An indicator of how out of order packets are.

As with the publisher, both current and lifetime statistics are reported. The following example demonstrates a typical listener report:

```
Lifetime:
Rx from publisher: /195.0.0.2:9999
             elapsed: 8770ms
         packet size: 1468
          throughput: 96 MB/sec
                      68415 packets/sec
            received: 600000 of 611400
             missing: 11400
        success rate: 0.9813543
        out of order: 2
          avg offset: 1


Now:
Rx from publisher: /195.0.0.2:9999
             elapsed: 1431ms
         packet size: 1468
```

```
  throughput: 98 MB/sec
              69881 packets/sec
    received: 100000 of 100000
     missing: 0
success rate: 1.0
out of order: 0
  avg offset: 0
```

The primary items of interest are the throughput and success rate. The goal is to find the highest throughput while maintaining a success rate as close to 1.0 as possible. A rate of around 10 MB/second should be achieved on a 100 Mb network setup. A rate of around 100 MB/second should be achieved on a 1 Gb network setup. Achieving these rates require some throttle tuning. If the network cannot achieve these rates or if the rates are considerably less, then it is very possible that there are network configuration issues. See Network Tuning.

Throttling

The publishing side of the test may be throttled to a specific data rate expressed in megabytes per second by including the `-txRate M` parameter, when `M` represents the maximum MB/second the test should put on the network.

Using the Message Bus Test Utility

The Coherence message bus test utility is used to test the performance characteristics of message bus implementations and the network on which they operate. The utility ensures that a network is optimally configured to support communication between clustered data services. In particular, the utility can be used to test the TCP message bus (TMB) implementation, which is the default transport for non-exalogic systems and the Infiniband message bus (IMB) implementation, which is the default transport on Exalogic systems. Tune your environment based on the results of these tests to achieve maximum performance. See TCP Considerations.
This section includes the following topics:

- Running the Message Bus Test Utility
- How to Test Message Bus Performance
- Understanding Message Bus Report Statistics

Running the Message Bus Test Utility

The message bus test utility is run from the command line using the `com.oracle.common.net.exabus.util.MessageBusTest` class. The following example demonstrates using the `MessageBusTest` class:

```
java -server -cp coherence.jar com.oracle.common.net.exabus.util.MessageBusTest
<command value> <command value> ...
```

Table 2-3 describes the available command line options for the message bus test utility.

Table 2-3 Command Line Options for the Message Bus Test Utility

Command	Required/ Optional	Description	Default
-bind	Required	List of one or more local end points to create	none
-peer	Required	List of one or more remote end points to send to	none
-rxThreads	Optional	Number of receive threads per bound EndPoint (negative for reentrant)	
-txThreads	Optional	Number of transmit threads per bound EndPoint	
-msgSize	Optional	Range of message sizes to send, expressed as `min[..max]`	4096
-chunkSize	Optional	Defines the number of bytes to process as a single unit; that is, 1 for byte, 8 for long, and 0 to disable	
-cached	Optional	Re-use message objects where possible, reducing buffer manager overhead	
-txRate	Optional	Target outbound data rate as MBps	
-txMaxBacklog	Optional	The maximum backlog the test should produce per tx thread.	
-rxRate	Optional	Target inbound data rate as MBps. Cannot be used if `-rxThreads` is less than or equal to 0.	
-flushFreq	Optional	Number of messages to send before flushing, or 0 for auto	0
-latencyFreq	Optional	Number of messages to send before sampling latency	100
-noReceipts	Optional	If specified, then receipts should not be used, relies on GC to reclaim messages	false
-manager	Optional	Buffer manager to utilize (net, direct, or heap)	net
-depotFactory	Optional	The fully qualified class name of a factory to use to obtain a `Depot` instance	
-reportInterval	Optional	The report interval	5 seconds
-polite	Optional	If specified, then this instance does not start sending until connected to	
-block	Optional	If specified, then a transmit thread blocks while awaiting a response	false
-relay	Optional	If specified, then the process relays any received messages to one of its peers	false
-ignoreFlowControl	Optional	If specified, then flow control events are ignored. If flow control events are to be ignored, use the `-txMaxBacklog` command to prevent out of memory errors	false
-poll	Optional	If specified, then a `PollingEventCollector` implementation is used that queues all events and returns them only when they are for. A polling collector generally requires the `-rxThreads` command set to 1.	
-prompt	Optional	If specified, then the user is prompted before each send	

Table 2-3 (Cont.) Command Line Options for the Message Bus Test Utility

Command	Required/ Optional	Description	Default
-tabular	Optional	If specified, then use tabular format for the output	
-warmup	Optional	Time duration or message count that are discarded for warmup	0
-verbose	Optional	If specified, then enable verbose debugging output	

How to Test Message Bus Performance

This section includes instructions for running a point-to-point message bus test and a distributed message bus test for the TMB transport. Both tests must be run successfully and show no significant performance issues or errors.

This section includes the following topics:

- Performing a Point-to-Point Message Bus Test
- Performing a Bidirectional Message Bus Test
- Performing a Distributed Message Bus Test

Performing a Point-to-Point Message Bus Test

The example in this section demonstrates how to test network performance between two servers— Server A with IP address `195.0.0.1` and Server B with IP address `195.0.0.2`. Server A acts as a server and Server B acts as a client.

First, start the listener on Server A. For example:

```
java -cp coherence.jar com.oracle.common.net.exabus.util.MessageBusTest -bind
tmb://servera:8000
```

After pressing ENTER, the utility displays that it is ready to receive messages. Example 2-3 illustrates sample output.

Example 2-3 Output from Starting a Server Listener

```
OPEN event for tmb://195.0.0.1:8000
```

Start the client on Server B and direct it to send messages to Server A. For example:

```
java -cp coherence.jar com.oracle.common.net.exabus.util.MessageBusTest -bind
tmb://serverb:8000 -peer tmb://servera:8000
```

The test instance on Server B starts both a client and a server listener. The message bus test always performs bi-directional communication. In it's default mode the client sends an endless stream of messages to the server, and the server periodically replies to the client. In this configuration most communication is client to server, while the occasional server to client communication allows for latency measurements. Example 2-4 demonstrates the sample output that displays in the Server B command window.

> **Note:**
>
> The performance results in Example 2-4 may not be indicative of your network environment.

Example 2-4 Message Bus Test—Starting a Client and Server

```
OPEN event for tmb://195.0.0.2:8001
CONNECT event for tmb://195.0.0.1:8000 on tmb://195.0.0.2:8001
now:  throughput(out 65426msg/s 2.14gb/s, in 654msg/s 21.4mb/s),
   latency(response(avg 810.71us, effective 1.40ms, min 37 .89us, max 19.59ms),
   receipt 809.61us), backlog(out 42% 1556/s 48KB, in 0% 0/s 0B), connections 1,
   errors 0
life: throughput(out 59431msg/s 1.94gb/s, in 594msg/s 19.4mb/s),
   latency(response(avg 2.12ms, effective 3.85ms, min 36.32us, max 457.25ms),
   receipt 2.12ms), backlog(out 45% 1497/s 449KB, in 0% 0/s 0B), connections 1,
   errors 0
```

The test, by default, tries to use the maximum bandwidth to push the maximum amount of messages, which results in increased latency. Use the `-block` command to switch the test from streaming data to request and response, which provides a better representation of the network minimum latency:

```
now:  throughput(out 17819msg/s 583mb/s, in 17820msg/s 583mb/s),
   latency(response(avg 51.06us, effective 51.06us, min 43.42us, max 143.68us),
   receipt 53.36us), backlog(out 0% 0/s 0B, in 0% 0/s 0B), connections 1, errors 0
life: throughput(out 16635msg/s 545mb/s, in 16635msg/s 545mb/s),
   latency(response(avg 56.49us, effective 56.49us, min 43.03us, max 13.91ms),
   receipt 59.43us), backlog(out 0% 0/s 2.18KB, in 0% 0/s 744B), connections 1,
   errors 0
```

Performing a Bidirectional Message Bus Test

The point-to-point test can also be run in bidirectional mode where servers act as both client and servers. Use the same test instances that were used in the point-to-point test. For example on Server A run:

```
java -cp coherence.jar com.oracle.common.net.exabus.util.MessageBusTest -bind tmb://
servera:8000 -peer tmb://serverb:8000 -polite
```

The `-polite` parameter instructs this test instance to not start publishing until it starts to receive data. On Server B run.

```
java -cp coherence.jar com.oracle.common.net.exabus.util.MessageBusTest -bind tmb://
serverb:8000 -peer tmb://servera:8000
```

Performing a Distributed Message Bus Test

A distributed test is used to test performance with more than two computers. This style of testing is far more realistic then simple one-to-one testing and may identify network bottlenecks that may not otherwise be apparent.

The following example runs a bidirectional message bus test among 4 computers:

On Server A:

```
java -cp coherence.jar com.oracle.common.net.exabus.util.MessageBusTest -bind
tmb://servera:8000 -peer tmb://serverb:8000 tmb://serverc:8000 tmb://
serverd:8000 -polite
```

On Server B:

```
java -cp coherence.jar com.oracle.common.net.exabus.util.MessageBusTest -bind
tmb://serverb:8000 -peer tmb://servera:8000 tmb://serverc:8000 tmb://
serverd:8000 -polite
```

On Server C:

```
java -cp coherence.jar com.oracle.common.net.exabus.util.MessageBusTest -bind
tmb://serverc:8000 -peer tmb://servera:8000 tmb://serverb:8000 tmb://
serverd:8000 -polite
```

On Server D:

```
java -cp coherence.jar com.oracle.common.net.exabus.util.MessageBusTest -bind
tmb://serverd:8000 -peer tmb://servera:8000 tmb://serverb:8000 tmb://
serverc:8000 -polite
```

It is important to start all but the last node using the `-polite` switch, as this causes all other nodes to delay testing until the final node is started.

Understanding Message Bus Report Statistics

Each side of the message bus test (client and server) periodically report performance statistics. The example output is from the client sending the requests:

```
throughput(out 17819msg/s 583mb/s, in 17820msg/s 583mb/s),
   latency(response(avg 51.06us, effective 51.06us, min 43.42us, max 143.68us),
   receipt 53.36us), backlog(out 0% 0/s 0B, in 0% 0/s 0B), connections 1, errors
0
```

The report includes both statistics since the last report (`now:`) and the aggregate lifetime statistics (`life:`).

Table 2-2 describes the message bus statistics.

Table 2-4 Message Bus Statistics

Element	Description
throughput	The amount of messages per second being sent and received and the transmission rate
latency	The time spent for message response and receipt
backlog	the number of messages waiting to be sent and to be processed
connections	The number of open connections between message listeners
errors	The number of messages which were detected as lost.

The primary items of interest are throughput and latency. The goal should be to utilize as much network bandwidth as possible without resulting in high latencies. If bandwidth usage is low or latencies are high, consider tuning TCP settings. A high backlog or error rate can also indicate network configuration issues. See Network Tuning.

3
Performing a Multicast Connectivity Test

Coherence includes a multicast test utility that checks whether a network environment supports multicast communication. Any production deployment should be preceded by a successful run of the multicast test.
This chapter includes the following sections:

- Running the Multicast Test Utility
- How to Test Multicast
- Troubleshooting Multicast Communications

Running the Multicast Test Utility

The Coherence multicast test utility is used to determine if multicast is enabled between two or more computers. The utility does not test load. Each instance, by default, only transmit a single multicast packet every two seconds. See Performing a Network Performance Test for network load testing.
The multicast test utility is run from the command line using either the `com.tangosol.net.MulticastTest` class or by running the `multicast-test` script that is provided in the *`COHERENCE_HOME`*`/bin` directory. A script is provided for both Windows and UNIX-based platforms.

The following example runs the utility using the `MulticastTest` class:

```
java com.tangosol.net.MulticastTest <command value> <command value> ...
```

The following example runs the utility using the script:

```
multicast-test <command value> <command value> ...
```

Table 3-1 describes the available command line options for the multicast test utility.

Table 3-1 Command Line Options for the Multicast Test Utility

Command	Required/ Optional	Description	Default
-local	Optional	The address of the NIC to transmit on, specified as an IP address	localhost
-group	Optional	The multicast address to use, specified as IP:port.	237.0.0.1:9000
-ttl	Optional	The time to live for multicast packets.	4
-delay	Optional	The delay between transmitting packets, specified in seconds.	2
-packetSize	Optional	The size of the packet to send. The default is based on the local MTU.	*MTU*
-display	Optional	The number of bytes to display from unexpected packets.	0

Table 3-1 (Cont.) Command Line Options for the Multicast Test Utility

Command	Required/ Optional	Description	Default
-translate	Optional	Listen to cluster multicast traffic and translate packets	*none*

How to Test Multicast

Testing multicast requires two steps: verify that multicast is functioning properly on individual servers; and ensure that multicast is functioning properly between servers. The example in this section demonstrates how to test if multicast address 237.0.0.1, port 9000 (the defaults for the test) can send messages between two servers: Server A with IP address 195.0.0.1 and Server B with IP address 195.0.0.2.

> **Note:**
>
> The default multicast address and port that are used by the test are different than the Coherence default address and port. The test should be performed using the same address and port that are being used in the actual Coherence processes. It is possible that the default address and port for the mutlicast test succeeds, but the Coherence defaults fail. This often due to local network policy configuration.

Starting with Server A, determine if it has multicast address 237.0.0.1 port 9000 available for 195.0.0.1 by first checking the computer or interface by itself as follows:

From a command prompt, enter the following command:

```
multicast-test.sh -ttl 0
```

After pressing ENTER, the utility display how it is sending sequential multicast packets and receiving them. Example 3-1 illustrates sample output.

Example 3-1 Sequential Multicast Packets Sent by the Multicast Test Utility

```
Starting test on ip=servera/195.0.0.1, group=/237.0.0.1:9000,ttl=0
Configuring multicast socket...
Starting listener...
Tue Mar 17 15:59:51 EST 2008: Sent packet 1.
Tue Mar 17 15:59:51 EST 2008: Received test packet 1 from self.
Tue Mar 17 15:59:53 EST 2008: Sent packet 2.
Tue Mar 17 15:59:53 EST 2008: Received test packet 2 from self.
...
```

Leave the test running for approximately 5 minutes to ensure there are no failures. Press CTRL-C to stop further testing.

If you do not see something similar to the above, then multicast is not working. Also, note that a TTL of 0 was specified to prevent the multicast packets from leaving Server A.

Repeat the same test on Server B to assure that it too has the multicast enabled for it's port combination.

Next, test multicast communications between Server A and Server B. For this test use a nonzero TTL which allows the packets to leave their respective servers. By default, the test uses a TTL of 4, if more network hops are required to route packets between Server A and Server B, specify a higher TTL value.

Start the test on Server A and Server B by entering the following command into each's respective command window and pressing ENTER:

```
multicast-test.sh
```

The following example demonstrates sample output for Server A:

```
Starting test on ip=servera/195.0.0.1, group=/237.0.0.1:9000, ttl=4
Configuring multicast socket...
Starting listener...
Tue Mar 17 16:11:03 EST 2008: Sent packet 1.
Tue Mar 17 16:11:03 EST 2008: Received test packet 1 from self.
Tue Mar 17 16:11:05 EST 2008: Sent packet 2.
Tue Mar 17 16:11:05 EST 2008: Received test packet 2 from self.
Tue Mar 17 16:11:07 EST 2008: Sent packet 3.
Tue Mar 17 16:11:07 EST 2008: Received test packet 3 from self.
Tue Mar 17 16:11:09 EST 2008: Sent packet 4.
Tue Mar 17 16:11:09 EST 2008: Received test packet 4 from self.
Tue Mar 17 16:11:10 EST 2008: Received test packet 1 from ip=serverb/195.0.0.2, group=/
237.0.0.1:9000, ttl=4.
Tue Mar 17 16:11:11 EST 2008: Sent packet 5.
Tue Mar 17 16:11:11 EST 2008: Received test packet 5 from self.
Tue Mar 17 16:11:12 EST 2008: Received test packet 2 from ip=serverb/195.0.0.2, group=/
237.0.0.1:9000, ttl=4.
Tue Mar 17 16:11:13 EST 2008: Sent packet 6.
Tue Mar 17 16:11:13 EST 2008: Received test packet 6 from self.
Tue Mar 17 16:11:14 EST 2008: Received test packet 3 from ip=serverb/195.0.0.2, group=/
237.0.0.1:9000, ttl=4.
Tue Mar 17 16:11:15 EST 2008: Sent packet 7.
Tue Mar 17 16:11:15 EST 2008: Received test packet 7 from self.
...
```

The following example demonstrates sample output for Server B:

```
Starting test on ip=serverb/195.0.0.2, group=/237.0.0.1:9000, ttl=4
Configuring multicast socket...
Starting listener...
Tue Mar 17 16:11:10 EST 2008: Sent packet 1.
Tue Mar 17 16:11:10 EST 2008: Received test packet 1 from self.
Tue Mar 17 16:11:11 EST 2008: Received test packet 5 from ip=servera/195.0.0.1, group=/
237.0.0.1:9000, ttl=4.
Tue Mar 17 16:11:12 EST 2008: Sent packet 2.
Tue Mar 17 16:11:12 EST 2008: Received test packet 2 from self.
Tue Mar 17 16:11:13 EST 2008: Received test packet 6 from ip=servera/195.0.0.1, group=/
237.0.0.1:9000, ttl=4.
Tue Mar 17 16:11:14 EST 2008: Sent packet 3.
Tue Mar 17 16:11:14 EST 2008: Received test packet 3 from self.
Tue Mar 17 16:11:15 EST 2008: Received test packet 7 from ip=servera/195.0.0.1, group=/
237.0.0.1:9000, ttl=4.
...
```

In the example both Server A and Server B are issuing multicast packets and seeing their own and each other's packets. This indicates that multicast is functioning properly between these servers using the default multicast address and port.

> **Note:**
>
> Server A sees only its own packets (1-4) until it receives packet 1 from Server B.

Troubleshooting Multicast Communications

There are many issues that can potentially cause bidirectional multicast communication to fail. If a failure is observed, then begin by reviewing the troubleshooting tips for the most common issues. If you are unable to resolve multicast issues after troubleshooting, then consult with a network administrator or sysadmin to determine the cause and to correct the situation.

- Firewalls—If any of the computers running the multicast test employ firewalls, the firewall may be blocking the traffic. Consult your operating system/firewall documentation for details on allowing multicast traffic.
- Switches—Ensure that the switches are configured to forward multicast traffic.
- If the multicast test fails after initially succeeding, try running the following on a Coherence node:

  ```
  tcpdump -i nic_device igmp
  ```

  Where *nic_device* is the NIC device name. Make sure that IGMP Query Messages (either v2 or v3) are seen in the `tcpdump` output. Make sure the switch is enabled to send and receive IGMP Query Messages. Also make sure that the NIC and OS networking is set to respond to periodic IGMP Query Messages. Lastly, check the switch to make sure it sees the Coherence servers do both "IGMP Join" and "IGMP Query Message" acknowledgements. The output should be similar to:

  ```
  07:58:33.452537 IP (tos 0xc0, ttl 1, id 0, offset 0, flags [DF], proto IGMP
   (2), length 40, options (RA))

  longcoh06a1-priv.emea.kuoni.int > igmp.mcast.net: igmp v3 report, 1 group
   record(s) [gaddr 192.168.0.5 to_ex, 0 source(s)]

  07:58:43.294453 IP (tos 0xc0, ttl 1, id 0, offset 0, flags [DF], proto IGMP
   (2), length 40, options (RA))

    longcoh06a1-priv.emea.kuoni.int > igmp.mcast.net: igmp v3 report, 1 group
   record(s) [gaddr 192.168.0.5 to_ex, 0 source(s)]

  07:58:51.782848 IP (tos 0xc0, ttl 1, id 3133, offset 0, flags [none], proto
   IGMP (2), length 36, options (RA))

    10.241.113.40 > all-systems.mcast.net: igmp query v3 [max resp time 10s]

  08:00:56.803800 IP (tos 0xc0, ttl 1, id 3134, offset 0, flags [none], proto
   IGMP (2), length 36, options (RA))

    10.241.113.40 > all-systems.mcast.net: igmp query v3 [max resp time 10s]
  …
  ```

The first two lines are servers "joining" the multicast group. The remaining output are the IGMP Query Messages originating at the switch, these are continuous, every few minutes - if the switch is configured to send them, and the NIC is configured to respond.

- IPv6—On operating systems which support IPv6, Java may be attempting to route the Multicast traffic over IPv6 rather than IPv4. Try specifying the following Java system property to force IPv4 networking `java.net.preferIPv4Stack=true`. Coherence cluster members must all use either IPv4 or IPv6 and cannot use a mix of both.
- Received ???—If the test reports receiving "???" this is an indication that it is receiving multicast packets which did not originate from an instance of the Multicast test. This occurs if the test is run with the same multicast address as an existing Coherence cluster, or any other multicast application.
- Multiple NICs—If the computers have multiple network interfaces, try specifying an explicit interface by using the `-local test` parameter. For instance if Server A has two interfaces with IP addresses 195.0.0.1 and 195.0.100.1, including `-local 195.0.0.1` on the test command line would ensure that the multicast packets used the first interface. In addition, the computer's routing table may require explicit configuration to forward multicast traffic through the desired network interface. This can be done by issuing the following command:

  ```
  route add -net 224.0.0.0 netmask 240.0.0.0 dev eth1
  ```

  Where `eth1` is the device that is designated to transmit multicast traffic.
- AIX—On AIX systems, the following multicast issues may be encountered:
  - IPv6—In addition to specifying `java.net.preferIPv4Stack=true`, the operating system may require additional configuration to perform IPv4 name resolution. Add `hosts=local,bind4` to the `/etc/netsvc.conf` file.
  - Virtual IP (VIPA)—AIX does not support multicast with VIPA. If using VIPA either bind multicast to a non-VIPA device, or run Coherence with multicast disabled. See Using Well Known Addresses in *Developing Applications with Oracle Coherence*.
  - MTU—Configure the MTU for the multicast device to 1500 bytes.
- Cisco Switches—See Deploying to Cisco Switches.
- Foundry Switches—See Deploying to Foundry Switches.

4
Performance Tuning

A critical part of successfully deploying Coherence solutions is to tune the application and production environment to achieve maximum performance.This chapter includes many guidelines and considerations that can be used to tune performance and possibly identify performance issues. However, as with any production deployment, always run performance and stress tests to validate how a particular application and production environment performs.
This chapter includes the following sections:

Operating System Tuning

Operating system settings, such as socket buffers, thread scheduling, and disk swapping can be configured to minimize latency.
This section includes the following topics:

Socket Buffer Sizes

Large operating system socket buffers can help minimize packet loss during garbage collection. Each Coherence socket implementation attempts to allocate a default socket buffer size. A warning message is logged for each socket implementation if the default size cannot be allocated. The following example is a message for the inbound UDP socket buffer:

```
UnicastUdpSocket failed to set receive buffer size to 16 packets (1023KB); actual
size is 12%, 2 packets (127KB). Consult your OS documentation regarding increasing
the maximum socket buffer size. Proceeding with the actual value may cause
sub-optimal performance.
```

It is recommended that you configure the operating system to allow for larger buffers. However, alternate buffer sizes for Coherence packet publishers and unicast listeners can be

configured using the `<packet-buffer>` element. See Configuring the Size of the Packet Buffers in *Developing Applications with Oracle Coherence*.

> **Note:**
>
> Most versions of UNIX have a very low default buffer limit, which should be increased to at least 2MB. Also, note that UDP recommendations are only applicable for configurations which explicitly configure UDP in favor of TCP as TCP is the default for performance sensitive tasks.

On Linux, execute (as root):

```
sysctl -w net.core.rmem_max=2097152
sysctl -w net.core.wmem_max=2097152
```

On Solaris, execute (as root):

```
ndd -set /dev/udp udp_max_buf 2097152
```

On AIX, execute (as root):

```
no -o rfc1323=1
no -o sb_max=4194304
```

> **Note:**
>
> Note that AIX only supports specifying buffer sizes of 1MB, 4MB, and 8MB.

On Windows:

Windows does not impose a buffer size restriction by default.

Other:

For information on increasing the buffer sizes for other operating systems, refer to your operating system's documentation.

High Resolution timesource (Linux)

Linux has several high resolution timesources to choose from, the fastest TSC (Time Stamp Counter) unfortunately is not always reliable. Linux chooses TSC by default and during startup checks for inconsistencies, if found it switches to a slower safe timesource. The slower time sources can be 10 to 30 times more expensive to query then the TSC timesource, and may have a measurable impact on Coherence performance. For more details on TSC, see

https://lwn.net/Articles/209101/

Note that Coherence and the underlying JVM are not aware of the timesource which the operating system is using. It is suggested that you check your system logs (`/var/log/dmesg`) to verify that the following is not present.

```
kernel: Losing too many ticks!
kernel: TSC cannot be used as a timesource.
kernel: Possible reasons for this are:
kernel:   You're running with Speedstep,
kernel:   You don't have DMA enabled for your hard disk (see hdparm),
kernel:   Incorrect TSC synchronization on an SMP system (see dmesg).
kernel: Falling back to a sane timesource now.
```

As the log messages suggest, this can be caused by a variable rate CPU (SpeedStep), having DMA disabled, or incorrect TSC synchronization on multi CPU computers. If present, work with your system administrator to identify and correct the cause allowing the TSC timesource to be used.

Datagram size (Microsoft Windows)

Microsoft Windows supports a fast I/O path which is used when sending "small" datagrams. The default setting for what is considered a small datagram is 1024 bytes; increasing this value to match your network maximum transmission unit (MTU), normally 1500, can significantly improve network performance.

To adjust this parameter:

1. Run Registry Editor (`regedit`)
2. Locate the following registry key
   `HKLM\System\CurrentControlSet\Services\AFD\Parameters`
3. Add the following new `DWORD` value `Name: FastSendDatagramThreshold Value: 1500` (decimal)
4. Restart.

> **Note:**
>
> The *`COHERENCE_HOME`*`/bin/optimize.reg` script can also perform this change. After running the script, restart the computer for the changes to take effect.

TCP Retransmission Timeout (Microsoft Windows)

Microsoft Windows includes a TCP retransmission timeout that is used for existing and new connections. The default retransmission timeout can abandon connections in a matter of seconds based on the Windows automatic tuning for TCP data transmission on the network. The short timeout can result in the false positive detection of cluster member death by the `TcpRing` process and can result in data loss. The default retransmission timeout can be configured to be more tolerant of short outages that may occur on the production network.

To increase the TCP retransmission timeout:

1. Run Registry Editor (`regedit`)
2. Locate the following registry key
   `HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`
3. Add the following new `DWORD` value `Name: TcpMaxConnectRetransmissions Value: 00000015` (Hex)

4. Add the following new `DWORD` value `Name: TcpMaxDataRetransmissions Value: 00000015` (Hex)
5. Restart.

> **Note:**
>
> The *COHERENCE_HOME*`/bin/optimize.reg` script can also perform this change. After running the script, restart the computer for the changes to take effect.

Thread Scheduling (Microsoft Windows)

Windows is optimized for desktop application usage. If you run two console ("DOS box") windows, the one that has the focus can use almost 100% of the CPU, even if other processes have high-priority threads in a running state. To correct this imbalance, you must configure the Windows thread scheduling to less-heavily favor foreground applications.

1. Open the Control Panel.
2. Open **System**.
3. Select the **Advanced** tab.
4. Under **Performance** select **Settings**.
5. Select the **Advanced** tab.
6. Under **Processor** scheduling, choose **Background services**.

> **Note:**
>
> The *COHERENCE_HOME*`/bin/optimize.reg` script performs this change. After running the script, restart the computer for the changes to take effect.

Swapping

Swapping, also known as paging, is the use of secondary storage to store and retrieve application data for use in RAM memory. Swapping is automatically performed by the operating system and typically occurs when the available RAM memory is depleted. Swapping can have a significant impact on Coherence's performance and should be avoided. Often, swapping manifests itself as Coherence nodes being removed from the cluster due to long periods of unresponsiveness caused by them having been swapped out of RAM. See Avoid using virtual memory (paging to disk).

To avoid swapping, ensure that sufficient RAM memory is available on the computer or that the number of running processes is accounted for and do not consume all the available RAM. Tools such as `vmstat` and `top` (on Unix and Linux) and `taskmgr` (on Windows) should be used to monitor swap rates.

Swappiness in Linux

Linux, by default, may choose to swap out a process or some of its heap due to low usage even if it is not running low on RAM. Swappiness is performed to be ready to handle eventual memory requests. Swappiness should be avoided for Coherence JVMs. The swappiness setting on Linux is a value between 0 and 100, where higher values encourage more optimistic swapping. The default value is 60. For Coherence, a lower value (0 if possible) should always be set.

To see the current swappiness value that is set, enter the following at the command prompt:

```
cat /proc/sys/vm/swappiness
```

To temporarily set the swappiness, as the root user echo a value onto `/proc/sys/vm/swappiness`. The following example sets the value to 0.

```
echo 0 > /proc/sys/vm/swappiness
```

To set the value permanently, modify the `/etc/sysctl.conf` file as follows:

```
vm.swappiness = 0
```

Load Balancing Network Interrupts (Linux)

Linux kernels have the ability to balance hardware interrupt requests across multiple CPUs or CPU cores. The feature is referred to as SMP IRQ Affinity and results in better system performance as well as better CPU utilization. For Coherence, significant performance can be gained by balancing ethernet card interrupts for all servers that host cluster members. Most Linux distributions also support irqbalance, which is aware of the cache topologies and power management features of modern multi-core and multi-socket systems.

Most Linux installations are not configured to balance network interrupts. The default network interrupt behavior uses a single processor (typically CPU0) to handle all network interrupts and can become a serious performance bottleneck with high volumes of network traffic. Balancing network interrupts among multiple CPUs increases the performance of network-based operations.

For detailed instructions on how to configure SMP IRQ Affinity, see the following document which is only summarized below:

http://www.mjmwired.net/kernel/Documentation/IRQ-affinity.txt

To view a list of the system's IRQs that includes which device they are assigned to and the number of interrupts each processor has handled for the device, run the following command:

```
# cat /proc/interrupts
```

The following example output snippet shows a single network interface card where all interrupts have been handled by the same processor (CPU0). This particular network card has multiple transmit and receive queues which have their own assigned IRQ. Systems that use multiple network cards will have additional IRQs assigned for each card.

```
           CPU0       CPU1       CPU2       CPU3
65:      20041      0          0          0      IR-PCI-MSI-edge  eth0-tx-0
66:      20232      0          0          0      IR-PCI-MSI-edge  eth0-tx-1
67:      20105      0          0          0      IR-PCI-MSI-edge  eth0-tx-2
68:      20423      0          0          0      IR-PCI-MSI-edge  eth0-tx-3
69:      21036      0          0          0      IR-PCI-MSI-edge  eth0-rx-0
```

```
70:      20201      0          0          0      IR-PCI-MSI-edge  eth0-rx-1
71:      20587      0          0          0      IR-PCI-MSI-edge  eth0-rx-2
72:      20853      0          0          0      IR-PCI-MSI-edge  eth0-rx-3
```

The goal is to have the interrupts balanced across the 4 processors instead of just a single processor. Ideally, the overall utilization of the processors on the system should also be evaluated to determine which processors can handle additional interrupts. Use `mpstat` to view statistics for a system's processors. The statistics show which processors are being over utilized and which are being under utilized and help determine the best ways to balance the network interrupts across the CPUs.

