Oracle® Coherence
Developer's Guide
Release 3.5
E14509-01

June 2009
## Contents

Preface .................................................................................................................................................... xvi

Audience ............................................................................................................................................... xvii
Documentation Accessibility ........................................................................................................ xvi
Related Documents ......................................................................................................................... xviii
Conventions ......................................................................................................................................... xviii

Part I  Coherence Features

1 Create and Use Coherence Caches

Creating a Cache in Your Application .............................................................................................. 1-1
Configuring the Caches ...................................................................................................................... 1-2
Cache Configuration Descriptor Location .......................................................................................... 1-5
Putting It all Together: Your First Coherence Cache Example ....................................................... 1-5
Setting Up Your Test Environment .................................................................................................. 1-5
Modifying the Cache Configuration .................................................................................................. 1-9

2 Implement Transactions, Locks, and Concurrency

Concurrency Options ........................................................................................................................... 2-1
Explicit Locking .................................................................................................................................. 2-1
Transactions ......................................................................................................................................... 2-2
Container Integration .......................................................................................................................... 2-4
    JCA .................................................................................................................................................. 2-4
    XA .................................................................................................................................................. 2-6
Entry Processors ................................................................................................................................... 2-6
Data Source Integration ....................................................................................................................... 2-8

3 Perform Continuous Query

Uses of Continuous Query Caching ..................................................................................................... 3-1
The Coherence Continuous Query Cache ............................................................................................ 3-2
Constructing a Continuous Query Cache ............................................................................................ 3-2
    Cleaning up the resources associated with a ContinuousQueryCache ........................................ 3-3
Caching only keys, or caching both keys and values .......................................................................... 3-3
    CacheValues Property and Event Listeners ................................................................................. 3-3
Listening to the ContinuousQueryCache ............................................................................................ 3-3
Priority Task Execution — Custom Objects ................................................................. 9-3
APIs for Creating Priority Task Objects ................................................................. 9-3
Errors Thrown by Task Timeouts ........................................................................... 9-4

10 Specifying a Custom Eviction Policy

11 Serialization Paged Cache

Understanding Serialization Paged Cache ............................................................ 11-1
Configuring Serialization Paged Cache ................................................................. 11-1
Optimizing a Partitioned Cache Service ............................................................... 11-2
Configuring for High Availability .......................................................................... 11-2
Configuring Load Balancing and Failover ............................................................. 11-2
Supporting Huge Caches ....................................................................................... 11-2

12 Pre-Loading the Cache

Performing Bulk Loading and Processing ............................................................ 12-1
Bulk Writing to a Cache ....................................................................................... 12-1
Efficient processing of filter results ....................................................................... 12-2
A Bulk Loading and Processing Example ............................................................. 12-4
Performing Distributed Bulk Loading ................................................................. 12-9
A Distributed Bulk Loading Example .................................................................. 12-9
Running a Distributed Bulk Loading Example .................................................. 12-11
Building the Sample Application ........................................................................ 12-11
Running the Sample Application ........................................................................ 12-11

13 Constraints on Re-entrant Calls

Re-entrancy, Services, and Service Threads ......................................................... 13-1
Parent-Child Object Relationships ....................................................................... 13-1
Avoiding Deadlock ............................................................................................... 13-2
Re-entrancy and Listeners ................................................................................... 13-2

Part II Testing and Tuning

14 Evaluating Performance and Scalability

Measuring Latency and Throughput ..................................................................... 14-1
Demonstrating Scalability .................................................................................... 14-1
Tuning Your Environment ................................................................................... 14-2
Measurements on a Large Cluster ....................................................................... 14-2

15 Performing a Multicast Connectivity Test

Running the Multicast Test Utility .................................................................... 15-1
Sample Commands .............................................................................................. 15-1
Multicast Test Example ....................................................................................... 15-2
Troubleshooting Multicast Communications ..................................................... 15-3
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Performing a Datagram Test for Network Performance</td>
</tr>
<tr>
<td>Running the Datagram Test Utility</td>
<td>16-1</td>
</tr>
<tr>
<td>Sample Commands for a Listener and a Publisher</td>
<td>16-2</td>
</tr>
<tr>
<td>Datagram Test Example</td>
<td>16-2</td>
</tr>
<tr>
<td>Reporting</td>
<td>16-3</td>
</tr>
<tr>
<td>Publisher Statistics</td>
<td>16-3</td>
</tr>
<tr>
<td>Listener Statistics</td>
<td>16-4</td>
</tr>
<tr>
<td>Throttling</td>
<td>16-5</td>
</tr>
<tr>
<td>Bidirectional Testing</td>
<td>16-5</td>
</tr>
<tr>
<td>Distributed Testing</td>
<td>16-5</td>
</tr>
<tr>
<td>17</td>
<td>Configuring and Using Coherence*Extend</td>
</tr>
<tr>
<td>General Instructions</td>
<td>17-1</td>
</tr>
<tr>
<td>Configuring and Using Coherence*Extend-TCP</td>
<td>17-2</td>
</tr>
<tr>
<td>Client-side Cache Configuration Descriptor</td>
<td>17-2</td>
</tr>
<tr>
<td>Cluster-side Cache (a.k.a Coherence Extend Proxy) Configuration Descriptor</td>
<td>17-4</td>
</tr>
<tr>
<td>Launching an Extend-TCP DefaultCacheServer Process</td>
<td>17-5</td>
</tr>
<tr>
<td>Launching an Extend-TCP Client Application</td>
<td>17-5</td>
</tr>
<tr>
<td>Sample Coherence*Extend Client Application</td>
<td>17-5</td>
</tr>
<tr>
<td>Coherence*Extend InvocationService</td>
<td>17-6</td>
</tr>
<tr>
<td>Advanced Configuration</td>
<td>17-6</td>
</tr>
<tr>
<td>Network Filters</td>
<td>17-6</td>
</tr>
<tr>
<td>Connection Error Detection and Failover</td>
<td>17-8</td>
</tr>
<tr>
<td>Read-only NamedCache Access</td>
<td>17-9</td>
</tr>
<tr>
<td>Client-side NamedCache Locking</td>
<td>17-9</td>
</tr>
<tr>
<td>Disabling Proxied Services</td>
<td>17-9</td>
</tr>
<tr>
<td>18</td>
<td>High Resolution Timesource (Linux)</td>
</tr>
<tr>
<td>19</td>
<td>Performance Tuning</td>
</tr>
<tr>
<td>Operating System Tuning</td>
<td>19-1</td>
</tr>
<tr>
<td>Socket Buffer Sizes</td>
<td>19-1</td>
</tr>
<tr>
<td>High Resolution timesource (Linux)</td>
<td>19-2</td>
</tr>
<tr>
<td>Datagram size (Microsoft Windows)</td>
<td>19-3</td>
</tr>
<tr>
<td>Thread Scheduling (Microsoft Windows)</td>
<td>19-3</td>
</tr>
<tr>
<td>Swapping</td>
<td>19-4</td>
</tr>
<tr>
<td>Network Tuning</td>
<td>19-4</td>
</tr>
<tr>
<td>Network Interface Settings</td>
<td>19-4</td>
</tr>
<tr>
<td>Bus Considerations</td>
<td>19-5</td>
</tr>
<tr>
<td>Network Infrastructure Settings</td>
<td>19-5</td>
</tr>
<tr>
<td>Ethernet Flow-Control</td>
<td>19-5</td>
</tr>
<tr>
<td>Path MTU</td>
<td>19-6</td>
</tr>
<tr>
<td>JVM Tuning</td>
<td>19-6</td>
</tr>
<tr>
<td>Server Mode</td>
<td>19-6</td>
</tr>
<tr>
<td>Sizing the Heap</td>
<td>19-6</td>
</tr>
<tr>
<td>GC Monitoring &amp; Tuning</td>
<td>19-7</td>
</tr>
</tbody>
</table>
## Coherence Network Tuning

Validation................................................................................................................ 19-7

### Setting Single Server Mode

Setting Single Server Mode in the Operation Configuration Descriptor ................. 20-1
Setting Single Server Mode on the Command Line.................................................. 20-2

## Part III Managing and Monitoring Oracle Coherence

### 21 How to Manage Coherence Using JMX

Add JMX libraries to the Coherence classpath ....................................................... 21-1
Configure the Coherence Management Framework .................................................. 21-2
Access Coherence MBeans ....................................................................................... 21-3
Using Coherence MBeanConnector to Access MBeans .......................................... 21-5
Configuring Management Refresh Methodology .................................................... 21-6

### 22 JMX Reporter

Basic Configuration ................................................................................................. 22-1
Administration ........................................................................................................ 22-1
Data Analysis .......................................................................................................... 22-3
Advanced Configuration .......................................................................................... 22-4
Creating Custom Reports......................................................................................... 22-4
Running Reporter in a Distributed Configuration .................................................... 22-4

### 23 How to Create a Custom Report

Configuring a Report File......................................................................................... 23-1
  file-name Element ............................................................................................... 23-1
  file-name Macros ............................................................................................... 23-1
  file-name Macro Examples ............................................................................... 23-2
Specifying Data Columns......................................................................................... 23-2
  How to Include an Attribute ............................................................................... 23-2
  How to Include Part of the Key ......................................................................... 23-3
  How to Include Information from Composite Attributes .................................... 23-3
  How to Include Information from Multiple MBeans ........................................... 23-3
  Including Multiple MBean Information Example .............................................. 23-3
  How to Use Report Macros ............................................................................... 23-4
  How to Include Constant Values ....................................................................... 23-5
Including Queries in a Report ................................................................................ 23-5
  Using Filters to Construct Reports .................................................................... 23-5
  Using Functions to Construct a Report ............................................................... 23-9
    Function Examples ........................................................................................... 23-9
  Using Aggregates to Construct a Report ............................................................ 23-10
    Aggregate Examples ......................................................................................... 23-11
Constructing Delta Functions .................................................................................. 23-11
  Delta Function Examples .................................................................................... 23-12
24 How to Modify Report Batch

Report Batch Deployment Descriptor .................................................................................. 24-1
  Document Location........................................................................................................ 24-1
  Document Root............................................................................................................ 24-1
  System Properties....................................................................................................... 24-1
  Document Format ........................................................................................................ 24-2

Report Batch Element Index ............................................................................................. 24-3
  frequency...................................................................................................................... 24-4
  location......................................................................................................................... 24-5
  init-param................................................................................................................... 24-6
  init-params.................................................................................................................. 24-7
  output-directory........................................................................................................ 24-8
  param-name ............................................................................................................... 24-9
  param-type.................................................................................................................. 24-10
  param-value............................................................................................................... 24-11
  report-config............................................................................................................. 24-12
  report-group.............................................................................................................. 24-13
  report-list ................................................................................................................... 24-14

25 Analyzing Reporter Content

Network Health .............................................................................................................. 25-1
Network Health Detail ................................................................................................... 25-1
Memory Status ............................................................................................................. 25-2
Cache Size ................................................................................................................... 25-3
Service Report ............................................................................................................ 25-4
Node List ..................................................................................................................... 25-5
Proxy Report ................................................................................................................. 25-5

26 How to Run a Report on Demand

How to Run ReportControl MBean at Node Startup ..................................................... 26-1
How to Configure the ReportControl MBean ............................................................... 26-2

27 Configuring Custom MBeans

Creating an MBean XML Configuration File ............................................................... 27-1
  Configuring Standard MBeans .................................................................................. 27-1
  Configuring MXBeans .............................................................................................. 27-1
  Configuring JMX MBeans ......................................................................................... 27-2

Enabling a Custom MBean Configuration File ............................................................ 27-3
  Setting a System Property ......................................................................................... 27-3
  Adding a Custom MBean Configuration File to the Class Path................................. 27-3

28 How to Manage Custom MBeans Within the Cluster

Custom MBean Configuration ......................................................................................... 28-1
  How to Add a Standard MBean to Coherence ......................................................... 28-1
  How to Programatically Add a Standard MBean to Coherence ......................... 28-1
A Production Checklist

Network ........................................................................................................... A-2
Hardware .......................................................................................................... A-4
Operating System .......................................................................................... A-7
JVM .................................................................................................................. A-8
Java Security Manager ................................................................................... A-10
Application Instrumentation ........................................................................ A-10
Coherence Editions and Modes ....................................................................... A-10
  Ensuring that RTC nodes don’t use Coherence TCMP .................................... A-11
Coherence Operational Configuration ............................................................. A-12
Coherence Cache Configuration ..................................................................... A-12
Large Cluster Configuration ......................................................................... A-14
Death Detection ............................................................................................. A-14
tangosol-license.xml Deprecated ................................................................ A-15

B Types of Caches in Coherence

Distributed Cache .......................................................................................... B-1
Replicated Cache ............................................................................................ B-1
Optimistic Cache ............................................................................................ B-1
Near Cache ...................................................................................................... B-1
Local Cache ..................................................................................................... B-2
Remote Cache .................................................................................................. B-2
Summary of Cache Types .............................................................................. B-2

C Cache Semantics

D Cache Configuration Elements

Cache Configuration Deployment Descriptor ................................................. D-1
  Document Location ......................................................................................... D-1
  Document Root .............................................................................................. D-1
  Document Format .......................................................................................... D-1
  Command Line Override .............................................................................. D-1
  Examples ....................................................................................................... D-2
Element Index ............................................................................................... D-3
  acceptor-config ............................................................................................. D-5
  address-provider ........................................................................................... D-7
  async-store-manager .................................................................................... D-8
  backing-map-scheme ..................................................................................... D-10
  backup-storage .............................................................................................. D-11
  bdb-store-manager ....................................................................................... D-13
  bundle-config ............................................................................................... D-14
  cache-config ................................................................................................ D-15
  cache-mapping ............................................................................................. D-16
  cache-service-proxy ..................................................................................... D-17
F Sample Cache Configurations

Local Caches (accessible from a single JVM)
In-memory Cache .................................................................................................................. F-2
NIO In-memory Cache ........................................................................................................... F-2
Size Limited In-memory Cache ............................................................................................ F-2
In-memory Cache with Expiring Entries ................................................................................ F-2
Cache on Disk ....................................................................................................................... F-3
Size Limited Cache on Disk .................................................................................................. F-3
Persistent Cache on Disk ..................................................................................................... F-3
In-memory Cache with Disk Based Overflow ....................................................................... F-4
Cache of a Database ............................................................................................................. F-4

**Clustered Caches (accessible from multiple JVMs)** ............................................................... F-5
Replicated Cache .................................................................................................................. F-5
Replicated Cache with Overflow .......................................................................................... F-5
Partitioned Cache .................................................................................................................. F-6
Partitioned Cache with Overflow ......................................................................................... F-6
Partitioned Cache of a Database ........................................................................................... F-6
Partitioned Cache with a Serializer ..................................................................................... F-7
Local Cache of a Partitioned Cache (Near cache) ................................................................ F-7

**G Sample CacheStores**

Sample CacheStore ............................................................................................................... G-1
Sample Controllable CacheStore ........................................................................................... G-6

**H Operational Configuration Elements**

Operational Configuration Deployment Descriptors ................................................................ H-1
  Document Location ............................................................................................................... H-1
  Document Root .................................................................................................................... H-1
  Document Format ............................................................................................................... H-1
  Operational Override File (tangosol-coherence-override.xml) ............................................. H-2
  Command Line Override ..................................................................................................... H-2

**Element Index** .................................................................................................................. H-3
  access-controller ................................................................................................................ H-5
  address-provider .................................................................................................................. H-6
  authorized-hosts .................................................................................................................. H-7
  burst-mode .......................................................................................................................... H-8
  callback-handler ................................................................................................................... H-9
  class-name ........................................................................................................................... H-10
  cluster-config ..................................................................................................................... H-11
  coherence ............................................................................................................................. H-12
  configurable-cache-factory-config .................................................................................. H-13
  filters .................................................................................................................................... H-14
  flow-control ........................................................................................................................ H-15
  host-range ............................................................................................................................ H-16
  incoming-message-handler ................................................................................................ H-17
  init-param ............................................................................................................................. H-18
  init-params .......................................................................................................................... H-19
  license-config .................................................................................................................... H-20
J POF User Type Configuration Elements

POF User Type Deployment Descriptor ......................................................... J-1
  Document Location .............................................................................. J-1
  Document Root .................................................................................... J-1
  Document Format ................................................................................ J-1
  Command Line Override ..................................................................... J-2
Element Index ......................................................................................... J-3
  allow-interfaces .............................................................................. J-4
  allow-subclasses ............................................................................. J-5
  class-name ......................................................................................... J-6
  include ............................................................................................... J-7
  init-param ......................................................................................... J-8
  init-params ....................................................................................... J-9
The PIF-POF Binary Format

Stream Format

Integer Values
Type Identifiers

Binary Formats for Predefined Types

Int
Coercion of Integer Types
Decimal
Floating Point
Boolean
Octet
Octet String
Char
Char String
Date
Year-Month Interval
Time
Time Interval
Date-Time
Coercion of Date and Time Types
Day-Time Interval
Collections
Arrays
Sparse Arrays
Key-Value Maps (Dictionaries)
Identity
Reference

Binary Format for User Types
Versioning of User Types

Log Message Glossary

TCMP Log Messages
Configuration Log Messages
Partitioned Cache Service Log Messages
List of Figures

1–1 Interacting with the Cache through a Browser................................................................. 1-9
1–2 Running SimpleCacheExplorer.jsp with a Distributed Cache ........................................ 1-12
1–3 Running SimpleCacheExplorer.jsp with an Eviction Policy ........................................ 1-15
14–1 Coherence Throughput versus Number of Machines.................................................. 14-2
14–2 Coherence Latency versus Number of Machines.......................................................... 14-3
21–1 Viewing the HttpAdapter Web Application in a Browser .......................................... 21-4
21–2 Using the JConsole Utility to Display and Manipulate Coherence MBeans................ 21-5
22–1 Reporter Attributes in JConsole............................................................................... 22-2
22–2 Reporter Operations in JConsole.................................................................................. 22-3
26–1 Reporter Operations in JConsole.................................................................................. 26-1
27–1 MBean Query Displayed in the JConsole................................................................. 27-3
28–1 JMX Query Run in JConsole......................................................................................... 28-2
O–1 Average Aggregation Time............................................................................................. O-6
O–2 Aggregation Scale-Out................................................................................................. O-7
Preface

Oracle Coherence is a JCache-compliant in-memory caching and data management solution for clustered J2EE applications and application servers. Coherence makes sharing and managing data in a cluster as simple as on a single server. It accomplishes this by coordinating updates to the data using clusterwide concurrency control, replicating and distributing data modifications across the cluster using the highest performing clustered protocol available, and delivering notifications of data modifications to any servers that request them. Developers can easily take advantage of Coherence features using the standard Java collections API to access and modify data, and use the standard JavaBean event model to receive data change notifications. Functionality such as HTTP Session Management is available out-of-the-box for applications deployed to WebLogic, WebSphere, Tomcat, Jetty and other Servlet 2.2, 2.3 and 2.3 compliant application servers.

Audience

This document is targeted at software developers and architects. It provides detailed technical information on creating and using the Coherence cache and for writing and deploying Java applications that interact with it.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.
Accessibility of Links to External Web Sites in Documentation
This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Deaf/Hard of Hearing Access to Oracle Support Services
To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html.

Related Documents
For more information, see the following documents that are included in the Oracle Coherence documentation set:
- Oracle Coherence Client Guide
- Oracle Coherence Getting Started Guide
- Oracle Coherence Integration Guide for Oracle Coherence
- Oracle Coherence Java API Reference
- Oracle Coherence Tutorial for Oracle Coherence
- Oracle Coherence User’s Guide for Oracle Coherence*Web

Conventions
The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>boldface</td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td>italic</td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
This section contains the following chapters:

- Chapter 1, "Create and Use Coherence Caches"
- Chapter 2, "Implement Transactions, Locks, and Concurrency"
- Chapter 3, "Perform Continuous Query"
- Chapter 4, "Managing Map Operations with Triggers"
- Chapter 5, "Data Affinity"
- Chapter 6, "Query the Cache"
- Chapter 7, "Security Framework"
- Chapter 8, "Network Filters"
- Chapter 9, "Priority Tasks"
- Chapter 10, "Specifying a Custom Eviction Policy"
- Chapter 11, "Serialization Paged Cache"
- Chapter 12, "Pre-Loading the Cache"
- Chapter 13, "Constraints on Re-entrant Calls"
Create and Use Coherence Caches

The simplest and most flexible way to create caches in Coherence is to use the cache configuration descriptor to define attributes and names for your application’s or cluster’s caches, and to instantiate the caches in your application code referring to them by name that matches the names or patterns as defined in the descriptor.

This approach to configuring and using Coherence caches has several very important benefits. It separates the cache initialization and access logic for the cache in your application from its attributes and characteristics. This way your code is written in a way that is independent of the cache type that will be used in your application deployment and changing the characteristics of each cache (such as Rich Text cache type, cache eviction policy, and cache type-specific attributes, and so on) can be done without making any changes to the code whatsoever. It enables you to create multiple configurations for the same set of named caches and to instruct your application to use the appropriate configuration at deployment time by specifying the descriptor to use in the java command line when the node JVM is started.

Creating a Cache in Your Application

To instantiate a cache in your application code, you need to:

1. Make sure that coherence.jar is in your classpath.
2. Use CacheFactory.getCache() to access the cache in your code.

Your code will look similar to the following:

```java
import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

NamedCache cache = CacheFactory.getCache("VirtualCache");
```

Now you can retrieve and store objects in the cache, using the NamedCache API, which extends the standard java.util.Map interface, adding several additional capabilities that provide concurrency control (ConcurrentMap interface), ability to listen for cache changes (ObservableMap interface) and ability to query the cache (QueryMap interface).

Example 1–1 illustrates typical cache operations in a Java program.

**Example 1–1  Typical Cache Operations in a Java Program**

```java
// simple retrieve and update cycle
String key = "key";
```
Configuring the Caches

The cache attributes and settings are defined in the cache configuration descriptor. Cache attributes determine the cache type (what means and resources the cache will use for storing, distributing and synchronizing the cached data) and cache policies (what happens to the objects in the cache based on cache size, object longevity and other parameters).