SMP IRQ affinity is configured in an `smp_affinity` file. Each IRQ has its own `smp_affinity` file that is located in the `/proc/irq/irq_#/` directory. To see the current affinity setting for an IRQ (for example 65), run:

```
# cat /proc/irq/65/smp_affinity
```

The returned hexadecimal value is a bitmask and represents the processors to which interrupts on IRQ 65 are routed. Each place in the value represents a group of 4 CPUs. For a 4 processor system, the hexadecimal value to represent a group of all four processors is f (or 15) and is 00000f as mapped below:

```
          Binary       Hex
CPU 0     0001          1
CPU 1     0010          2
CPU 2     0100          4
CPU 3     1000          8
-----------------------
all       1111          f
```

To target a single processor or group of processors, the bitmask must be changed to the appropriate hexadecimal value. Based on the system in the example above, to direct all interrupts on IRQ 65 to CPU1 and all interrupts on IRQ 66 to CPU2, change the `smp_affinity` files as follows:

```
echo 000002 > /proc/irq/65/smp_affinity # eth0-tx-0
echo 000004 > /proc/irq/66/smp_affinity # eth0-tx-1
```

To direct all interrupts on IRQ 65 to both CPU1 and CPU2, change the `smp_affinity` file as follows:

```
echo 000006 > /proc/irq/65/smp_affinity # eth0-tx-0
```

To direct all interrupts on each IRQ to all CPUs, change the `smp_affinity` files as follows:

```
echo 00000f > /proc/irq/65/smp_affinity # eth0-tx-0
echo 00000f > /proc/irq/66/smp_affinity # eth0-tx-1
echo 00000f > /proc/irq/67/smp_affinity # eth0-tx-2
echo 00000f > /proc/irq/68/smp_affinity # eth0-tx-3
echo 00000f > /proc/irq/69/smp_affinity # eth0-rx-0
echo 00000f > /proc/irq/70/smp_affinity # eth0-rx-1
echo 00000f > /proc/irq/71/smp_affinity # eth0-rx-2
echo 00000f > /proc/irq/72/smp_affinity # eth0-rx-3
```

Network Tuning

Network settings, such as network link speeds, Ethernet flow-control, and path MTU can be configured to maximize network throughput.

This section includes the following topics:

Network Interface Settings

Verify that your Network card (NIC) is configured to operate at it's maximum link speed and at full duplex. The process for doing this varies between operating systems.

On Linux execute (as root):

```
ethtool eth0
```

See the man page on `ethtool` for further details and for information on adjust the interface settings.

On Solaris execute (as root):

```
kstat ce:0 | grep link_
```

This displays the link settings for interface 0. Items of interest are `link_duplex` (2 = full), and `link_speed` which is reported in Mbps.

> **Note:**
>
> If running on Solaris 10, review issues `1000972.1` and `1000940.1` which relate to packet corruption and multicast disconnections. These often manifest as either `EOFExceptions`, "Large gap" warnings while reading packet data, or frequent packet timeouts. It is highly recommend that the patches for both issues be applied when using Coherence on Solaris 10 systems.

On Windows:

1. Open the Control Panel.
2. Open **Network Connections**.
3. Open the **Properties** dialog for desired network adapter.
4. Select **Configure**.
5. Select the **Advanced** tab.
6. Locate the driver specific property for **Speed** & **Duplex**.
7. Set it to either auto or to a specific speed and duplex setting.

Network Infrastructure Settings

If you experience frequent multi-second communication pauses across multiple cluster nodes, try increasing your switch's buffer space. These communication pauses can be identified by a series of Coherence log messages identifying communication delays with multiple nodes which are not attributable to local or remote GCs.

Example 4-1 Message Indicating a Communication Delay

```
Experienced a 4172 ms communication delay (probable remote GC) with Member(Id=7,
Timestamp=2006-10-20 12:15:47.511, Address=192.168.0.10:8089, MachineId=13838);
320 packets rescheduled, PauseRate=0.31, Threshold=512
```

Some switches such as the Cisco 6500 series support configuring the amount of buffer space available to each Ethernet port or ASIC. In high load applications it may be necessary to increase the default buffer space. On Cisco, this can be accomplished by executing:

```
fabric buffer-reserve high
```

See Cisco's documentation for additional details on this setting.

Switch and Subnet Considerations

Cluster members may be split across multiple switches and may be part of multiple subnets. However, such topologies can overwhelm inter-switch links and increase the chances of a split cluster if the links fail.Typically, the impact materializes as communication delays that affect cluster and application performance. If possible, consider always locating all cluster members on the same switch and subnet to minimize the impact. See Evaluate the Production Network's Speed for both UDP and TCP.

Ethernet Flow-Control

Full duplex Ethernet includes a flow-control feature which allows the receiving end of a point to point link to slow down the transmitting end. This is implemented by the receiving end sending an Ethernet PAUSE frame to the transmitting end, the transmitting end then halts transmissions for the interval specified by the PAUSE frame. Note that this pause blocks **all** traffic from the transmitting side, even traffic destined for computers which are not overloaded. This can induce a head of line blocking condition, where one overloaded computer on a switch effectively slows down all other computers. Most switch vendors recommend that Ethernet flow-control be disabled for inter switch links, and at most be used on ports which are directly connected to computers. Even in this setup head of line blocking can still occur, and thus it is advisable to disable Ethernet flow-control. Higher level protocols such as TCP/IP and Coherence TCMP include their own flow-control mechanisms which are not subject to head of line blocking, and also negate the need for the lower level flow-control.

Path MTU

By default Coherence assumes a 1500 byte network MTU, and uses a default packet size of 1468 based on this assumption. Having a packet size which does not fill the MTU results in an under used network. If your equipment uses a different MTU, then

configure Coherence by specifying the `<packet-size>` setting, which is 32 bytes smaller then the network path's minimal MTU.

If you are unsure of your equipment's MTU along the full path between nodes, you can use either the standard `ping` or `traceroute` utilities to determine the MTU. For example, execute a series of ping or traceroute operations between the two computers. With each attempt, specify a different packet size, starting from a high value and progressively moving downward until the packets start to make it through without fragmentation.

On Linux execute:

```
ping -c 3 -M do -s 1468 serverb
```

On Solaris execute:

```
traceroute -F serverb 1468
```

On Windows execute:

```
ping -n 3 -f -l 1468 serverb
```

On other operating systems: Consult the documentation for the `ping` or `traceroute` command to see how to disable fragmentation, and specify the packet size.

If you receive a message stating that packets must be fragmented then the specified size is larger then the path's MTU. Decrease the packet size until you find the point at which packets can be transmitted without fragmentation. If you find that you must use packets smaller then 1468, you may want to contact your network administrator to get the MTU increased to at least 1500.

10GbE Considerations

Many 10 Gigabit Ethernet (10GbE) switches and network interface cards support frame sizes that are larger than the 1500 byte ethernet frame standard. When using 10GbE, make sure that the MTU is set to the maximum allowed by the technology (approximately 16KB for ethernet) to take full advantage of the greater bandwidth. Coherence automatically detects the MTU of the network and selects a UDP socket buffer size accordingly. UDP socket buffer sizes of 2MB, 4MB, or 8MB are selected for MTU sizes of 1500 bytes (standard), 9000 bytes (jumbo), and over 9000 (super jumbo), respectively. Also, make sure to increase the operating system socket buffers to 8MB to accommodate the larger sizes. A warning is issued in the Coherence logs if a significantly small operating system buffer is detected. Lastly, always run the datagram test to validate the performance and throughput of the network. See Performing a Network Performance Test.

TCP Considerations

Coherence utilizes a TCP message bus (TMB) to transmit messages between clustered data services. Therefore, a network must be optimally tuned for TCP. Coherence inherits TCP settings, including buffer settings, from the operating system. Most servers already have TCP tuned for the network and should not require additional configuration. The recommendation is to tune the TCP stack for the network instead of tuning Coherence for the network.

Coherence includes a message bus test utility that can be used to test throughput and latency between network nodes. See Running the Message Bus Test Utility. If a network shows poor performance, then it may not be properly configured, use the following

recommendations (note that these settings are demonstrated on Linux but can be translated to other operating systems):

```
#!/bin/bash
#
# aggregate size limitations for all connections, measured in pages; these values
# are for 4KB pages (getconf PAGESIZE)

/sbin/sysctl -w net.ipv4.tcp_mem='   65536   131072   262144'

# limit on receive space bytes per-connection; overridable by SO_RCVBUF; still
# goverened by core.rmem_max

/sbin/sysctl -w net.ipv4.tcp_rmem=' 262144  4194304  8388608'

# limit on send space bytes per-connection; overridable by SO_SNDBUF; still
# goverered by core.wmem_max

/sbin/sysctl -w net.ipv4.tcp_wmem='  65536  1048576  2097152'

# absolute limit on socket receive space bytes per-connection; cannot be
# overriden programatically

/sbin/sysctl -w net.core.rmem_max=16777216

# absolute limit on socket send space bytes per-connection; cannot be overriden;
# cannot be overriden programatically

/sbin/sysctl -w net.core.wmem_max=16777216
```

Each connection consumes a minimum of 320KB, but under normal memory pressure, consumes 5MB per connection and ultimately the operating system tries to keep the entire system buffering for TCP under 1GB. These are recommended defaults based on tuning for fast (>= 10gbe) networks and should be acceptable on 1gbe.

JVM Tuning

JVM runtime settings, such as heap size and garbage collection can be configured to ensure the right balance of resource utilization and performance.
This section includes the following topics:

- Basic Sizing Recommendation
- Heap Size Considerations
- Garbage Collection Monitoring

Basic Sizing Recommendation

The recommendations in this section are sufficient for general use cases and require minimal setup effort. The primary issue to consider when sizing your JVMs is a balance of available RAM versus garbage collection (GC) pause times.

Cache Servers

The standard, safe recommendation for Coherence cache servers is to run a fixed size heap of up to 8GB. In addition, use an incremental garbage collector to minimize GC pause durations. Lastly, run all Coherence JVMs in server mode, by specifying the -

`server` on the JVM command line. This allows for several performance optimizations for long running applications.

For example:

```
java -server -Xms8g -Xmx8g -Xloggc: -jar coherence.jar
```

This sizing allows for good performance without the need for more elaborate JVM tuning. See Garbage Collection Monitoring.

Larger heap sizes are possible and have been implemented in production environments; however, it becomes more important to monitor and tune the JVMs to minimize the GC pauses. It may also be necessary to alter the storage ratios such that the amount of scratch space is increased to facilitate faster GC compactions. Additionally, it is recommended that you make use of an up-to-date JVM version to ensure the latest improvements for managing large heaps. See Heap Size Considerations.

TCMP Clients

Coherence TCMP clients should be configured similarly to cache servers as long GCs could cause them to be misidentified as being terminated.

Extends Clients

Coherence Extend clients are not technically speaking cluster members and, as such, the effect of long GCs is less detrimental. For extend clients it is recommended that you follow the existing guidelines as set forth by the application in which you are embedding coherence.

Heap Size Considerations

Use this section to decide:

- How many CPUs are need for your system
- How much memory is need for each system
- How many JVMs to run per system
- How much heap to configure with each JVM

Since all applications are different, this section should be read as guidelines. You must answer the following questions to choose the configuration that is right for you:

- How much data is to be stored in Coherence caches?
- What are the application requirements in terms of latency and throughput?
- How CPU or Network intensive is the application?

Sizing is an imprecise science. There is no substitute for frequent performance and stress testing.

This section includes the following topics:

- General Guidelines
- Moving the Cache Out of the Application Heap

General Guidelines

Running with a fixed sized heap saves the JVM from having to grow the heap on demand and results in improved performance. To specify a fixed size heap use the `-Xms` and `-Xmx` JVM options, setting them to the same value. For example:

```
java -server -Xms4G -Xmx4G ...
```

A JVM process consumes more system memory then the specified heap size. The heap size settings specify the amount of heap which the JVM makes available to the application, but the JVM itself also consumes additional memory. The amount consumed differs depending on the operating system and JVM settings. For instance, a HotSpot JVM running on Linux configured with a 1GB JVM consumes roughly 1.2GB of RAM. It is important to externally measure the JVMs memory utilization to ensure that RAM is not over committed. Tools such as `top`, `vmstat`, and Task Manager are useful in identifying how much RAM is actually being used.

Storage Ratios

The basic starting point for how much data can be stored within a cache server of a given size is to use a 1/3rd of the heap for primary cache storage. This leaves another 1/3rd for backup storage and the final 1/3rd for scratch space. Scratch space is then used for things such as holding classes, temporary objects, network transfer buffers, and GC compaction. However, this recommendation is considered a basic starting point and should not be considered a rule. A more precise, but still conservative starting point, is to assume your cache data can occupy no more than the total heap minus two times the young generation size of a JVM heap (for example, 32GB – (2 * 4GB) = 24GB). In this case, cache data can occupy 75% of the heap. Note that the resulting percentage depends on the configured young generation size. In addition, you may instruct Coherence to limit primary storage on a per-cache basis by configuring the `<high-units>` element and specifying a `BINARY` value for the `<unit-calculator>` element. These settings are automatically applied to backup storage as well.

Ideally, both the primary and backup storage also fits within the JVMs tenured space (for HotSpot-based JVMs). See Sizing the Generations in *Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning*.

Cache Topologies and Heap Size

For large data sets, partitioned or near caches are recommended. Varying the number of Coherence JVMs does not significantly affect cache performance because the scalability of the partitioned cache is linear for both reading and writing. Using a replicated cache puts significant pressure on GC.

Planning Capacity for Data Grid Operations

Data grid operations (such as queries and aggregations) have additional heap space requirements and their use must be planned for accordingly. During data grid operations, binary representations of the cache entries that are part of the result set are held in-memory. Depending on the operation, the entries may also be held in deserialized form for the duration of the operation. Typically, this doubles the amount of memory for each entry in the result set. In addition, a second binary copy is maintained when using RAM or flash journaling as the backing map implementation due to differences in how the objects are stored. The second binary copy is also held

for the duration of the operation and increases the total memory for each entry in the result set by 3x.

Heap memory usage depends on the size of the result set on which the operations are performed and the number of concurrent requests being handled. The result set size is affected by both the total number of entries as well as the size of each entry. Moderately sized result sets that are maintained on a storage cache server would not likely exhaust the heap's memory. However, if the result set is sufficiently large, the additional memory requirements can cause the heap to run out of memory. Data grid aggregate operations typically involve larger data sets and are more likely to exhaust the available memory than other operations.

The JVMs heap size should be increased on storage enabled cache servers whenever large result sets are expected. For example, if a third of the heap has been reserved for scratch space, then the scratch space must also support the projected result set sizes. Alternatively, data grid operations can use the `PartitionedFilter` API. The API reduces memory consumption by executing grid operations against individual partition sets.

Deciding How Many JVMs to Run Per System

The number of JVMs (nodes) to run per system depends on the system's number of processors/cores and amount of memory. As a starting point, plan to run one JVM for every four cores. This recommendation balances the following factors:

- Multiple JVMs per server allow Coherence to make more efficient use of high-bandwidth (>1gb) network resources.
- Too many JVMs increases contention and context switching on processors.
- Too few JVMs may not be able to handle available memory and may not fully use the NIC.
- Especially for larger heap sizes, JVMs must have available processing capacity to avoid long GC pauses.

Depending on your application, you can add JVMs up toward one per core. The recommended number of JVMs and amount of configured heap may also vary based on the number of processors/cores per socket and on the computer architecture.

> **Note:**
>
> Applications that use Coherence as a basic cache (get, put and remove operations) and have no application classes (entry processors, aggregators, queries, cachestore modules, and so on) on the cache server can sometimes go beyond 1 JVM per core. They should be tested for both health and failover scenarios.

Sizing Your Heap

When considering heap size, it is important to find the right balance. The lower bound is determined by per-JVM overhead (and also, manageability of a potentially large number of JVMs). For example, if there is a fixed overhead of 100MB for infrastructure software (for example, JMX agents, connection pools, internal JVM structures), then the use of JVMs with 256MB heap sizes results in close to 40% overhead for non-cache data. The upper bound on JVM heap size is governed by memory management overhead, specifically the maximum duration of GC pauses and the percentage of CPU allocated to GC (and other memory management tasks).

GC can affect the following:

- The latency of operations against Coherence. Larger heaps cause longer and less predictable latency than smaller heaps.
- The stability of the cluster. With very large heaps, lengthy long garbage collection pauses can trick TCMP into believing a cluster member is terminated since the JVM is unresponsive during GC pauses. Although TCMP takes GC pauses into account when deciding on member health, at some point it may decide the member is terminated.

The following guideline is provided:

- For Java, a conservative heap size of 8GB is recommended for most applications and is based on throughput, latency, and stability. However, larger heap sizes, are suitable for some applications where the simplified management of fewer, larger JVMs outweighs the performance benefits of many smaller JVMs. A core-to-heap ratio of roughly 4 cores: 8GB is ideal, with some leeway to manage more GBs per core. Every application is different and GC must be monitored accordingly.

The length of a GC pause scales worse than linearly to the size of the heap. That is, if you double the size of the heap, pause times due to GC more than double (in general). GC pauses are also impacted by application usage:

- Pause times increase as the amount of live data in the heap increases. Do not exceed 70% live data in your heap. This includes primary data, backup data, indexes, and application data.
- High object allocation rates increase pause times. Even "simple" Coherence applications can cause high object allocation rates since every network packet generates many objects.
- CPU-intensive computations increase contention and may also contribute to higher pause times.

Depending on your latency requirements, you can increase allocated heap space beyond the above recommendations, but be sure to stress test your system.

Moving the Cache Out of the Application Heap

Using dedicated Coherence cache server instances for Partitioned cache storage minimizes the heap size of application JVMs because the data is no longer stored locally. As most Partitioned cache access is remote (with only `1/N` of data being held locally), using dedicated cache servers does not generally impose much additional overhead. Near cache technology may still be used, and it generally has a minimal impact on heap size (as it is caching an even smaller subset of the Partitioned cache). Many applications are able to dramatically reduce heap sizes, resulting in better responsiveness.

Local partition storage may be enabled (for cache servers) or disabled (for application server clients) with the `coherence.distributed.localstorage` Java property (for example, `-Dcoherence.distributed.localstorage=false`).

It may also be disabled by modifying the `<local-storage>` setting in the `tangosol-coherence.xml` (or `tangosol-coherence-override.xml`) file as follows:

Example 4-2 Disabling Partition Storage

```
<?xml version='1.0'?>
```

```
<coherence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-operational-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-operational-config
   coherence-operational-config.xsd">
   <cluster-config>
      <services>
         <!--
         id value must match what's in tangosol-coherence.xml for DistributedCache
         service
         -->
         <service id="3">
            <init-params>
               <init-param id="4">
                  <param-name>local-storage</param-name>
                  <param-value system-property="coherence.distributed.
                     localstorage">false</param-value>
               </init-param>
            </init-params>
         </service>
      </services>
   </cluster-config>
</coherence>
```

At least one storage-enabled JVM must be started before any storage-disabled clients access the cache.

Garbage Collection Monitoring

Lengthy GC pause times can negatively impact the Coherence cluster and are typically indistinguishable from node termination. A Java application cannot send or receive packets during these pauses. As for receiving the operating system buffered packets, the packets may be discarded and must be retransmitted. For these reasons, it is very important that cluster nodes are sized and tuned to ensure that their GC times remain minimal. As a general rule, a node should spend less than 10% of its time paused in GC, normal GC times should be under 100ms, and maximum GC times should be around 1 second. See Introduction in *Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning*.

Log messages are generated when one cluster node detects that another cluster node has been unresponsive for a period of time, generally indicating that a target cluster node was in a GC cycle.

Example 4-3 Message Indicating Target Cluster Node is in Garbage Collection Mode

```
Experienced a 4172 ms communication delay (probable remote GC) with Member(Id=7,
Timestamp=2006-10-20 12:15:47.511, Address=192.168.0.10:8089, MachineId=13838); 320
packets rescheduled, PauseRate=0.31, Threshold=512
```

`PauseRate` indicates the percentage of time for which the node has been considered unresponsive since the statistics were last reset. Nodes reported as unresponsive for more then a few percent of their lifetime may be worth investigating for GC tuning.

GC activity can be monitored in many ways; some Oracle HotSpot mechanisms include:

- `-verbose:gc` (writes GC log to standard out; use `-Xloggc` to direct it to some custom location)
- `-XX:+PrintGCDetails`, `-XX:+PrintGCTimeStamps`, `-XX:+PrintHeapAtGC`, `-XX:+PrintTenuringDistribution`, `-XX:+PrintGCApplicationStoppedTime` `-XX:+PrintGCApplicationConcurrentTime`

- `-Xprof`: enables profiling. Profiling activities should be distinguished between testing and production deployments and its effects on resources and performance should always be monitored
- JConsole and VisualVM (including VisualGC plug-in) that are included with the JDK.

Data Access Patterns

Understanding how an application uses a cache can help you determine how to configure the cache for maximum performance.
This section includes the following topics:

- Data Access Distribution (hot spots)
- Cluster-node Affinity
- Read/Write Ratio and Data Sizes
- Interleaving Cache Reads and Writes
- Concurrent Near Cache Misses on a Specific Hot Key

Data Access Distribution (hot spots)

When caching a large data set, typically a small portion of that data set is responsible for most data accesses. For example, in a 1000 object datasets, 80% of operations may be against a 100 object subset. The remaining 20% of operations may be against the other 900 objects. Obviously the most effective return on investment is gained by caching the 100 most active objects; caching the remaining 900 objects provides 25% more effective caching while requiring a 900% increase in resources.

However, if every object is accessed equally often (for example in sequential scans of the datasets), then caching requires more resources for the same level of effectiveness. In this case, achieving more than 0% effectiveness requires caching 100% of the data. (Note that sequential scans of partially cached data sets generally defeat MRU, LFU and MRU-LFU eviction policies). In practice, most non-synthetic (benchmark) data access patterns are uneven, and respond well to caching subsets of data.

In cases where a subset of data is active, and a smaller subset is particularly active, Near caching can be very beneficial when used with the `all` invalidation strategy (this is effectively a two-tier extension of the above rules).

Cluster-node Affinity

Coherence's Near cache technology transparently takes advantage of cluster-node affinity, especially when used with the `present` invalidation strategy. This topology is particularly useful when used with a sticky load-balancer. Note that the `present` invalidation strategy results in higher overhead (as opposed to `all`) when the front portion of the cache is "thrashed" (very short lifespan of cache entries); this is due to the higher overhead of adding/removing key-level event listeners. In general, a cache should be tuned to avoid thrashing and so this is usually not an issue.

Read/Write Ratio and Data Sizes

Generally speaking, the following cache topologies are best for the following use cases:

- Replicated cache—small amounts of read-heavy data (for example, metadata)
- Partitioned cache—large amounts of read/write data (for example, large data caches)
- Near cache—similar to Partitioned, but has further benefits from read-heavy tiered access patterns (for example, large data caches with hotspots) and "sticky" data access (for example, sticky HTTP session data). Depending on the synchronization method (expiry, asynchronous, synchronous), the worst case performance may range from similar to a Partitioned cache to considerably worse.

Interleaving Cache Reads and Writes

Interleaving refers to the number of cache reads between each cache write. The Partitioned cache is not affected by interleaving (as it is designed for 1:1 interleaving). The Replicated and Near caches by contrast are optimized for read-heavy caching, and prefer a read-heavy interleave (for example, 10 reads between every write). This is because they both locally cache data for subsequent read access. Writes to the cache forces these locally cached items to be refreshed, a comparatively expensive process (relative to the near-zero cost of fetching an object off the local memory heap). Note that with the Near cache technology, worst-case performance is still similar to the Partitioned cache; the loss of performance is relative to best-case scenarios.

Note that interleaving is related to read/write ratios, but only indirectly. For example, a Near cache with a 1:1 read/write ratio may be extremely fast (all writes followed by all reads) or much slower (1:1 interleave, write-read-write-read...).

Concurrent Near Cache Misses on a Specific Hot Key

Frequent cache misses by multiple clients, concurrently on a specific hot key that will never have a value in a near cache, result in all the misses being serialized across the cluster. The backlog of multiple cache lookups on the same hot key results in cluster slowdown around registering/unregistering of the message listener for the hot key in question. The more concurrent clients request the same key, the longer the delay will be for the clients.

To avoid this issue, Oracle recommends you to completely avoid requesting a value for a key that will never have a value from a near cache.

If that is not possible, use negative caching to resolve performance slow down. Negative caching turns the expensive misses that cannot be satisfied, into a cheap miss that returns a value, which by application convention, indicates no value for the key. However, it is a value that is kept in the near cache to avoid the overhead of frequent misses on the same key.

Distributed Tracing

During development, there may be unexpected latencies involved in request processing. OpenTracing is the ideal tool to help diagnose such issues. See Using Distributed Tracing.

5

Production Checklist

There are many production related issues to consider when moving a Coherence solution from a development or test environment to a production environment. The production checklist provides a comprehensive set of best practices that can be implemented as required to ensure a smooth transition to a production environment. Always test your Coherence solution in the production environment to identify potential resource and performance issues.
This chapter includes the following sections:

Network Performance Test and Multicast Recommendations

Configured and test network communication.

Test TCP Network Performance

Run the message bus test utility to test the actual network speed and determine its capability for pushing large amounts TCP messages. Any production deployment should be preceded by a successful run of the message bus test. See Running the Message Bus Test Utility. A TCP stack is typically already configured for a network and requires no additional configuration for Coherence. If TCP performance is unsatisfactory, consider changing TCP settings. See TCP Considerations.

Test Datagram Network Performance

Run the datagram test utility to test the actual network speed and determine its capability for pushing datagram messages. Any production deployment should be preceded by a

successful run of both tests. See Performing a Network Performance Test. Furthermore, the datagram test utility must be run with an increasing ratio of publishers to consumers, since a network that appears fine with a single publisher and a single consumer may completely fall apart as the number of publishers increases.

Consider the Use of Multicast

The term multicast refers to the ability to send a packet of information from one server and to have that packet delivered in parallel by the network to many servers. Coherence supports both multicast and multicast-free clustering. The use of multicast can be used to ease cluster configuration. However, the use of multicast may not always be possible for several reasons:

- Some organizations disallow the use of multicast.
- Multicast cannot operate over certain types of network equipment; for example, many WAN routers disallow or do not support multicast traffic.
- Multicast is occasionally unavailable for technical reasons; for example, some switches do not support multicast traffic.

Run the multicast test to verify that multicast is working and to determine the correct (the minimum) TTL value for the production environment. Any production deployment should be preceded by a successful run of the multicast test. See Performing a Multicast Connectivity Test.

Applications that cannot use multicast for deployment must use unicast and the well known addresses feature. See Using Well Known Addresses in *Developing Applications with Oracle Coherence*.

Configure Network Devices

Network devices may require configuration even if all network performance tests and the multicast test pass without incident and the results are perfect. See Network Tuning.

Changing the Default Cluster Port

The default cluster port is 7574 and for most use cases does not need to be changed. This port number, or any other selected port number, must not be within the operating system ephemeral port range. Ephemeral ports can be randomly assigned to other processes and can result in Coherence not being able to bind to the port during startup. On most operating systems, the ephemeral port range typically starts at 32,768 or higher. Some versions of Linux, such as Red Hat, have a much lower ephemeral port range and additional precautions must be taken to avoid random bind failures.

On Linux the ephemeral port range can be queried as follows:

```
sysctl net.ipv4.ip_local_port_range

sysctl net.ipv4.ip_local_reserved_ports
```

The first command shows the range as two space separated values indicating the start and end of the range. The second command shows exclusions from the range as a comma separated list of reserved ports, or reserved port ranges (for example, (1,2,10-20,40-50, and so on).

If the desired port is in the ephemeral range and not reserved, you can modify the reserved set and optionally narrow the ephemeral port range. This can be done as root be editing `/etc/sysctl.conf`. For example:

```
net.ipv4.ip_local_port_range = 9000 65000
net.ipv4.ip_local_reserved_ports = 7574
```

After editing the file you can then trigger a reload of the settings by running:

```
sysctl -p
```

Network Recommendations

Test the production network.

Ensure a Consistent IP Protocol

It is suggested that cluster members share the same setting for the `java.net.preferIPv4Stack` property. In general, this property does not need to be set. If there are multiple clusters running on the same machine and they share a cluster port, then the clusters must also share the same value for this setting. In rare circumstances, such as running multicast over the loopback address, this setting may be required.

Test in a Clustered Environment

After the POC or prototype stage is complete, and until load testing begins, it is not out of the ordinary for an application to be developed and tested by engineers in a non-clustered form. Testing primarily in the non-clustered configuration can hide problems with the application architecture and implementation that appear later in staging or even production.

Make sure that the application has been tested in a clustered configuration before moving to production. There are several ways for clustered testing to be a natural part of the development process; for example:

- Developers can test with a locally clustered configuration (at least two instances running on their own computer). This works well with the `TTL=0` setting, since clustering on a single computer works with the `TTL=0` setting.
- Unit and regression tests can be introduced that run in a test environment that is clustered. This may help automate certain types of clustered testing that an individual developer would not always remember (or have the time) to do.

Evaluate the Production Network's Speed for both UDP and TCP

Most production networks are based on 10 Gigabit Ethernet (10GbE), with some still built on Gigabit Ethernet (GbE) and 100Mb Ethernet. For Coherence, GbE and 10GbE are suggested and 10GbE is recommended. Most servers support 10GbE, and switches are economical, highly available, and widely deployed.

It is important to understand the topology of the production network, and what the devices are used to connect all of the servers that run Coherence. For example, if there are ten different switches being used to connect the servers, are they all the same type (make and model) of switch? Are they all the same speed? Do the servers support the network speeds that are available?

In general, all servers should share a reliable, fully switched network. This generally implies sharing a single switch (ideally, two parallel switches and two network cards per server for availability). There are two primary reasons for this. The first is that using multiple switches

almost always results in a reduction in effective network capacity. The second is that multi-switch environments are more likely to have network partitioning events where a partial network failure results in two or more disconnected sets of servers. While partitioning events are rare, Coherence cache servers ideally should share a common switch.

To demonstrate the impact of multiple switches on bandwidth, consider several servers plugged into a single switch. As additional servers are added, each server receives dedicated bandwidth from the switch backplane. For example, on a fully switched gigabit backplane, each server receives a gigabit of inbound bandwidth and a gigabit of outbound bandwidth for a total of 2Gbps full duplex bandwidth. Four servers would have an aggregate of 8Gbps bandwidth. Eight servers would have an aggregate of 16Gbps. And so on up to the limit of the switch (in practice, usually in the range of 160-192Gbps for a gigabit switch). However, consider the case of two switches connected by a 4Gbps (8Gbps full duplex) link. In this case, as servers are added to each switch, they have full mesh bandwidth up to a limit of four servers on each switch (that is, all four servers on one switch can communicate at full speed with the four servers on the other switch). However, adding additional servers potentially create a bottleneck on the inter-switch link. For example, if five servers on one switch send data to five servers on the other switch at 1Gbps per server, then the combined 5Gbps is restricted by the 4Gbps link. Note that the actual limit may be much higher depending on the traffic-per-server and also the portion of traffic that must actually move across the link. Also note that other factors such as network protocol overhead and uneven traffic patterns may make the usable limit much lower from an application perspective.

Avoid mixing and matching network speeds: make sure that all servers connect to the network at the same speed and that all of the switches and routers between those servers run at that same speed or faster.

Plan for Sustained Network Outages

The Coherence cluster protocol can detect and handle a wide variety of connectivity failures. The clustered services are able to identify the connectivity issue and force the offending cluster node to leave and re-join the cluster. In this way the cluster ensures a consistent shared state among its members. See Death Detection Recommendations and Deploying to Cisco Switches.

Plan for Firewall Port Configuration

Coherence clusters members that are located outside of a firewall must be able to communicate with cluster members that are located within the firewall. Configure the firewall to allow Coherence communication as required. The following list shows common default ports and additional areas where ports are configured.

> **Note:**
>
> In general, using a firewall within a cluster (even between TCMP clients and TCMP servers) is an anti-pattern as it is very easy to mis-configure and prone to reliability issues that can be hard to troubleshoot in a production environment. By definition, any member within a cluster should be considered trusted. Untrusted members should not be allowed into the cluster and should connect as Coherence extend clients or using a services layer (HTTP, SOA, and so on).

- cluster port: The default cluster port is 7574. The cluster port should be open in the firewall for both UDP and TCP traffic.
- unicast ports: Unicast uses TMB (default) and UDP. Each cluster member listens on one UDP and one TCP port and both ports need to be opened in the firewall. The default unicast ports are automatically assigned from the operating system's available ephemeral port range. For clusters that need to communicate across a firewall, a range of ports can be specified for coherence to operate within. Using a range rather then a specific port allows multiple cluster members to reside on the same machine and use a common configuration. See Specifying a Cluster Member's Unicast Address in *Developing Applications with Oracle Coherence*.
- port 7: The default TCP port of the IpMonitor component that is used for detecting hardware failure of cluster members. Coherence doesn't bind to this port, it only tries to connect to it as a means of pinging remote machines. The port needs to be open in order for Coherence to do health monitoring checks.
- Proxy service ports: The proxy listens by default on the same TCP port as the unicast port. For firewall-based configurations, this can be restricted to a range of ports which can then be opened in the firewall. Using a range of ports allows multiple cluster members to be run on the same machine and share a single common configuration. See Defining a Single Proxy Service Instance in *Developing Remote Clients for Oracle Coherence*.
- Coherence REST ports: Any number of TCP ports that are used to allow remote connections from Coherence REST clients. See Deploying Coherence REST in *Developing Remote Clients for Oracle Coherence*.