The structure of the cache configuration descriptor (described in detail by the cache-config.dtd included in the coherence.jar) consists of two primary sections: caching-schemes section and caching-scheme-mapping section.

The caching-schemes section is where the attributes of a cache or a set of caches get defined. The caching schemes can be of several types, each with its own set of attributes. The caching schemes can be defined completely from scratch, or can incorporate attributes of other existing caching schemes, referring to them by their scheme-names (using a scheme-ref element) and optionally overriding some of their attributes to create new caching schemes. This flexibility enables you to create caching scheme structures that are easy to maintain, foster reuse and are very flexible.

The caching-scheme-mapping section is where the specific cache name or a naming pattern is attached to the cache scheme that defines the cache configuration to use for the cache that matches the name or the naming pattern. So if we would like to define the cache descriptor for the cache we mentioned in the previous section (VirtualCache), it may look something like the following:

Example 1–2  Sample Cache Configuration File

```xml
<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">
<cache-config>
  <caching-scheme-mapping>
    <!-- Caches with any name will be created as default replicated. -->
    <cache-mapping>
      <cache-name>*</cache-name>
      <scheme-name>default-replicated</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>
  <caching-schemes>
    <!-- Default Replicated caching scheme. -->
    <replicated-scheme>
      <scheme-name>default-replicated</scheme-name>
    </replicated-scheme>
  </caching-schemes>
</cache-config>
```
Configuring the Caches

Create and Use Coherence Caches

The above cache configuration descriptor specifies that all caches will be created (including our VirtualCache cache) using the default-replicated caching scheme. It defines the default-replicated caching scheme as a replicated-scheme, using a service named ReplicatedCache and using the backing map named default-backing-map, which is defined as a class com.tangosol.util.SafeHashMap (the default backing map storage that Coherence uses when no eviction policies are required).

Then, at a later point, let's say we decide that, since the number of entries that our cache is holding is too large and updates to the objects too frequent to use a replicated cache, we want our VirtualCache cache to become a distributed cache instead (while keeping all other caches replicated). To accommodate these new circumstances, we can change the cache configuration by adding the following cache-scheme definition for the distributed cache to the caching-schemes section:

Example 1–3  cache-scheme Definition for a Distributed Cache

```
<!--
Default Distributed caching scheme.
-->  
<distributed-scheme>
  <scheme-name>default-distributed</scheme-name>
  <service-name>DistributedCache</service-name>
  <backing-map-scheme>
    <class-scheme>
      <scheme-ref>default-backing-map</scheme-ref>
    </class-scheme>
  </backing-map-scheme>
</distributed-scheme>
```

Then mapping the VirtualCache cache to it in the caching-schemes-mapping section:

Example 1–4  Mapping to a Distributed Cache

```
<cache-mapping>
  <cache-name>VirtualCache</cache-name>
  <scheme-name>default-distributed</scheme-name>
</cache-mapping>
```

The resulting cache definition descriptor will look similar to Example 1–5:
Example 1–5  Configuration for a Distributed Cache

<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">

<cache-config>
  <caching-scheme-mapping>
    <!--
    Caches with any name will be created as default replicated.
    -->
    <cache-mapping>
      <cache-name>*</cache-name>
      <scheme-name>default-replicated</scheme-name>
    </cache-mapping>
    <cache-mapping>
      <cache-name>VirtualCache</cache-name>
      <scheme-name>default-distributed</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <!--
    Default Replicated caching scheme.
    -->
    <replicated-scheme>
      <scheme-name>default-replicated</scheme-name>
      <service-name>ReplicatedCache</service-name>

      <backing-map-scheme>
        <class-scheme>
          <scheme-ref>default-backing-map</scheme-ref>
        </class-scheme>
      </backing-map-scheme>
    </replicated-scheme>

    <!--
    Default Distributed caching scheme.
    -->
    <distributed-scheme>
      <scheme-name>default-distributed</scheme-name>
      <service-name>DistributedCache</service-name>

      <backing-map-scheme>
        <class-scheme>
          <scheme-ref>default-backing-map</scheme-ref>
        </class-scheme>
      </backing-map-scheme>
    </distributed-scheme>

    <!--
    Default backing map scheme definition used by all
    The caches that do not require any eviction policies
    -->
    <class-scheme>
      <scheme-name>default-backing-map</scheme-name>
      <class-name>com.tangosol.util.SafeHashMap</class-name>
    </class-scheme>
  </caching-schemes>
</cache-config>
When we revise and deploy the descriptor and restart the cluster, the VirtualCache cache will be a distributed cache instead of replicated, all without any changes to the code we wrote.

**Cache Configuration Descriptor Location**

A few words about how to instruct Coherence where to find the cache configuration descriptor. Without specifying anything in the Java command line, Coherence will attempt to use the cache configuration descriptor named coherence-cache-config.xml that it finds in the classpath. Since Coherence ships with this file packaged into the coherence.jar, unless you place another file with the same name in the classpath location preceding coherence.jar, that is the one that Coherence will use. You can tell Coherence to use a different default descriptor by using the `-Dtangosol.coherence.cacheconfig` java command line property as follows:

```java
java -Dtangosol.coherence.cacheconfig=/cfg/my-config.xml AppServer
```

The above command instructs Coherence to use `my-config.xml` file in `/cfg` directory as the default cache configuration descriptor. As you can see, this capability can give you the flexibility to modify the cache configurations of your applications without making any changes to the application code and by simply specifying different cache configuration descriptors at application deployment or start-up.

**Putting It all Together: Your First Coherence Cache Example**

Let's try walking through creating a working example cache using the caches and the cache configuration descriptor we described in the previous section. The easiest way to initially do that is to use the Coherence command line application. A couple of general comments regarding this example before we get started:

- In the examples we refer to the 'nodes' or 'JVMs'. We make no assumption regarding where they will run - you can run all of them on the same machine multiple machines or a combination of multiple nodes per machine and multiple machines. To see the clustered cache in action you will need at least 2 nodes to see the JVMs sharing data (all the following examples were captured with 2 JVMs on a single machine).

- This example uses Windows conventions and commands but it will work equally well in any of the UNIX environments (with the appropriate adjustments for the UNIX commands and conventions) and we encourage you to try it on multiple machines with different operating systems, as this is the way Coherence is designed to function: on multiple platforms simultaneously.

**Setting Up Your Test Environment**

To set up the test environment, you will need install Coherence by unzipping the software distribution in the desired location on one or more machines.

The coherence/examples directory contains the following examples that will be used in this exercise:

- examples/config/explore-config.xml

  is the configuration descriptor used by the test environment example.
Setting Up Your Test Environment

- examples/java/com/tangosol/examples/explore/SimpleCacheExplorer.java

  is the Java class that demonstrates how you can access the cache from a command line.

To deploy and run it, you need to execute the following Java command line (from the coherence directory):

- **In Windows:**
  
  ```
  java -cp .\\lib\coherence.jar;\\examples\\java
  -Dtangosol.coherence.cacheconfig=\\examples\\config\\explore-config.xml
  com.tangosol.examples.explore.SimpleCacheExplorer
  ```

- **In UNIX:**
  
  ```
  java -cp ./lib/coherence.jar:./examples/java
  -Dtangosol.coherence.cacheconfig=./examples/config/explore-config.xml
  com.tangosol.examples.explore.SimpleCacheExplorer
  ```

You should see something like the following when you bring it up:

**Example 1–6  Output from Starting a Coherence Server**

```
D:\coherence>java -cp .\\lib\coherence.jar;\\examples\\java
-Dtangosol.coherence.cacheconfig=\\examples\\config\\explore-config.xml
com.tangosol.examples.explore.SimpleCacheExplorer
2008-09-15 16:54:18.745 Oracle Coherence 3.4/405(thread=main, member=n/a): Loaded operational configuration from
resource "jar:file:/D:/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2008-09-15 16:54:18.745 Oracle Coherence 3.4/405 (thread=main, member=n/a): Loaded operational overrides from
resource "jar:file:/D:/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2008-09-15 16:54:18.755 Oracle Coherence 3.4/405 (thread=main, member=n/a): Optional configuration override
"/tangosol-coherence-override.xml" is not specified
2008-09-15 16:54:18.755 Oracle Coherence 3.4/405 (thread=main, member=n/a): Optional configuration override
"/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.4/405
Grid Edition: Development mode
Copyright (c) 2000-2008 Oracle. All rights reserved.

2008-09-15 16:54:18.945 Oracle Coherence GE 3.4/405(thread=main, member=n/a): Loaded cache configuration from
file "D:\coherence\examples\config\explore-config.xml"
2008-09-15 16:54:19.716 Oracle Coherence GE 3.4/405(thread=Cluster, member=n/a): Service Cluster joined the
cluster with senior service member n/a
"cluster:0x19DB" with Member[Id=1, Timestamp=2008-09-15 16:54:19.396,
Address=xxx.xxx.xxx.xxx:8088, MachineId=6522,
Location=site:mydomain.com, machine:mycomputer, process:3500,
CpuCount=1, SocketCount=1] UID=0x0A8F9C7A0000011BE70BDE04197A1F98
2008-09-15 16:54:23.001 Oracle Coherence GE 3.4/405(thread=ReplicatedCache,
member=1): Service ReplicatedCache
joined the cluster with senior service member 1
```
Setting Up Your Test Environment

Create and Use Coherence Caches

Command:

Type `Help` to view the `SimpleCacheExplorer` command line options. You may need to press `Enter` to display the Command: prompt.

**Example 1–7  Output from the help Command**

Command: help

clear
get
keys
info
put
quit
remove

Command:

Type `info` to display configuration and member information (please note that in the following example there are 2 cluster members active):

**Example 1–8  Output from the info Command**

Command: info

>> VirtualCache cache is using a cache-scheme named ‘default-replicated’ defined as:
<replicated-scheme>
</replicated-scheme>

>> The following member nodes are currently active:

Command:

You can also put a value into the cache:

**Example 1–9  Putting a Value into the Cache**

Command: put 1 One

>> Put Complete

Command:

And retrieve a value from the cache:
Example 1–10   Retrieving a Value from the Cache

Command: get 1

>> Value is One

Command:

Try these commands from multiple sessions and see the results. The examples/jsp/explore/SimpleCacheExplorer.jsp is the JSP file that can be used with your favorite application server:

- To deploy and run it, you will need to deploy the JSP to the default web applications directory of your application server (along with the contents of the examples/jsp/images directory), modify the server start-up script to make sure that the classpath includes coherence.jar, and specify the location of the cache configuration file on the Java command line using the -Dtangosol.coherence.cacheconfig option (for example, -Dtangosol.coherence.cacheconfig=$COHERENCE_HOME/examples/config/explore-config.xml).

- You can then start one or more instances of the application server (on different machines or different ports) and access the SimpleCacheExplorer.jsp from the browser. You should see something like the following when you bring it up:
As with the command line application try adding, updating, and removing entries from multiple instances of the application server. Also please notice the information about the cache configuration and cluster membership at the bottom of the page. As cluster members are added and removed, this information will change.