Ensure that IP Masquerading (IPMASQ) is Not Enabled

IP masquerading rules block some types of traffic that Coherence requires to form clusters. If you are not able to form clusters, check for the issue using the following command:

```
# iptables -t nat -v  -L POST_public_allow -n
Chain POST_public_allow (1 references)
pkts bytes target     prot opt in     out     source
destination
164K   11M MASQUERADE  all  --  *      !lo     0.0.0.0/0            0.0.0.0/0
   0     0 MASQUERADE  all  --  *      !lo     0.0.0.0/0            0.0.0.0/0
```

If you see an output similar to the above example, you need to remove them. You can remove the entries using this command:

```
# iptables -t nat -v -D POST_public_allow 1
```

You will need to run the command for each line. So in the example above, you will need to run it twice. After you are done, run the previous command again to verify that the output is an empty list. After you make this change, restart the cluster. You can now form the Coherence cluster correctly.

Cache Size Calculation Recommendations

Calculate the approximate size of a cache. Understanding what size cache is required can help determine how many JVMs, how much physical memory, and how many CPUs and servers are required. Hardware and JVM recommendations are provided later in this chapter.

The recommendations in this section are only guidelines: an accurate view of size can only be validated through specific tests that take into account an application's load and use cases that simulate expected users volumes, transactions profiles, processing operations, and so on.

As a starting point, allocate at least 3x the physical heap size as the data set size, assuming that you are going to keep 1 backup copy of primary data. To make a more accurate calculation, the size of a cache can be calculated as follows and also assumes 1 backup copy of primary data:

*Cache Capacity = Number of entries * 2 * Entry Size*

Where:

Entry Size = Serialized form of the key + Serialized form of the Value + 150 bytes

For example, consider a cache that contains 5 million objects, where the value and key serialized are 100 bytes and 2kb, respectively.

Calculate the entry size:

```
100 bytes + 2048 bytes + 150 bytes = 2298 bytes
```

Then, calculate the cache capacity:

```
5000000 * 2 * 2298 bytes = 21,915 MB
```

If indexing is used, the index size must also be taken into account. Un-ordered cache indexes consist of the serialized attribute value and the key. Ordered indexes include additional forward and backward navigation information.

Indexes are stored in memory. Each node will require 2 additional maps (instances of `java.util.HashMap`) for an index: one for a reverse index and one for a forward index. The reverse index size is a cardinal number for the value (size of the value domain, that is, the number of distinct values). The forward index size is of the key set size. The extra memory cost for the `HashMap` is about 30 bytes. Extra cost for each extracted indexed value is 12 bytes (the object reference size) plus the size for the value itself.

For example, the extra size for a `Long` value is 20 bytes (12 bytes + 8 bytes) and for a String is 12 bytes + the string length. There is also an additional reference (12 bytes) cost for indexes with a large cardinal number and a small additional cost (about 4 bytes) for sorted indexes. Therefore, calculate an approximate index cost as:

Index size = forward index map + backward index map + reference + value size

For an indexed `Long` value of large cardinal, it's going to be approximately:

```
30 bytes + 30 bytes + 12 bytes + 8 bytes = 80 bytes
```

For an indexed String of an average length of 20 chars it's going to be approximately:

```
30 bytes + 30 bytes + 12 bytes + (20 bytes * 2) = 112 bytes
```

The index cost is relatively high for small objects, but it's constant and becomes less and less expensive for larger objects.

Sizing a cache is not an exact science. Assumptions on the size and maximum number of objects have to be made. A complete example follows:

- *Estimated average entry size* = `1k`
- *Estimated maximum number of cache objects* = `100k`

- *String indexes* of 20 chars = `5`

Calculate the index size:

```
5 * 112 bytes * 100k = 56MB
```

Then, calculate the cache capacity:

```
100k * 2 * 1k + 56MB = ~312MB
```

Each JVM stores on-heap data itself and requires some free space to process data. With a 1GB heap this will be approximately 300MB or more. The JVM process address space for the JVM – outside of the heap is also approximately 200MB. Therefore, to store 312MB of data requires the following memory for each node in a 2 node JVM cluster:

`312MB` (for data) `+ 300MB` (working JVM heap) `+ 200MB` (JVM executable) `= 812MB` (of physical memory)

Note that this is the minimum heap space that is required. It is prudent to add additional space, to take account of any inaccuracies in your estimates, about 10%, and for growth (if this is anticipated). Also, adjust for M+N redundancy. For example, with a 12 member cluster that needs to be able to tolerate a loss of two servers, the aggregate cache capacity should be based on 10 servers and not 12.

With the addition of JVM memory requirements, the complete formula for calculating memory requirements for a cache can be written as follows:

*Cache Memory Requirement = (Size of cache entries * 2 (for primary and backup)) + Size of indexes + JVM working memory (~30% of 1GB JVM)*

Hardware Recommendations

Understand the hardware requirements and test the hardware accordingly.

Plan Realistic Load Tests

Development typically occurs on relatively fast workstations. Moreover, test cases are usually non-clustered and tend to represent single-user access (that is, only the developer). In such environments the application may seem extraordinarily responsive.

Before moving to production, ensure that realistic load tests have been routinely run in a cluster configuration with simulated concurrent user load.

Develop on Adequate Hardware Before Production

Coherence is compatible with all common workstation hardware. Most developers use PC or Apple hardware, including notebooks, desktops and workstations.

Developer systems should have a significant amount of RAM to run a modern IDE, debugger, application server, database and at least two cluster instances. Memory utilization varies widely, but to ensure productivity, the suggested minimum memory configuration for developer systems is 2GB.

Select a Server Hardware Platform

Oracle works to support the hardware that the customer has standardized on or otherwise selected for production deployment.

- Oracle has customers running on virtually all major server hardware platforms. The majority of customers use "commodity x86" servers, with a significant number deploying Oracle SPARC and IBM Power servers.
- Oracle continually tests Coherence on "commodity x86" servers, both Intel and AMD.
- Intel, Apple and IBM provide hardware, tuning assistance and testing support to Oracle.

If the server hardware purchase is still in the future, the following are suggested for Coherence:

It is strongly recommended that servers be configured with a minimum of 32GB of RAM. For applications that plan to store massive amounts of data in memory (tens or hundreds of gigabytes, or more), evaluate the cost-effectiveness of 128GB or even 256GB of RAM per server. Also, note that a server with a very large amount of RAM likely must run more Coherence nodes (JVMs) per server to use that much memory, so having a larger number of CPU cores helps. Applications that are data-heavy require a higher ratio of RAM to CPU, while applications that are processing-heavy require a lower ratio.

A minimum of 1000Mbps for networking (for example, Gigabit Ethernet or better) is strongly recommended. NICs should be on a high bandwidth bus such as PCI-X or PCIe, and not on standard PCI.

Plan the Number of Servers

Coherence is primarily a scale-out technology. The natural mode of operation is to span many servers (for example, 2-socket or 4-socket commodity servers). However, Coherence can also effectively scale-up on a small number of large servers by using multiple JVMs per server. Failover and failback are more efficient the more servers that are present in the cluster and the impact of a server failure is lessened. A cluster should contain a minimum of four physical servers to minimize the possibility of data loss during a failure. In most WAN configurations, each data center has independent clusters (usually interconnected by Extend-TCP). This increases the total number of discrete servers (four servers per data center, multiplied by the number of data centers).

Coherence is often deployed on smaller clusters (one, two or three physical servers) but this practice has increased risk if a server failure occurs under heavy load. In addition, Coherence clusters are ideally confined to a single switch (for example, fewer than 96 physical servers). In some use cases, applications that are compute-bound or memory-bound applications (as opposed to network-bound) may run acceptably on larger clusters. See Evaluate the Production Network's Speed for both UDP and TCP.

Also, given the choice between a few large JVMs and a lot of small JVMs, the latter may be the better option. There are several production environments of Coherence that span hundreds of JVMs. Some care is required to properly prepare for clusters of this size, but smaller clusters of dozens of JVMs are readily achieved.

Decide How Many Servers are Required Based on JVMs Used

The following rules should be followed in determining how many servers are required for reliable high availability configuration and how to configure the number of *storage-enabled* JVMs.

- There must be more than two servers. A grid with only two servers stops being machine-safe as soon as several JVMs on one server are different than the

number of JVMs on the other server; so, even when starting with two servers with equal number of JVMs, losing one JVM forces the grid out of machine-safe state. If the number of JVMs becomes unequal it may be difficult for Coherence to assign partitions in a way that ensures both equal per-member utilization as well as the placement of primary and backup copies on different machines. As a result, the recommended best practice is to use more than two physical servers.

- For a server that has the largest number of JVMs in the cluster, that number of JVMs must not exceed the total number of JVMs on all the other servers in the cluster.
- A server with the smallest number of JVMs should run at least half the number of JVMs as a server with the largest number of JVMs; this rule is particularly important for smaller clusters.
- The margin of safety improves as the number of JVMs tends toward equality on all computers in the cluster; this is more of a general practice than the preceding rules.

Operating System Recommendations

Select and configure an operating system.

Selecting an Operating System

Oracle tests on and supports the following operating systems:

- Various Linux distributions
- Sun Solaris
- IBM AIX
- Windows
- Mac
- OS/400
- z/OS
- HP-UX
- Various BSD UNIX distributions

For commodity x86 servers, Linux distributions (Linux 2.6 kernel or higher) are recommended. While it is expected that most Linux distributions provide a good environment for running Coherence, the following are recommended by Oracle: Oracle Linux (including Oracle Linux with the Unbreakable Enterprise Kernel), Red Hat Enterprise Linux (version 4 or later), and Suse Linux Enterprise (version 10 or later).

Review and follow the instructions in Platform-Specific Deployment Considerations for the operating system on which Coherence is deployed.

> **Note:**
>
> The development and production operating systems may be different. Make sure to regularly test the target production operating system.

Avoid using virtual memory (paging to disk)

In a Coherence-based application, primary data management responsibilities (for example, Dedicated Cache Servers) are hosted by Java-based processes. Modern Java distributions do not work well with virtual memory. In particular, garbage collection (GC) operations may slow down by several orders of magnitude if memory is paged to disk. A properly tuned JVM can perform full GCs in less than a second. However, this may grow to many minutes if the JVM is partially resident on disk. During garbage collection, the node appears unresponsive for an extended period and the choice for the rest of the cluster is to either wait for the node (blocking a portion of application activity for a corresponding amount of time) or to consider the unresponsive node as failed and perform failover processing. Neither of these outcomes are a good option, and it is important to avoid excessive pauses due to garbage collection. JVMs should be configured with a set heap size to ensure that the heap does not deplete the available RAM memory. Also, periodic processes (such as daily backup programs) should be monitored to ensure that memory usage spikes do not cause Coherence JVMs to be paged to disk.

See also: Swapping.

Increase Socket Buffer Sizes

The operating system socket buffers must be large enough to handle the incoming network traffic while your Java application is paused during garbage collection. Most versions of UNIX have a very low default buffer limit, which should be increased to 2MB.

See also: Socket Buffer Sizes.

JVM Recommendations

Select and configure a JVM. During development, developers typically use the latest Oracle HotSpot JVM or a direct derivative such as the Mac OS X JVM. The main issues related to using a different JVM in production are:

- Command line differences, which may expose problems in shell scripts and batch files;
- Logging and monitoring differences, which may mean that tools used to analyze logs and monitor live JVMs during development testing may not be available in production;
- Significant differences in optimal garbage collection configuration and approaches to tuning;
- Differing behaviors in thread scheduling, garbage collection behavior and performance, and the performance of running code.

Make sure that regular testing has occurred on the JVM that is used in production.

Selecting a JVM

For the minimum supported JVM version, refer to System Requirements in *Installing Oracle Coherence*.

Often the choice of JVM is also dictated by other software. For example:

- IBM only supports IBM WebSphere running on IBM JVMs. Most of the time, this is the IBM "Sovereign" or "J9" JVM, but when WebSphere runs on Oracle Solaris/Sparc, IBM builds a JVM using the Oracle JVM source code instead of its own.
- Oracle WebLogic and Oracle Exalogic include specific JVM versions.
- Apple Mac OS X, HP-UX, IBM AIX and other operating systems only have one JVM vendor (Apple, HP, and IBM respectively).
- Certain software libraries and frameworks have minimum Java version requirements because they take advantage of relatively new Java features.

On commodity x86 servers running Linux or Windows, use the Oracle HotSpot JVM. Generally speaking, the recent update versions should be used.

> **Note:**
>
> Test and deploy using the latest supported Oracle HotSpot JVM based on your platform and Coherence version.

Before going to production, a JVM vendor and version should be selected and well tested, and absent any flaws appearing during testing and staging with that JVM, that should be the JVM that is used when going to production. For applications requiring continuous availability, a long-duration application load test (for example, at least two weeks) should be run with that JVM before signing off on it.

Review and follow the instructions in Platform-Specific Deployment Considerations for the JVM on which Coherence is deployed.

Setting the JVM Options

JVM configuration options vary over versions and between vendors, but the following are generally suggested.

- Using the `-server` option results in substantially better performance.
- Using identical heap size values for both `-Xms` and `-Xmx` yields substantially better performance on Oracle HotSpot JVM and "fail fast" memory allocation.
- Using Garbage First Garbage Collector (G1GC) results in better garbage collection performance: `-XX:+UseG1GC`.
- Monitor garbage collection– especially when using large heaps: `-verbose:gc`, `-XX:+PrintGCDetails`, `-XX:+PrintGCTimeStamps`, `-XX:+PrintHeapAtGC`, `-XX:+PrintTenuringDistribution`, `-XX:+PrintGCApplicationStoppedTime` `-XX:+PrintGCApplicationConcurrentTime`
- JVMs that experience an `OutOfMemoryError` can be left in an indeterministic state which can have adverse effects on a cluster. Configure JVMs to exit upon encountering an `OutOfMemoryError` instead of allowing the JVM to attempt recovery: On Linux, `-XX:OnOutOfMemoryError="kill -9 %p"`; on Windows, `-XX:OnOutOfMemoryError="taskkill /F /PID %p"`.
- Capture a heap dump if the JVM experiences an out of memory error: `-XX:+HeapDumpOnOutOfMemoryError`.

Plan to Test Mixed JVM Environments

Coherence is pure Java software and can run in clusters composed of any combination of JVM vendors and versions and Oracle tests such configurations.

Note that it is *possible* for different JVMs to have slightly different serialization formats for Java objects, meaning that it is possible for an incompatibility to exist when objects are serialized by one JVM, passed over the wire, and a different JVM (vendor, version, or both) attempts to deserialize it. Fortunately, the Java serialization format has been very stable for several years, so this type of issue is extremely unlikely. However, it is highly recommended to test mixed configurations for consistent serialization before deploying in a production environment.

See also:

- Deploying to Oracle HotSpot JVMs
- Deploying to IBM JVMs

Oracle Exalogic Elastic Cloud Recommendations

Configure Coherence accordingly when using Oracle Exalogic Elastic Cloud software. Oracle Exalogic and the Oracle Exalogic Elastic Cloud software provide a foundation for extreme performance, reliability, and scalability. Coherence has been optimized to take advantage of this foundation especially in its use of Oracle Exabus technology. Exabus consists of unique hardware, software, firmware, device drivers, and management tools and is built on Oracle's Quad Data Rate (QDR) InfiniBand technology. Exabus forms the high-speed communication (I/O) fabric that ties all Oracle Exalogic system components together.
Oracle Coherence includes the following optimizations:

- Transport optimizations

  Oracle Coherence uses the Oracle Exabus messaging API for message transport. The API is optimized on Exalogic to take advantage of InfiniBand. The API is part of the Oracle Exalogic Elastic Cloud software and is only available on Oracle Exalogic systems.

  In particular, Oracle Coherence uses the InfiniBand Message Bus (IMB) provider. IMB uses a native InfiniBand protocol that supports zero message copy, kernel bypass, predictive notifications, and custom off-heap buffers. The result is decreased host processor load, increased message throughput, decreased interrupts, and decreased garbage collection pauses.

  The default Coherence setup on Oracle Exalogic uses IMB for service communication (transferring data) and for cluster communication. Both defaults can be changed and additional protocols are supported. See Changing the Reliable Transport Protocol.

- Elastic data optimizations

  The Elastic Data feature is used to store backing map and backup data seamlessly across RAM memory and devices such as Solid State Disks (SSD). The feature enables near memory speed while storing and reading data from SSDs. The feature includes dynamic tuning to ensure the most efficient use of SSD memory on Exalogic systems. See Using the Elastic Data Feature to Store Data in *Developing Applications with Oracle Coherence*.

- Coherence*Web optimizations

  Coherence*Web naturally benefits on Exalogic systems because of the increased performance of the network between WebLogic Servers and Coherence servers. Enhancements also include less network usage and better performance by enabling optimized session state management when locking is disabled (`coherence.session.optimizeModifiedSessions=true`). See Coherence*Web Context Parameters in *Administering HTTP Session Management with Oracle Coherence*Web*.

Consider Using Fewer JVMs with Larger Heaps

The IMB protocol requires more CPU usage (especially at lower loads) to achieve lower latencies. If you are using many JVMs, JVMs with smaller heaps (under 12GB), or many JVMs and smaller heaps, then consider consolidating the JVMs as much as possible. Large heap sizes up to 20GB are common and larger heaps can be used depending on the application and its tolerance to garbage collection. See JVM Tuning.

Changing the Reliable Transport Protocol

On Oracle Exalogic, Coherence automatically selects the best reliable transport available for the environment. The default Coherence setup uses the InfiniBand Message Bus (IMB) for service communication (transferring data) and for cluster communication unless SSL is enabled, in which case SDMBS is used. You can use a different transport protocol and check for improved performance. However, you should only consider changing the protocol after following the previous recommendations in this section.

> **Note:**
>
> The only time the default transport protocol may need to be explicitly set is in a Solaris Super Cluster environment. The recommended transport protocol is SDMB or (if supported by the environment) IMB.

The following transport protocols are available on Exalogic:

- `datagram` – Specifies the use of UDP.
- `tmb` – Specifies the TCP Message Bus (TMB) protocol. TMB provides support for TCP/IP.
- `tmbs` – TCP/IP message bus protocol with SSL support. TMBS requires the use of an SSL socket provider. See socket-provider in *Developing Applications with Oracle Coherence*.
- `sdmb` – Specifies the Sockets Direct Protocol Message Bus (SDMB). The Sockets Direct Protocol (SDP) provides support for stream connections over the InfiniBand fabric. SDP allows existing socket-based implementations to transparently use InfiniBand.

  > **Note:**
  >
  > When running with JDK11 or higher, specifying `sdmb` protocol will result in a log message stating SDP classes are unavailable, and the client or cache server will not start.

- `sdmbs` – SDP message bus with SSL support. SDMBS requires the use of an SSL socket provider.

> **Note:**
>
> When running with JDK11 or higher, specifying `sdmbs` protocol will result in a log message stating SDP classes are unavailable, and the client or cache server will not start.

- `imb` – InfiniBand message bus (IMB). IMB is automatically used on Exalogic systems as long as TCMP has not been configured with SSL.

> **Note:**
>
> `imb` protocol is removed as of release 12.2.1.4.0. If `imb` protocol is specified, it is mapped to the `tmb` protocol.

To configure a reliable transport for all cluster (unicast) communication, edit the operational override file and within the `<unicast-listener>` element add a `<reliable-transport>` element that is set to a protocol:

> **Note:**
>
> By default, all services use the configured protocol and share a single transport instance. In general, a shared transport instance uses less resources than a service-specific transport instance.

```
<?xml version="1.0"?>
<coherence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-operational-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-operational-
config
   coherence-operational-config.xsd">
   <cluster-config>
      <unicast-listener>
         <reliable-transport
            system-property="coherence.transport.reliable">imb
         </reliable-transport>
      </unicast-listener>
   </cluster-config>
</coherence>
```

The `coherence.transport.reliable` system property also configures the reliable transport. For example:

```
-Dcoherence.transport.reliable=imb
```

To configure reliable transport for a service, edit the cache configuration file and within a scheme definition add a `<reliable-transport>` element that is set to a protocol. The following example demonstrates setting the reliable transport for a partitioned cache service instance called `ExampleService`:

> **Note:**
>
> Specifying a reliable transport for a service results in the use of a service-specific transport instance rather then the shared transport instance that is defined by the `<unicast-listener>` element. A service-specific transport instance can result in higher performance but at the cost of increased resource consumption and should be used sparingly for select, high priority services. In general, a shared transport instance uses less resource consumption than service-specific transport instances.

```
<?xml version="1.0"?>
<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-cache-config
   coherence-cache-config.xsd">

   <caching-scheme-mapping>
      <cache-mapping>
         <cache-name>example</cache-name>
         <scheme-name>distributed</scheme-name>
      </cache-mapping>
   </caching-scheme-mapping>

   <caching-schemes>
      <distributed-scheme>
         <scheme-name>distributed</scheme-name>
         <service-name>ExampleService</service-name>
         <reliable-transport>imb</reliable-transport>
         <backing-map-scheme>
           <local-scheme/>
         </backing-map-scheme>
         <autostart>true</autostart>
      </distributed-scheme>
   </caching-schemes>
</cache-config>
```

Each service type also has a system property that sets the reliable transport, respectively. The system property sets the reliable transport for all instances of a service type. The system properties are:

```
coherence.distributed.transport.reliable
coherence.replicated.transport.reliable
coherence.optimistic.transport.reliable
coherence.invocation.transport.reliable
coherence.proxy.transport.reliable
```

Security Recommendations

Ensure security has been configured properly.

Ensure Security Privileges

The minimum set of privileges required for Coherence to function are specified in the `security.policy` file which is included as part of the Coherence installation. This file can be found in `coherence/lib/security/security.policy`. If using the Java Security Manager, these privileges must be granted in order for Coherence to function properly.

Plan for SSL Requirements

Coherence-based applications may chose to implement varying levels of security as required, including SSL-based security between cluster members and between Coherence*Extend clients and the cluster. If SSL is a requirement, ensure that all servers have a digital certificate that has been verified and signed by a trusted certificate authority and that the digital certificate is imported into the servers' key store and trust store as required. Coherence*Extend clients must include a trust key store that contains the certificate authority's digital certificate that was used to sign the proxy's digital certificate. See Using SSL to Secure Communication in *Securing Oracle Coherence* .

Persistence Recommendations

Follow persistence best practices.

Plan for SAN/NFS Persistence Storage

Persisting caches to a remote or shared disk may require additional planning and configuration of the underlying persistence store. The persistence store currently used by Coherence is Oracle Berkeley DB (BDB) Java Edition (JE). General recommendations are provided in the FAQ entry: Can Berkeley DB Java Edition use a NFS, SAN, or other remote/shared/network filesystem for an environment?.

> **Note:**
>
> The current persistence store implementation may change in the future.

The documentation lists a number of issues and recommendations related to the use of remote file systems. You should evaluate those recommendations to help avoid database corruption. In particular, one of the issues relates to having multiple clients (cluster members in the case of Coherence) pointing at the same remote filesystem (BDB JE Environment). The documentation indicates that this should never be done due to issues with faulty remote implementations of `flock()`. Coherence persistence maintains a distinct BDB JE Environment for each persisted partition and helps addresses the `flock` issue by using Coherence clustering to enforce that each of the BDB JE Environments is only ever accessed by single cluster members at a given time, that is, by the partition owner. The caveat is that if a cluster encounters a split brain condition, then there is temporarily multiple clusters and multiple logical owners for each partition trying to access the same BDB JE Environment. When using a remote file system, the best practice is to either configure Coherence to prevent split brain by using a Cluster Quorum, ensure that your remote file system properly supports `flock()`, or point each cluster storage member at a different remote directory.

Application Instrumentation Recommendations

Some Java-based management and monitoring solutions use instrumentation (for example, bytecode-manipulation and `ClassLoader` substitution). Oracle has observed issues with such solutions in the past. Use application instrumentation solutions cautiously even though there are no current issues reported with the major vendors.

Coherence Modes and Editions

Verify that Coherence is configured to run in production mode and is using the correct edition settings.

Select the Production Mode

Coherence may be configured to operate in either evaluation, development, or production mode. These modes do not limit access to features, but instead alter some default configuration settings. For instance, development mode allows for faster cluster startup to ease the development process.

The development mode is used for all pre-production activities, such as development and testing. This is an important safety feature because development nodes are restricted from joining with production nodes. Development mode is the default mode. Production mode must be explicitly specified when using Coherence in a production environment. To change the mode to production mode, edit the `tangosol-coherence.xml` (located in `coherence.jar`) and enter `prod` as the value for the `<license-mode>` element. For example:

```
...
<license-config>
   ...
   <license-mode system-property="coherence.mode">prod</license-mode>
</license-config>
...
```

The `coherence.mode` system property is used to specify the license mode instead of using the operational deployment descriptor. For example:

```
-Dcoherence.mode=prod
```

In addition to preventing mixed mode clustering, the `license-mode` also dictates the operational override file to use. When in `eval` mode the `tangosol-coherence-override-eval.xml` file is used; when in `dev` mode the `tangosol-coherence-override-dev.xml` file is used; whereas, the `tangosol-coherence-override-prod.xml` file is used when the `prod` mode is specified. A `tangosol-coherence-override.xml` file (if it is included in the classpath before the `coherence.jar` file) is used no matter which mode is selected and overrides any mode-specific override files.

Select the Edition

> **Note:**
>
> The edition switches no longer enforce license restrictions. Do not change the default setting (`GE`).

All nodes within a cluster must use the same license edition and mode. The default edition is grid edition (GE). Be sure to obtain enough licenses for the all the cluster members in the production environment. The servers hardware configuration (number or type of processor sockets, processor packages, or CPU cores) may be verified using `ProcessorInfo` utility included with Coherence. For example:

```
java -cp coherence.jar com.tangosol.license.ProcessorInfo
```

If the result of the `ProcessorInfo` program differs from the licensed configuration, send the program's output and the actual configuration as a support issue.

> **Note:**
>
> Clusters that run different editions may connect by using Coherence*Extend as a Data Client.

Ensuring that RTC Nodes do Not Use Coherence TCMP

Real-Time client nodes can connect to clusters using either Coherence TCMP or Coherence*Extend. If the intention is to use extend clients, disable TCMP on the client to ensure that it only connects to a cluster using Coherence*Extend. Otherwise, The client may become a member of the cluster. See Disabling TCMP Communication in *Developing Remote Clients for Oracle Coherence*.

Coherence Operational Configuration Recommendations

Verify that the operational configuration file is setup correctly.

Operational configuration relates to cluster-level configuration that is defined in the `tangosol-coherence.xml` file and includes such items as:

- Cluster and cluster member settings
- Network settings
- Management settings
- Security settings

Coherence Operational aspects are typically configured by using a `tangosol-coherence-override.xml` file. See Specifying an Operational Configuration File in *Developing Applications with Oracle Coherence*.

The contents of this file often differs between development and production. It is recommended that these variants be maintained independently due to the significant differences between these environments. The production operational configuration file should be maintained by systems administrators who are far more familiar with the workings of the production systems.

All cluster nodes should use the same operational configuration override file and any node-specific values should be specified by using system properties. See System Property Overrides in *Developing Applications with Oracle Coherence*. A centralized configuration file may be maintained and accessed by specifying a URL as the value of the `coherence.override` system property on each cluster node. For example:

```
-Dcoherence.override=/net/mylocation/tangosol-coherence-override.xml
```

The override file need only contain the operational elements that are being changed. In addition, always include the `id` and `system-property` attributes if they are defined for an element.

Coherence Cache Configuration Recommendations

Verify that the cache configuration file is setup correctly.

Cache configuration relates to cache-level configuration and includes such things as:

- Cache topology (<distributed-scheme>, <near-scheme>, and so on)
- Cache capacities (<high-units>)
- Cache redundancy level (<backup-count>)

Coherence cache configuration aspects are typically configured by using a coherence-cache-config.xml file. See Specifying a Cache Configuration File in *Developing Applications with Oracle Coherence*.

The default coherence-cache-config.xml file included within coherence.jar is intended only as an example and is not suitable for production use. Always use a cache configuration file with definitions that are specific to the application.

All cluster nodes should use the same cache configuration descriptor if possible. A centralized configuration file may be maintained and accessed by specifying a URL as the value the coherence.cacheconfig system property on each cluster node. For example:

```
-Dcoherence.cacheconfig=/net/mylocation/coherence-cache-config.xml
```

Caches can be categorized as either partial or complete. In the former case, the application does not rely on having the entire data set in memory (even if it expects that to be the case). Most caches that use cache loaders or that use a side cache pattern are partial caches. Complete caches require the entire data set to be in cache for the application to work correctly (most commonly because the application is issuing non-primary-key queries against the cache). Caches that are partial should always have a size limit based on the allocated JVM heap size. The limits protect an application from OutOfMemoryExceptions errors. Set the limits even if the cache is not expected to be fully loaded to protect against changing expectations. See JVM Tuning. Conversely, if a size limit is set for a complete cache, it may cause incorrect results.

It is important to note that when multiple cache schemes are defined for the same cache service name, the first to be loaded dictates the service level parameters. Specifically the <partition-count>, <backup-count>, and <thread-count> subelements of <distributed-scheme> are shared by all caches of the same service. It is recommended that a single service be defined and inherited by the various cache-schemes. If you want different values for these items on a cache by cache basis then multiple services may be configured.

For partitioned caches, Coherence evenly distributes the storage responsibilities to all cache servers, regardless of their cache configuration or heap size. For this reason, it is recommended that all cache server processes be configured with the same heap size. For computers with additional resources multiple cache servers may be used to effectively make use of the computer's resources.

To ensure even storage responsibility across a partitioned cache the <partition-count> subelement of the <distributed-scheme> element, should be set to a prime number which is at least the square of the number of expected cache servers.

A clustered service can perform all tasks on the service thread, a caller's thread if possible, and any number of daemon (worker) threads managed by a dynamic thread pool. The dynamic thread pool is automatically enabled for these services. You can use <thread-count-min> and <thread-count-max> to control the minimum and maximum number of threads in a dynamic thread pool. By default, the value of <thread-count-min> is 1 and <thead-count-max> is Integer.MAX_VALUE. The dynamic thread pool is started with the number of threads specified by <thread-count-min>.

For caches which are backed by a cache store, Oracle recommends configuring the parent service with a thread pool of `<thread-count-min>` greater than `1` as requests to the cache store may block on I/O. Such thread pools are also recommended for caches that perform CPU-intensive operations on the cache server (queries, aggregations, some entry processors, and so on). For non-CacheStore-based caches, more threads are unlikely to improve performance. Therefore, you may leave the `<thread-count-min>` at its default value of `1`.

Unless explicitly specified, all cluster nodes are storage enabled, that is, they act as cache servers. It is important to control which nodes in your production environment are storage enabled and storage disabled. The `coherence.distributed.localstorage` system property may be used to control storage, setting it to either `true` or `false`. Generally, only dedicated cache servers (including proxy servers) should have storage enabled. All other cluster nodes should be configured as storage disabled. This is especially important for short lived processes which may join the cluster to perform some work and then exit the cluster. Having these nodes as storage enabled introduces unneeded re-partitioning.

Large Cluster Configuration Recommendations

Configure Coherence accordingly when deploying a large cluster.

- Distributed caches on large clusters of more than 16 cache servers require more partitions to ensure optimal performance. The default partition count is 257 and should be increased relative to the number of cache servers in the cluster and the amount of data being stored in each partition. See Changing the Number of Partitions in *Developing Applications with Oracle Coherence*.
- The maximum packet size on large clusters of more than 400 cluster members must be increased to ensure better performance. The default of 1468 should be increased relative to the size of the cluster, that is, a 600 node cluster would need the maximum packet size increased by 50%. A simple formula is to allow four bytes per node, that is, `maximum_packet_size >= maximum_cluster_size * 4B`. The maximum packet size is configured as part of the coherence operational configuration file. See Adjusting the Maximum Size of a Packet in *Developing Applications with Oracle Coherence*.
- Multicast communication, if supported by the network, can be used instead of point-to-point communication for cluster discovery. This is an ease-of-use recommendation and is not a requirement for large clusters. Multicast is enabled in an operational configuration file. See Configuring Multicast Communication in *Developing Applications with Oracle Coherence*.

Death Detection Recommendations

Test scenarios that include node failure and configure Coherence accordingly. The Coherence death detection algorithms are based on sustained loss of connectivity between two or more cluster nodes.
When a node identifies that it has lost connectivity with any other node, it consults with other cluster nodes to determine what action should be taken. In attempting to consult with others, the node may find that it cannot communicate with any other nodes and assumes that it has been disconnected from the cluster. Such a condition could be triggered by physically unplugging a node's network adapter. In such an event, the isolated node restarts its clustered services and attempts to rejoin the cluster.

If connectivity with other cluster nodes remains unavailable, the node may (depending on well known address configuration) form a new isolated cluster, or continue searching for the larger cluster. In either case, the previously isolated cluster nodes rejoins the running cluster when connectivity is restored. As part of rejoining the cluster, the nodes former cluster state is discarded, including any cache data it may have held, as the remainder of the cluster has taken on ownership of that data (restoring from backups).

It is obviously not possible for a node to identify the state of other nodes without connectivity. To a single node, local network adapter failure and network wide switch failure looks identical and are handled in the same way, as described above. The important difference is that for a switch failure all nodes are attempting to re-join the cluster, which is the equivalent of a full cluster restart, and all prior state and data is dropped.

Dropping all data is not desirable and, to avoid this as part of a sustained switch failure, you must take additional precautions. Options include:

- Increase detection intervals: The cluster relies on a deterministic process-level death detection using the TcpRing component and hardware death detection using the IpMonitor component. Process-level detection is performed within milliseconds and network or machine failures are detected within 15 seconds by default. Increasing these value allows the cluster to wait longer for connectivity to return. Death detection is enabled by default and is configured within the `<tcp-ring-listener>` element. See Configuring Death Detection in *Developing Applications with Oracle Coherence*.
- Persist data to external storage: By using a Read Write Backing Map, the cluster persists data to external storage, and can retrieve it after a cluster restart. So long as write-behind is disabled (the `<write-delay>` subelement of `<read-write-backing-map-scheme>`) no data would be lost if a switch fails. The downside here is that synchronously writing through to external storage increases the latency of cache update operations, and the external storage may become a bottleneck.
- Decide on a cluster quorum: The cluster quorum policy mandates the minimum number of cluster members that must remain in the cluster when the cluster service is terminating suspect members. During intermittent network outages, a high number of cluster members may be removed from the cluster. Using a cluster quorum, a certain number of members are maintained during the outage and are available when the network recovers. See Using the Cluster Quorum in *Developing Applications with Oracle Coherence*.

  > **Note:**
  >
  > To ensure that Windows does not disable a network adapter when it is disconnected, add the following Windows registry `DWORD` and set it to `1:HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\DisableDHCPMediaSense`. See How to disable the Media Sensing feature for TCP/IP in Windows. This setting also affects static IPs despite the name.

- Add network level fault tolerance: Adding a redundant layer to the cluster's network infrastructure allows for individual pieces of networking equipment to fail without disrupting connectivity. This is commonly achieved by using at least two network adapters per computer, and having each adapter connected to a separate switch. This is not a feature of Coherence but rather of the underlying operating system or network driver. The only change to Coherence is that it should be configured to bind to the virtual rather then physical network adapter. This form of network redundancy goes by different names depending on the operating system: Linux bonding, Solaris trunking and Windows teaming.

Part II
Advanced Administration

Persistence and federation are advanced features that are used for solutions that require continuous availability and redundancy. In many cases, these features can be enabled without any changes to an application.

Part II contains the following chapters:

- Persisting Caches
- Federating Caches Across Clusters

6
Persisting Caches

The Coherence persistence feature is used to save a cache to disk and ensures that cache data can always be recovered.
This chapter includes the following sections:

Overview of Persistence

Coherence persistence is a set of tools and technologies that manage the persistence and recovery of Coherence distributed caches. Cached data is persisted so that it can be quickly recovered after a catastrophic failure or after a cluster restart due to planned maintenance. Persistence and federated caching can be used together as required. See Federating Caches Across Clusters .
This section includes the following topics:

Persistence Modes

Persistence can operate in two modes:

- On-Demand persistence mode – a cache service is manually persisted and recovered upon request using the persistence coordinator. The persistence coordinator is exposed as an MBean interface that provides operations for creating, archiving, and recovering snapshots of a cache service.
- Active persistence mode – In this mode, cache contents are automatically persisted on all mutations and are automatically recovered on cluster/service startup. The persistence coordinator can still be used in active persistence mode to perform on-demand snapshots.

Disk-Based Persistence Storage

Persistence uses a database for the persistence store. The database is used to store the backing map partitions of a partitioned service. The locations of the database files can be stored on the local disk of each cache server or on a shared disk: Storage Area Network (SAN) or Network File System (NFS). See Plan for SAN/NFS Persistence Storage.

> **Note:**
>
> Database files should never be manually edited. Editing the database files can lead to persistence errors.

The local disk option allows each cluster member to access persisted data for the service partitions that it owns. Persistence is coordinated across all storage member using a list of cache server host addresses. The address list ensures that all persisted partitions are discovered during recovery. Local disk storage provides a high throughput and low latency storage mechanism; however, a partition service must still rely on in-memory backup (`backup-count` value greater than zero) to remain machine safe.

The shared disk option, together with active persistence mode, allows each cluster member to access persisted data for all service partitions. An advantage to using a shared disk is that partitioned services do not require in-memory backup (`backup-count` value can be equal to zero) to remain machine-safe; because, all storage-enabled members can recover partitions from the shared storage. Disabling in-memory backup increases the cache capacity of the cluster at the cost of higher latency recovery during node failure. In general, the use of a shared disk can potentially affect throughput and latencies and should be tested and monitored accordingly.

> **Note:**
>
> The service statusHA statistic shows an `ENDAGERED` status when the backup count is set to zero even if persistence is being used to replace in-memory backup.

Both the local disk and shared disk approach can rely on a quorum policy that controls how many cluster members must be present to perform persistence operations and before recovery can begin. Quorum policies allow time for a cluster to start before data recovery begins.

Persistence Configuration

Persistence is declaratively configured using Coherence configuration files and requires no changes to application code. An operational override file is used to configure the underlying persistence implementation if the default settings are not acceptable. A cache configuration file is used to set persistence properties on a distributed cache.

Management and Monitoring

Persistence can be monitored and managed using MBean attributes and operations. Persistence operations such as creating and archiving snapshots are performed using the `PersistenceManagerMBean` MBean. Persistence attributes are included as part of the attributes of a service and can be viewed using the `ServiceMBean` MBean.

Persistence attributes and statistics are aggregated in the `persistence` and `persistence-details` reports. Persistence statistics are also aggregated in the VisualVM plug-in. Both tools can help troubleshoot possible resource and performance issues.

Persistence Dependencies

Persistence is only available for distributed caches and requires the use of a centralized partition assignment strategy.

> **Note:**
>
> Transactional caches do not support persistence.

Distributed caches use a centralized partitioned assignment strategy by default. Although uncommon, it is possible that an autonomous or a custom partition assignment strategy is being used. Check the `StrategyName` attribute on the `PartitionAssignment` MBean to verify the strategy that is currently configured for a distributed cache. See Changing the Partition Distribution Strategy in *Developing Applications with Oracle Coherence*.

Persisting Caches on Demand

Caches can be persisted to disk at any point in time and recovered as required.
To persist caches on demand:

1. Use the persistence coordinator to create, recover, and remove snapshots. See Using Snapshots to Persist a Cache Service.
2. Optionally, change the location where persistence files are written to disk. See Changing the Pre-Defined Persistence Directory.
3. Optionally, configure the number of storage members that are required to perform recovery. See Using Quorum for Persistence Recovery.

Actively Persisting Caches

Caches can be automatically persisted to disk and automatically recovered when a cluster is restarted.
To actively persist caches:

1. Enable active persistence. See Enabling Active Persistence Mode.
2. Optionally, change the location where persistence files are written to disk. See Changing the Pre-Defined Persistence Directory.
3. Optionally, change how a service responds to possible failures during active persistence. See Changing the Active Persistence Failure Response.
4. Optionally, configure the number of storage members that are required to perform recovery. See Using Quorum for Persistence Recovery.

Using Snapshots to Persist a Cache Service

Snapshots are a backup of the contents of a cache service that must be manually managed using the `PersistenceManagerMBean` MBean.

The MBean includes asynchronous operations to create, recover, and remove snapshots. When a snapshot is recovered, the entire service is automatically restored to the state of the snapshot. To use the MBean, JMX must be enabled on the cluster. See Using JMX to Manage Oracle Coherence in *Managing Oracle Coherence*.

> **Note:**
>
> The instructions in this section were created using the VisualVM-MBeans plug-in for the Coherence VisualVM tool. The Coherence VisualVM Plug-in can also be used to perform snapshot operations.

This section includes the following topics:

- Create a Snapshot
- Recover a Snapshot
- Remove a Snapshot

Create a Snapshot

Creating snapshots writes the contents of a cache service to the snapshot directory that is specified within the persistence environment definition in the operational override configuration file. A snapshot can be created either on a running service (a

service that is accepting and processing requests) or on a suspended service. The former provides consistency at a partition level while the latter provides global consistency.

By default, the snapshot operation assumes partition level consistency. To achieve global consistency, the service must be explicitly suspended which causes any requests into the service to be blocked.

Snapshot with Partition Consistency

To create a snapshot with partition consistency:

1. From the list of MBeans, select and expand the Persistence node.
2. Expand a service for which you want to create a snapshot and select **PersistenceCoordinator**.
3. From the Operations tab, enter a name for the snapshot in the field for the createSnapshot operation.
4. Click **createSnapshot**.

Snapshot with Global Consistency

To create a snapshot with global consistency:

1. From the list of MBeans, select the ClusterMBean node.
2. From the Operations tab, enter the name of the service that you want to suspend in the field for the suspendService operation.
3. Click **suspendService**.
4. From the list of MBeans, select and expand the Persistence node.
5. Expand the service (now suspended) for which you want to create a snapshot and select **PersistenceCoordinator**.
6. From the Operations tab, enter a name for the snapshot in the field for the createSnapshot operation.
7. Click **createSnapshot**.

> **Note:**
>
> Applications can be notified when the operation completes by subscribing to the snapshot JMX notifications. See Subscribing to Persistence JMX Notifications.

8. From the list of MBeans, select the ClusterMBean node.
9. From the Operations tab, enter the name of the service that you want to resume in the field for the resumeService operation.
10. Click **resumeService**.

Recover a Snapshot

Recovering snapshots restores the contents of a cache service from a snapshot.

> **Note:**
>
> A Coherence service recovered from a persistent snapshot is not propagated to federated clusters. The data on the originating cluster is recovered but the cache data on the destination cluster remains unaffected and may still contain the data that was present prior to the recovery. To propagate the snapshot data, a federation `ReplicateAll` operation is required after the snapshot recovery is completed. The `ReplicateAll` operation is available on the `FederationManagerMBean` MBean. See FederationManagerMBean Operations in *Managing Oracle Coherence*.

To recover a snapshot:

1. From the list of MBeans, select and expand the Persistence node.
2. Expand a service for which you want to recover a snapshot and select **PersistenceCoordinator**.
3. From the Operations tab, enter the name of a snapshot in the field for the recoverSnapshot operation.
4. Click **recoverSnapshot**.

   After the operation has returned, check the `OperationStatus` or `Idle` attributes on the persistence coordinator to determine when the operation has completed. Applications can be notified when the operation completes by subscribing to the snapshot JMX notifications.

Remove a Snapshot

Removing a snapshot deletes the snapshot from the snapshot directory. The cache service remains unchanged.

To remove a snapshot:

1. From the list of MBeans, select and expand the Persistence node.
2. Expand a service for which you want to remove a snapshot and select **PersistenceCoordinator**.
3. From the Operations tab, enter the name of a snapshot in the field for the removeSnapshot operation.
4. Click **removeSnapshot**.

Archiving Snapshots

Snapshots can be archived to a central location and then later retrieved and restored. Archiving snapshots requires defining the directory where archives are stored and configuring cache services to use an archive directory. Archiving operations are performed using the `PersistenceManagerMBean` MBean. An archive is slower to create than snapshots but, unlike snapshots, the archive is portable.
This section includes the following topics:

- Defining a Snapshot Archive Directory

Defining a Snapshot Archive Directory

The directory where snapshots are archived is defined in the operational override file using a directory snapshot archiver definition. Multiple definitions can be created as required.

> **Note:**
>
> The archive directory location and name must be the same across all members. The archive directory location must be a shared directory and must be accessible to all members.

To define a snapshot archiver directory, include the `<directory-archiver>` element within the `<snapshot-archivers>` element. Use the `<archiver-directory>` element to enter the directory where snapshot archives are stored. Use the `id` attribute to provide a unique name for the definition. For example:

```
<snapshot-archivers>
   <directory-archiver id="archiver1">
      <archive-directory>/mydirectory</archive-directory>
   </directory-archiver>
</snapshot-archivers>
```

Specifying a Directory Snapshot Archiver

To specify a directory snapshot archiver, edit the persistence definition within a distributed scheme and include the name of a directory snapshot archiver that is defined in the operational override configuration file. For example:

```
<distributed-scheme>
   <scheme-name>distributed</scheme-name>
   <service-name>Service1</service-name>
   <backing-map-scheme>
      <local-scheme/>
   </backing-map-scheme>
   <persistence>
      <archiver>archiver1</archiver>
   </persistence>
   <autostart>true</autostart>
</distributed-scheme>
```

Performing Snapshot Archiving Operations

Snapshot archiving is manually managed using the `PersistenceManagerMBean` MBean. The MBean includes asynchronous operations to archive and retrieve snapshot archives and also includes operations to list and remove archives.

This section includes the following topics:

Archiving a Snapshot

To archive a snapshot:

1. From the list of MBeans, select and expand the Persistence node.
2. Expand a service for which you want to archive a snapshot and select **PersistenceCoordinator**.
3. From the Operations tab, enter a name for the archive in the field for the archiveSnapshot operation.
4. Click **archiveSnapshot**. The snapshot is archived to the location that is specified in the directory archiver definition defined in the operational override configuration file.

   Check the `OperationStatus` on the persistence coordinator to determine when the operation has completed.

Retrieving Archived Snapshots

To retrieve an archived snapshot:

1. From the list of MBeans, select and expand the Persistence node.
2. Expand a service for which you want to retrieve an archived snapshot and select **PersistenceCoordinator**.
3. From the Operations tab, enter the name of an archived snapshot in the field for the retrieveArchivedSnapshot operation.
4. Click **retrieveArchivedSnapshot**. The archived snapshot is copied from the directory archiver location to the snapshot directory and is available to be recovered to the service backing map. See Recover a Snapshot.

Removing Archived Snapshots

To remove an archived snapshot:

1. From the list of MBeans, select and expand the Persistence node.
2. Expand a service for which you want to purge an archived snapshot and select **PersistenceCoordinator**.
3. From the Operations tab, enter the name of an archived snapshot in the field for the removeArchivedSnapshot operation.
4. Click **removeArchivedSnapshot**. The archived snapshot is removed from the archive directory.

Listing Archived Snapshots

To get a list of the current archived snapshots:

1. From the list of MBeans, select and expand the Persistence node.
2. Expand a service for which you want to list archived snapshots and select **PersistenceCoordinator**.
3. From the Operations tab, click the `listArchivedSnapshots` operation. A list of archived snapshots is returned.

Listing Archived Snapshot Stores

To list the individual stores, or parts of and archived snapshot:

1. From the list of MBeans, select and expand the Persistence node.
2. Expand a service for which you want to list archived snapshot stores and select **PersistenceCoordinator**.
3. From the Operations tab, enter the name of an archived snapshot in the field for the listArchivedSnapshotStores operation.
4. Click **listArchivedSnapshotStores**. A list of stores for the archived snapshots is returned.

Creating a Custom Snapshot Archiver

Custom snapshot archiver implementations can be created as required to store archives using an alternative technique than the default directory snapshot archiver implementation. For example, you may want to persist archives to an external database, use a web service to store archives to a storage area network, or store archives in a content repository.

This section includes the following topics:

- Create a Custom Snapshot Archiver Implementation
- Create a Custom Snapshot Archiver Definition
- Specifying a Custom Snapshot Archiver

Create a Custom Snapshot Archiver Implementation

To create a custom snapshot archiver implementation, create a class that extends the `AbstractSnapshotArchiver` class.

Create a Custom Snapshot Archiver Definition

To create a custom snapshot archiver definition, include the `<custom-archiver>` element within the `<snapshot-archivers>` element and use the `id` attribute to provide a unique name for the definition. Add the `<class-name>` element within the `<custom-archiver>` element that contains the fully qualified name of the implementation class. The following example creates a definition for a custom implementation called `MyCustomArchiver`:

```
<snapshot-archivers>
   <custom-archiver id="custom1">
      <class-name>package.MyCustomArchiver</class-name>
   </custom-archiver>
</snapshot-archivers>
```

Use the `<class-factory-name>` element if your implementation uses a factory class that is responsible for creating archiver instances. Use the `<method-name>` element to specify the

static factory method on the factory class that performs object instantiation. The following example gets a snapshot archiver instance using the `getArchiver` method on the `MyArchiverFactory` class.

```
<snapshot-archivers>
   <custom-archiver id="custom1">
      <class-factory-name>package.MyArchiverFactory</class-factory-name>
      <method-name>getArchiver</method-name>
   </custom-archiver>
</snapshot-archivers>
```

Any initialization parameters that are required for an implementation can be specified using the `<init-params>` element. The following example sets the `UserName` parameter to `Admin`.

```
<snapshot-archivers>
   <custom-archiver id="custom1">
      <class-name>package.MyCustomArchiver</class-name>
      <init-params>
         <init-param>
            <param-name>UserName</param-name>
            <param-value>Admin</param-value>
         </init-param>
      </init-params>
   </custom-archiver>
</snapshot-archivers>
```

Specifying a Custom Snapshot Archiver

To specify a custom snapshot archiver, edit the persistence definition within a distributed scheme and include the name of a custom snapshot archiver that is defined in the operational override configuration file. For example:

```
<distributed-scheme>
   <scheme-name>distributed</scheme-name>
   <service-name>Service1</service-name>
   <backing-map-scheme>
      <local-scheme/>
   </backing-map-scheme>
   <persistence>
      <archiver>custom1</archiver>
   </persistence>
   <autostart>true</autostart>
</distributed-scheme>
```

Using Active Persistence Mode

You can enable and configure active persistence mode to have the contents of a cache automatically persisted and recovered.
This section includes the following topics:

- Enabling Active Persistence Mode
- Changing the Active Persistence Failure Response
- Changing the Partition Count When Using Active Persistence

Enabling Active Persistence Mode

Active persistence can be enabled for all services or for specific services. To enable active persistence for all services, set the `coherence.distributed.persistence.mode` system property to `active`. For example:

```
-Dcoherence.distributed.persistence.mode=active
```

The default value if no value is specified is `on-demand`, which enables on-demand persistence. The persistence coordinator can still be used in active persistence mode to take snapshots of a cache.

To enable active persistence for a specific service, modify a distributed scheme definition and include the `<environment>` element within the `<persistence>` element. Set the value of the `<environment>` element to `default-active`. For example:

```
<distributed-scheme>
   <scheme-name>distributed</scheme-name>
   <service-name>Service1</service-name>
   <backing-map-scheme>
      <local-scheme/>
   </backing-map-scheme>
   <persistence>
      <environment>default-active</environment>
   </persistence>
   <autostart>true</autostart>
</distributed-scheme>
```

The default value if no value is specified is `default-on-demand`, which enables on-demand persistence for the service.

Changing the Active Persistence Failure Response

You can change the way a partitioned cache service responds to possible persistence failures during active persistence operations. The default response is to immediately stop the service. This behavior is ideal if persistence is critical for the service (for example, a cache depends on persistence for data backup). However, if persistence is not critical, you can chose to let the service continue servicing requests.

To change the active persistence failure response for a service, edit the distributed scheme definition and include the `<active-failure-mode>` element within the `<persistence>` element and set the value to `stop-persistence`. If no value is specified, then the default value (`stop-service`) is automatically used. The following example changes the active persistence failure response to `stop-persistence`.

```
<distributed-scheme>
   <scheme-name>distributed</scheme-name>
   <service-name>Service1</service-name>
   <backing-map-scheme>
      <local-scheme/>
   </backing-map-scheme>
   <persistence>
      <active-failure-mode>stop-persistence</active-failure-mode>
   </persistence>
   <autostart>true</autostart>
</distributed-scheme>
```

Changing the Partition Count When Using Active Persistence

The partition count cannot be changed when using active persistence. If you change a services partition count, then on restart of the services all active data is moved to the persistence trash and must be recovered after the original partition count is restored. Data that is persisted can only be recovered only to services that are running with the same partition count, or you can select one of the available workarounds. See Workarounds to Migrate a Persistent Service to a Different Partition Count.

Ensure that the partition count is not modified if active persistence is being used. If the partition count is changed, then a message similar to the following is displayed when the services are started:

```
<Warning> (thread=DistributedCache:DistributedCachePersistence, member=1):
Failed to recover partition 0 from SafeBerkeleyDBStore(...); partition-count
mismatch 501(persisted) != 277(service); reinstate persistent store from
trash once validation errors have been resolved
```

The message indicates that the change in the partition-count is not supported and the current active data has been copied to the trash directory. To recover the data:

1. Shutdown the entire cluster.
2. Remove the current active directory contents for the cluster and service affected on each cluster member.
3. Copy (recursively) the contents of the trash directory for each service to the active directory.
4. Restore the partition count to the original value.
5. Restart the cluster.

- Workarounds to Migrate a Persistent Service to a Different Partition Count

Workarounds to Migrate a Persistent Service to a Different Partition Count

There are two possible workarounds when changing the partition count with persistent services:

- Using Coherence Federation as a means to replicate data to a service with a different partition count.
- Defining a new persistent service and transferring the data manually.

For instructions to use the two options, see Using Federation and Using a New Service. Oracle recommends using federation because it ensures that data is migrated and available as quickly as possible.

If the existing persistent cache service is federated, migration is trivial, as illustrated in the following steps:

1. Stop one of the destination clusters in the federation.
2. Move the persistence directory to a backup storage.
3. Create a new cache config that differs only in the partition count for the related service.
4. Invoke the Mean operation `replicateAll` from the newly started cluster with the different partition count.

If the existing persistence service is not a federated cache service, the upgrade will include a few additional steps, but is still fairly simple.

- Using Federation
- Using a New Service

Using Federation

If the existing persistent cache service is federated, migration is trivial. To migrate, complete the following steps:

1. Create a new cache config, and change the existing cache service from distributed scheme to federated scheme.

> **Note:**
>
> It is important to use the exact same service name.

For example, if this is the existing cache config:

```
<caching-scheme-mapping>
    <cache-mapping>
      <cache-name>*</cache-name>
      <scheme-name>federated-active</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <distributed-scheme>
      <scheme-name>federated-active</scheme-name>
      <service-name>DistributedCachePersistence</service-name>
      <thread-count>5</thread-count>
      <partition-count system-property="test.partitioncount">5</partition-
count>
      <backing-map-scheme>
        <local-scheme>
        </local-scheme>
      </backing-map-scheme>
      <persistence>
          <environment>simple-bdb-environment</environment>
      </persistence>
      <autostart>true</autostart>
    </distributed-scheme>
  </caching-schemes>
</cache-config>
```

Then the new config should be:

```
<caching-scheme-mapping>
    <cache-mapping>
      <cache-name>*</cache-name>
      <scheme-name>federated-active</scheme-name>
```

```
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <federated-scheme>
      <scheme-name>federated-active</scheme-name>
      <service-name>DistributedCachePersistence</service-name>
      <thread-count>5</thread-count>
      <partition-count system-property="test.partitioncount">5</
partition-count>
      <backing-map-scheme>
        <local-scheme>
        </local-scheme>
      </backing-map-scheme>
      <persistence>
          <environment>simple-bdb-environment</environment>
      </persistence>
      <autostart>true</autostart>
      <topologies>
        <topology>
          <name>EastCoast</name>
        </topology>
      </topologies>
    </federated-scheme>
  </caching-schemes>
```

Ensure that you change the override config file to include federation. For example, add the following section to override the file:

```
<federation-config>
    <participants>
      <participant>
        <name>BOSTON</name>
        <remote-addresses>
          <socket-address>
            <address>192.168.1.5</address>
            <port system-
property="test.federation.port.boston">7574</port>
          </socket-address>
        </remote-addresses>
      </participant>

      <participant>
        <name>NEWYORK</name>
        <remote-addresses>
          <socket-address>
            <address>192.168.1.5</address>
            <port system-
property="test.federation.port.newyork">7574</port>
          </socket-address>
        </remote-addresses>
      </participant>

    </participants>
```

```
    <topology-definitions>
      <active-passive>
        <name>EastCoast</name>
        <active>NEWYORK</active>
        <passive>BOSTON</passive>
      </active-passive>
    </topology-definitions>

  </federation-config>
```

See Federating Caches Across Clusters .

2. Stop the running cluster.
3. Start the active side of the cluster. In this example, the NEWYORK cluster. The persistent stores from the previous cluster should be recovered successfully.
4. Start the passive cluster. In this example, the BOSTON cluster.
5. Invoke the Mean operation `replicateAll` to the passive, BOSTON, cluster.

Now, all the data is persisted to the cluster with the target partition count.

Oracle recommends that you use the federated cache scheme to facilitate any future upgrade. However, if desired, you can also go back to the distributed scheme with the new partition count. Simply point the active persistent directory to the one that is created by BOSTON.

Using a New Service

If the existing persistent service is not a federated cache service, the upgrade will include a few additional steps, but is still fairly simple.
If you do not want to use a federated service, perform the following steps to upgrade:

1. Create a new cache config, duplicate the existing cache service with a different service name and targeted partition count. For example:

```
<caching-scheme-mapping>
    <cache-mapping>
      <cache-name>dist*</cache-name>
      <scheme-name>simple-persistence</scheme-name>
    </cache-mapping>
    <cache-mapping>
      <cache-name>new*</cache-name>
      <scheme-name>new-persistence</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <distributed-scheme>
      <scheme-name>simple-persistence</scheme-name>
      <service-name>DistributedCachePersistence</service-name>
      <partition-count system-property="test-partitioncount">257</
partition-count>
      <backing-map-scheme>
        <local-scheme/>
```

```
      </backing-map-scheme>
      <persistence>
        <environment>simple-bdb-environment</environment>
      </persistence>
      <autostart>true</autostart>
    </distributed-scheme>

    <distributed-scheme>
      <scheme-name>new-persistence</scheme-name>
      <service-name>DistributedCachePersistenceNew</service-name>
      <partition-count system-property="new-partitioncount">457</
partition-count>
      <backing-map-scheme>
        <local-scheme/>
      </backing-map-scheme>
      <persistence>
        <environment>simple-bdb-environment</environment>
      </persistence>
      <autostart>true</autostart>
    </distributed-scheme>
  </caching-schemes>
</cache-config>
```

2. Stop the cluster.
3. Start the cluster with the new config. Now, you have two distributed services (old and new), with different partition counts.
4. Transfer the data from the “old” service to the “new” service. Here is an example of client code for the data transfer:

```
public static void main(String[] args)
    {

System.setProperty("tangosol.coherence.distributed.localstorage",
"false");
    System.setProperty("coherence.cacheconfig", "path to new cache
config file");
    System.setProperty("coherence.override", “path to override
config file");

    NamedCache cacheOld  = CacheFactory.getCache("dist");
    NamedCache cacheTemp = CacheFactory.getCache("new");

    DistributedCacheService serviceOld = (DistributedCacheService)
cacheOld.getCacheService();
    DistributedCacheService servicenNew = (DistributedCacheService)
cacheTemp.getCacheService();

    NamedCache   cache       = servicenNew.ensureCache("dist",
null);   // the cache name must be same as the old one
    int          cPartitions = serviceOld.getPartitionCount();
    PartitionSet parts       = new PartitionSet(cPartitions);

    for (int iPartition = 0; iPartition < cPartitions; iPartition++)
        {
```

```
            parts.add(iPartition);

            Filter filter = new PartitionedFilter(AlwaysFilter.INSTANCE,
parts);

            Set<Map.Entry> setPart  = cacheOld.entrySet(filter);

            cache.putAll(new EntrySetMap(setPart));

            parts.remove(iPartition);
            }

        System.out.println("CacheOld.size " + cacheOld.size());
        System.out.println("CacheNew.size " + cache.size());

        cacheTemp.destroy();
```

Now, all the data is persisted in the new service with the targeted partition count.

5. Shut down the cluster.
6. Go to the active persistent directory of the cluster. You will see two directories. In our example, **DistributedCachePersistence** and **DistributedCachePersistenceNew**. Move the directory of the old service, **DistributedCachePersistence**, to a backup storage.
7. Now, remove the old service from the cache config and restart the cluster. All partitions should be recovered successfully with new partition count and new service name.
8. If you want to use exactly the same service name, simply rename the new persistent directory to the existing service name, and restart the cluster with the old cache config with the targeted partition count.

To avoid loss of new live data while doing the data transfer to the new service, block the client requests temporarily.

Using Asynchronous Persistence Mode

You can enable and configure asynchronous persistence mode to have the storage servers to persist data asynchronously. See Using Asynchronous Persistence.

Modifying the Pre-Defined Persistence Environments

Persistence uses a set of directories for storage. You can choose to use the default storage directories or change the directories as required.

This section includes the following topics:

- Overview of the Pre-Defined Persistence Environment
- Changing the Pre-Defined Persistence Directory

Overview of the Pre-Defined Persistence Environment

The operational deployment descriptor includes two pre-defined persistence environment definitions:

- `default-active` – used when active persistence is enabled.

- `default-on-demand` – used when on-demand persistence is enabled.

The operational override file or system properties are used to override the default settings of the pre-defined persistence environments. The pre-defined persistence environments have the following configuration:

```
<persistence-environments>
   <persistence-environment id="default-active">
      <persistence-mode>active</persistence-mode>
      <active-directory
        system-property="coherence.distributed.persistence.active.dir">
      </active-directory>
      <snapshot-directory
        system-
property="coherence.distributed.persistence.snapshot.dir">
      </snapshot-directory>
      <trash-directory
        system-property="coherence.distributed.persistence.trash.dir">
      </trash-directory>
   </persistence-environment>
   <persistence-environment-environment id="default-on-demand">
      <persistence-mode>on-demand</persistence-mode>
      <active-directory
        system-property="coherence.distributed.persistence.active.dir">
      </active-directory>
      <snapshot-directory
        system-
property="coherence.distributed.persistence.snapshot.dir">
      </snapshot-directory>
      <trash-directory
        system-property="coherence.distributed.persistence.trash.dir">
      </trash-directory>
   </persistence-environment>
</persistence-environments>
```

Changing the Pre-Defined Persistence Directory

The pre-defined persistence environments use a base directory called `coherence` within the `USER_HOME` directory to save persistence files. The location includes directories for active persistence files, snapshot persistence files, and trash files. The locations can be changed to a different local directory or a shared directory on the network.

> **Note:**
>
> - Persistence directories and files (including the `meta.properties` files) should never be manually edited. Editing the directories and files can lead to persistence errors.