Modifying the Cache Configuration

When you are comfortable with the test setup, let's change the cache configuration and test our changes, using this simple test harness. Please remember that after each cache configuration change all the cluster members need to be shut down and then restarted (whether you are using application server instances or just plain Java JVMs). All our tests are configured to use coherence/examples/config/explore-config.xml, so this is the file that must be edited to make cache configuration changes. Let's make the first change we described previously, changing the VirtualCache to be a distributed cache by adding the following (bolded) sections:

```xml
<cache-name>VirtualCache</cache-name>
<distribution-mode>true</distribution-mode>
```
Example 1–11 Specifying a Distributed Cache in the cache-config File

<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">

<cache-config>
  <caching-scheme-mapping>
    <!--
    Caches with any name will be created as default replicated.
    -->
    <cache-mapping>
      <cache-name>*</cache-name>
      <scheme-name>default-replicated</scheme-name>
    </cache-mapping>
    <cache-mapping>
      <cache-name>VirtualCache</cache-name>
      <scheme-name>default-distributed</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <!--
    Default Replicated caching scheme.
    -->
    <replicated-scheme>
      <scheme-name>default-replicated</scheme-name>
      <service-name>ReplicatedCache</service-name>

      <backing-map-scheme>
        <class-scheme>
          <scheme-ref>default-backing-map</scheme-ref>
        </class-scheme>
      </backing-map-scheme>
    </replicated-scheme>

    <!--
    Default Distributed caching scheme.
    -->
    <distributed-scheme>
      <scheme-name>default-distributed</scheme-name>
      <service-name>DistributedCache</service-name>

      <backing-map-scheme>
        <class-scheme>
          <scheme-ref>default-backing-map</scheme-ref>
        </class-scheme>
      </backing-map-scheme>
    </distributed-scheme>

    <!--
    Default backing map scheme definition used by all
    The caches that do not require any eviction policies
    -->
    <class-scheme>
      <scheme-name>default-backing-map</scheme-name>

      <class-name>com.tangosol.util.SafeHashMap</class-name>
    </class-scheme>
  </caching-schemes>
</cache-config>
After the changes are saved, the test instances are restarted and you have had a chance to do some test data entry to see how the cache behaves, you should see the following in the cache configuration section of the tests:

- **SimpleCacheExplorer.java:**

**Example 1–12 Running SimpleCacheExplorer.java with a Distributed Cache**

Command: info

```
>> VirtualCache cache is using a cache-scheme named 'default-distributed' defined as:
   <distributed-scheme>
     <scheme-name>default-distributed</scheme-name>
     <service-name>DistributedCache</service-name>
     <backing-map-scheme>
       <class-scheme>
         <scheme-ref>default-backing-map</scheme-ref>
       </class-scheme>
     </backing-map-scheme>
   </distributed-scheme>

>> The following member nodes are currently active:
Member(Id=1, Timestamp=2008-09-15 17:53:22.701, Address=xxx.xxx.xxx.xxx:8088,
MachineId=6522, Location=site:mydomain.com,
machine:mycomputer,process:3156, Role=TangosolSimpleCacheExplorer) <-- this node
Member(Id=2, Timestamp=2008-09-15 17:54:37.619, Address=xxx.xxx.xxx.xxx:8089,
MachineId=6522, Location=site:mydomain.com,
machine:mycomputer,process:916, Role=TangosolSimpleCacheExplorer)
```

Command:

- **SimpleCacheExplorer.jsp:**
As you can see, our VirtualCache cache is now distributed according to the cache configuration descriptor.

Now let's add an eviction policy for our default distributed cache, limiting its size to 5 entries (per node) and setting the entry expiry to 60 seconds with an LRU eviction policy. To do that we need to make the following (bolded) changes to our descriptor:

Example 1–13 Adding an Eviction Policy to a cache-config File

```xml
<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">
<cache-config>
  <caching-scheme-mapping>
    <!-- Caches with any name will be created as default replicated. -->
    <cache-mapping>
      <cache-name>*</cache-name>
      <scheme-name>default-replicated</scheme-name>
    </cache-mapping>
    <class-schema>
      <default-eviction-map>
        <cache-name>*</cache-name>
        <scheme-name>default-replicated</scheme-name>
      </default-eviction-map>
    </class-schema>
  </caching-scheme-mapping>
</cache-config>
```
Note that we defined a general purpose local-scheme 'default-eviction' (with no size limit, 5 minute expiry and a HYBRID eviction policy) and then used it by reference (using scheme-ref) for our default-distributed scheme definition, overriding it's configuration settings to match our requirements.
After the changes are saved, the test instances are restarted and you have had a chance to do some test data entry to see how the cache behaves, you should see the following in the cache configuration section of the tests:

- SimpleCacheExplorer.java:

**Example 1–14 Running SimpleCacheExplorer.java with an Eviction Policy**

Command: info

>> VirtualCache cache is using a cache-scheme named 'default-distributed' defined as:

```xml
<distributed-scheme>
  <scheme-name>default-distributed</scheme-name>
  <service-name>DistributedCache</service-name>
  <backing-map-scheme>
    <local-scheme>
      <scheme-ref>default-eviction</scheme-ref>
      <eviction-policy>LRU</eviction-policy>
      <high-units>5</high-units>
      <expiry-delay>60</expiry-delay>
    </local-scheme>
  </backing-map-scheme>
</distributed-scheme>
```

>> The following member nodes are currently active:

Member (Id=1, Timestamp=2008-09-15 18:10:23.148, Address=xxx.xxx.xxx.xxx:8088, MachineId=6522, Location=site:mydomain.com, machine:mycomputer, process:2960, Role=TangosolSimpleCacheExplorer) <-- this node


Command:

- SimpleCacheExplorer.jsp:
Try doing some puts and gets, carefully noting the time you last updated the specific entries. You should see that the number of entries does not exceed 5 entries per node (so if you have 2 nodes running the number of entries should not exceed 10, for 3 nodes - 15, and so on) and entries either expire after they have not been updated for 60 seconds, or when you add the 6th entry (with the least recently touched entries being ‘evicted’ from the cache first. (Hint: use the keys command in the SimpleCacheExplorer.java to see the list of keys in the cache.)

These examples show you the general approach to modifying the cache configurations without making any code changes (as you no doubt noticed we did not touch our test application's code). Please refer to the cache-config.dtd, which can be found in the coherence.jar for full details on the available cache configuration descriptor settings and the explanation of their meaning and possible settings.
Implement Transactions, Locks, and Concurrency

Coherence provides several different options to support locking, transactions, and concurrent access to data.

Concurrency Options

Coherence provides several options for managing concurrent access to data.

Table 2–1 Coherence Concurrent Access Options

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit Locking</td>
<td>The ConcurrentHashMap interface (part of the NamedCache interface) supports explicit locking operations. Many developers find this simple locking API to be the most natural approach.</td>
</tr>
<tr>
<td>Transactions</td>
<td>The TransactionMap API builds on top of the explicit locking operations to support ACID-style transactions.</td>
</tr>
<tr>
<td>Container Integration</td>
<td>For transaction management in a Java EE container, Coherence provides a JCA resource adaptor to allow transactions to be managed by using JTA. Although Coherence does not currently support XA transactions, it can participate in XA transactions as the last resource.</td>
</tr>
<tr>
<td>Entry Processors</td>
<td>Coherence also supports a lock-free programming model through the EntryProcessor API. For many transaction types, this minimizes contention and latency and improves system throughput, without compromising the fault-tolerance of data operations.</td>
</tr>
<tr>
<td>Data Source Integration</td>
<td>Guidelines on maintaining caches with local (non XA) data resources.</td>
</tr>
</tbody>
</table>

Explicit Locking

The standard NamedCache interface extends the ConcurrentHashMap interface which includes basic locking methods. Locking operations are applied at the entry level by requesting a lock against a specific key in a NamedCache:

Example 2–1 Applying Locking Operations on a Cache

```java
... 
NamedCache cache = CacheFactory.getCache("dist-cache"); 
Object key = 'example_key'; 
cache.lock(key, -1); 
try
```
Coherence lock functionality is similar to the Java `synchronized` keyword and the C# `lock` keyword: locks only block locks. Threads must cooperatively coordinate access to data through appropriate use of locking. If a thread has locked the key to an item, another thread can read the item without locking.

Locks are unaffected by server failure (and will failover to a backup server.) Locks are immediately released when the lock owner (client) fails.

Locking behavior varies depending on the timeout requested and the type of cache. A timeout of -1 will block indefinitely until a lock can be obtained, 0 will return immediately, and a value greater than 0 will wait the specified number of milliseconds before timing out. The boolean return value should be examined to ensure the caller has actually obtained the lock. See `ConcurrentMap.lock()` for more details. Note that if a timeout value is not passed to `lock()` the default is 0. With replicated caches, the entire cache can be locked by using `ConcurrentMap.LOCK_ALL` as the key, although this is usually not recommended. This operation is not supported with partitioned caches.

In both replicated and partitioned caches, gets are permitted on keys that are locked. In a replicated cache, puts are blocked, but they are not blocked in a partitioned cache. When a lock is in place, it is the responsibility of the caller (either in the same thread or the same cluster node, depending on the `lease-granularity` configuration) to release the lock. This is why locks should always be released with a finally clause (or equivalent). If this is not done, unhandled exceptions may leave locks in place indefinitely. For more information on `lease-granularity` configuration, see "DistributedCache Service Parameters".

Transactions

A `TransactionMap` can be used to update multiple items in a Coherence cache in a transaction. To perform transactions with a `TransactionMap`, the Java Transaction API (JTA) libraries must be present in the classpath. `TransactionMaps` are created using the `CacheFactory`:

```java
NamedCache cache = CacheFactory.getCache("dist-cache");
TransactionMap tmap = CacheFactory.getLocalTransaction(cache);
```

Before using a `TransactionMap`, concurrency and isolation levels should be set to the desired level. For most short lived, highly concurrent transactions, a concurrency level of `CONCUR_PESSIMISTIC` and an isolation level of `TRANSACTION_REPEATABLE_GET` is necessary. For most long lived transactions that don't occur as often, `CONCUR_OPTIMISTIC` and `TRANSACTION_REPEATABLE_GET` are good defaults. The combination of concurrency level `CONCUR_PESSIMISTIC` and isolation level `TRANSACTION_SERIALIZABLE` will lock the entire cache. As mentioned
Previously, partitioned caches do not allow the entire cache to be locked, thus this mode will not work for partitioned caches. (In general, this level of isolation is not required for reliable transaction processing.) Queries against caches (calling keySet(Filter)) or entrySet(Filter)) are always performed with READ_COMMITTED isolation. For more information about concurrency and isolation levels, see the TransactionMap API.