To change the pre-defined location of persistence files, include the `<active-directory>`, `<snapshot-directory>`, and `<trash-directory>` elements that are each

set to the respective directories where persistence files are saved. The following example modifies the pre-defined on-demand persistence environment and changes the location of all directories to the `/persistence` directory:

```
<persistence-environments>
   <persistence-environment id="default-on-demand">
      <active-directory
        system-property="coherence.distributed.persistence.active.dir">
        /persistence/active</active-directory>
      <snapshot-directory
        system-property="coherence.distributed.persistence.snapshot.dir">
        /persistence/snapshot</snapshot-directory>
      <trash-directory
        system-property="coherence.distributed.persistence.trash.dir">
        /persistence</trash</trash-directory>
   </persistence-environment>
</persistence-environments>
```

The following system properties are used to change the pre-defined location of the persistence files instead of using the operational override file:

```
-Dcoherence.distributed.persistence.active.dir=/persistence/active
-Dcoherence.distributed.persistence.snapshot.dir=/persistence/snapshot
-Dcoherence.distributed.persistence.trash.dir=/persistence/trash
```

Use the `coherence.distributed.persistence.base.dir` system property to change the default directory off the `USER_HOME` directory:

```
-Dcoherence.distributed.persistence.base.dir=persistence
```

Creating Persistence Environments

You can choose to define and use multiple persistence environments to support different cache scenarios. Persistence environments are defined in the operational override configuration file and are referred within a distributed scheme or `paged-topic-scheme` definition in the cache configuration file.
This section includes the following topics:

- Define a Persistence Environment
- Configure a Persistence Mode
- Configure Persistence Directories
- Configure a Cache Service to Use a Persistence Environment

Define a Persistence Environment

To define a persistence environment, include the `<persistence-environments>` element that contains a `<persistence-environment>` element. The `<persistence-environment>` element includes the configuration for a persistence environment. Use the `id` attribute to name the environment. The `id` attribute is used to refer to the persistence environment from a distributed scheme definition. The following example creates a persistence environment with the name `environment1`:

```
<persistence-environments>
   <persistence-environment id="enviornment1">
      <persistence-mode></persistence-mode>
      <active-directory></active-directory>
```

```
      <snapshot-directory></snapshot-directory>
      <trash-directory></trash-directory>
   </persistence-environment>
</persistence-environments>
```

Configure a Persistence Mode

A persistence environment supports two persistence modes: on-demand and active. On-demand persistence requires the use of the persistence coordinator to persist and recover cache services. Active persistence automatically persists and recovers cache services. You can still use the persistence coordinator in active persistence mode to periodically persist a cache services.

To configure the persistence mode, include the `<persistence-mode>` element set to either `on-demand` or `active`. The default value if no value is specified is `on-demand`. The following example configures active persistence.

```
<persistence-environments>
   <persistence-environment id="enviornment1">
      <persistence-mode>active</persistence-mode>
      <persistence-mode></persistence-mode>
      <active-directory></active-directory>
      <snapshot-directory></snapshot-directory>
      <trash-directory></trash-directory>
   </persistence-environment>
</persistence-environments>
```

Configure Persistence Directories

A persistence environment saves cache service data to disk. The location can be configured as required and can be either on a local drive or on a shared network drive. When configuring a local drive, only the partitions that are owned by a cache server are persisted to the respective local disk. When configuring a shared network drive, all partitions are persisted to the same shared disk.

> **Note:**
>
> - Persistence directories and files (including the `meta.properties` files) should never be manually edited. Editing the directories and files can lead to persistence errors.
> - If persistence is configured to use an NFS mounted file system, then the NFS mount should be configured to use synchronous IO and not asynchronous IO, which is the default on many operating systems. The use of asynchronous IO can lead to data loss if the file system becomes unresponsive due to an outage. For details on configuration, refer to the `mount` documentation for your operating system.

Different directories are used for active, snapshot and trash files and are named accordingly. Only the top-level directory must be specified. To configure persistence directories, include the `<active-directory>`, `<snapshot-directory>`, and `<trash-directory>` elements that are each set to a directory path where persistence files are

saved. The default value if no value is specified is the *USER_HOME* directory. The following example configures the `/env1` directory for all persistence files:

```
<persistence-environments>
   <persistence-environment id="enviornment1">
      <persistence-mode>on-demand</persistence-mode>
      <active-directory>/env1</active-directory>
      <snapshot-directory>/env1</snapshot-directory>
      <trash-directory>/env1</trash-directory>
   </persistence-environment>
</persistence-environments>
```

Configure a Cache Service to Use a Persistence Environment

To change the persistence environment used by a cache service, modify the distributed scheme definition and include the `<environment>` element within the `<persistence>` element. Set the value of the `<environment>` element to the name of a persistence environment that is defined in the operational override configuration file. For example:

```
<distributed-scheme>
   <scheme-name>distributed</scheme-name>
   <service-name>Service1</service-name>
   <backing-map-scheme>
      <local-scheme/>
   </backing-map-scheme>
   <persistence>
      <environment>environment1</environment>
   </persistence>
   <autostart>true</autostart>
</distributed-scheme>
```

Using Quorum for Persistence Recovery

Coherence includes a quorum policy that enables Coherence to defer recovery until a suitable point. Suitability is based on availability of all partitions and sufficient capacity to initiate recovery across storage members.

This section includes the following topics:

- Overview of Persistence Recovery Quorum
- Using the Dynamic Recovery Quorum Policy
- Explicit Persistence Quorum Configuration

Overview of Persistence Recovery Quorum

The partitioned cache recover quorum uses two inputs to determine whether the requirements of partition availability and storage capacity are met to commence persistence recovery. The partition availability requirement is met through a list of recovery host names (machines that will contain the persistent stores), while the storage capacity requirement is met by specifying the number of storage nodes. Using the Dynamic Quorum policy both of these inputs can be inferred by the 'last known good' state of the service, and is the recommended approach for configuring a recovery quorum. If the rules used by the Dynamic Quorum policy are insufficient, you can explicitly specify the recovery-hosts list and the recover-quorum. The use of the quorum allows time for a cluster to start and ensures that

partitions are recovered gracefully without overloading too few storage members or without inadvertently deleting orphaned partitions.

If the recover quorum is not satisfied, then persistence recovery does not proceed and the service or cluster may appear to be blocked. To check for this scenario, view the `QuorumPolicy` attribute in the `ServiceMBean` MBean to see if `recover` is included in the list of actions. If data has not been recovered after cluster startup, the following log message is emitted (each time a new service member starts up) to indicate that the quorum has not been satisfied:

```
<Warning> (thread=DistributedCache:DistributedCachePersistence, member=1):
 Action recover disallowed; all-disallowed-actions: recover(4)
```

After the quorum is satisfied, the following message is emitted:

```
<Warning> (thread=DistributedCache:DistributedCachePersistence, member=1):
 All actions allowed
```

For active persistence, the recover quorum is enabled by default and automatically uses the dynamic recovery quorum policy. See Using the Dynamic Recovery Quorum Policy.

For general details about partitioned cache quorums, see Using the Partitioned Cache Quorums in *Developing Applications with Oracle Coherence*.

Using the Dynamic Recovery Quorum Policy

The dynamic recovery quorum policy is used with active persistence and automatically configures the persistence recovery quorum based on a predefined algorithm. The dynamic recovery quorum policy is the default quorum policy for active persistence mode and does not need to be explicitly enabled. The policy is automatically used if either the `<recover-quorum>` value is not specified or if the value is set to `0`. The following example explicitly enables the dynamic recovery quorum policy and is provided here for clarity.

```
<distributed-scheme>
   <scheme-name>distributed</scheme-name>
   <service-name>Service1</service-name>
   <backing-map-scheme>
      <local-scheme/>
   </backing-map-scheme>
   <partitioned-quorum-policy-scheme>
      <recover-quorum>0</recover-quorum>
   </partitioned-quorum-policy-scheme>
   <autostart>true</autostart>
</distributed-scheme>
```

> **Note:**
>
> When using the dynamic recovery quorum policy, the `<recovery-hosts>` element should not be used within the `<partitioned-quorum-policy-scheme>` element. All other quorum polices (for example the read quorum policy) are still valid.

Understanding the Dynamic Recovery Algorithm

The dynamic recovery quorum policy works by recording cluster membership information each time a member joins the cluster and partition distribution stabilizes. Membership is only recorded if the service is not suspended and all other partitioned cache actions (such as read, write, restore, and distribute) are allowed by the policy. JMX notifications are sent to subscribers of the `PersistenceManagerMBean` MBean every time the cluster membership changes.

During recovery scenarios, a service only recovers data if the following conditions are satisfied:

- the persistent image of all partitions is accessible by the cluster members
- the number of storage-enabled nodes is at least 2/3 of the last recorded membership
- if the persistent data is being stored to a local disk (not shared and visible by all hosts), then there should be at least 2/3 of the number of members for each host as there was when the last membership was recorded

The partitioned cache service blocks any client side requests if any of the conditions are not satisfied. However, if an administrator determines that the full recovery is impossible due to missing partitions or that starting the number of servers that is expected by the quorum is unnecessary, then the recovery can be forced by invoking the `forceRecovery` operation on the `PersistenceManagerMBean` MBean.

The recovery algorithm can be overridden by using a custom quorum policy class that extends the `com.tangosol.net.ConfigurableQuorumPolicy.PartitionedCacheQuorumPolicy` class. To change the hard-coded 2/3 ratio, override the `PartitionedCacheQuorumPolicy.calculateMinThreshold` method. See Using Custom Action Policies in *Developing Applications with Oracle Coherence*.

Explicit Persistence Quorum Configuration

To configure the recover quorum for persistence, modify a distributed scheme definition and include the `<recover-quorum>` element within the `<partitioned-quorum-policy-scheme>` element. Set the `<recover-quorum>` element value to the number of storage members that must be available before recovery starts. For example:

```
<distributed-scheme>
   <scheme-name>distributed</scheme-name>
   <service-name>Service1</service-name>
   <backing-map-scheme>
      <local-scheme/>
   </backing-map-scheme>
   <partitioned-quorum-policy-scheme>
      <recover-quorum>2</recover-quorum>
   </partitioned-quorum-policy-scheme>
   <autostart>true</autostart>
</distributed-scheme>
```

> **Note:**
>
> In active persistence mode, setting the `<recover-quorum>` element to `0`, enables the dynamic recovery quorum policy. See Using the Dynamic Recovery Quorum Policy.

In shared disk scenarios, all partitions are persisted and recovered from a single location. For local-disk scenarios, each storage member recovers its partitions from a local disk. However, if you use a non-dynamic recovery quorum with local-disk based storage, you must define a list of storage-enabled hosts in the cluster that are required to recover orphaned partition from the persistent storage, otherwise empty partitions will be assigned.

> **Note:**
>
> Recovery hosts must be specified to ensure that recovery does not commence prior to all persisted state being available.

To define a list of addresses, edit the operational override configuration file and include the `<address-provider>` element that contains a list of addresses each defined using an `<address>` element. Use the `id` attribute to name the address provider list. The `id` attribute is used to refer to the list from a distributed scheme definition. The following example creates an address provider list that contains two member addresses and is named `persistence_hosts`:

```
<address-providers>
   <address-provider id="persistence_hosts">
      <address>HOST_NAME1</address>
      <address>HOST_NAME2</address>
   </address-provider>
</address-providers>
```

To refer to the address provider list, modify a distributed scheme definition and include the `<recovery-hosts>` element within the `<partitioned-quorum-policy-scheme>` element and set the value to the name of an address provider list. For example:

```
<distributed-scheme>
   <scheme-name>distributed</scheme-name>
   <service-name>Service1</service-name>
   <backing-map-scheme>
      <local-scheme/>
   </backing-map-scheme>
   <partitioned-quorum-policy-scheme>
      <recover-quorum>2</recover-quorum>
      <recovery-hosts>persistence_hosts</recovery-hosts>
   </partitioned-quorum-policy-scheme>
   <autostart>true</autostart>
</distributed-scheme>
```

Subscribing to Persistence JMX Notifications

The `PersistenceManagerMBean` MBean includes a set of notification types that applications can use to monitor persistence operations. See PersistenceManagerMBean in *Managing Oracle Coherence*.
To subscribe to persistence JMX notifications, implement the JMX `NotificationListener` interface and register the listener. The following code snippet demonstrates registering a notification listener. Refer to the Coherence examples for the complete example, which includes a sample listener implementation.

```
...
MBeanServer server = MBeanHelper.findMBeanServer();
```

```
Registry registry = cluster.getManagement();
try
   {
   for (String sServiceName : setServices)
      {
      logHeader("Registering listener for " + sServiceName);
      String sMBeanName = getMBeanName(sServiceName);

      ObjectName           oBeanName = new ObjectName(sMBeanName);
      NotificationListener listener  = new
         PersistenceNotificationListener(sServiceName);
      server.addNotificationListener(oBeanName, listener, null, null);
      }
   ...
```

Managing Persistence

Persistence should be managed to ensure there is enough disk space and to ensure persistence operations do not add significant latency to cache operations. Latency is specific to active persistence mode and can affect cache performance because persistence operations are being performed in parallel with cache operations.
This section includes the following topics:

- Plan for Persistence Storage
- Plan for Persistence Memory Overhead
- Monitor Persistence Storage Usage
- Monitoring Persistence Latencies

Plan for Persistence Storage

An adequate amount of disk space is required to persist data. Ensure enough space is provisioned to persist the expected amount of cached data. The following guidelines should be used when sizing disks for persistence:

- The approximate overhead for active persistence data storage is an extra 10-30% per partition. The actual overhead may vary depending upon data access patterns, the size of keys and values, and other factors such as block sizes and heavy system load.
- Use the Coherence VisualVM plug-in and persistence reports to monitor space availability and usage. See Monitor Persistence Storage Usage. Specifically, use the `PersistenceActiveSpaceUsed` attribute on the `ServiceMBean` MBean to monitor the actual persistence space used for each service and node.
- Persistence configurations that use a shared disk for storage should plan for the potential maximum size of the cache because all partitions are persisted to the same location. For example, if the maximum capacity of a cache is 8GB, then the shared disk must be able to accommodate at least 8GB of persisted data plus overhead.
- Persistence configurations that use a local disk for storage should plan for the potential maximum cache capacity of the cache server because only the partitions owned by a cache server are persisted to the local disk. For example, if the maximum cache capacity of a cache server is 2GB, then the local disk must be able to accommodate at least 2GB of persisted data plus overhead.
- Plan additional space when creating snapshots in either active or on-demand mode. Each snapshot of a cache duplicates the size of the persistence files on disk.

- Plan additional space for snapshot archives. Each archive of a snapshot is slightly less than the size of the snapshot files on disk.

> **Note:**
>
> The underlying Berkeley DB (BDB) storage which is used for persistence, is "append only". Insertions, deletions, and updates are always added at the end of the current file. The first file is named `00000000.jdb`. When this file grows to a certain size (10 MB, by default), then a new file named `00000001.jdb` becomes the current file, and so on. As the files reach the 10MB limit and roll over to a new file, the background tasks are run automatically to cleanup, compress, and remove the unused files. As a result, you will have at least 10MB storage overhead per service per partition. This overhead is taken into account in the 10-30% figure quoted above.

Plan for Persistence Memory Overhead

In addition to affecting disk usage, active persistence requires additional data structures and memory within each JVM's heap for managing persistence. The amount of memory required varies based on the partition-count and data usage patterns but as a guide, you should allocate an additional 20-35% of memory to each JVM that is running active persistence.

Monitor Persistence Storage Usage

Monitor persistence storage to ensure that there is enough space available on the file system to persist cached data.

Coherence VisualVM Plug-in

Use the Persistence tab in the Coherence VisualVM plug-in to view the amount of space being used by a service for active persistence. The space is reported in both Bytes and Megabytes. The tab also reports the current number of snapshots available for a service. The snapshot number can be used to estimate the additional space usage and to determine whether snapshots should be deleted to free up space.

Coherence Reports

Use the persistence detail report (`persistence-detail.txt`) to view the amount of space being used by a service for both active persistence and for persistence snapshots. The amount of available disk space is also reported and allows you to monitor if a disk is reaching capacity.

Coherence MBeans

Use the persistence attributes on the `ServiceMBean` MBean to view all the persistence storage statistics for a service. The MBean includes statistics for both active persistence and persistence snapshots.

Monitoring Persistence Latencies

Monitor persistence latencies when using active persistence to ensure that persistence operations are not adversely affecting cache operations. High latencies can be a sign that network issues are delaying writing persistence files to a shared disk or delaying coordination between local disks.

Coherence VisualVM Plug-In

Use the Persistence tab in the Coherence VisualVM plug-in to view the amount of latency that persistence operations are adding to cache operations. The time is reported in milliseconds. Statistics are reported for each service and provide the average latency of all persistence operations and for the highest recorded latency.

Coherence Reports

Use the persistence detail report (`persistence-detail.txt`) to view the amount of latency that persistence operations are adding to cache operations. The time is reported in milliseconds. Statistics are provided for the average latency of all persistence operations and for the highest recorded latency on each cluster node of a service. The statistics can be used to determine if some nodes are experiencing higher latencies than other nodes.

Coherence MBeans

Use the persistence attributes on the `ServiceMBean` MBean to view the amount of latency that persistence operations are adding to cache operations. The time is reported in milliseconds. Statistics are provided for the average latency of all persistence operations and for the highest recorded latency on each cluster nodes of a service. The statistics can be used to determine if some nodes are experiencing higher latencies than other nodes.

Configuring Caches as Transient

Caches that do not require persistence can be configured as transient. Caches that are transient are not recovered during persistence recovery operations.

> **Note:**
>
> During persistence recovery operations, the entire cache service is recovered from the persisted state and any caches that are configured as transient are reset.

Caches are configured as transient using the `<transient>` element within the `<backing-map-scheme>` element of a distributed scheme definition. However, because persistence is always enabled on a service, a parameter macro is used to configure the transient setting for each cache. For example:

```
<caching-scheme-mapping>
   <cache-mapping>
      <cache-name>nonPersistedCache</cache-name>
      <scheme-name>distributed</scheme-name>
      <init-params>
         <init-param>
            <param-name>transient</param-name>
```

```
            <param-value>true</param-value>
         </init-param>
      </init-params>
   </cache-mapping>
   <cache-mapping>
      <cache-name>persistedCache</cache-name>
      <scheme-name>distributed</scheme-name>
   </cache-mapping>
</caching-scheme-mapping>

<distributed-scheme>
   <scheme-name>distributed</scheme-name>
   <service-name>DistributedService</service-name>
   <backing-map-scheme>
      <transient>{transient false}</transient>
      <local-scheme/>
   </backing-map-scheme>
   <autostart>true</autostart>
</distributed-scheme>
```

Note:

The default value of the `<transient>` element is `false` and indicates that cache data is persisted.

7

Federating Caches Across Clusters

The Coherence federated caching feature is used to link multiple clusters so that cache data is automatically synchronized between clusters.
This chapter includes the following sections:

Overview of Federated Caching

Federated caching federates cache data asynchronously across multiple geographically dispersed clusters. Cached data is federated across clusters to provide redundancy, off-site backup, and multiple points of access for application users in different geographical locations.

> **Note:**
>
> Care should be taken when issuing the following cache operations because they are not supported or federated to the destination participants and will cause inconsistencies in your federated data:
>
> - Destroy Cache – If you issue this operation, the local cluster's cache will be destroyed, but any destination caches will be left intact.
> - Truncate Cache – If you issue this operation, the local cluster's cache will be truncated, but any destination caches will be left intact.

This section includes the following topics:

Multiple Federation Topologies

Federated caching supports multiple federation topologies. These include:

- active-active
- active-passive
- hub-spoke
- central-federation

The topologies define common federation strategies between clusters and support a wide variety of use cases. Custom federation topologies can also be created as required.

Conflict Resolution

Federated caching provides applications with the ability to accept, reject, or modify cache entries being stored locally or remotely. Conflict resolution is application specific to allow the greatest amount of flexibility when defining federation rules.

Federation Configuration

Federated caching is configured using Coherence configuration files and requires no changes to application code. An operational override file is used to configure federation participants and the federation topology. A cache configuration file is used to create federated caches schemes. A federated cache is a type of partitioned cache service and is managed by a federated cache service instance.

Management and Monitoring

Federated caching is managed using attributes and operations from the `FederationManagerMBean`, `DestinationMBean`, `OriginMBean` and `TopologyMBean` MBeans. These MBeans make it easy to perform administrative operations, such as starting and stopping federation and to monitor federation configuration and performance statistics. Many of these statistics and operations are also available from the Coherence VisualVM plug-in.

Federation attributes and statistics are aggregated in the `federation-status`, `federation-origin`, and `federation-destination` reports. Federation statistics are also aggregated in the Coherence VisualVM plug-in. Both tools can help troubleshoot possible resource and performance issues.

In addition, as with any distributed cache, federated services and caches can be managed and monitored using the attributes operations of the `ServiceMBean` MBean and `CacheMBean` MBean and related reports and the VisualVM plug-in tabs.

For more information about managing Coherence federation, see Managing Federated Caching.

General Steps for Setting Up Federated Caching

Federated caching is configuration based and in most cases requires no application changes. Setup includes configuring federation participants, topologies, and caches.
To set up federated caching:

1. Ensure that all clusters that are participating in the federation are operational and that you know the address (host and cluster port) of at least one cache server in each cluster.
2. Configure each cluster with a list of the cluster participants that are in the federation. See Defining Federation Participants.
3. Configure each cluster with a topology definition that specifies how data is federated among cluster participants. See Defining Federation Topologies.
4. Configure each cluster with a federated cache scheme that is used to store cached data. See Defining Federated Cache Schemes.
5. Configure the federated cache on each cluster to use a defined federation topology. See Associating a Federated Cache with a Federation Topology.

Defining Federation Participants

Each Coherence cluster in a federation must be defined as a federation participant. Federation participants are defined in an operational override file. The operational override file for each cluster in the federation must include the list of participants to be federated. The list of participants must include the local cluster participant and remote cluster participants. To define federation participants, include any number of `<participant>` elements within the `<participants>` element. Use the `<name>` element to define a name for the participant and the `<remote-addresses>` element to define the address and port of at least one cache server or proxy that is located in the participant cluster. Enter the cluster port if you are using the `NameService` service to look up ephemeral ports. Entering an exact port is typically only used for environments which cannot use the `NameService` service for address lookups. The following example defines multiple participants and demonstrates both methods for specifying a remote address:

```
<federation-config>
   <participants>
      <participant>
         <name>LocalClusterA</name>
         <remote-addresses>
            <socket-address>
               <address>192.168.1.7</address>
               <port>7574</port>
            </socket-address>
         </remote-addresses>
      </participant>
      <participant>
         <name>RemoteClusterB</name>
         <remote-addresses>
            <socket-address>
               <address>192.168.10.16</address>
               <port>9001</port>
            </socket-address>
         </remote-addresses>
      </participant>
      <participant>
```

```
            <name>RemoteClusterC</name>
            <remote-addresses>
               <socket-address>
                  <address>192.168.19.25</address>
                  <port>9001</port>
               </socket-address>
            </remote-addresses>
         </participant>
      </participants>
   </federation-config>
```

The `<address>` element also supports external NAT addresses that route to local addresses; however the external and local addresses must use the same port number.

Changing the Default Settings of Federation Participants

Federation participants can be explicitly configured to override their default settings. The default settings include:

- The federation state that a cluster participant is in when the cluster is started.
- The connect time-out to a destination cluster.
- The send time-out for acknowledgement messages from a destination cluster.
- The maximum bandwidth, per member, for sending federated data to a destination participant. This value is loaded from the source member's configuration of the destination participant.

> **Note:**
>
> The value of maximum bandwidth can be specified as a combination of a decimal factor and a unit descriptor such as Mbps, KBps, and so on. If no unit is specified, a unit of bps (bits per second) is assumed.

- The location meta-data for the participant

See participant in *Developing Applications with Oracle Coherence*, for more information.
To change the default settings of federation participants, edit the operational override file for the cluster and modify the `<participant>` definition. Update the value of each setting as required. For example:

```
<participant>
   <name>ClusterA</name>
   <initial-action>start</initial-action>
   <connect-timeout>1m</connect-timeout>
   <send-timeout>5m</send-timeout>
   <max-bandwidth>100Mbps</max-bandwidth>
   <geo-ip>Philadelphia</geo-ip>
   <remote-addresses>
      <socket-address>
         <address>192.168.1.7</address>
         <port>7574</port>
      </socket-address>
   </remote-addresses>
</participant>
```

Understanding Federation Topologies

Federation topologies determine how data is federated and synchronized between cluster participants in a federation. The federation topology defines which clusters can send cached data, which clusters can receive cached data, and which clusters can re-send cached data. These roles are well-defined and ensure that data is not missed or sent multiples times. The supported federation topologies are:

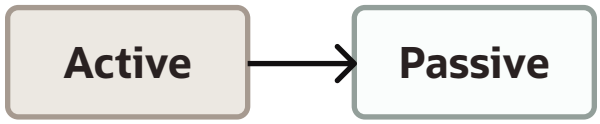
- Active-Passive Topologies
- Active-Active Topologies
- Hub and Spoke Topologies
- Central Federation Topologies
- Custom Federation Topologies

Active-Passive Topologies

Active-passive topologies are used to federate data from an active cluster to a passive cluster. Data that is put into active cluster is federated to the passive cluster. If data is put into the passive cluster, then it does not get federated to the active cluster. Consider using active-passive topologies when a copy of cached data is required for read-only operations or an off-site backup is required.

Figure 7-1 provides conceptual view of an active-passive topology.

Figure 7-1 Active-Passive Topology

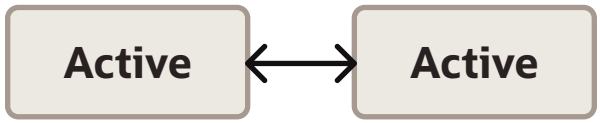


Active-Active Topologies

Active-active topologies are used to federate data between active clusters. Data that is put into one active cluster, is federated at the other active clusters. The active-active topology ensures that cached data is always synchronized between clusters. Consider using an active-active topology to provide applications in multiple geographical location with access to a local cluster instance.

Figure 7-2 provides a conceptual view of an active-active topology.

Figure 7-2 Active-Active Topology

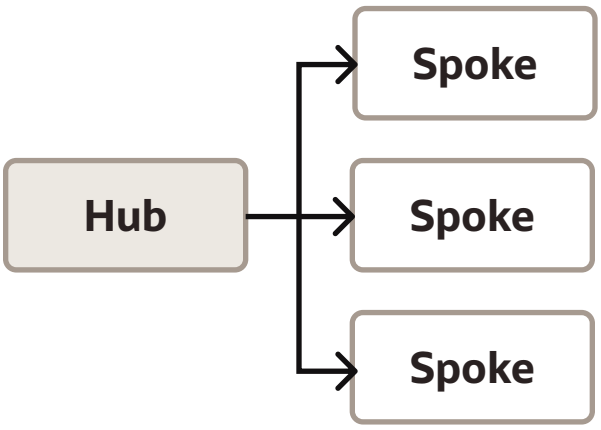


Hub and Spoke Topologies

Hub and spoke topologies are used to federate data from a single hub cluster to multiple spoke clusters. The hub cluster can only send data and spoke clusters can only receive data. Consider using a hub and spoke topology when multiple geographically dispersed copies of a cluster are required. Each spoke cluster can be used by local applications to perform read-only operations.

Figure 7-3 provides a conceptual view of a hub and spoke topology.

Figure 7-3 Hub and Spoke Topology

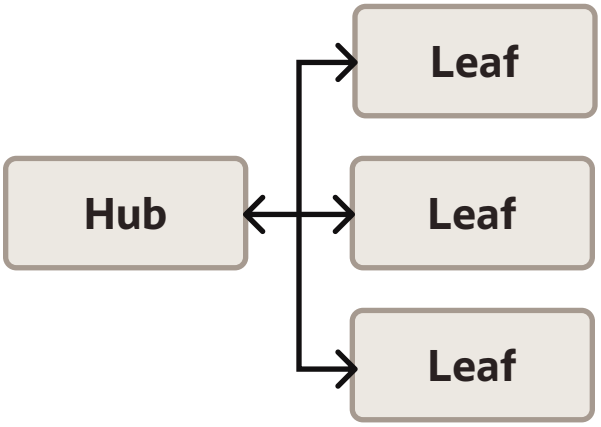


Central Federation Topologies

Central federation topologies are used to federate data from a single hub to multiple leaf clusters. In addition, each leaf can send data to the hub cluster and the hub cluster re-sends (repeats) the data to all the other leaf clusters. Consider using a central federation topology to provide applications in multiple geographical location with access to a local cluster instance.

Figure 7-4 provides a conceptual view of a central federation topology.

Figure 7-4 Central Federation Topology

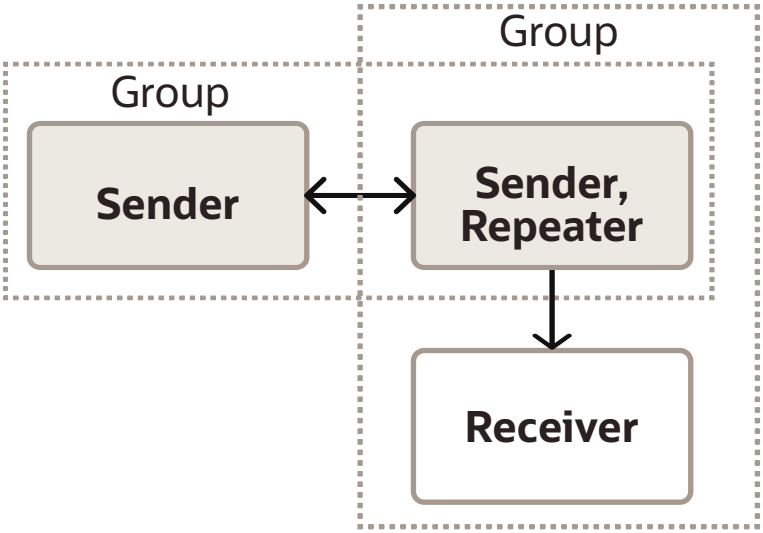


Custom Federation Topologies

Custom federation topologies are used to create free-from topologies. Clusters are organized into groups and each cluster is designated with a role in the group. The roles include: sender, receiver, or repeater. A sender participant only federates changes occurring on the local cluster. A repeater federates both local cluster changes as well changes it receives from other participants. Only sender and repeater clusters can federate data to other clusters in the group. Consider creating a custom federation topology if the pre-defined federation topologies do not address the federation requirements of a cache.

Figure 7-5 provides a conceptual view of a custom federation topology in one possible configuration.

Figure 7-5 Custom Federation Topology



Defining Federation Topologies

A topology definition includes the federation roles that each cluster participant performs in the topology. Multiple topologies can be defined and participants can be part of multiple topologies. Each cluster in the federation should have a corresponding federation topology definition to ensure that data is federated between participants in an expected manner. Federation topologies are defined in an operational override file within the `<federation-config>` element. If you are unsure about which federation topology to use, then see Understanding Federation Topologies before completing the instructions in this section.

> **Note:**
>
> If no topology is defined, then all the participants are assumed to be in an active-active topology.

This section includes the following topics:

Defining Active-Passive Topologies

To configure active-passive topologies edit the operational override file and include an `<active-passive>` element within the `<topology-definitions>` element. Use the `<name>` element to include a name that is used to reference this topology. Use the `<active>` element to define active participants and the `<passive>` element to define passive participants. For example:

```
<federation-config>
   ...
   <topology-definitions>
      <active-passive>
         <name>MyTopology</name>
         <active>LocalClusterA</active>
         <passive>RemoteClusterB</passive>
      </active-passive>
   </topology-definitions>
</federation-config>
```

With this topology, changes that are made on LocalClusterA are federated to RemoteClusterB, but changes that are made on RemoteClusterB are not federated to LocalClusterA.

Defining Active-Active Topologies

To configure active-active topologies edit the operational override file and include an `<active-active>` element within the `<topology-definitions>` element. Use the `<name>` element to include a name that is used to reference this topology. Use the `<active>` element to define active participants. For example:

```
<federation-config>
   ...
   <topology-definitions>
      <active-active>
         <name>MyTopology</name>
         <active>LocalClusterA</active>
         <active>RemoteClusterB</active>
      </active-active>
   </topology-definitions>
</federation-config>
```

With this topology, changes that are made on LocalClusterA are federated to RemoteClusterB and changes that are made on RemoteClusterB are federated to LocalClusterA.