Here is how to set the isolation and concurrency levels:

tmap.setTransactionIsolation(TransactionMap.TRANSACTION_REPEATABLE_GET);
tmap.setConcurrency(TransactionMap.CONCUR_PESSIMISTIC);

Now the TransactionMap can be used to update the cache in a transaction:

**Example 2–2  Updating the Cache in a Transaction**

```java
import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.tangosol.util.Base;
import com.tangosol.util.TransactionMap;
import java.util.Collection;
import java.util.Collections;

public class TxMapSample extends Base {
    public static void main(String[] args) {
        // populate the cache
        NamedCache cache = CacheFactory.getCache("dist-cache");
        String key1 = "key1";
        String key2 = "key2";
        cache.put(key1, new Integer(1));
        cache.put(key2, new Integer(1));
        out("Initial value for key 1: " + cache.get(key1));
        out("Initial value for key 2: " + cache.get(key2));

        // create one TransactionMap per NamedCache
        TransactionMap mapTx = CacheFactory.getLocalTransaction(cache);
        mapTx.setTransactionIsolation(TransactionMap.TRANSACTION_REPEATABLE_GET);
        mapTx.setConcurrency(TransactionMap.CONCUR_PESSIMISTIC);

        // gather the cache(s) into a Collection
        Collection txnCollection = Collections.singleton(mapTx);
        boolean fTxSucceeded = false;

        try {
            // start the transaction
            mapTx.begin();
            int i1 = ((Integer)mapTx.get(key1)).intValue();
            mapTx.put(key1, new Integer(++i1));
            int i2 = ((Integer)mapTx.get(key2)).intValue();
        }
    }
}
```
mapTx.put(key2, new Integer(++i2));

// commit the changes
fTxSucceeded = CacheFactory.commitTransactionCollection(txnCollection, 1);
}
}
catch (Throwable t)
{
// rollback
CacheFactory.rollbackTransactionCollection(txnCollection);
}

int v1 = ((Integer) cache.get(key1)).intValue();
int v2 = ((Integer) cache.get(key2)).intValue();

out("Transaction " + (fTxSucceeded ? "succeeded" : "did not succeed"));

out("Updated value for key 1: " + v1);
out("Updated value for key 2: " + v2);
azzert(v1 == 2, "Expected value for key1 == 2");
azzert(v2 == 2, "Expected value for key2 == 2");
}

The CacheFactory API provides helper methods for committing and rolling back a collection of TransactionMap instances. The commit method uses a conventional two-phase commit (2PC) protocol. However, as with any 2PC implementation, if one of the resources fails to commit during the second phase ("commit"), the transaction may end up partially committed. Unlike traditional 2PC implementations, Coherence can guarantee that a given server will not enter an "in doubt" state during the commit phase, but other failures are possible (for example, write-through caching can cause persistent failures). Additionally, as the transaction log is stored only on the client, a client-side failure during the "commit" phase might result in a partial commit. As the "commit" phase is nonblocking (any required locks are acquired before the start of the "commit" phase), the "commit" phase is much shorter (usually no more than a few milliseconds) than the "prepare" phase and thus the exposure, while nonzero, is minimal for typical workloads.

MapListeners registered with caches that participate in a transaction will receive a MapEvent as each item is committed. There are no guarantees that events will be fired in the order that they appear in the transaction. Additionally, if the transaction updates an item multiple times, only one event will be dispatched, reflecting the final state of the item.

**Container Integration**

**JCA**

Coherence ships with a JCA 1.0 compliant resource adaptor that can be used to manage transactions in a Java EE container. It is packaged in a resource adaptor archive (RAR) file that can be deployed to any Java EE container compatible with JCA 1.0. When deployed, JTA can be used to execute the transaction:
Example 2–3  Configuration for a JCA Container

```java
String key = "key";
Context ctx = new InitialContext();
UserTransaction tx = null;
try {
    // the transaction manager from container
    tx = (UserTransaction) ctx.lookup("java:comp/UserTransaction");
    tx.begin();

    // the try-catch-finally block below is the block of code
    // that could be on an EJB and therefore automatically within
    // a transactional context
    CacheAdapter adapter = null;
    try {
        // the transaction manager from container
        adapter = new CacheAdapter(ctx, "tangosol.coherenceTx",
                                   CacheAdapter.CONCUR_OPTIMISTIC,
                                   CacheAdapter.TRANSACTION_GET_COMMITTED, 0);

        NamedCache cache = adapter.getNamedCache("dist-test",
                                                getClass().getClassLoader());

        int n = ((Integer)cache.get(key)).intValue();
        cache.put(key, new Integer(++n));
    }
    catch (Throwable t)
    {
        String sMsg = "Failed to connect: " + t;
        System.err.println(sMsg);
        t.printStackTrace(System.err);
    }
    finally {
        try {
            adapter.close();
        }
        catch (Throwable ex)
        {
            System.err.println("SHOULD NOT HAPPEN: " + ex);
        }
    }
} finally
{
    try {
        tx.commit();
    }
    catch (Throwable t)
    {
        String sMsg = "Failed to commit: " + t;
        System.err.println(sMsg);
    }
}
```
XA

Coherence can participate in an XA transaction as the last resource. This feature is supported by most transaction managers and is known by various names, such as "Last Resource Commit" or "Last Participant." In this scenario, the completion of a transaction would involve the following steps:

- prepare is called on all XA resources
- commit is called on the Coherence transaction
- if the commit is successful, commit is called on the other XA participants in the transaction.

Refer to your transaction manager's documentation on XA last resource configuration for further details on this technique.

Entry Processors

The InvocableMap superinterface of NamedCache allows for concurrent lock-free execution of processing code within a cache. This processing is performed by an EntryProcessor. In exchange for reduced flexibility compared to the more general purpose TransactionMap and ConcurrentMap explicit locking APIs, EntryProcessors provide the highest levels of efficiency without compromising data reliability.

Since EntryProcessors perform an implicit low level lock on the entries they are processing, the end user can place processing code in an EntryProcessor without having to worry about concurrency control. Note that this is not the same as the explicit lock(key) functionality provided by ConcurrentMap!

In a replicated cache or a partitioned cache running under Caching Edition, execution will happen locally on the initiating client. In partitioned caches running under Enterprise Edition or greater, the execution occurs on the node that is responsible for primary storage of the data.

InvocableMap provides three methods of starting EntryProcessors:

- Invoke an EntryProcessor on a specific key. Note that the key need not exist in the cache to invoke an EntryProcessor on it.
- Invoke an EntryProcessor on a collection of keys.
- Invoke an EntryProcessor on a Filter. In this case, the Filter will be executed against the cache entries. Each entry that matches the Filter criteria will have the EntryProcessor executed against it. For more information on Filters, see Chapter 6, "Query the Cache".

In partitioned caches running under Enterprise Edition or greater, EntryProcessors will be executed in parallel across the cluster (on the nodes that own the individual entries.) This provides a significant advantage over having a client lock all affected keys, pull all required data from the cache, process the data, place the data back in the cache, and unlock the keys. The processing occurs in parallel across multiple machines (as opposed to serially on one machine) and the network overhead of obtaining and releasing locks is eliminated.

**Note:** EntryProcessor classes must be available in the classpath for each cluster node.
Here is a sample of high level concurrency control. Code that will require network access is commented:

**Example 2–4  Concurrency Control without Using EntryProcessors**

```java
final NamedCache cache = CacheFactory.getCache("dist-test");
final String key = "key";

cache.put(key, new Integer(1));

// begin processing

// *requires network access*
if (cache.lock(key, 0))
{
  try
  {
    // *requires network access*
    Integer i = (Integer) cache.get(key);
    // *requires network access*
    cache.put(key, new Integer(i.intValue() + 1));
  }
  finally
  {
    // *requires network access*
    cache.unlock(key);
  }
}

// end processing
```

The following is an equivalent technique using an Entry Processor. Again, network access is commented:

**Example 2–5  Concurrency Control Using EntryProcessors**

```java
final NamedCache cache = CacheFactory.getCache("dist-test");
final String key = "key";

cache.put(key, new Integer(1));

// begin processing

// *requires network access*
cache.invoke(key, new MyCounterProcessor());

// end processing

...
```

```java
public static class MyCounterProcessor
extends AbstractProcessor
{
  // this is executed on the node that owns the data,
  // no network access required
  public Object process(InvocableMap.Entry entry)
  {
    Integer i = (Integer) entry.getValue();
  }
}
```
entry.setValue(new Integer(i.intValue() + 1));
return null;
}
)

EntryProcessors are individually executed atomically, however multiple EntryProcessor invocations by using InvocableMap.invokeAll() will not be executed as one atomic unit. As soon as an individual EntryProcessor has completed, any updates made to the cache will be immediately visible while the other EntryProcessors are executing. Furthermore, an uncaught exception in an EntryProcessor will not prevent the others from executing. Should the primary node for an entry fail while executing an EntryProcessor, the backup node will perform the execution instead. However if the node fails after the completion of an EntryProcessor, the EntryProcessor will not be invoked on the backup.

Note that in general, EntryProcessors should be short lived. Applications with longer running EntryProcessors should increase the size of the distributed service thread pool so that other operations performed by the distributed service are not blocked by the long running EntryProcessor. For more information on the distributed service thread pool, see "DistributedCache Service Parameters".

Coherence includes several EntryProcessor implementations for common use cases. Further details on these EntryProcessors, along with additional information on parallel data processing, can be found in "Provide a Data Grid".

**Data Source Integration**

When using write-behind and write-through to a database in a Coherence cache, transactional behavior must be taken into account. With write-through enabled, the put will succeed if the item is successfully stored in the database. Otherwise, the exception that occurred in the CacheStore will be rethrown to the client. (Note: to enable this behavior, set <rollback-cachestore-failures> to true. See "read-write-backing-map-scheme" on page D-76 for more details.) This only applies when updating one cache item at a time; if two cache items are updated at a time then each CacheStore operation will be a distinct database transaction. This limitation will be addressed in a future release of Coherence.

Write-behind caching provides much higher throughput and performance. However, writes to the database are performed after the cache has been updated. Therefore care must be taken to ensure that writes to the database will not fail. Write-behind should only be used in applications where:

- data constraints will be managed by the application, not the database
- no other application will update the database

See "Read-Through, Write-Through, Write-Behind Caching and Refresh-Ahead" for more information on cache stores.

If multiple updates to cache entries must be persisted to the database in a transaction, it may be more suitable to implement a cache-aside pattern where the client is responsible for updating the database and the cache. Note that a CacheLoader may still be used to load cache misses from the data source.
Perform Continuous Query

While it is possible to obtain a point in time query result from a Coherence cache to, and it is possible to receive events that would change the result of that query, Coherence provides a feature that combines a query result with a continuous stream of related events to maintain an up-to-date query result in a real-time fashion. This capability is called Continuous Query, because it has the same effect as if the desired query had zero latency and the query were being executed several times every millisecond! For more information on point in time query results and events, see Chapter 6, “Query the Cache” and “Deliver Events for Changes as they Occur”.

Coherence implements the Continuous Query functionality by materializing the results of the query into a Continuous Query Cache, and then keeping that cache up-to-date in real-time using event listeners on the query. In other words, a Coherence Continuous Query is a cached query result that never gets out-of-date.

Uses of Continuous Query Caching

There are several different general use categories for Continuous Query Caching:

- It is an ideal building block for Complex Event Processing (CEP) systems and event correlation engines.
- It is ideal for situations in which an application repeats a particular query, and would benefit from always having instant access to the up-to-date result of that query.
- A Continuous Query Cache is analogous to a materialized view, and is useful for accessing and manipulating the results of a query using the standard NamedCache API, and receiving an ongoing stream of events related to that query.
- A Continuous Query Cache can be used in a manner similar to a Near Cache, because it maintains an up-to-date set of data locally where it is being used, for example on a particular server node or on a client desktop; note that a Near Cache is invalidation-based, but the Continuous Query Cache actually maintains its data in an up-to-date manner.

An example use case is a trading system desktop, in which a trader’s open orders and all related information must be maintained in an up-to-date manner at all times. By combining the Coherence*Extend functionality with Continuous Query Caching, an application can support literally tens of thousands of concurrent users.
The Coherence Continuous Query Cache

The Coherence implementation of Continuous Query is found in the com.tangosol.net.cache.ContinuousQueryCache class. This class, like all Coherence caches, implements the standard NamedCache interface, which includes the following capabilities:

■ Cache access and manipulation using the Map interface: NamedCache extends the standard Map interface from the Java Collections Framework, which is the same interface implemented by the JDK’s HashMap and Hashtable classes.

■ Events for all objects modifications that occur within the cache: NamedCache extends the ObservableMap interface.

■ Identity-based clusterwide locking of objects in the cache: NamedCache extends the ConcurrentHashMap interface.

■ Querying the objects in the cache: NamedCache extends the QueryMap interface.

■ Distributed Parallel Processing and Aggregation of objects in the cache: NamedCache extends the InvocableMap interface.

Since the ContinuousQueryCache implements the NamedCache interface, which is the same API provided by all Coherence caches, it is extremely simple to use, and it can be easily substituted for another cache when its functionality is called for.

Constructing a Continuous Query Cache

There are two items that define a Continuous Query Cache:

1. The underlying cache that it is based on;
2. A query of that underlying cache that produces the sub-set that the Continuous Query Cache will cache.

The underlying cache is any Coherence cache, including another Continuous Query Cache. A cache is usually obtained from a CacheFactory, which allows the developer to simply specify the name of the cache and have it automatically configured based on the application’s cache configuration information; for example:

```java
NamedCache cache = CacheFactory.getCache("orders");
```

See Appendix D, "Cache Configuration Elements" for more information on specifying cache configuration information.

The query is the same type of query that would be used to; for example:

```java
Filter filter = new AndFilter(new EqualsFilter("getTrader", traderid),
                           new EqualsFilter("getStatus", Status.OPEN));
```

See Chapter 6, "Query the Cache" for more information on queries.

Note: Continuous Query Caches are useful in almost every type of application, including both client-based and server-based applications, because they provide the ability to very easily and efficiently maintain an up-to-date local copy of a specified sub-set of a much larger and potentially distributed cached data set.
Normally, to query a cache, one of the methods from the QueryMap is used; for examples, to obtain a snap-shot of all open trades for this trader:

**Example 3–2  Getting Data for the Continuous Query Cache**

```java
Set setOpenTrades = cache.entrySet(filter);
```

Similarly, the Continuous Query Cache is constructed from those same two pieces:

**Example 3–3  Constructing the Continuous Query Cache**

```java
ContinuousQueryCache cacheOpenTrades = new ContinuousQueryCache(cache, filter);
```

**Cleaning up the resources associated with a ContinuousQueryCache**

A Continuous Query Cache places one or more event listeners on its underlying cache. If the Continuous Query Cache is used for the duration of the application, then the resources will be cleaned up when the node is shut down or otherwise stops. However, if the Continuous Query Cache is only used for a period, then when the application is done using it, the application must call the `release()` method on the `ContinuousQueryCache`.

**Caching only keys, or caching both keys and values**

When constructing a Continuous Query Cache, it is possible to specify that the cache should only keep track of the keys that result from the query, and obtain the values from the underlying cache only when they are asked for. This feature may be useful for creating a Continuous Query Cache that represents a very large query result set, or if the values are never or rarely requested. To specify that only the keys should be cached, use the constructor that allows the `CacheValues` property to be configured; for example:

**Example 3–4  A Constructor that Allows the CacheValues Property**

```java
ContinuousQueryCache cacheOpenTrades = new ContinuousQueryCache(cache, filter, false);
```

If necessary, the `CacheValues` property can also be modified after the cache has been instantiated; for example:

**Example 3–5  Setting the CacheValues Property**

```java
cacheOpenTrades.setCacheValues(true);
```

**CacheValues Property and Event Listeners**

If the Continuous Query Cache has any standard (non-lite) event listeners, or if any of the event listeners are filtered, then the `CacheValues` property will automatically be set to true, because the Continuous Query Cache uses the locally cached values to filter events and to supply the old and new values for the events that it raises.

**Listening to the ContinuousQueryCache**

Since the Continuous Query Cache is itself observable, it is possible for the client to place one or more event listeners onto it. For example:
Listening to the ContinuousQueryCache

**Example 3–6  Adding a Listener to a Continuous Query Cache**

ContinuousQueryCache cacheOpenTrades = new ContinuousQueryCache(cache, filter);
cacheOpenTrades.addMapListener(listener);

Assuming some processing has to occur against every item that is already in the cache and every item added to the cache, there are two approaches. First, the processing could occur then a listener could be added to handle any later additions:

**Example 3–7  Processing Continuous Query Cache Entries and Adding a Listener**

ContinuousQueryCache cacheOpenTrades = new ContinuousQueryCache(cache, filter);
for (Iterator iter = cacheOpenTrades.entrySet().iterator(); iter.hasNext(); )
{
    Map.Entry entry = (Map.Entry) iter.next();
    // .. process the cache entry
}

However, that code is incorrect because it allows events that occur in the split second after the iteration and before the listener is added to be missed! The alternative is to add a listener first, so no events are missed, and then do the processing:

**Example 3–8  Adding a Listener Before Processing Continuous Query Cache Entries**

ContinuousQueryCache cacheOpenTrades = new ContinuousQueryCache(cache, filter);
cacheOpenTrades.addMapListener(listener);
for (Iterator iter = cacheOpenTrades.entrySet().iterator(); iter.hasNext(); )
{
    Map.Entry entry = (Map.Entry) iter.next();
    // .. process the cache entry
}

However, it is possible that the same entry will show up in both an event an in the Iterator, and the events can be asynchronous, so the sequence of operations cannot be guaranteed.

The solution is to provide the listener during construction, and it will receive one event for each item that is in the Continuous Query Cache, whether it was there to begin with (because it was in the query) or if it got added during or after the construction of the cache:

**Example 3–9  Providing a Listener When Constructing the Continuous Query Cache**

ContinuousQueryCache cacheOpenTrades = new ContinuousQueryCache(cache, filter, listener);

Achieving a Stable Materialized View

The ContinuousQueryCache implementation faced the same challenge: How to assemble an exact point-in-time snapshot of an underlying cache while receiving a stream of modification events from that same cache. The solution has several parts. First, Coherence supports an option for synchronous events, which provides a set of ordering guarantees. See "Deliver Events for Changes as they Occur" for more information on this option.

Secondly, the ContinuousQueryCache has a two-phase implementation of its initial population that allows it to first query the underlying cache and then subsequently resolve all of the events that came in during the first phase. Since achieving these guarantees of data visibility without any missing or repeated events is fairly complex,
the **ContinuousQueryCache** allows a developer to pass a listener during construction, thus avoiding exposing these same complexities to the application developer.

### Support for Synchronous and Asynchronous Listeners

By default, listeners to the **ContinuousQueryCache** will have their events delivered asynchronously. However, the **ContinuousQueryCache** does respect the option for synchronous events as provided by the **SynchronousListener** interface. See “**Deliver Events for Changes as they Occur**” for more information on this option.

### Making the **ContinuousQueryCache** Read-Only

The **ContinuousQueryCache** can be made into a read-only cache; for example:

**Example 3–10 Making the Continuous Query Cache Read-Only**

```java
cacheOpenTrades.setReadOnly(true);
```

A read-only **ContinuousQueryCache** will not allow objects to be added to, changed in, removed from or locked in the cache.

When a **ContinuousQueryCache** has been set to read-only, it cannot be changed back to read/write.
Map triggers supplement the standard capabilities of Oracle Coherence to provide a highly customized cache management system. For example, map triggers can be used to prevent invalid transactions, enforce complex security authorizations or complex business rules, provide transparent event logging and auditing, and gather statistics on data modifications. Other possible use for triggers include restricting operations against a cache to those issued during application re-deployment time.

For example, assume that you have code that is working with a NamedCache, and you want to change an entry’s behavior or contents before the entry is inserted into the map. The addition of a map trigger will allow you to make this change, without having to modify all the exiting code.

Map triggers could also be used as part of an upgrade process. The addition of a map trigger could prompt inserts to be diverted from one cache into another.

A map trigger in the Oracle Coherence cache is somewhat similar to a trigger that might be applied to a database. It is a functional agent represented by the MapTrigger interface that will be run in response to a pending change (or removal) of the corresponding map entry. The pending change is represented by the MapTrigger.Entry interface. This interface inherits from the InvocableMap.Entry interface, so it provides methods to retrieve, update, and remove values in the underlying map.

The MapTrigger interface contains the process method that is used to validate, reject, or modify the pending change in the map. This method is called before an operation that intends to change the underlying map content is committed. An implementation of this method can evaluate the pending change by analyzing the original and the new value and produce any of the following results:

- override the requested change with a different value
- undo the pending change by resetting the original value
- remove the entry from the underlying map
- reject the pending change by throwing a RuntimeException
- do nothing, and allow the pending change to be committed

MapTrigger functionality is typically added as part of an application start-up process. It can be added programmatically as described in the MapTrigger API, or it can be configured using the class-factory mechanism in the coherence-cache-config.xml configuration file. In this case, a MapTrigger will be registered during the very first CacheFactory.getCache(...) call for the corresponding cache. Example 4-1 assumes that the createMapTrigger method would return a new MapTriggerListener(new MyCustomTrigger());
Example 4–1  Creating a MapTriggerListener in the coherence-cache-config.xml File

```xml
<cache-config>
  ...
  <listener>
    <class-scheme>
      <class-factory-name>package.MyFactory</class-factory-name>
      <method-name>createTriggerListener</method-name>
      <init-params>
        <init-param>
          <param-type>string</param-type>
          <param-value>{cache-name}</param-value>
        </init-param>
      </init-params>
    </class-scheme>
  </listener>
  ...
</cache-config>
```

In addition to the `MapTrigger.Entry` and `MapTrigger` interfaces, Oracle Coherence provides the `FilterTrigger` and `MapTriggerListener` classes. The `FilterTrigger` is a generic `MapTrigger` implementation that will perform a predefined action if a pending change is rejected by the associated `Filter`. The `FilterTrigger` can either reject the pending operation, ignore the change and restore the entry's original value, or remove the entry itself from the underlying map.

The `MapTriggerListener` is a special purpose `MapListener` implementation that is used to register a `MapTrigger` with a corresponding `NamedCache`. In Example 4–2, `MapTriggerListener` is used to register the `PersonMapTrigger` with the `People` named cache.