Defining Hub and Spoke Topologies

To configure hub and spoke topologies edit the operational override file and include a <hub-spoke> element within the <topology-definitions> element. Use the <name> element to include a name that is used to reference this topology. Use the <hub> element to define the hub participant and the <spoke> element to define the spoke participants. For example:

```
<federation-config>
   ...
   <topology-definitions>
      <hub-spoke>
         <name>MyTopology</name>
         <hub>LocalClusterA</hub>
         <spoke>RemoteClusterB</spoke>
         <spoke>RemoteClusterC</spoke>
      </hub-spoke>
   </topology-definitions>
</federation-config>
```

With this topology, changes that are made on LocalClusterA are federated to RemoteClusterB and RemoteClusterC, but changes that are made on RemoteClusterB and RemoteClusterC are not federated to LocalClusterA.

Defining Central Federation Topologies

To configure central federation topologies edit the operational override file and include a <central-replication> element within the <topology-definitions> element. Use the <name> element to include a name that is used to reference this topology. Use the <hub> element to define the hub participant and the <leaf> element to define the leaf participants. For example:

```
<federation-config>
   ...
   <topology-definitions>
      <central-replication>
         <name>MyTopology</name>
         <hub>LocalClusterA</hub>
         <leaf>RemoteClusterB</leaf>
         <leaf>RemoteClusterC</leaf>
      </central-replication>
   </topology-definitions>
</federation-config>
```

With this topology, changes that are made on LocalClusterA are federated to RemoteClusterB and RemoteClusterC. Changes that are made on RemoteClusterB or RemoteClusterC are federated to LocalClusterA, which re-sends the data to the other cluster participant.

Defining Custom Topologies

To configure custom topologies edit the operational override file and include a <custom-topology> element within the <topology-definitions> element. Use the <name> element to include a name that is used to reference this topology. Use the <group> element within the <groups> element to define the role (sender, repeater, or receiver) for each the participant in the group. For example:

```
<federation-config>
   ...
   <topology-definitions>
      <custom-topology>
         <name>MyTopology</name>
         <groups>
            <group>
               <sender>LocalClusterA</sender>
               <sender>RemoteClusterB</sender>
            </group>
            <group>
               <repeater>LocalClusterA</repeater>
               <receiver>RemoteClusterC</receiver>
            </group>
         </groups>
      </custom-topology>
   </topology-definitions>
</federation-config>
```

With this topology, changes that are made on LocalClusterA or RemoteClusterB are federated to RemoteClusterC. Any changes made on RemoteClusterC are not federated to LocalCluster A or RemoteClusterB.

Defining Federated Cache Schemes

Each participant in the cluster must include a federated cache scheme in their respective cache configuration file.

The `<federated-scheme>` element is used to define federated caches. Any number of federated caches can be defined in a cache configuration file. See federated-scheme in *Developing Applications with Oracle Coherence*.
The federated caches on all participants must be managed by the same federated service instance. The service is specified using the `<service-name>` element.

Example 7-1 defines a basic federated cache scheme that uses `federated` as the scheme name and `federated` as the service instance name. The scheme is mapped to the cache name `example`. The `<autostart>` element is set to `true` to start the federated cache service on a cache server node.

Example 7-1 Sample Federated Cache Definition

```
<?xml version="1.0" encoding="windows-1252"?>
<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-cache-config
   coherence-cache-config.xsd">
   <caching-scheme-mapping>
      <cache-mapping>
         <cache-name>example</cache-name>
         <scheme-name>federated</scheme-name>
      </cache-mapping>
   </caching-scheme-mapping>
   <caching-schemes>
      <federated-scheme>
         <scheme-name>federated</scheme-name>
         <service-name>federated</service-name>
         <backing-map-scheme>
            <local-scheme/>
         </backing-map-scheme>
```

```
         <autostart>true</autostart>
      </federated-scheme>
   </caching-schemes>
</cache-config>
```

Associating a Federated Cache with a Federation Topology

A federated cache must be associated with a topology for data to be federated to federation participants.

Topologies are defined in an operational override file and referenced from a federated cache definition. See Defining Federation Topologies.

> **Note:**
>
> If no topology is defined (all participants are assumed to be in an active-active topology) or if only one topology is defined, then a topology name does not need to be specified in a federated scheme definition.

To associate a federated cache with a federation topology, include a `<topology>` element within the `<topologies>` element and use the `<name>` element to references a federation topology that is defined in an operational override file. For example:

```
<federated-scheme>
   <scheme-name>federated</scheme-name>
   <service-name>federated</service-name>
   <backing-map-scheme>
      <local-scheme />
   </backing-map-scheme>
   <autostart>true</autostart>
   <topologies>
      <topology>
         <name>MyTopology</name>
      </topology>
   </topologies>
</federated-scheme>
```

A federated cache can be associated with multiple federation topologies. For example:

```
<federated-scheme>
   <scheme-name>federated</scheme-name>
   <service-name>federated</service-name>
   <backing-map-scheme>
      <local-scheme />
   </backing-map-scheme>
   <autostart>true</autostart>
   <topologies>
      <topology>
         <name>MyTopology1</name>
      </topology>
      <topology>
         <name>MyTopology2</name>
      </topology>
   </topologies>
</federated-scheme>
```

Overriding the Destination Cache

The default behavior of the federation service is to federate data to a cache on the remote participant using same cache name that is defined on the local participant. A different remote cache can be explicitly specified if required. However, each cache should still be managed by the same federation service; that is, the caches should specify the same value in the `<service-name>` element.
To override the default destination cache, include a `<cache-name>` element and set the value to the name of the destination cache on the remote participant. For example:

```
<federated-scheme>
   <scheme-name>federated</scheme-name>
   <service-name>federated</service-name>
   <backing-map-scheme>
      <local-scheme />
   </backing-map-scheme>
   <autostart>true</autostart>
   <topologies>
      <topology>
         <name>MyTopology</name>
         <cache-name>fed-remote</cache-name>
      </topology>
   </topologies>
</federated-scheme>
```

Excluding Caches from Being Federated

Federated caches can be exclude from being federated to other participants. Excluding a cache allows an application to perform cross-cache transactions on the same partition and service, but keep some information always local (not federated to other clusters).
To exclude a cache from being federated, add the `<federated>` element set to `false` as part of the cache mapping definition. For example:

```
<caching-scheme-mapping>
   <cache-mapping>
      <cache-name>example</cache-name>
      <scheme-name>federated</scheme-name>
   </cache-mapping>
   <cache-mapping>
      <cache-name>excluded-example</cache-name>
      <scheme-name>federated</scheme-name>
      <federated>false</federated>
   </cache-mapping>
</caching-scheme-mapping>
...
```

Limiting Federation Service Resource Usage

The federation service relies on an internal cache and journal to hold entries during federation. The internal cache can consume all the available resources on a cluster node depending on the memory limit, amount and size of the entries being federated. This can in turn adversely affect all clusters in the federation. To guard against such scenarios, the internal cache can be configured to limit the size of the internal cache.

Once the limit is reached, the federation service moves the destination participants to `ERROR` state and removes all pending entries from federation's internal backlog cache.
To limit federation service resources usage, edit a federated cache scheme and set the `<journalcache-highunits>` elements to the memory limit or number of cache entries allowed in the internal cache before the limit is reached. For example:

```
<federated-scheme>
   <scheme-name>federated</scheme-name>
   <service-name>federated</service-name>
   <backing-map-scheme>
      <local-scheme />
   </backing-map-scheme>
   <autostart>true</autostart>
   <journalcache-highunits>2G</journalcache-highunits>
</federated-scheme>
```

> **Note:**
>
> Valid values for `<journalcache-highunits>` are memory values in G, K, M, (for example, 1G, 2K, 3M) or positive integers and zero. A memory value is treated as a memory limit on federation's backlog. If no units are specified, then the value is treated as a limit on the number of entries in the backlog. Zero implies no limit. The default value is 0.

Resolving Federation Conflicts

Applications can implement any custom logic that is needed to resolve conflicts that may arise between concurrent updates of the same entry. Conflicts are resolved by creating interceptors to capture federation-specific event types and performing custom logic as required. Conflict resolution makes use of Coherence live events. See Using Live Events in *Developing Applications with Oracle Coherence*.
This section includes the following topics:

- Processing Federated Connection Events
- Processing Federated Change Events
- Federating Events to Custom Participants

Processing Federated Connection Events

Federated connection events (`FederatedConnectionEvent`) represent the communication between participants of a federated service. Event types include: `CONNECTING`, `DISCONNECTED`, `BACKLOG_EXCESSIVE`, `BACKLOG_NORMAL`, and `ERROR` events. See Federated Connection Events in *Developing Applications with Oracle Coherence*.

To process federated connection events:

1. Create an event interceptor to process the desired event types and implement any custom logic as required. See Handling Live Events in *Developing Applications with Oracle Coherence*. The following example shows an interceptor that processes `ERROR` events and prints the participant name to the console.

> **Note:**
>
> Federated connection events are raised on the same thread that caused the event. Interceptors that handle these events must never perform blocking operations.

```
package com.examples

import
com.tangosol.internal.federation.service.FederatedCacheServiceDispatcher;
import com.tangosol.net.events.EventDispatcher;
import com.tangosol.net.events.EventDispatcherAwareInterceptor;
import com.tangosol.net.events.federation.FederatedConnectionEvent;
import com.tangosol.net.events.annotation.Interceptor;
import java.util.Map;

@Interceptor(identifier = "testConnection", federatedConnectionEvents =
   FederatedConnectionEvent.Type.ERROR)
public class ConnectionInterceptorImp implements
   EventDispatcherAwareInterceptor<FederatedConnectionEvent>
   {
   @Override
   public void onEvent(FederatedConnectionEvent event)
      {
      System.out.println("Participant in Error: " +
event.getParticipantName());
      }

   @Override
   public void introduceEventDispatcher(String sIdentifier, EventDispatcher
      dispatcher)
      {
      if (dispatcher instanceof FederatedCacheServiceDispatcher)
         {
         dispatcher.addEventInterceptor(sIdentifier, this);
         }
      }
   }
```

2. Register the interceptor in a federated cache scheme. See Registering Event Interceptors in *Developing Applications with Oracle Coherence*. For example:

```
<federated-scheme>
   <scheme-name>federated</scheme-name>
   <service-name>federated</service-name>
   <backing-map-scheme>
      <local-scheme />
   </backing-map-scheme>
   <autostart>true</autostart>
   <interceptors>
      <interceptor>
         <name>MyInterceptor</name>
         <instance>
            <class-name>
               com.examples.ConnectionInterceptorImp
            </class-name>
         </instance>
      </interceptor>
```

```
      </interceptors>
      <topologies>
         <topology>
            <name>MyTopology</name>
         </topology>
      </topologies>
   </federated-scheme>
```

3. Ensure the interceptor implementation is found on the classpath at runtime.

Processing Federated Change Events

Federated change events (`FederatedChangeEvent`) represent a transactional view of all the changes that occur on the local participant. All changes that belong to a single partition are captured in a single `FederatedChangeEvent` object. From the event, a map of `ChangeRecord` objects that are indexed by cache name is provided and the participant name to which the change relates is also accessible. Through the `ChangeRecord` map, you can accept the changes, modify the values, or reject the changes. The object also provides methods to extract or update POF entries using the `PofExtractor` and `PofUpdater` APIs.

Event types include: `COMMITTING_LOCAL`, `COMMITTING_REMOTE`, and `REPLICATING` events. `REPLICATING` events are dispatched before local entries are federated to remote participants. This event is used to perform changes to the entries prior to federation. Any changes performed in the `REPLICATING` event interceptor are not reflected in the local caches. `COMMITTING_LOCAL` events are dispatched before entries are inserted locally. It is designed to resolve any local conflicts. `COMMITTING_REMOTE` events are dispatched before entries from other participants are inserted locally. It is designed to resolve the conflicts between federating entries and local entries. Any changes performed when processing `COMMITTING_LOCAL` and `COMMITTING_REMOTE` events are reflected in the local participant caches.

> **Note:**
>
> - In an active-active federation topology, modifications that are made to an entry when processing `COMMITTING_REMOTE` events are sent back to the originating participant. This can potentially end up in a cyclic loop where changes keep looping through the active participants.
> - Interceptors that capture `COMMITTING_LOCAL` events are not called for passive spoke participants.
> - Synthetic operations are not included in federation change events.

To process federated change events:

1. Create an event interceptor to process the desired event types and implement any custom logic as required. See Handling Live Events in *Developing Applications with Oracle Coherence*. The following example shows an interceptor that processes `REPLICATING` events and assigns a key name before the entry is federated.

```
package com.examples

import com.tangosol.coherence.federation.events.AbstractFederatedInterceptor;
import com.tangosol.coherence.federation.ChangeRecord;
```

```
import com.tangosol.coherence.federation.ChangeRecordUpdater;
import com.tangosol.net.events.annotation.Interceptor;
import com.tangosol.net.events.federation.FederatedChangeEvent;

@Interceptor(identifier = "yourIdentifier", federatedChangeEvents =
   FederatedChangeEvent.Type.REPLICATING)
public static class MyInterceptor extends
   AbstractFederatedInterceptor<String, String>
   {
   public ChangeRecordUpdater getChangeRecordUpdater()
      {
      return updater;
      }

   public class ChangeRecordUpdate implements ChangeRecordUpdater<String,
String>
      {
      @Override
      public void update(String sParticipant, String sCacheName,
         ChangeRecord<String, String> record)
         {
         if (sParticipant.equals("NewYork") &&
(record.getKey()).equals("key"))
            {
            record.setValue("newyork-key");
            }
         }
      }

      private ChangeRecordUpdate updater = new ChangeRecordUpdate();
   }
```

2. Register the interceptor in a federated cache scheme. See Registering Event Interceptors in *Developing Applications with Oracle Coherence*. For example:

```
<federated-scheme>
   <scheme-name>federated</scheme-name>
   <service-name>federated</service-name>
   <backing-map-scheme>
      <local-scheme />
   </backing-map-scheme>
   <autostart>true</autostart>
   <interceptors>
      <interceptor>
         <name>MyInterceptor</name>
         <instance>
            <class-name>
               com.examples.MyInterceptor
            </class-name>
         </instance>
      </interceptor>
   </interceptors>
   <topologies>
      <topology>
         <name>MyTopology</name>
      </topology>
   </topologies>
</federated-scheme>
```

3. Ensure the interceptor implementation is found on the classpath at runtime.

Federating Events to Custom Participants

Federated `ChangeRecord` objects can be federated to custom, non-cluster participants in addition to other cluster members. For example, `ChangeRecord` objects can be saved to a log, message queue, or perhaps one or more databases. Custom participants are implemented as event interceptors for the change records. Custom participants are only receiver participants.

To federate `ChangeRecord` objects to custom participants:

1. Create a `FederatedChangeEvent` interceptor to process `REPLICATING` event types and implement any custom logic for `ChangeRecord` objects. See Handling Live Events in *Developing Applications with Oracle Coherence*. The following example shows an interceptor for `REPLICATING` events that processes federation change records. Note that the `Map` of `ChangeRecord` objects can be from multiple caches. For each entry in the `Map`, the key is the cache name and the value is a list of `ChangeRecord` objects in that cache.

```
@Interceptor(identifier = "MyInterceptor", federatedChangeEvents =
FederatedChangeEvent.Type.REPLICATING)
public class MyInterceptorImplChangeEvents implements
EventInterceptor<FederatedChangeEvent>
   {
   @Override
   public void onEvent(FederatedChangeEvent event)
      {
      final String sParticipantName = "ForLogging";
      if (sParticipantName.equals(event.getParticipant()))
         {
         Map<String, Iterable<ChangeRecord<Object, Object>>> mapChanges =
event.getChanges();
         switch (event.getType())
            {
            case REPLICATING:
               m_cEvents++;
               for (Map.Entry<String, Iterable<ChangeRecord<Object, Object>>>
entry : mapChanges.entrySet())
                  {
                  for (ChangeRecord<Object, Object> record : entry.getValue())
                     {
                     if (record.isDeleted())
                        {
                        System.out.printf("deleted key: " + record.getKey() +
"\n");
                        }
                     else
                        {
                        System.out.printf("modified entry, key: " +
record.getKey() + ", value: " +
                           record.getModifiedEntry().getValue());
                        }
                     }
                  }
            break;
         default:
            throw new IllegalStateException("Expected event of type " +
FederatedChangeEvent.Type.REPLICATING
               + ", but got event of type: " + event.getType());
         }
```

```
        }
    }

    public long getMessageCount()
        {
        return m_cEvents;
        }

    private volatile long m_cEvents;
}
```

2. Configure a custom participant in the operational configuration file using `interceptor` as the participant type and register the interceptor class using the `<interceptor>` element. For example:

```
<participant>
   <name>ForLogging</name>
   <send-timeout>5s</send-timeout>
   <participant-type>interceptor</participant-type>
   <interceptors>
      <interceptor>
         <name>MyInterceptor</name>
         <instance>
            <class-name>example.MyInterceptorImplChangeEvents</class-name>
         </instance>
      </interceptor>
   </interceptors>
</participant>
```

> **Note:**
>
> You can either register the interceptor class in the participant configuration (as shown) or in a federated cache schema. If you register the interceptor class in the participant configuration, then it applies to all the federated cache services that use the participant. Specify the interceptor in a federated cache scheme if you want to control which services use the interceptor. See federated-scheme in *Developing Applications with Oracle Coherence*.

3. Include the custom participant as part of the federation topology for which you want to federate events. For example:

```
<topology-definitions>
   <active-active>
      <name>Active</name>
      <active>BOSTON</active>
      <active>NEWYORK</active>
      <interceptor>ForLogging</interceptor>
   </active-active>
</topology-definitions>
```

4. Ensure the interceptor implementation is found on the classpath at runtime.

Using a Specific Network Interface for Federation Communication

Federation communication can be configured to use a network interface that is different than the interface used for cluster communication.
To use a different network configuration for federation communication:

1. Edit the operational override file on each cluster participant and include an `<address-provider>` element that defines a `NameService` address on a separate IP address and port that is bound to the desired network interface. For example

```
<cluster-config>
   <address-providers>
      <address-provider id="NameServiceAddress">
         <socket-address>
            <address system-property="coherence.nameservice.ip">
               192.168.1.5</address>
            <port system-property="coherence.nameservice.port">
               10100</port>
         </socket-address>
      </address-provider>
   </address-providers>
</cluster-config>
```

2. Modify the participant definition to use the remote address. For example:

```
<federation-config>
   <participants>
      <participant>
         <name>LocalClusterA</name>
         <remote-addresses>
            <address-provider>NameServiceAddress</address-provider>
         </remote-addresses>
      </participant>
      ...
```

3. When starting cluster members (for the LocalClusterA participant in the above example), use the `coherence.nameservice.addressprovider` system property and reference the address provider definition for the name service. For example:

```
-Dcoherence.nameservice.addressprovider=NameServiceAddress
```

Load Balancing Federated Connections

Connections between federated service members are load balanced. By default, a federation-based strategy is used that distributes connections to federated service members that are being utilized the least. Custom strategies can be created or the default strategy can be modified as required. As an alternative, a client-based load balance strategy can be implemented by creating an address provider implementation or by relying on randomized connections to federated service members. The random approach provides minimal balancing as compared to federated-based load balancing.
Connections between federated service members are distributed equally across federated service members based upon existing connection count and incoming message backlog. Typically, this algorithm provides the best load balancing strategy. However, you can choose to implement a different load balancing strategy as required.

This section includes the following topics:

- Using Federation-Based Load Balancing
- Implementing a Custom Federation-Based Load Balancing Strategy
- Using Client-Based Load Balancing

Using Federation-Based Load Balancing

federation-based load balancing is the default strategy that is used to balance connections between two or more members of the same federation service. The strategy distribute connections equally across federated service members based upon existing connection count and incoming message backlog.

The federation-based load balancing strategy is configured within a `<federated-scheme>` definition using a `<load-balancer>` element that is set to `federation`. For clarity, the following example explicitly specifies the strategy. However, the strategy is used by default if no strategy is specified and is not required in a federated scheme definition.

```
<federated-scheme>
   <scheme-name>federated</scheme-name>
   <service-name>federated</service-name>
   <backing-map-scheme>
      <local-scheme />
   </backing-map-scheme>
   <autostart>true</autostart>
   <load-balancer>federation</load-balancer>
   <topologies>
      <topology>
         <name>MyTopology</name>
      </topology>
   </topologies>
</federated-scheme>
```

Implementing a Custom Federation-Based Load Balancing Strategy

The `com.tangosol.coherence.net.federation` package includes the APIs that are used to balance client load across federated service members.

A custom strategy must implement the `FederatedServiceLoadBalancer` interface. New strategies can be created or the default strategy (`DefaultFederatedServiceLoadBalancer`) can be extended and modified as required.

To enable a custom federation-based load balancing strategy, edit a federated scheme and include an `<instance>` subelement within the `<load-balancer>` element and provide the fully qualified name of a class that implements the `FederatedServiceLoadBalancer` interface. The following example enables a custom federation-based load balancing strategy that is implemented in the `MyFederationServiceLoadBalancer` class:

```
...
<load-balancer>
   <instance>
      <class-name>package.MyFederationServiceLoadBalancer</class-name>
   </instance>
</load-balancer>
...
```

In addition, the `<instance>` element also supports the use of a `<class-factory-name>` element to use a factory class that is responsible for creating `FederatedServiceLoadBalancer` instances, and a `<method-name>` element to specify the static factory method on the factory class that performs object instantiation. See instance in *Developing Applications with Oracle Coherence*.

Using Client-Based Load Balancing

The client-based load balancing strategy relies upon a `com.tangosol.net.AddressProvider` implementation to dictate the distribution of connections across federated service members. If no address provider implementation is provided, each configured cluster participant member is tried in a random order until a connection is successful. See address-provider in *Developing Applications with Oracle Coherence*.

The client-based load balancing strategy is configured within a `<federated-scheme>` definition using a `<load-balancer>` element that is set to `client`. For example:

```
<federated-scheme>
   <scheme-name>federated</scheme-name>
   <service-name>federated</service-name>
   <backing-map-scheme>
      <local-scheme />
   </backing-map-scheme>
   <autostart>true</autostart>
   <load-balancer>client</load-balancer>
   <topologies>
      <topology>
         <name>MyTopology</name>
      </topology>
   </topologies>
</federated-scheme>
```

Managing Federated Caching

Federated caching should be managed on each cluster participant in the same manner as any non-federated cluster and distributed cache to ensure optimal performance and resource usage. A poorly performing cluster is likely to cause performance issues when it is included as part of a federation. In addition, federated caching should also be managed to ensure efficient federation performance and throughput among cluster participants in the federation. Monitoring federation performance is especially important due to the possible issues that are inherent in wide area network topologies.
The following tools are available to manage and monitor federation:

- Coherence VisualVM Plug-in: See Using the Coherence VisualVM Plug-In in *Managing Oracle Coherence*.
- Coherence Reports: See Understanding the Federation Destination Report in *Managing Oracle Coherence*.
- Coherence MBeans: See Destination MBean in *Managing Oracle Coherence*.
- Coherence Command Line Interface: See Coherence CLI.
- Coherence Grafana Dashboards: See Visualizing Metrics in Grafana in *Managing Oracle Coherence*.

This section includes the following topics:

- Monitoring the Cluster Participant Status

- Monitor Federation Performance and Throughput

Monitoring the Cluster Participant Status

Monitor the status of each cluster participant in the federation to ensure that there are no issues.

Coherence VisualVM Plug-in

Use the Federation tab in the Coherence VisualVM plug-in to view the status of each cluster participant from the context of the local cluster participant. That is, each destination cluster participant is listed and its status is shown. In addition, the federation state of each node in the local cluster participant is reported in the Outbound tab. Check the Error Description field to view an error message, if the status of cluster participant is `Error`.

Coherence Reports

Use the federation destination report (`federation-destination.txt`) to view the status of each destination cluster participant and the federation state of each node over time.

Coherence MBeans

Use the federation attributes on the `DestinationMBean` MBean to view the status of each destination cluster participants and the federation state of each node of the local cluster participant.

Coherence Command Line Interface

Use the Coherence Command Line Interface (CLI) to monitor and manage Coherence federation clusters from a terminal based interface. For more information on federation commands, see Federation. For more information about CLI, see coherence-cli.

Grafana Dashboards

In Grafana, navigate to the Coherence Federation Summary Dashboard and view the **Status** column for each Service and Participant. If this value is either a **WARNING** or an **ERROR**, you should investigate the Coherence log file in the sending nodes to determine the reason for the status.

Monitor Federation Performance and Throughput

Monitor federation performance and throughput to ensure that the local cluster participant is federating data to each participant without any substantial delays or lost data. Issues with performance and throughput can be a sign that there is a problem with the network connect between cluster participants or that there is a problem on the local cluster participant.

Coherence VisualVM Plug-in

Use the Federation tab in the Coherence VisualVM plug-in to view the current federation performance statistics and throughput from the local participant to each destination cluster participant. Select a destination cluster participant and view its federation performance statistics, then view the Current Throughput column on the Outbound tab to see the throughput to the selected participant from each node in the

local cluster. Select an individual node in the Outbound tab to see its bandwidth utilization and federation performance in the graph tabs, respectively. Lastly, select the Inbound tab to view how efficiently the local cluster participant is receiving data from destination cluster participants.

Coherence Reports

Use the federation destination report (`federation-destination.txt`) and the federation origin report (`federation-origin.txt`) to view federation performance statistics. The destination report shows how efficiently each node in the local cluster participant is sending data to each destination cluster participant. The federation origin reports shows how efficiently each node in the local cluster participant is receiving data from destination cluster participants.

Coherence MBeans

Use the persistence attributes on the `DestinationMBean` MBean and the `OriginMBean` MBean to view federation performance statistics. The `DestinationMBean` MBean attribute shows how efficiently each node in the local cluster participant is sending data to each destination cluster participant. The `OriginMBean` MBean shows how efficiently the local cluster participant is receiving data from destination cluster participants.

Grafana Dashboards

In Grafana, navigate to the Coherence Federation Summary Dashboard and select the service and destination to view detailed information about a participant. In the Coherence Federation Details Dashboard, there are various metrics to view and determine the health of a participant. These metrics include the following:

- The destinations table shows the metrics for each sending node for that destination as well as the status of the connection. For a detailed information about each status value, see Destination MBean in *Managing Oracle Coherence*.
- The current envelope size indicates the number of messages waiting to be sent to the destination for all nodes.
- The current RAM journal and flash journal that are in use.
- The average apply time, roundtrip delay, and backlog delay for each node.

> **Note:**
>
> The current envelope size will never be zero because there is at least one entry per partition. You should monitor this value. If the value is trending upwards continuously over time (along with the RAM or flash journal), there may be issues with the ability of the members to send data to the destination participant. These values could trend upwards due to a replicate-all operation but should return to normal levels after the operation completes. If they do not, further investigation of the health of the destination participants should be carried out to determine the root cause.

A

Platform-Specific Deployment Considerations

Coherence can be deployed to many different platforms, each of which may require specific deployment considerations. This appendix identifies issues that should be considered when deploying Coherence to various platforms and offers solutions if available.
This appendix includes the following sections:

Deploying to Oracle HotSpot JVMs

Issues to be aware of when deploying Coherence on Oracle HotSpot JVMs

Heap Sizes

Coherence recommends keeping heap sizes at 1-8GB per JVM. However, larger heap sizes, up to 20GB, are suitable for some applications where the simplified management of fewer, larger JVMs outweighs the performance benefits of many smaller JVMs. Using multiple cache servers allows a single computer to achieve higher capacities. With Oracle's JVMs, heap sizes beyond 8GB are reasonable, though GC tuning is still advisable to minimize long GC pauses. See Introduction in *Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide*. It is also advisable to run with fixed sized heaps as this generally lowers GC times. See JVM Tuning.

AtomicLong

When available Coherence uses the highly concurrent AtomicLong class, which allows concurrent atomic updates to long values without requiring synchronization.

It is suggested to run in server mode to ensure that the stable and highly concurrent version can be used. To run the JVM in server mode include the -server option on the Java command line.

OutOfMemoryError

JVMs that experience an `OutOfMemoryError` can be left in an indeterministic state which can have adverse effects on a cluster. We recommend configuring JVMs to exit upon encountering an `OutOfMemoryError` instead of allowing the JVM to attempt recovery. Here is the parameter to configure this setting on Sun JVMs:

UNIX:

```
-XX:OnOutOfMemoryError="kill -9 %p"
```

Windows:

```
-XX:OnOutOfMemoryError="taskkill /F /PID %p"
```

Additionally, it is recommended to configure the JVM to generate a heap dump if an `OutOfMemoryError` is thrown to assist the investigation into the root cause for the error. Use the following flag to enable this feature on the Sun JVM:

```
-XX:+HeapDumpOnOutOfMemoryError
```

Deploying to IBM JVMs

Issues to be aware of when deploying Coherence on IBM JVMs

OutOfMemoryError

JVMs that experience an `OutOfMemoryError` can be left in an indeterministic state which can have adverse effects on a cluster. We recommend configuring JVMs to exit upon encountering an `OutOfMemoryError` instead of allowing the JVM to attempt recovery. Here is the parameter to configure this setting on IBM JVMs:

UNIX:

```
-Xdump:tool:events=throw,filter=java/lang/OutOfMemoryError,exec="kill -9 %pid"
```

Windows:

```
-Xdump:tool:events=throw,filter=java/lang/
OutOfMemoryError,exec="taskkill /F /PID %pid"
```

Heap Sizing

IBM does not recommend fixed size heaps for JVMs. In many cases, it is recommended to use the default for `-Xms` (in other words, omit this setting and only set `-Xmx`). See this link for more details:

http://www.ibm.com/developerworks/java/jdk/diagnosis/

It is recommended to configure the JVM to generate a heap dump if an `OutOfMemoryError` is thrown to assist the investigation into the root cause for the error. IBM JVMs generate a heap dump on `OutOfMemoryError` by default; no further configuration is required.

Deploying to Linux

Issues to be aware of when deploying Coherence on Linux

- TSC High Resolution Timesource

TSC High Resolution Timesource

Linux has several high resolution timesources to choose from, the fastest TSC (Time Stamp Counter) unfortunately is not always reliable. Linux chooses TSC by default, and during startup checks for inconsistencies, if found it switches to a slower safe timesource. The slower time sources can be 10 to 30 times more expensive to query then the TSC timesource, and may have a measurable impact on Coherence performance. For more details on TSC, see

https://lwn.net/Articles/209101/

Coherence and the underlying JVM are not aware of the timesource which the operating system is using. It is suggested that you check your system logs (/var/log/dmesg) to verify that the following is not present.

```
kernel: Losing too many ticks!
kernel: TSC cannot be used as a timesource.
kernel: Possible reasons for this are:
kernel:   You're running with Speedstep,
kernel:   You don't have DMA enabled for your hard disk (see hdparm),
kernel:   Incorrect TSC synchronization on an SMP system (see dmesg).
kernel: Falling back to a sane timesource now.
```

As the log messages suggest, this can be caused by a variable rate CPU (SpeedStep), having DMA disabled, or incorrect TSC synchronization on multi CPU computers. If present, it is suggested that you work with your system administrator to identify the cause and allow the TSC timesource to be used.

Deploying to Solaris

Issues to be aware of when deploying Coherence on Solaris

- Solaris 10 (x86 and SPARC)
- Solaris 10 Networking
- Solaris Network Interface Cards

Solaris 10 (x86 and SPARC)

When running on Solaris 10, there are known issues relate to packet corruption and multicast disconnections. These most often manifest as either EOFExceptions, "Large gap" warnings while reading packet data, or frequent packet timeouts. It is highly recommend that the patches for both issues below be applied when using Coherence on Solaris 10 systems.

Possible Data Integrity Issues on Solaris 10 Systems Using the e1000g Driver for the Intel Gigabit Network Interface Card (NIC)

https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=ALERT&id=1000972.1

IGMP(1) Packets do not Contain IP Router Alert Option When Sent From Solaris 10 Systems With Patch 118822-21 (SPARC) or 118844-21 (x86/x64) or Later Installed

https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=ALERT&id=1000940.1

Solaris 10 Networking

If running on Solaris 10, review the above Solaris 10 (x86 and SPARC) issues which relate to packet corruption and multicast disconnections. These most often manifest as either EOFExceptions, "Large gap" warnings while reading packet data, or frequent packet timeouts. It is highly recommend that the patches for both issues be applied when using Coherence on Solaris 10 systems.