Example 4–2  A MapTriggerListener Registering a MapTrigger with a Named Cache

```java
NamedCache person = CacheFactory.getCache("People");
MapTrigger trigger = new PersonMapTrigger();
person.addMapListener(new MapTriggerListener(trigger));
```

These API reside in the `com.tangosol.util` package. For more information on these API, see the Javadoc pages for `MapTrigger`, `MapTrigger.Entry`, `FilterTrigger`, and `MapTriggerListener`.

A Map Trigger Example

The code in Example 4–3 illustrates a map trigger and how it can be called. In the `PersonMapTrigger` class in Example 4–3, the `process` method is implemented to modify an entry before it is placed in the map. In this case, the last name attribute of a `Person` object is converted to upper case characters. The object is then returned to the entry.

Example 4–3  A MapTrigger Class

```java
public class PersonMapTrigger implements MapTrigger
{
  public PersonMapTrigger()
```
public void process(MapTrigger.Entry entry) {
    Person person = (Person) entry.getValue();
    String sName = person.getLastName();
    String sNameUC = sName.toUpperCase();

    if (!sNameUC.equals(sName)) {
        person.setLastName(sNameUC);
        System.out.println("Changed last name of \[" + sName + "\] to \[" +
                        person.getLastName() + "]\)
        entry.setValue(person);
    }
}

// ---- hashCode() and equals() must be implemented
public boolean equals(Object o) {
    return o != null && o.getClass() == this.getClass();
}
public int hashCode() {
    return getClass().getName().hashCode();
}

The MapTrigger in Example 4–4, calls the PersonMapTrigger. The new
MapTriggerListener passes the PersonMapTrigger to the People
NamedCache.

Example 4–4  Calling a MapTrigger and Passing it to a Named Cache

...
public static void main(String[] args) {
    NamedCache cache = CacheFactory.getCache("People");
    cache.addMapListener(createTriggerListener("People"));

    System.out.println("Installed MapTrigger into cache People");
}
Data affinity describes the concept of ensuring that a group of related cache entries is contained within a single cache partition. This ensures that all relevant data is managed on a single primary cache node (without compromising fault-tolerance).

Affinity may span multiple caches (if they are managed by the same cache service, which will generally be the case). For example, in a master-detail pattern such as an "Order-LineItem", the Order object may be co-located with the entire collection of LineItem objects that are associated with it.

The benefit is two-fold. First, only a single cache node is required to manage queries and transactions against a set of related items. Second, all concurrency operations can be managed locally, avoiding the need for clustered synchronization.

Several standard Coherence operations can benefit from affinity, including cache queries, InvocableMap operations and the getAll, putAll, and removeAll methods.

**Note:** Data affinity is specified in terms of entry keys (not values). As a result, the association information must be present in the key class. Similarly, the association logic applies to the key class, not the value class.

### Specifying Affinity

Affinity is specified in terms of a relationship to a partitioned key. In the Order-LineItem example above, the Order objects would be partitioned normally, and the LineItem objects would be associated with the appropriate Order object.

The association does not need to be directly tied to the actual parent key - it only must be a functional mapping of the parent key. It could be a single field of the parent key (even if it is non-unique), or an integer hash of the parent key. All that matters is that all child keys return the same associated key; it does not matter whether the associated key is an actual key (it is simply a "group id"). This fact may help minimize the size impact on the child key classes that don’t already contain the parent key information (as it is derived data, the size of the data may be decided explicitly, and it also will not affect the behavior of the key). Note that making the association too general (having too many keys associated with the same "group id") can cause a "lumpy" distribution (if all child keys return the same association key regardless of what the parent key is, the child keys will all be assigned to a single partition, and will not be spread across the cluster).

There are two ways to ensure that a set of cache entries are co-located. Note that association is based on the cache key, not the value (otherwise updating a cache entry
could cause it to change partitions). Also, note that while the Order will be co-located with the child LineItems, Coherence does not currently support composite operations that span multiple caches (for example, updating the Order and the collection of LineItems within a single invocation request com.tangosol.util.InvocableMap.EntryProcessor).

**Specifying Data Affinity with a KeyAssociation**

For application-defined keys, the class (of the cache key) may implement com.tangosol.net.cache.KeyAssociation as follows:

**Example 5–1 Creating a Key Association**

```java
import com.tangosol.net.cache.KeyAssociation;

public class LineItemId implements KeyAssociation
{
    // {...}

    public Object getAssociatedKey()
    {
        return getOrderId();
    }

    // {...}
}
```

**Specifying Data Affinity with a KeyAssociator**

Applications may also provide a custom KeyAssociator:

**Example 5–2 A Custom KeyAssociator**

```java
import com.tangosol.net.partition.KeyAssociator;

public class LineItemAssociator implements KeyAssociator
{
    public Object getAssociatedKey(Object oKey)
    {
        if (oKey instanceof LineItemId)
        {
            return ((LineItemId) oKey).getOrderId();
        }
        else if (oKey instanceof OrderId)
        {
            return oKey;
        }
        else
        {
            return null;
        }
    }

    public void init(PartitionedService service)
    {
    }
}
```
The key associator may be configured for a NamedCache in the associated `<distributed-scheme>` element:

**Example 5–3 Configuring a Key Associator**

```xml
<distributed-scheme>
    <!-- ... -->
    <key-associator>
        <class-name>LineItemAssociator</class-name>
    </key-associator>
</distributed-scheme>
```

**Example of Using Affinity**

**Example 5–4** illustrates how to use affinity to create a more efficient query (NamedCache.entrySet(Filter)) and cache access (NamedCache.getAll(Collection)).

**Example 5–4 Using Affinity for a More Efficient Query**

```java
OrderId orderId = new OrderId(1234);

// this Filter will be applied to all LineItem objects to fetch those
// for which getOrderId() returns the specified order identifier
// "select * from LineItem where OrderId = :orderId" Filter filterEq = new
EqualsFilter("getOrderId", orderId);

// this Filter will direct the query to the cluster node that currently owns
// the Order object with the given identifier
Filter filterAsc = new KeyAssociatedFilter(filterEq, orderId);

// run the optimized query to get the ChildKey objects
Set setLineItemKeys = cacheLineItems.keySet(filterAsc);

// get all the Child objects immediately
Set setLineItems = cacheLineItems.getAll(setLineItemKeys);

// Or remove all immediately
cacheLineItems.keySet().removeAll(setLineItemKeys);
```
Coherence can perform queries and indexes against currently cached data that meets a given set of criteria. Queries and indexes can be simple, employing filters packaged with Coherence, or they can be run against multi-value attributes such as collections and arrays.

### Query Functionality

Coherence provides the ability to search for cache entries that meet a given set of criteria. The result set may be sorted if desired. Queries are evaluated with Read Committed isolation.

It should be noted that queries apply only to currently cached data (and will not use the CacheLoader interface to retrieve additional data that may satisfy the query). Thus, the dataset should be loaded entirely into cache before queries are performed. In cases where the dataset is too large to fit into available memory, it may be possible to restrict the cache contents along a specific dimension (for example, "date") and manually switch between cache queries and database queries based on the structure of the query. For maintainability, this is usually best implemented inside a cache-aware data access object (DAO).

Indexing requires the ability to extract attributes on each Partitioned cache node; in the case of dedicated CacheServer instances, this implies (usually) that application classes must be installed in the CacheServer classpath.

For Local and Replicated caches, queries are evaluated locally against unindexed data. For Partitioned caches, queries are performed in parallel across the cluster, using indexes if available. Coherence includes a Cost-Based Optimizer (CBO). Access to unindexed attributes requires object deserialization (though indexing on other attributes can reduce the number of objects that must be evaluated).

### Simple Queries

Querying cache content is very simple:

#### Example 6–1 Querying the Cache with a Filter

```java
Filter filter = new GreaterEqualsFilter("getAge", 18);
for (Iterator iter = cache.entrySet(filter).iterator(); iter.hasNext(); )
{
    Map.Entry entry = (Map.Entry) iter.next();
    Integer key = (Integer) entry.getKey();
    Person person = (Person) entry.getValue();
    System.out.println("key= + key + " person= + person);
}
```

---

**Query the Cache**

Coherence can perform queries and indexes against currently cached data that meets a given set of criteria. Queries and indexes can be simple, employing filters packaged with Coherence, or they can be run against multi-value attributes such as collections and arrays.

### Query Functionality

Coherence provides the ability to search for cache entries that meet a given set of criteria. The result set may be sorted if desired. Queries are evaluated with Read Committed isolation.

It should be noted that queries apply only to currently cached data (and will not use the CacheLoader interface to retrieve additional data that may satisfy the query). Thus, the dataset should be loaded entirely into cache before queries are performed. In cases where the dataset is too large to fit into available memory, it may be possible to restrict the cache contents along a specific dimension (for example, "date") and manually switch between cache queries and database queries based on the structure of the query. For maintainability, this is usually best implemented inside a cache-aware data access object (DAO).

Indexing requires the ability to extract attributes on each Partitioned cache node; in the case of dedicated CacheServer instances, this implies (usually) that application classes must be installed in the CacheServer classpath.

For Local and Replicated caches, queries are evaluated locally against unindexed data. For Partitioned caches, queries are performed in parallel across the cluster, using indexes if available. Coherence includes a Cost-Based Optimizer (CBO). Access to unindexed attributes requires object deserialization (though indexing on other attributes can reduce the number of objects that must be evaluated).

### Simple Queries

Querying cache content is very simple:

#### Example 6–1 Querying the Cache with a Filter

```java
Filter filter = new GreaterEqualsFilter("getAge", 18);
for (Iterator iter = cache.entrySet(filter).iterator(); iter.hasNext(); )
{
    Map.Entry entry = (Map.Entry) iter.next();
    Integer key = (Integer) entry.getKey();
    Person person = (Person) entry.getValue();
    System.out.println("key= + key + " person= + person);
}
```
Coherence provides a wide range of filters in the `com.tangosol.util.filter` package.

A `LimitFilter` may be used to limit the amount of data sent to the client, and also to provide "paging" for users. This is illustrated in Example 6–2:

**Example 6–2  Using LimitFilter Class to Limit the Amount of Data Sent to the Client**

```java
int pageSize = 25;
Filter filter = new GreaterEqualsFilter("getAge", 18);

// get entries 1-25
Filter limitFilter = new LimitFilter(filter, pageSize);
Set entries = cache.entrySet(limitFilter);

// get entries 26-50
limitFilter.nextPage();
entries = cache.entrySet(limitFilter);
```

Any queryable attribute may be indexed with the `addIndex` method of the `QueryMap` class. This is illustrated in Example 6–3:

**Example 6–3  Indexing a Queryable Attribute**

```java
// addIndex(ValueExtractor extractor, boolean fOrdered, Comparator comparator)
cache.addIndex(extractor, true, null);
```

The `fOrdered` argument specifies whether the index structure is sorted. Sorted indexes are useful for range queries, including "select all entries that fall between two dates" and "select all employees whose family name begins with 'S'". For "equality" queries, an unordered index may be used, which may have better efficiency in terms of space and time.

The `comparator` argument can be used to provide a custom `java.util.Comparator` for ordering the index.