Solaris Network Interface Cards

Solaris M series systems include an on-board NIC (`bge`) and PCI connected NIC (`nxge`). The on-board gigabit ethernet ports are used for low-bandwidth administrative networking connectivity for the domain and are not intended for high-performance cluster interconnect workloads with high traffic demands. Coherence cluster members must always use the dedicated PCIe NICs for high bandwidth cluster interconnects.

Solaris Link Aggregation

Solaris 11 supports two types of NIC link aggregation: trunk aggregation and Datalink multipathing (DLMP) aggregations. Trunk aggregation requires the use of a network switch, which must support the Link Aggregation Control Protocol (LCAP). DLMP aggregation requires the use of at least one network switch. However, when using DLMP aggregations, make sure any switches are not configured to use trunk aggregation with LCAP. If you change from a trunk aggregation to a DLMP aggregation, you must remove the switch configuration that was previously created for the trunk aggregation. Failure to do so, can result in packet loss and under utilization of the network bandwidth.

Deploying to Windows

Issues to be aware of when deploying Coherence on Windows

Performance Tuning

The default Windows configuration is not optimized for background processes, heavy network loads, and network interruptions. This may be addressed by running the `optimize.reg` script included in the Coherence installation's `bin` directory. See Operating System Tuning.

Personal Firewalls

If running a firewall on a computer you may have difficulties in forming a cluster consisting of multiple computers. This can be resolved by either:

- Disabling the firewall, though this is generally not recommended.
- Granting full network access to the Java executable which runs Coherence.
- Opening up individual address and ports for Coherence. See Configuring Firewalls for Cluster Members in *Developing Applications with Oracle Coherence*.

Disconnected Network Interface

On Microsoft Windows, if the Network Interface Card (NIC) is unplugged from the network, the operating system invalidates the associated IP address. The effect of this is that any socket which is bound to that IP address enters an error state. This results in the Coherence nodes exiting the cluster and residing in an error state until the NIC is reattached to the network. In cases where it is desirable to allow multiple collocated JVMs to remain clustered during a physical outage Windows must be configured to not invalidate the IP address.

To adjust this parameter:

1. Run Registry Editor (`regedit`)
2. Locate the following registry key

   ```
   HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
   ```

3. Add or reset the following new DWORD value

   ```
   Name: DisableDHCPMediaSense
   Value: 1 (boolean)
   ```

4. Reboot

While the name of the keyword includes DHCP, the setting effects both static and dynamic IP addresses. See Microsoft Windows TCP/IP Implementation Details for additional information:

`http://technet.microsoft.com/en-us/library/bb726981.aspx#EDAA`

Deploying to OS X

Issues to be aware of when deploying Coherence on OS X

Multicast and IPv6

OS X defaults to running multicast over IPv6 rather than IPv4. If you run in a mixed IPv6/IPv4 environment, configure your JVMs to explicitly use IPv4. This can be done by setting the `java.net.preferIPv4Stack` system property to `true` on the Java command line.

Socket Buffer Sizing

Generally, Coherence prefers 2MB or higher buffers, but for OS X this may result in unexpectedly high kernel CPU time, which in turn reduces throughput. For OS X, the suggested buffers size is 768KB, though your own tuning may find a better size. See Configuring the Size of the Packet Buffers in *Developing Applications with Oracle Coherence*.

Deploying to z/OS

Issues to be aware of when deploying Coherence on z/OS

EBCDIC

When deploying Coherence into environments where the default character set is EBCDIC rather than ASCII, ensure that Coherence configuration files which are loaded from JAR files or off of the classpath are in ASCII format. Configuration files loaded directly from the file system should be stored in the systems native format of EBCDIC.

Multicast

Under some circumstances, Coherence cluster nodes that run within the same logical partition (LPAR) on z/OS on IBM zSeries cannot communicate with each other. (This problem does not occur on the zSeries when running on Linux.)

The root cause is that z/OS may bind the MulticastSocket that Coherence uses to an automatically-assigned port, but Coherence requires the use of a specific port in order for cluster discovery to operate correctly. (Coherence does explicitly initialize the `java.net.MulitcastSocket` to use the necessary port, but that information appears to be ignored on z/OS when there is an instance of Coherence running within that same LPAR.)

The solution is to run only one instance of Coherence within a z/OS LPAR; if multiple instances are required, each instance of Coherence should be run in a separate z/OS LPAR. Alternatively, well known addresses may be used. See Using Well Known Addresses in *Developing Applications with Oracle Coherence*.

Deploying to AIX

Issues to be aware of when deploying Coherence on AIX

Multicast and IPv6

AIX 5.2 and above default to running multicast over IPv6 rather than IPv4. If you run in a mixed IPv6/IPv4 environment, configure your JVMs to explicitly use IPv4. This can be done by setting the `java.net.preferIPv4Stack` system property to `true` on the Java command line. See the IBM 32-bit SDK for AIX User Guide for details.

Deploying to Virtual Machines

Issues to be aware of when deploying Coherence on virtual machines
Oracle Coherence follows the support policies of Oracle Fusion Middleware. See Supported Virtualization and Partitioning Technologies for Oracle Fusion Middleware.

Multicast Connectivity

Virtualization adds another layer to your network topology and it must be properly configured to support multicast networking. See Configuring Multicast Communication in *Developing Applications with Oracle Coherence*.

Performance

It is less likely that a process running in a virtualized operating system can fully use gigabit Ethernet. This is not specific to Coherence and is visible on most network intensive virtualized applications.

Fault Tolerance

Additional configuration is required to ensure that cache entry backups reside on physically separate hardware. See Specifying a Cluster Member's Identity in *Developing Applications with Oracle Coherence*.

Deploying to Cisco Switches

Issues to be aware of when deploying Coherence with Cisco switches

Buffer Space and Packet Pauses

Some Cisco switches may run out of buffer space and exhibit frequent multi-second communication pauses under heavy packet load some. These communication pauses can be identified by a series of Coherence log messages referencing communication delays with multiple nodes which cannot be attributed to local or remote GCs.

```
Experienced a 4172 ms communication delay (probable remote GC) with Member(Id=7,
Timestamp=2008-09-15 12:15:47.511, Address=xxx.xxx.x.xx:8089, MachineId=13838);
320 packets rescheduled, PauseRate=0.31, Threshold=512
```

The Cisco 6500 series support configuring the amount of buffer space available to each Ethernet port or ASIC. In high load applications it may be necessary to increase the default buffer space. This can be accomplished by executing:

```
fabric buffer-reserve high
```

See Cisco's documentation for additional details on this setting.

Multicast Connectivity on Large Networks

Cisco's default switch configuration does not support proper routing of multicast packets between switches due to the use of IGMP snooping. See the Cisco's documentation regarding the issue and solutions.

Multicast Outages

Some Cisco switches have shown difficulty in maintaining multicast group membership resulting in existing multicast group members being silently removed from the multicast group. This cause a partial communication disconnect for the associated Coherence node(s) and they are forced to leave and rejoin the cluster. This type of outage can most often be identified by the following Coherence log messages indicating that a partial communication problem has been detected.

```
A potential network configuration problem has been detected. A packet has failed
to be delivered (or acknowledged) after 60 seconds, although other packets were
acknowledged by the same cluster member (Member(Id=3, Timestamp=Sat Sept 13
12:02:54 EST 2008, Address=192.168.1.101, Port=8088, MachineId=48991)) to this
member (Member(Id=1, Timestamp=Sat Sept 13 11:51:11 EST 2008,
Address=192.168.1.101, Port=8088, MachineId=49002)) as recently as 5 seconds ago.
```

To confirm the issue, use the same multicast address and port as the running cluster. If the issue affects a multicast test node, its logs show that it suddenly stopped receiving multicast test messages. See Performing a Multicast Connectivity Test.

The following test logs show the issue:

Example A-1 Log for a Multicast Outage

```
Test Node 192.168.1.100:

Sun Sept 14 16:44:22 GMT 2008: Received 83 bytes from a Coherence cluster node
at 182.168.1.100: ???
Sun Sept 14 16:44:23 GMT 2008: Received test packet 76 from ip=/192.168.1.101,
group=/224.3.2.0:32367, ttl=4.
Sun Sept 14 16:44:23 GMT 2008: Received 83 bytes from a Coherence cluster node
at 182.168.1.100: ???
```

```
Sun Sept 14 16:44:23 GMT 2008: Sent packet 85. Sun Sept 14 16:44:23 GMT 2008: Received
test packet 85 from self.
Sun Sept 14 16:44:24 GMT 2008: Received 83 bytes from a Coherence cluster node at
182.168.1.100: ???
Sun Sept 14 16:44:25 GMT 2008: Received test packet 77 from ip=/192.168.1.101, group=/
224.3.2.0:32367, ttl=4.
Sun Sept 14 16:44:25 GMT 2008: Received 83 bytes from a Coherence cluster node at
182.168.1.100: ???
Sun Sept 14 16:44:25 GMT 2008: Sent packet 86.
Sun Sept 14 16:44:25 GMT 2008: Received test packet 86 from self. Sun Sept 14 16:44:26
GMT 2008: Received 83 bytes from a Coherence cluster node at 182.168.1.100: ???
Sun Sept 14 16:44:27 GMT 2008: Received test packet 78 from ip=/192.168.1.101, group=/
224.3.2.0:32367, ttl=4.
Sun Sept 14 16:44:27 GMT 2008: Received 83 bytes from a Coherence cluster node at
182.168.1.100: ???
Sun Sept 14 16:44:27 GMT 2008: Sent packet 87.
Sun Sept 14 16:44:27 GMT 2008: Received test packet 87 from self.
Sun Sept 14 16:44:28 GMT 2008: Received 83 bytes from a Coherence cluster node at
182.168.1.100: ???
Sun Sept 14 16:44:29 GMT 2008: Received 83 bytes from a Coherence cluster node at
182.168.1.100: ???
Sun Sept 14 16:44:29 GMT 2008: Sent packet 88.
Sun Sept 14 16:44:29 GMT 2008: Received test packet 88 from self.
 Sun Sept 14 16:44:30 GMT 2008: Received 83 bytes from a Coherence cluster node at
182.168.1.100: ???
Sun Sept 14 16:44:31 GMT 2008: Received 83 bytes from a Coherence cluster node at
182.168.1.100: ???
Sun Sept 14 16:44:31 GMT 2008: Sent packet 89.
Sun Sept 14 16:44:31 GMT 2008: Received test packet 89 from self.
Sun Sept 14 16:44:32 GMT 2008: Received 83 bytes from a Coherence cluster node at
182.168.1.100: ???
Sun Sept 14 16:44:33 GMT 2008: Received 83 bytes from a Coherence cluster node at
182.168.1.100: ???

Test Node 192.168.1.101:

Sun Sept 14 16:44:22 GMT 2008: Sent packet 76.
Sun Sept 14 16:44:22 GMT 2008: Received test packet 76 from self.
Sun Sept 14 16:44:22 GMT 2008: Received 83 bytes from a Coherence cluster node at
192.168.1.100: ???
Sun Sept 14 16:44:22 GMT 2008: Received test packet 85 from ip=/192.168.1.100, group=/
224.3.2.0:32367, ttl=4.
Sun Sept 14 16:44:23 GMT 2008: Received 83 bytes from a Coherence cluster node at
192.168.1.100: ??? Sun Sept 14 16:44:24 GMT 2008: Sent packet 77.
Sun Sept 14 16:44:24 GMT 2008: Received test packet 77 from self.
Sun Sept 14 16:44:24 GMT 2008: Received 83 bytes from a Coherence cluster node at
192.168.1.100: ???
Sun Sept 14 16:44:24 GMT 2008: Received test packet 86 from ip=/192.168.1.100, group=/
224.3.2.0:32367, ttl=4.
Sun Sept 14 16:44:25 GMT 2008: Received 83 bytes from a Coherence cluster node at
192.168.1.100: ???
Sun Sept 14 16:44:26 GMT 2008: Sent packet 78.Sun Sept 14 16:44:26 GMT 2008: Received
test packet 78 from self.
Sun Sept 14 16:44:26 GMT 2008: Received 83 bytes from a Coherence cluster node at
192.168.1.100: ???
Sun Sept 14 16:44:26 GMT 2008: Received test packet 87 from ip=/192.168.1.100, group=/
224.3.2.0:32367, ttl=4.
Sun Sept 14 16:44:27 GMT 2008: Received 83 bytes from a Coherence cluster node at
192.168.1.100: ???
Sun Sept 14 16:44:28 GMT 2008: Sent packet 79.
Sun Sept 14 16:44:28 GMT 2008: Received test packet 79 from self.
```

```
Sun Sept 14 16:44:28 GMT 2008: Received 83 bytes from a Coherence cluster node
at 192.168.1.100: ???
Sun Sept 14 16:44:28 GMT 2008: Received test packet 88 from ip=/192.168.1.100,
group=/224.3.2.0:32367, ttl=4.
Sun Sept 14 16:44:29 GMT 2008: Received 83 bytes from a Coherence cluster node
at 192.168.1.100: ???
Sun Sept 14 16:44:30 GMT 2008: Sent packet 80.
Sun Sept 14 16:44:30 GMT 2008: Received test packet 80 from self.
Sun Sept 14 16:44:30 GMT 2008: Received 83 bytes from a Coherence cluster node
at 192.168.1.100: ???
Sun Sept 14 16:44:30 GMT 2008: Received test packet 89 from ip=/192.168.1.100,
group=/224.3.2.0:32367, ttl=4.
Sun Sept 14 16:44:31 GMT 2008: Received 83 bytes from a Coherence cluster node
at 192.168.1.100: ???
Sun Sept 14 16:44:32 GMT 2008: Sent packet 81.Sun Sept 14 16:44:32 GMT 2008:
Received test packet 81 from self.
Sun Sept 14 16:44:32 GMT 2008: Received 83 bytes from a Coherence cluster node
at 192.168.1.100: ???
Sun Sept 14 16:44:32 GMT 2008: Received test packet 90 from ip=/192.168.1.100,
group=/224.3.2.0:32367, ttl=4.
Sun Sept 14 16:44:33 GMT 2008: Received 83 bytes from a Coherence cluster node
at 192.168.1.100: ???
Sun Sept 14 16:44:34 GMT 2008: Sent packet 82.
```

Note that at 16:44:27 the first test node stops receiving multicast packets from other computers. The operating system continues to properly forward multicast traffic from other processes on the same computer, but the test packets (79 and higher) from the second test node are not received. Also note that both the test packets and the cluster's multicast traffic generated by the first node do continue to be delivered to the second node. This indicates that the first node was silently removed from the multicast group.

If you encounter this multicast issue it is suggested that you contact Cisco technical support, or you may consider changing your configuration to unicast-only by using the well known addresses. See Using Well Known Addresses in *Developing Applications with Oracle Coherence*.

Multicast Time-to-Live

The Cisco 6500 series router may become overloaded if too many packets with a time-to-live (TTL) value of 1 are received. In addition, a low TTL setting may overload single group members. Set the Coherence multicast TTL setting to at least the size of the multicast domain (127 or 255) and make sure that clusters do not use overlapping groups. See Specifying the Multicast Time-to-Live in *Developing Applications with Oracle Coherence*.

Deploying to Foundry Switches

Issues to be aware of when deploying Coherence with Foundry switches

Multicast Connectivity

Foundry switches have shown to exhibit difficulty in handing multicast traffic. When deploying on with Foundry switches, ensure that all computers that are part of the

Coherence cluster can communicate over multicast. See Performing a Multicast Connectivity Test.

If you encounter issues with multicast you may consider changing your configuration to unicast-only by using the well-known-addresses feature. See Using Well Known Addresses in *Developing Applications with Oracle Coherence*.

Deploying to IBM BladeCenters

Issues to be aware of when deploying Coherence on IBM BladeCenters

- MAC Address Uniformity and Load Balancing

MAC Address Uniformity and Load Balancing

A typical deployment on a BladeCenter may include blades with two NICs where one is used for administration purposes and the other for cluster traffic. By default, the MAC addresses assigned to the blades of a BladeCenter are uniform enough that the first NIC generally has an even MAC address and the second has an odd MAC address. If the BladeCenter's uplink to a central switch also has an even number of channels, then layer 2 (MAC based) load balancing may prevent one set of NICs from making full use of the available uplink bandwidth as they are all bound to either even or odd channels. This issue arises due to the assumption in the switch that MAC addresses are essentially random, which in BladeCenter's is untrue. Remedies to this situation include:

- Use layer 3 (IP based) load balancing (if the IP addresses do not follow the same even/odd pattern).
  - This setting must be applied across all switches carrying cluster traffic.
- Randomize the MAC address assignments by swapping them between the first and second NIC on alternating computers.
  - Linux enables you to change a NIC's MAC address using the `ifconfig` command.
  - For other operating systems custom tools may be available to perform the same task.

B

Log Message Glossary

Coherence emits many log messages that identify important information, which includes potential issues. This appendix provides a reference to common Coherence log messages and includes the cause of the message and possible actions to take.
This appendix includes the following sections:

- TCMP Log Messages
- Configuration Log Messages
- Partitioned Cache Service Log Messages
- Service Thread Pool Log Messages
  Log messages that pertain to the service thread pool
- TMB Log Messages
  Log messages that pertain to TMB.
- Cluster Service Exceptions
  Exceptions thrown by the cluster service.
- Guardian Service Log Messages
  Log messages that pertain to the Guardian Service.

TCMP Log Messages

Log messages that pertain to TCMP

Experienced a %n1 ms communication delay (probable remote GC) with Member %s
%n1 - the latency in milliseconds of the communication delay; %s - the full Member information. Severity: 2-Warning or 5-Debug Level 5 or 6-Debug Level 6 depending on the length of the delay.

Cause: This node detected a delay in receiving acknowledgment packets from the specified node, and has determined that is it likely due to a remote GC (rather than a local GC). This message indicates that the overdue acknowledgment has been received from the specified node, and that it has likely emerged from its GC. Any slowdown in the network or the remote server can trigger this, but the most common cause is GC, which should be investigated first.

Action: Prolonged and frequent garbage collection can adversely affect cluster performance and availability. If these warnings are seen frequently, review your JVM heap and GC configuration and tuning. See Performance Tuning.

Failed to satisfy the variance: allowed=%n1 actual=%n2
%n1 - the maximum allowed latency in milliseconds; %n2 - the actual latency in milliseconds. Severity: 3-Informational or 5-Debug Level 5 depending on the message frequency.

Cause: One of the first steps in the Coherence cluster discovery protocol is the calculation of the clock difference between the new and the senior nodes. This step assumes a relatively small latency for peer-to-peer round trip communication between the nodes. By default, the configured maximum allowed latency (the value of the `<maximum-time-variance>`

configuration element) is 16 milliseconds. See incoming-message-handler in *Developing Applications with Oracle Coherence*. Failure to satisfy that latency causes this message to be logged and increases the latency threshold, which is reflected in a follow up message.

Action: If the latency consistently stays very high (over 100 milliseconds), consult your network administrator and see Performing a Network Performance Test.

Created a new cluster "%s1" with Member(%s2)
%s1 - the cluster name; %s2 - the full Member information. Severity: 3-Informational.

Cause: This Coherence node attempted to join an existing cluster in the configured amount of time (specified by the `<join-timeout-milliseconds>` element), but did not receive any responses from any other node. As a result, it created a new cluster with the specified name (either configured by the `<cluster-name>` element or calculated based on the multicast listener address and port, or the well known addresses list). The Member information includes the node id, creation timestamp, unicast address and port, location, process id, role, and so on.

Action: None, if this node is expected to be the first node in the cluster. Otherwise, the operational configuration has to be reviewed to determine the reason that this node does not join the existing cluster.

This Member(%s1) joined cluster "%s2" with senior Member(%s3)
%s1 - the full Member information for this node; %s2 - the cluster name; %s3 - the full Member information for the cluster senior node. Severity: 3-Informational.

Cause: This Coherence node has joined an existing cluster.

Action: None, if this node is expected to join an existing cluster. Otherwise, identify the running cluster and consider corrective actions.

Member(%s) joined Cluster with senior member %n
%s - the full Member information for a new node that joined the cluster this node belongs to; %n - the node id of the cluster senior node. Severity: 5-Debug Level 5.

Cause: A new node has joined an existing Coherence cluster.

Action: None.

there appears to be other members of the cluster "%s" already running with an incompatible network configuration, aborting join with "%n"
%s - the cluster name to which a join attempt was made; %n - information for the cluster. Severity: 5-Debug Level 5.

Cause:The joining member has a network configuration that conflicts with existing members of the cluster.

Action:The operational configuration has to be reviewed to determine the reason that this node does not join the existing cluster.

Member(%s) left Cluster with senior member %n
%s - the full Member information for a node that left the cluster; %n - the node id of the cluster senior node. Severity: 5-Debug Level 5.

Cause: A node has left the cluster. This departure could be caused by the programmatic shutdown, process termination (normal or abnormal), or any other communication failure (for example, a network disconnect or a very long GC pause). This message reports the node's departure.

Action: None, if the node departure was intentional. Otherwise, the departed node logs should be analyzed.

MemberLeft notification for Member %n received from Member(%s)
%n - the node id of the departed node; %s - the full Member information for a node that left the cluster. Severity: 5-Debug Level 5.

Cause: When a Coherence node terminates, this departure is detected by some nodes earlier than others. Most commonly, a node connected through the TCP ring connection ("TCP ring buddy") would be the first to detect it. This message provides the information about the node that detected the departure first.

Action: None, if the node departure was intentional. Otherwise, the logs for both the departed and the detecting nodes should be analyzed.

Received cluster heartbeat from the senior %n that does not contain this %s ; stopping cluster service.
%n - the senior service member id; %s - a cluster service member's id. Severity: 1-Error.

Cause: A heartbeat is broadcast from the senior cluster service member that contains a cluster member set. If this cluster service member is not part of the broadcast set, then it is assumed that the senior member believes this service member to be dead and the cluster service is stopped on the member. This typically occurs if a member lost communication with the cluster for an extended period of time (possibly due to network issues or extended garbage collection) and was ejected from the cluster.

Action: Corrective action is not necessarily required, since the rest of the cluster presumably is continuing its operation. However, it may warrant an investigation into root causes of the problem (especially if it is recurring with some frequency).

Service %s joined the cluster with senior service member %n
%s - the service name; %n - the senior service member id. Severity: 5-Debug Level 5.

Cause: When a clustered service starts on a given node, Coherence initiates a handshake protocol between all cluster nodes running the specified service. This message serves as an indication that this protocol has been initiated. If the senior node is not currently known, it is shown as "n/a".

Action: None.

This node appears to have partially lost the connectivity: it receives responses from MemberSet(%s1) which communicate with Member(%s2), but is not responding directly to this member; that could mean that either requests are not coming out or responses are not coming in; stopping cluster service.
%s1 - set of members that can communicate with the member indicated in %s2; %s2 - member that can communicate with set of members indicated in %s1. Severity: 1-Error.

Cause: The communication link between this member and the member indicated by %s2 has been broken. However, the set of witnesses indicated by %s1 report no communication issues with %s2. It is therefore assumed that this node is in a state of partial failure, thus resulting in the shutdown of its cluster threads.

Action: Corrective action is not necessarily required, since the rest of the cluster presumably is continuing its operation and this node may recover and rejoin the cluster. On the other hand, it may warrant an investigation into root causes of the problem (especially if it is recurring with some frequency).

validatePolls: This senior encountered an overdue poll, indicating a dead member, a significant network issue or an Operating System threading library bug (e.g. Linux NPTL): Poll
Severity: 2-Warning

Cause: When a node joins a cluster, it performs a handshake with each cluster node. A missing handshake response prevents this node from joining the service. The log message following this one indicates the corrective action taken by this node.

Action: If this message reoccurs, further investigation into the root cause may be warranted.

Received panic from junior member %s1 caused by %s2
%s1 - the cluster member that sent the panic; %s2 - a member claiming to be the senior member. Severity 2-Warning

Cause: This occurs after a cluster is split into multiple cluster islands (usually due to a network link failure). This message indicates that this senior member has no information about the other member that is claiming to be the senior member and will ignore the panic from the junior member until it can communicate with the other senior member.

Action: If this issue occurs frequently, the root cause of the cluster split should be investigated.

Received panic from senior Member(%s1) caused by Member(%s2)
%s1 - the cluster senior member as known by this node; %s2 - a member claiming to be the senior member. Severity: 1-Error.

Cause: This occurs after a cluster is split into multiple cluster islands (usually due to a network link failure). When a link is restored and the corresponding island seniors see each other, the panic protocol is initiated to resolve the conflict.

Action: If this issue occurs frequently, the root cause of the cluster split should be investigated.

Member %n1 joined Service %s with senior member %n2
%n1 - an id of the Coherence node that joins the service; %s - the service name; %n2 - the senior node for the service. Severity: 5-Debug Level 5.

Cause: When a clustered service starts on any cluster node, Coherence initiates a handshake protocol between all cluster nodes running the specified service. This message serves as an indication that the specified node has successfully completed the handshake and joined the service.

Action: None.

Member %n1 left Service %s with senior member %n2
%n1 - an id of the Coherence node that joins the service; %s - the service name; %n2 - the senior node for the service. Severity: 5-Debug Level 5.

Cause: When a clustered service terminates on some cluster node, all other nodes that run this service are notified about this event. This message serves as an indication that the specified clustered service at the specified node has terminated.

Action: None.

Service %s: received ServiceConfigSync containing %n entries
%s - the service name; %n - the number of entries in the service configuration map. Severity: 5-Debug Level 5.

Cause: As a part of the service handshake protocol between all cluster nodes running the specified service, the service senior member updates every new node with the full content of the service configuration map. For the partitioned cache services that map includes the full partition ownership catalog and internal ids for all existing caches. The message is also sent for an abnormal service termination at the senior node when a new node assumes the service seniority. This message serves as an indication that the specified node has received that configuration update.

Action: None.

TcpRing: connecting to member %n using TcpSocket{%s}
%s - the full information for the TcpSocket that serves as a TcpRing connector to another node; %n - the node id to which this node has connected. Severity: 5-Debug Level 5.

Cause: For quick process termination detection Coherence utilizes a feature called TcpRing, which is a sparse collection of TCP/IP-based connection between different nodes in the cluster. Each node in the cluster is connected to at least one other node, which (if at all possible) is running on a different physical box. This connection is not used for any data transfer; only trivial "heartbeat" communications are sent once a second per each link. This message indicates that the connection between this and specified node is initialized.

Action: None.

Rejecting connection to member %n using TcpSocket{%s}
%n - the node id that tries to connect to this node; %s - the full information for the TcpSocket that serves as a TcpRing connector to another node. Severity: 4-Debug Level 4.

Cause: Sometimes the TCP Ring daemons running on different nodes could attempt to join each other or the same node at the same time. In this case, the receiving node may determine that such a connection would be redundant and reject the incoming connection request. This message is logged by the rejecting node when this happens.

Action: None.

Timeout while delivering a packet; requesting the departure confirmation for Member(%s1) by MemberSet(%s2)
%s1 - the full Member information for a node that this node failed to communicate with; %s2 - the full information about the "witness" nodes that are asked to confirm the suspected member departure. Severity: 2-Warning.

Cause: Coherence uses TMB for all data communications (mostly peer-to-peer unicast), which by itself does not have any delivery guarantees. Those guarantees are built into the cluster management protocol used by Coherence (TCMP). The TCMP daemons are responsible for acknowledgment (ACK or NACK) of all incoming communications. If one or more packets are not acknowledged within the ACK interval ("ack-delay-milliseconds"), they are resent. This repeats until the packets are finally acknowledged or the timeout interval

elapses ("timeout-milliseconds"). At this time, this message is logged and the "witness" protocol is engaged, asking other cluster nodes whether they experience similar communication delays with the non-responding node. The witness nodes are chosen based on their roles and location.

Action: Corrective action is not necessarily required, since the rest of the cluster presumably is continuing its operation and this node may recover and rejoin the cluster. On the other hand, it may warrant an investigation into root causes of the problem (especially if it is recurring with some frequency).

This node appears to have become disconnected from the rest of the cluster containing %n nodes. All departure confirmation requests went unanswered. Stopping cluster service.
%n - the number of other nodes in the cluster this node was a member of. Severity: 1-Error.

Cause: Sometime a node that lives within a valid Java process, stops communicating to other cluster nodes. (Possible reasons include: a network failure; extremely long GC pause; swapped out process.) In that case, other cluster nodes may choose to revoke the cluster membership for the paused node and completely shun any further communication attempts by that node, causing this message be logged when the process attempts to resume cluster communications.

Action: Corrective action is not necessarily required, since the rest of the cluster presumably is continuing its operation and this node may recover and rejoin the cluster. On the other hand, it may warrant an investigation into root causes of the problem (especially if it is recurring with some frequency).

A potential communication problem has been detected. A packet has failed to be delivered (or acknowledged) after %n1 seconds, although other packets were acknowledged by the same cluster member (Member(%s1)) to this member (Member(%s2)) as recently as %n2 seconds ago. Possible causes include network failure, poor thread scheduling (see FAQ if running on Windows), an extremely overloaded server, a server that is attempting to run its processes using swap space, and unreasonably lengthy GC times.
%n1 - The number of seconds a packet has failed to be delivered or acknowledged; %s1 - the recipient of the packets indicated in the message; %s2 - the sender of the packets indicated in the message; %n2 - the number of seconds since a packet was delivered successfully between the two members indicated above. Severity: 2-Warning.

Cause: Possible causes are indicated in the text of the message.

Action: If this issue occurs frequently, the root cause should be investigated.

Node %s1 is not allowed to create a new cluster; WKA list: [%s2]
%s1 - Address of node attempting to join cluster; %s2 - List of WKA addresses. Severity: 1-Error.

Cause: The cluster is configured to use WKA, and there are no nodes present in the cluster that are in the WKA list.

Action: Ensure that at least one node in the WKA list exists in the cluster, or add this node's address to the WKA list.

This member is configured with a compatible but different WKA list then the senior Member(%s). It is strongly recommended to use the same WKA list for all cluster members.
%s - the senior node of the cluster. Severity: 2-Warning.

Cause: The WKA list on this node is different than the WKA list on the senior node. Using different WKA lists can cause different cluster members to operate independently from the rest of the cluster.

Action: Verify that the two lists are intentionally different or set them to the same values.

***<socket implementation>* failed to set receive buffer size to %n1 packets (%n2 bytes); actual size is %n3 packets (%n4 bytes). Consult your OS documentation regarding increasing the maximum socket buffer size. Proceeding with the actual value may cause sub-optimal performance.**
%n1 - the number of packets that fits in the buffer that Coherence attempted to allocate; %n2 - the size of the buffer Coherence attempted to allocate; %n3 - the number of packets that fits in the actual allocated buffer size; %n4 - the actual size of the allocated buffer. Severity: 2-Warning.

Cause: See Operating System Tuning.

Action: See Operating System Tuning.

The timeout value configured for IpMonitor pings is shorter than the value of 5 seconds. Short ping timeouts may cause an IP address to be wrongly reported as unreachable on some platforms.
Severity: 2-Warning

Cause: The ping timeout value is less than 5 seconds.

Action: Ensure that the ping timeout that is configured within the `<tcp-ring-listener>` element is greater than 5 seconds.

Network failure encountered during InetAddress.isReachable(): %s
%n - a stack trace. Severity: 5-Debug Level 5.

Cause: The IpMonitor component is unable to ping a member and has reached the configured timeout interval.

Action: Ensure that the member is operational or verify a network outage. The ping timeout that is configured within the `<tcp-ring-listener>` element can be increased to allow for a longer timeout as required by the network.

TcpRing has been explicitly disabled, this is not a recommended practice and will result in a minimum death detection time of %n seconds for failed processes.
%n - the number of seconds that is specified by the packet publisher's resend timeout which is 5 minutes by default. Severity: 2-Warning.

Cause: The TcpRing Listener component has been disabled.

Action: Enable the TcpRing listener within the `<tcp-ring-listener>` element.

IpMonitor has been explicitly disabled, this is not a recommended practice and will result in a minimum death detection time of %n seconds for failed machines or networks.
%n - the number of seconds that is specified by the packet publisher's resend timeout which is 5 minutes by default. Severity: 2-Warning.

Cause: The IpMonitor component has been disabled.

Action: The IpMonitor component is enabled when the TcpRing listener is enabled within the `<tcp-ring-listener>` element.

TcpRing connecting to %s
%s - the cluster member to which this member has joined to form a TCP-Ring. Severity: 6-Debug Level 6.

Cause: This message indicates that the connection between this and the specified member is initialized. The TCP-Ring is used for quick process termination detection and is a sparse collection of TCP/IP-based connection between different nodes in the cluster.

Action: none.

TcpRing disconnected from %s to maintain ring
%s - the cluster member from which this member has disconnected. Severity: 6-Debug Level 6.

Cause: This message indicates that this member has disconnected from the specified member and that the specified member is no longer a member of the TCP-Ring. The TCP-Ring is used for quick process termination detection and is a sparse collection of TCP/IP-based connection between different nodes in the cluster.

Action: If the member was intentionally stopped, no further action is required. Otherwise, the member may have been released from the cluster due to a failure or network outage. Restart the member.

TcpRing disconnected from %s due to a peer departure; removing the member.
%s - the cluster member from which this member has disconnected. Severity: 5-Debug Level 5.

Cause: This message indicates that this member has disconnected from the specified member and that the specified member is no longer a member of the TCP-Ring. The TCP-Ring is used for quick process termination detection and is a sparse collection of TCP/IP-based connection between different nodes in the cluster.

Action: If the member was intentionally stopped, no further action is required. Otherwise, the member may have been released from the cluster due to a failure or network outage. Restart the member.

TcpRing connection to "%s" refused ("%s1"); removing the member.
%s - the cluster member to which this member was refused a connection; %s1- the refusal message. Severity: 5-Debug Level 5.

Cause: The specified member has refused a TCP connection from this member and has subsequently been removed from the TCP-Ring.

Action: If the member was intentionally stopped, no further action is required. Otherwise, the member may have been released from the cluster due to a failure or network outage. Restart the member.

Configuration Log Messages

Log messages that pertain to configuration

java.io.IOException: Configuration file is missing: "tangosol-coherence.xml"
Severity: 1-Error.

Cause: The operational configuration descriptor cannot be loaded.

Action: Make sure that the `tangosol-coherence.xml` resource can be loaded from the class path specified in the Java command line.

Loaded operational configuration from resource "%s"
%s - the full resource path (URI) of the operational configuration descriptor. Severity: 3-Informational.

Cause: The operational configuration descriptor is loaded by Coherence from the specified location.

Action: If the location of the operational configuration descriptor was explicitly specified using system properties or programmatically, verify that the reported URI matches the expected location.

Loaded operational overrides from "%s"
%s - the URI (file or resource) of the operational configuration descriptor override. Severity: 3-Informational.

Cause: The operational configuration descriptor points to an override location, from which the descriptor override has been loaded.

Action: If the location of the operational configuration descriptor was explicitly specified using system properties, descriptor override or programmatically, verify that the reported URI matches the expected location.

Optional configuration override "%s" is not specified
%s - the URI of the operational configuration descriptor override. Severity: 3-Informational.

Cause: The operational configuration descriptor points to an override location which does not contain any resource.

Action: Verify that the operational configuration descriptor override is not supposed to exist.

java.io.IOException: Document "%s1" is cyclically referenced by the 'xml-override' attribute of element %s2
%s1 - the URI of the operational configuration descriptor or override; %s2 - the name of the XML element that contains an incorrect reference URI. Severity: 1-Error.

Cause: The operational configuration override points to itself or another override that point to it, creating an infinite recursion.

Action: Correct the invalid `xml-override` attribute's value.

java.io.IOException: Exception occurred during parsing: %s
%s - the XML parser error. Severity: 1-Error.

Cause: The specified XML is invalid and cannot be parsed.

Action: Correct the XML document.

Loaded cache configuration from "%s"
%s - the URI (file or resource) of the cache configuration descriptor. Severity: 3-Informational.

Cause: The operational configuration descriptor or a programmatically created `ConfigurableCacheFactory` instance points to a cache configuration descriptor that has been loaded.

Action: Verify that the reported URI matches the expected cache configuration descriptor location.

Partitioned Cache Service Log Messages

Log messages that pertain to the partitioned cache service

Application code running on "%s1" service thread(s) should not call ensureCache as this may result in a deadlock
The most common case is a `CacheFactory` call from a custom `CacheStore` implementation.
%s1 - the service which is executing the application code
Cause: This message indicates that an `ensureCache` operation has been called for a cache on the same service that is executing the application code. This code can be from any user application code such as an interceptor, entry processor, or cache store. This is to protect against potential deadlock through a callback into the same service.

Action: In your application code, you should execute the code that does the `ensureCache` on an Executor so that the code executes on a different thread which is not managed by Coherence. Depending upon the nature and expected load of your requests, you may choose to use a thread pool or a single Executor.

Asking member %n1 for %n2 primary partitions
%n1 - the node id this node asks to transfer partitions from; %n2 - the number of partitions this node is willing to take. Severity: 4-Debug Level 4.

Cause: When a storage-enabled partitioned service starts on a Coherence node, it first receives the configuration update that informs it about other storage-enabled service nodes and the current partition ownership information. That information allows it to calculate the "fair share" of partitions that each node is supposed to own after the re-distribution process. This message demarcates the beginning of the transfer request to a specified node for its "fair" ownership distribution.

Action: None.

Transferring %n1 out of %n2 primary partitions to member %n3 requesting %n4
%n1 - the number of primary partitions this node transferring to a requesting node; %n2 - the total number of primary partitions this node currently owns; %n3 - the node

id that this transfer is for; %n4 - the number of partitions that the requesting node asked for. Severity: 4-Debug Level 4.

Cause: During the partition distribution protocol, a node that owns less than a "fair share" of primary partitions requests any of the nodes that own more than the fair share to transfer a portion of owned partitions. The owner may choose to send any number of partitions less or equal to the requested amount. This message demarcates the beginning of the corresponding primary data transfer.

Action: None.

Transferring %n1 out of %n2 partitions to a machine-safe backup 1 at member %n3 (under %n4)
%n1 - the number of backup partitions this node transferring to a different node; %n2 - the total number of partitions this node currently owns that are "endangered" (do not have a backup); %n3 - the node id that this transfer is for; %n4 - the number of partitions that the transferee can take before reaching the "fair share" amount. Severity: 4-Debug Level 4.

Cause: After the primary partition ownership is completed, nodes start distributing the backups, ensuring the "strong backup" policy, that places backup ownership to nodes running on computers that are different from the primary owners' computers. This message demarcates the beginning of the corresponding backup data transfer.

Action: None.

Transferring backup%n1 for partition %n2 from member %n3 to member %n4
%n1 - the index of the backup partition that this node transferring to a different node; %n2 - the partition number that is being transferred; %n3 the node id of the previous owner of this backup partition; %n4 the node id that the backup partition is being transferred to. Severity: 5-Debug Level 5.

Cause: During the partition distribution protocol, a node that determines that a backup owner for one of its primary partitions is overloaded may choose to transfer the backup ownership to another, underloaded node. This message demarcates the beginning of the corresponding backup data transfer.

Action: None.

Failed backup transfer for partition %n1 to member %n2; restoring owner from: %n2 to: %n3
%n1 the partition number for which a backup transfer was in-progress; %n2 the node id that the backup partition was being transferred to; %n3 the node id of the previous backup owner of the partition. Severity: 4-Debug Level 4.

Cause: This node was in the process of transferring a backup partition to a new backup owner when that node left the service. This node is restoring the backup ownership to the previous backup owner.

Action: None.

Deferring the distribution due to %n1 pending configuration updates
%n1- the number of configuration updates. Severity: 5-Debug Level 5.

Cause: This node is in the process of updating the global ownership map (notifying other nodes about ownership changes) when the periodic scheduled distribution check takes place. Before the previous ownership changes (most likely due to a previously completed

transfer) are finalized and acknowledged by the other service members, this node postpones subsequent scheduled distribution checks.

Action: None.

Limiting primary transfer to %n1 KB (%n2 partitions)
%n1 - the size in KB of the transfer that was limited; %n2 the number of partitions that were transferred. Severity: 4-Debug Level 4.

Cause: When a node receives a request for some number of primary partitions from an underloaded node, it may transfer any number of partitions (up to the requested amount) to the requester. The size of the transfer is limited by the `<transfer-threshold>` element. This message indicates that the distribution algorithm limited the transfer to the specified number of partitions due to the transfer-threshold.

Action: None.

DistributionRequest was rejected because the receiver was busy. Next retry in %n1 ms
%n1 - the time in milliseconds before the next distribution check is scheduled. Severity: 6-Debug Level 6.

Cause: This (underloaded) node issued a distribution request to another node asking for one or more partitions to be transferred. However, the other node declined to initiate the transfer as it was in the process of completing a previous transfer with a different node. This node waits at least the specified amount of time (to allow time for the previous transfer to complete) before the next distribution check.

Action: None.

Restored from backup %n1 partitions
%n1 - the number of partitions being restored. Severity: 3-Informational.

Cause: The primary owner for some backup partitions owned by this node has left the service. This node is restoring those partitions from backup storage (assuming primary ownership). This message is followed by a list of the partitions that are being restored.

Action: None.

Re-publishing the ownership for partition %n1 (%n2)
%n1 the partition number whose ownership is being re-published; %n2 the node id of the primary partition owner, or 0 if the partition is orphaned. Severity: 4-Debug Level 4.

Cause: This node is in the process of transferring a partition to another node when a service membership change occurred, necessitating redistribution. This message indicates this node re-publishing the ownership information for the partition whose transfer is in-progress.

Action: None.

%n1> Ownership conflict for partition %n2 with member %n3 (%n4!=%n5)
%n1 - the number of attempts made to resolve the ownership conflict; %n2 - the partition whose ownership is in dispute; %n3 - the node id of the service member in disagreement about the partition ownership; %n4 - the node id of the partition's

primary owner in this node's ownership map; %n5 - the node id of the partition's primary owner in the other node's ownership map. Severity: 4-Debug Level 4.

Cause: If a service membership change occurs while the partition ownership is in-flux, it is possible for the ownership to become transiently out-of-sync and require reconciliation. This message indicates that such a conflict was detected, and denotes the attempts to resolve it.

Action: None.

Unreconcilable ownership conflict; conceding the ownership
Severity: 1-Error

Cause: If a service membership change occurs while the partition ownership is in-flux, it is possible for the ownership to become transiently out-of-sync and require reconciliation. This message indicates that an ownership conflict for a partition could not be resolved between two service members. To resolve the conflict, one member is forced to release its ownership of the partition and the other member republishes ownership of the partition to the senior member.

Action: None

Multi-way ownership conflict; requesting a republish of the ownership
Severity: 1-Error

Cause: If a service membership change occurs while the partition ownership is in-flux, it is possible for the ownership to become transiently out-of-sync and require reconciliation. This message indicates that a service member and the most senior storage-enabled member have conflicting views about the owner of a partition. To resolve the conflict, the partitioned is declared an orphan until the owner of the partition republishes its ownership of the partition.

Action: None

Assigned %n1 orphaned primary partitions
%n1 - the number of orphaned primary partitions that were re-assigned. Severity: 2-Warning.

Cause: This service member (the most senior storage-enabled) has detected that one or more partitions have no primary owner (orphaned), most likely due to several nodes leaving the service simultaneously. The remaining service members agree on the partition ownership, after which the storage-senior assigns the orphaned partitions to itself. This message is followed by a list of the assigned orphan partitions. This message indicates that data in the corresponding partitions may have been lost.

Action: None.

validatePolls: This service timed-out due to unanswered handshake request. Manual intervention is required to stop the members that have not responded to this Poll
Severity: 1-Error.

Cause: When a node joins a clustered service, it performs a handshake with each clustered node running the service. A missing handshake response prevents this node from joining the service. Most commonly, it is caused by an unresponsive (for example, deadlocked) service thread.

Action: Corrective action may require locating and shutting down the JVM running the unresponsive service. See Note 845363.1 at My Oracle Support for more details.

com.tangosol.net.RequestPolicyException: No storage-enabled nodes exist for service *service_name*
Severity: 1-Error.

Cause: A cache request was made on a service that has no storage-enabled service members. Only storage-enabled service members may process cache requests, so there must be at least one storage-enabled member.

Action: Check the configuration/deployment to ensure that members that are intended to store cache data are configured to be storage-enabled. Storage is enabled on a member using the `<local-storage>` element or by using the `-Dcoherence.distributed.localstorage` command-line override.

An entry was inserted into the backing map for the partitioned cache "%s" that is not owned by this member; the entry will be removed."
%s - the name of the cache into which insert was attempted. Severity: 1-Error.

Cause: The backing map for a partitioned cache may only contain keys that are owned by that member. Cache requests are routed to the service member owning the requested keys, ensuring that service members only process requests for keys which they own. This message indicates that the backing map for a cache detected an insertion for a key which is not owned by the member. This is most likely caused by a direct use of the backing-map as opposed to the exposed cache APIs (for example, `NamedCache`) in user code running on the cache server. This message is followed by a Java exception stack trace showing where the insertion was made.

Action: Examine the user-code implicated by the stack-trace to ensure that any backing-map operations are safe. This error can be indicative of an incorrect implementation of `KeyAssociation`.

Exception occurred during filter evaluation: %s; removing the filter...
%s - the description of the filter that failed during evaluation. Severity: 1-Error.

Cause: An exception was thrown while evaluating a filter for a `MapListener` implementation that is registered on this cache. As a result, some map events may not have been issued. Additionally, to prevent further failures, the filter (and associated `MapListener` implementation) are removed. This message is followed by a Java exception stack trace showing where the failure occurred.

Action: Review filter implementation and the associated stack trace for errors.

Exception occurred during event transformation: %s; removing the filter...
%s - the description of the filter that failed during event transformation. Severity: 1-Error.

Cause: An Exception was thrown while the specified filter was transforming a `MapEvent` for a `MapListener` implementation that is registered on this cache. As a result, some map events may not have been issued. Additionally, to prevent further failures, the Filter implementation (and associated `MapListener` implementation) are removed. This message is followed by a Java exception stack trace showing where the failure occurred.

Action: Review the filter implementation and the associated stack trace for errors.

Exception occurred during index rebuild: %s
%s - the stack trace for the exception that occurred during index rebuild. Severity: 1-Error.

Cause: An Exception was thrown while adding or rebuilding an index. A likely cause of this is a faulty `ValueExtractor` implementation. As a result of the failure, the associated index is removed. This message is followed by a Java exception stack trace showing where the failure occurred.

Action: Review the `ValueExtractor` implementation and associated stack trace for errors.

Exception occurred during index update: %s
%s - the stack trace for the exception that occurred during index update. Severity: 1-Error.

Cause: An Exception was thrown while updating an index. A likely cause of this is a faulty `ValueExtractor` implementation. As a result of the failure, the associated index is removed. This message is followed by a Java exception stack trace showing where the failure occurred.

Action: Review the `ValueExtractor` implementation and associated stack trace for errors.

Exception occurred during query processing: %s
%s - the stack trace for the exception that occurred while processing a query. Severity: 1-Error.

Cause: An Exception was thrown while processing a query. A likely cause of this is an error in the filter implementation used by the query. This message is followed by a Java exception stack trace showing where the failure occurred.

Action: Review the filter implementation and associated stack trace for errors.

BackingMapManager %s1: returned "null" for a cache: %s2
%s1 - the classname of the BackingMapManager implementation that returned a null backing-map; %s2 - the name of the cache for which the BackingMapManager returned null. Severity: 1-Error.

Cause: A `BackingMapManager` returned `null` for a backing-map for the specified cache.

Action: Review the specified BackingMapManager implementation for errors and to ensure that it properly instantiates a backing map for the specified cache.

BackingMapManager %s1: failed to instantiate a cache: %s2
%s1 - the classname of the BackingMapManager implementation that failed to create a backing-map; %s2 - the name of the cache for which the BackingMapManager failed. Severity: 1-Error.

Cause: A `BackingMapManager` unexpectedly threw an Exception while attempting to instantiate a backing-map for the specified cache.

Action: Review the specified BackingMapManager implementation for errors and to ensure that it properly instantiates a backing map for the specified cache.

BackingMapManager %s1: failed to release a cache: %s2
%s1 - the classname of the BackingMapManager implementation that failed to release a backing-map; %s2 - the name of the cache for which the BackingMapManager failed. Severity: 1-Error.

Cause: A `BackingMapManager` unexpectedly threw an Exception while attempting to release a backing-map for the specified cache.

Action: Review the specified `BackingMapManager` implementation for errors and to ensure that it properly releases a backing map for the specified cache.

Unexpected event during backing map operation: key=%s1; expected=%s2; actual=%s3
%s1 - the key being modified by the cache; %s2 - the expected backing-map event from the cache operation in progress; %s3 - the actual MapEvent received. Severity: 6-Debug Level 6.

Cause: While performing a cache operation, an unexpected MapEvent was received on the backing-map. This indicates that a concurrent operation was performed directly on the backing-map and is most likely caused by direct manipulation of the backing-map as opposed to the exposed cache APIs (for example, `NamedCache`) in user code running on the cache server.

Action: Examine any user-code that may directly modify the backing map to ensure that any backing-map operations are safe.

Application code running on "%s1" service thread(s) should not call %s2 as this may result in deadlock. The most common case is a CacheFactory call from a custom CacheStore implementation.
%s1 - the name of the service which has made a re-entrant call; %s2 - the name of the method on which a re-entrant call was made. Severity: 2-Warning.

Cause: While executing application code on the specified service, a re-entrant call (a request to the same service) was made. Coherence does not support re-entrant service calls, so any application code (CacheStore, EntryProcessor, and so on...) running on the service thread(s) should avoid making cache requests.

Action: Remove re-entrant calls from application code running on the service thread(s) and consider using alternative design strategy. See Constraints on Re-entrant Calls in *Developing Applications with Oracle Coherence*.

Repeating %s1 for %n1 out of %n2 items due to re-distribution of %s2
%s1 - the description of the request that must be repeated; %n1 - the number of items that are outstanding due to re-distribution; %n2 - the total number of items requested; %s2 - the list of partitions that are in the process of re-distribution and for which the request must be repeated. Severity: 5-Debug Level 5.

Cause: When a cache request is made, the request is sent to the service members owning the partitions to which the request refers. If one or more of the partitions that a request refers to is in the process of being transferred (for example, due to re-distribution), the request is rejected by the (former) partition owner and is automatically resent to the new partition owner.

Action: None.

Error while starting cluster: com.tangosol.net.RequestTimeoutException: Timeout during service start: ServiceInfo(%s)
%s - information on the service that could not be started. Severity: 1-Error.

Cause: When joining a service, every service in the cluster must respond to the join request. If one or more nodes have a service that does not respond within the timeout period, the join times out.

Action: See 845363.1 at My Oracle Support for more details.

Failed to restart services: com.tangosol.net.RequestTimeoutException: Timeout during service start: ServiceInfo(%s)
%s - information on the service that could not be started. Severity: 1-Error.

Cause: When joining a service, every service in the cluster must respond to the join request. If one or more nodes have a service that does not respond within the timeout period, the join times out.

Action: See My Oracle Support Note
See 845363.1 at My Oracle Support for more details.

Failed to recover partition 0 from SafeBerkeleyDBStore(...); partition-countmismatch 501(persisted) != 277(service); reinstate persistent store fromtrash once validation errors have been resolved
Cause: The partition-count is changed while active persistence is enabled. The current active data is copied to the trash directory.

Action: Complete the following steps to recover the data:

1. Shutdown the entire cluster.
2. Remove the current active directory contents for the cluster and service affected on each cluster member.
3. Copy (recursively) the contents of the trash directory for each service to the active directory.
4. Restore the partition count to the original value.
5. Restart the cluster.

Service Thread Pool Log Messages

Log messages that pertain to the service thread pool

DaemonPool "%s" increasing the pool size from %n1 to %n2 thread(s) due to the pool being shaken
%s - the service name; %n1 - the current thread pool count; %n2 - the new thread pool count. Severity: 3 - Informational.
Cause: The thread pool count will be increased intermittently and the thread pool throughput will be measured to determine if the increase is effective. The thread count will be increased only if dynamic thread pooling is enabled and the new thread count does not exceed the maximum configured value.
Action: None. This is part of the process for determining the most effective thread pool count.

DaemonPool "%s" increasing the pool size from %n1 to %n2 thread(s) due to a decrease in throughput of %n3op/sec
%s - the service name; %n1 - the current thread pool count; %n2 - the new thread pool count; %n3 - the change in operations per second. Severity: 3 - Informational.
Cause: The thread pool task throughput was reduced with a lower thread count. The thread count is being increased to improve throughput. The thread count will be increased only if

dynamic thread pooling is enabled and the new thread count does not exceed the maximum configured value.
Action: None. This is part of the process for determining the most effective thread pool count.

DaemonPool "%s" decreasing the pool size from %n1 to %n2 thread(s) due to a decrease in throughput of %n3op/sec
%s - the service name; %n1 - the current thread pool count; %n2 - the new thread pool count; %n3 - the change in operations per second. Severity: 3 - Informational.
Cause: The thread pool task throughput was reduced with a higher thread count. The thread count is being decreased to improve throughput. The thread count will be decreased only if dynamic thread pooling is enabled and the new thread count does not drop below the configured minimum value.
Action: None. This is part of the process for determining the most effective thread pool count.

TMB Log Messages

Log messages that pertain to TMB.

%s1 rejecting connection from %s2 using incompatible protocol id %s3, required %s4
%s1 - the local endpoint; %s2 - the socket address; %s3 - the connection protocol id; %s4 - the required protocol Id. Severity: 2 -Warning.
Cause: A Coherence node with incompatible protocol identifier has attempted to establish a connection with this node. This should not happen unless the request is from a malicious connect attempt or the message header is corrupted.
Action: Restart remote node. If problem persists, send all related information to Oracle Support for investigation.

%s1 rejecting connection from %s2, bus is closing
%s1 – the local endpoint; %s2 – the socket address. Severity: 5-Debug.
Cause: The local message bus connection received a connection request while it is being closed. This is likely to occur during the local node shutdown.
Action: None.

%s1 deferring reconnect attempt from %s2 on %s3, pending release
%s1 – the local endpoint; %s2 – the peer's endpoint; %s3 – the associated channel socket. Severity: 5-Debug.
Cause: The local message bus connection received a reconnect request from the same remote endpoint while the current connection is waiting for an application to fully release.
Action: None.

%s1 replacing deferred reconnect attempt from %s2 on %s3, pending release
%s1 – the local endpoint; %s2 – the peer's endpoint; %s3 – the associated channel socket. Severity: 5-Debug.
Cause: The local TCP socket received a subsequent reconnect request, thus replacing a previous reconnect attempt, while waiting for the application to fully release. This is expected due to the possibility of concurrent connect initiation from both the remote and local endpoint, which is part of the connection handshake protocol.
Action: None.

%s1 initiating connection migration with %s2 after %n ack timeout %s3
%s1 – the local endpoint; %s2 - the peer's endpoint; %n – the ack timeout value; %s3 – debug info. Severity: 2-Warning.
Cause: A message was sent, but a logical ack for that message was not received for more than the configured ack timeout. The default value of this timeout is 15s which, from a Coherence perspective, is an eternity to not deliver a message. This is a means to detect a stalled connection and initiate the remedial actions to resolve the situation.
Action: If a stalled connection was correctly inferred, then the remedial action of migrating the connection to a new TCP connection should resolve the issue. To resolve the stalled connections, ensure that you have the latest version of the OS as stalls have been observed in certain Linux Kernel versions. The connection may not have been installed and may be due to process unresponsiveness. Therefore, also ensure that network connectivity to the machine mentioned in %s2 looks reasonable and the process seems responsive (not in a GC loop). Frequent migration will severely impact performance and availability. If the message perpetually repeats, collect a heap dump from both the local and peer, any network reports available, and all coherence logs. Send the information to Oracle Support for investigation.

%s1 accepting connection migration with %s2, replacing %s3 with %s4:%s5
%s1 – the local endpoint; %s2 – the peer endpoint; %s3 – the old SocketChannel; %s4 – the new SocketChannel; %s5 – the old message bus connection. Severity: 2-Warning.
Cause: The peer initiated a connection migration while local was not aware of the connection issue. The local message bus accepted the request and replaced the old socket channel with the new one. The migration can be caused by TCP connection stalls, GC, or a network issue.
Action: If the problem persists, collect heap dumps from local and remote server, any available network reports, and all coherence logs. Send the information to Oracle Support for investigation. Also, enabling TCP captures provides significant insight into whether the messages are being received by the peer and the sender.

%s1 migrating connection with %s2 off %s3 on %s4
%s1 – the local endpoint; %s2 – the peer endpoint; %s3 – the socket channel; %s4 – the string representation of the message bus connection. Severity: 6-Debug.
Cause: The local message bus initiated a connection migration due to ack timeout or another error. If the message is seen frequently while the application is still functioning, it indicates process unresponsiveness (often due to GC) or a network issue, which is likely to impact cluster performance.
Action: Investigate the remote GC or network issue. If the problem persists, send heap dumps of both the local and peer as well as all Coherence logs to Oracle Support for investigation. Also, enabling TCP captures provides significant insight into whether the messages are being received by the peer and by the sender.

%s1 synchronizing migrated connection with %s2 will result in %n1 skips and %n2 re-deliveries: %s3
%s1 – the local endpoint; %s2 – the peer's endpoint %n1 – number of messages to skip; %n2 – number of messages to re-deliver; %s3 – the string representation of the local bus connection. Severity: 5-Debug.
Cause: This is informational only. The migrated connection needs to skip or redeliver the queued messages, depending on whether acks for the messages are received.
Action: None.

%s1 rejecting connection migration from %s2 on %s3, no existing connection %s4/%s5
%s1 – the local endpoint; %s2 – the peer endpoint; %s3 – the local socket address; %s4 – the current connection identifier; %s5 – the old connection identifier or 0 if old connection does not exist. Severity: 5-Debug.
Cause: The local message bus received a migration request on a connection that does not exist. Hence, reject the request. Most probably, the connection has been released.
Action: None.

%s1, %s2 accepted connection migration with %s3:%s4
%s1 – the local endpoint; %s2 – the peer endpoint; %s3 – the socket channel; %s4 – the string representation of message bus connection. Severity: 2-Warning.
Cause: This is informational message. The connection migration has successfully finished the handshake protocol.
Action: None.

%s1 resuming migrated connection with %s2
%s1 – the local endpoint; %s2 – the string representation of bus connection. Severity: 5-Debug.
Cause: This is informational message. The connection was successfully migrated; now resuming normal processing with the new migrated socket channel.
Action: None.

%s1 ServerSocket failure; no new connection will be accepted.
%s2 – the local endpoint. Severity: 1-Error.
Cause: This message indicates that the server socket channel, on which the message bus accepts new connections, has failed to register with a selection service.
Action: This is an unexpected state and may require a node restart if the process continues to appear unhealthy.

%s1 disconnected connection with %s2
%s1 – local endpoint; %s2 – remote endpoint. Severity: 3-Info.
Cause: The connection with the mentioned remote endpoint was disconnected.
Action: None.

%s1 close due to exception during handshake phase %s2 on %s3
%s1 – the local endpoint; %s2 – the phase of handshake; %s3 – socket associated with the connection channel. Severity: 2-Warning.
Cause: The connection request was rejected during the mentioned handshake phase due to SSLException; the associated socket channel was closed.
Action: The error message should indicate why the handshake failed and should provide sufficient information to resolve (for example, an expired cert). If the issue persists, contact Oracle Support.

%s1 dropping connection with %s2 after %s3 fatal ack timeout %s4
%s1 – the local endpoint; %s2 – the remote endpoint; %s3 – the fatal ack timeout value in ms; %s4 – info for debugging purpose. Severity: 2-Warning.
Cause: The local bus connection has failed to hear from the peer for the configure fatal ack timeout; the connection will be dropped as it is unrecoverable. It is likely caused by extended process unresponsiveness (potential GC issues) or a network issue.
Action: Investigate remote GC logs or network logs (TCP captures / network monitoring). If the problem persists, send heap dumps from both the local and peer, as well as all Coherence logs to Oracle Support for investigation.

%s unexpected exception during Bus accept, ignoring
%s – the local endpoint. Severity: 3-Info.
Cause: An exception has occurred during server socket accepting a connection request. It is safe to ignore the exception and continue to accept the request as the server socket channel is still open.
Action: None.

%s ServerSocket failure; no new connection will be accepted
%s – the local endpoint. Severity: 1-Error.
Cause: An exception has occurred during server socket accepting a connection request and the server socket was closed unexpectedly.
Action: Restart the node. If the problem persists, contact Oracle Support.

Unhandled exception in %s, attempting to continue
%s – the selection service. Severity: 1-Error.
Cause: Unexpected error while running the selector thread. The thread will continue to select and process messages.
Action: None.

%s1 disconnected connection with %s2
%s1 – the local endpoint; %s2 – the string representation of bus connection. Severity: 2-Warning if the reason for disconnect is SSLException, otherwise 6-Debug.
Cause: The mentioned connection was closed. This could be due to various reasons, including an encountered exception/error or an expected release.
Action: None.

Cluster Service Exceptions

Exceptions thrown by the cluster service.

IllegalStateException: The cluster has been halted and is not restartable.
Cause: When a service guardian fails to terminate its non-responsive service thread and the guardian service failure policy is `exit-cluster`, the cluster is halted. Any attempt to use a Coherence cluster operation after the cluster has been halted results in this exception.
Action: Restart the server process when the cluster is halted. To avoid the need to restart the server process in the future, review if another guardian service failure policy option is more appropriate for your environment. See Setting the Guardian Service Failure Policy in *Developing Applications with Oracle Coherence*.
The default guardian service failure policy is `exit-cluster`. To identify what time this failure occurred, search in the server logs for the guardian service log message Oracle Coherence <Error>: Halted the cluster: Cluster is not running: State=5 .

Guardian Service Log Messages

Log messages that pertain to the Guardian Service.

Detected hard timeout after %s1 of {WrapperGuardable Guard{Daemon=%s2 } Service=%s3{Name=%s4, State=(SERVICE_STARTED), …}}
%s1 –time duration; %s2 - cache scheme:service name; %s3 – serviceKind: %s4 – service name
Cause: The guardian service did not receive a heartbeat from service for time duration of %s1. The service will attempt to terminate the non-responding thread.

Action: Informational. Typically, the guardian is able to recover the service thread and proceed.

onServiceFailed: Failed to stop service %s with state=%n, isAlive=%b1, stop service thread isAlive=%b2
%s – service name; %n – service state; %b1 – true if service is still alive; %b2 – true if terminating service thread is still alive
Cause: The guardian service failed to interrupt hung service thread %s.
Action: Analyze the hung service's stack trace (see the next message), and find the next **Full Thread Dump** and **Outstanding Polls** in the server log to analyze what the service thread was waiting for to get stuck.

onServiceFailed: Service thread: Stack trace for thread Thread[%s1:%s2,%n,Cluster]: <%stackTrace>
%s1 – cache kind; %s2 – service name; %n – thread identifier; <%stackTrace> - multi line stack trace of uninterruptible thread
Cause: The guardian service failed to interrupt service %s2.
Action: Analyze the stack trace of the hung service and search for **Full Thread Dump** to see if there was a deadlock between the stuck thread and another running thread.

Oracle Coherence <Error>: Halted the cluster: Cluster is not running: State=5
Cause: When the guardian service fails to recover a stuck thread and the guardian service failure policy is `exit-cluster`, the cluster is halted.
Action: Restart the server process when the cluster is halted. To avoid the need to restart the server process in the future, review if another guardian service failure policy option is more appropriate for your environment. See Setting the Guardian Service Failure Policy in *Developing Applications with Oracle Coherence*. The default guardian service failure policy is `exit-cluster`. To understand the failure that resulted in the cluster being halted, look for the logs messages listed in this section in the server log. The timestamp on this message provides the exact time the cluster is halted.

(thread=Recovery Thread, member=%n): Full Thread Dump: (excluding deadlock analysis)
`%n` = short Coherence member id
Cause: The guardian service recovery thread prints this diagnostic when recovery of the stuck service thread fails.
Action: Analyze if the stuck service thread that could not be stopped was blocked waiting for a lock held by another active thread at the time of the failure.