This method is only intended as a hint to the cache implementation, and as such it may be ignored by the cache if indexes are not supported or if the desired index (or a similar index) already exists. It is expected that an application will call this method to suggest an index even if the index may already exist, just so that the application is certain that index has been suggested. For example in a distributed environment, each server will likely suggest the same set of indexes when it starts, and there is no downside to the application blindly requesting those indexes regardless of whether another server has already requested the same indexes.

Indexes are a feature of Coherence Enterprise Edition or higher. This method will have no effect when using Coherence Standard Edition.

Note that queries can be combined by Coherence if necessary, and also that Coherence includes a cost-based optimizer (CBO) to prioritize the usage of indexes. To take advantage of an index, queries must use extractors that are equal `((Object.equals()))` to the one used in the query.

A list of applied indexes can be retrieved from the `StorageManagerMBean` by using JMX. For more information, see Chapter 21, "How to Manage Coherence Using JMX".
Querying Partitioned Caches

The Partitioned Cache implements this method using the Parallel Query feature, which is only available in Coherence Enterprise Edition or higher. When working with a Partitioned Cache in Coherence Standard Edition, this method will retrieve the data set to the client for processing.

Querying Near Caches

Although queries can be executed through a near cache, the query will not use the front portion of a near cache. If using a near cache with queries, the best approach is to use the sequence in Example 6–4:

Example 6–4 Querying the Near Cache

```java
Set setKeys = cache.keySet(filter);
Map mapResult = cache.getAll(setKeys);
```

Query Concepts

This section goes into more detail on the design of the query interface, building up from the core components.

The concept of querying is based on the `ValueExtractor` interface. A value extractor is used to extract an attribute from a given object for querying (and similarly, indexing). Most developers will need only the `ReflectionExtractor` implementation of this interface. The ReflectionExtractor uses reflection to extract an attribute from a value object by referring to a method name, typically a "getter" method like `getName()`.

```java
ValueExtractor extractor = new ReflectionExtractor("getName");
```

Any "void argument" method can be used, including `Object` methods like `toString()` (useful for prototyping/debugging). Indexes may be either traditional "field indexes" (indexing fields of objects) or "functional indexes" (indexing "virtual" object attributes). For example, if a class has field accessors `getFirstName` and `getLastName`, the class may define a function `getFullName` which concatenates those names, and this function may be indexed.

To query a cache that contains objects with `getName` attributes, a `Filter` must be used. A filter has a single method which determines whether a given object meets a criterion.

Example 6–5 Equality Filter

```java
Filter filter = new EqualsFilter(extractor, "Bob Smith");
```

Note that the filters also have convenience constructors that accept a method name and internally construct a `ReflectionExtractor`:

Example 6–6 Filter that Constructs a `ReflectionExtractor`

```java
Filter filter = new EqualsFilter("getName", "Bob Smith");
```

Example 6–7 illustrates a routine to select the entries of a cache that satisfy a particular filter:
**Example 6–7  Selecting Cache Entries that Satisfy a Filter**

for (Iterator iter = cache.entrySet(filter).iterator(); iter.hasNext(); )
{
    Map.Entry entry = (Map.Entry)iter.next();
    Integer key = (Integer)entry.getKey();
    Person person = (Person)entry.getValue();
    System.out.println("key=\" + key + \	\" person=\" + person);
}

**Example 6–8 illustrates using a filter to select and sort cache entries:**

**Example 6–8  Selecting and Sorting Cache Entries that Satisfy a Filter**

// entrySet(Filter filter, Comparator comparator)
Iterator iter = cache.entrySet(filter, null).iterator();

The additional null argument specifies that the result set should be sorted using the "natural ordering" of Comparable objects within the cache. The client may explicitly specify the ordering of the result set by providing an implementation of Comparator. Note that sorting places significant restrictions on the optimizations that Coherence can apply, as sorting requires that the entire result set be available before sorting.

**Example 6–9 illustrates using the keySet form of the queries, combined with getAll(). This technique may provide more control over memory usage:**

**Example 6–9 Using a keySet Query Format**

// keySet(Filter filter)
Set setKeys = cache.keySet(filter);
Set setPageKeys = new HashSet();
int PAGE_SIZE = 100;
for (Iterator iter = setKeys.iterator(); iter.hasNext();)
{
    setPageKeys.add(iter.next());
    if (setKeyPage.size() == PAGE_SIZE || !iter.hasNext())
    {
        // get a block of values
        Map mapResult = cache.getAll(setPageKeys);

        // process the block
        // ...

        setPageKeys.clear();
    }
}

**Queries Involving Multi-Value Attributes**

Coherence supports indexing and querying of multi-value attributes including collections and arrays. When an object is indexed, Coherence will verify if it is a multi-value type, and will then index it as a collection rather than a singleton. The ContainsAllFilter, ContainsAnyFilter and ContainsFilter are used to query against these collections.

**Example 6–10 Querying on Multi-Value Attributes**

Set searchTerms = new HashSet();
searchTerms.add("java");
searchTerms.add("clustering");
searchTerms.add('books');

// The cache contains instances of a class "Document" which has a method
// 'getWords' which returns a Collection<String> containing the set of
// words that appear in the document.
Filter filter = new ContainsAllFilter("getWords", searchTerms);

Set entrySet = cache.entrySet(filter);

// iterate through the search results
// ...

**ChainedExtractor**

The ChainedExtractor implementation allows chained invocation of
zero-argument (accessor) methods. In Example 6–11, the extractor will first use
reflection to call `getName()` on each cached `Person` object, and then use reflection to
call `length()` on the returned `String`.

**Example 6–11  Chaining Invocation Methods**

ValueExtractor extractor = new ChainedExtractor("getName.length");

This extractor could be passed into a query, allowing queries (for example) to select all
people with names not exceeding 10 letters. Method invocations may be chained
indefinitely, for example `getName.trim.length`. 
This chapter describes the following security features:
- Transport Layer Security
- Access Controller
- Proof of Identity
- Proof of Trustworthiness
- Default Access Controller implementation
- Working in applications with installed security manager

**Transport Layer Security**
For information on transport layer security, see "Encryption Filters" on page 8-1.

**Access Controller**
Security Framework in Coherence is based on a concept of Clustered Access Controller, which can be turned on (activated) by a configurable parameter or command line attribute.

The Access Controller manages access to the "clustered resources", such as clustered services and caches and controls operations that include (but not limited to) the following:
- creating a new clustered cache or service;
- joining an existing clustered cache or service;
- destroying an existing clustered cache.

The Access Controller serves three purposes:
- grant or deny access to a protected clustered resource based on the caller’s permissions
- encrypt outgoing communications based on the caller’s private credentials
- decrypt incoming communications based on the caller’s public credentials

Coherence uses a local LoginModule (see JAAS Reference Guide for details) to authenticate the caller and an Access Controller on one or more cluster nodes to verify the caller’s access rights.
The Access Controller is a pluggable component that could be declared in the Coherence deployment descriptor, `tangosol-coherence.xml`. The specified class should implement the `com.tangosol.net.security.AccessController` interface.

Coherence provides a default Access Controller implementation that is based on the Key Management infrastructure that is shipped as a standard part of Sun's JDK.

Each clustered service in Coherence maintains a concept of a "senior" service member (cluster node), which serves as a controlling agent for a particular service. While the senior member does not have to consult anyone when accessing a clustered resource, any junior node willing to join that service has to request and receive a confirmation from the senior member, which in turn notifies all other cluster nodes about the joining node.

Since Coherence is a system providing distributed data management and computing, the security subsystem is designed to operate in a partially hostile environment. We assume that when there is data shared between two cluster nodes either node could be a malicious one - lacking sufficient credentials to join a clustered service or obtain access to a clustered resource.

Let's call a cluster node that may try to gain unauthorized access to clustered resources by using nonstandard means as a "malicious" node. The means of such an access could vary. They could range from attempts to get protected or private class data using reflection, replacing classes in the distribution (coherence.jar or other application binaries), modifying classes on-the-fly using custom `ClassLoader(s)` and so on.

Alternatively, a cluster node that never attempts to gain unauthorized access to clustered resources by using nonstandard means will be called a "trusted" node. It's important to note that even a trusted node may attempt to gain access to resources without having sufficient rights, but it does so in a standard way by using the exposed standard API.

File system mechanisms (the same that is used to protect the integrity of the Java runtime libraries) and standard Java security policy could be used to resolve an issue of guarantying the trustworthiness of a given single node. In a case of inter-node communications there are two dangers that we have to consider:

- A malicious node surpasses the local access check and attempts to join a clustered service or gain access to a clustered resource controlled by a trusted node;
- A malicious node creates a clustered service or clustered resource becoming its controller.

To prevent either of these two scenarios from occurring Coherence uses two-ways encryption algorithm: all client requests must be accompanied by the proof of identity and all service responses must be accompanied by the proof of trustworthiness.

**Proof of Identity**

In a case of an active Access Controller the client code could use the following construct to authenticate the caller and perform necessary actions:

```java
import com.tangosol.net.security.Security;
import java.security.PrivilegedAction;
import javax.security.auth.Subject;
...

Subject subject = Security.login(sName, acPassword);
PrivilegedAction action = new PrivilegedAction()
{
    ...
}
```

---

7-2 Oracle Coherence Developer's Guide
public Object run()
{
    // all processing here is taking place with access
    // rights assigned to the corresponding Subject
    ...
}

Security.runAs(subject, action);

During the "login" call Coherence uses JAAS that runs on the caller's node to authenticate the caller. In a case of successful authentication, it uses the local Access Controller to:

1. Determine whether the local caller has sufficient rights to access the protected clustered resource (local access check);
2. Encrypt the outgoing communications regarding the access to the resource with the caller's private credentials retrieved during the authentication phase;
3. Decrypt the result of the remote check using the requester's public credentials;
4. In the case that access is granted verify whether the responder had sufficient rights to do so.

Step 2 (above) serves a role of the proof of identity for the responder preventing a malicious node pretending to pass the local access check phase.

There are two alternative ways to provide the client authentication information. First, a reference to a CallbackHandler could be passed instead of the user name and password. Second, a previously authenticated Subject could be used, which could become handy when Coherence is used by a Java EE application that could retrieve an authenticated Subject from the application container.

If a caller's request comes without any authentication context, Coherence will instantiate and call a CallbackHandler implementation declared in the Coherence operational descriptor to retrieve the appropriate credentials. However that "lazy" approach is much less efficient, since without externally defined call scope, every access to a protected clustered resource will force repetitive authentication calls.

Proof of Trustworthiness

Every clustered resource in Coherence is created by an explicit API call. A senior service member retains the private credentials that are presented during that call as a proof of trustworthiness. When the senior service member receives an access request to a protected clustered resource, it use the local Access Controller to:

1. Decrypt the incoming communication using the remote caller's public credentials;
2. Determine whether the remote caller has sufficient rights to access the protected clustered resource (remote access check);
3. Encrypt the response of access check using the private credentials of the service.

Since the requester will accept the response as valid only after decrypting it, step 3) in this cycle serves a role of the proof of trustworthiness for the requester preventing a malicious node pretending to be a valid service senior.

Default Access Controller implementation

Coherence ships with an Access Controller implementation that uses a standard Java KeyStore. The implementation class is com.tangosol.net.security.DefaultController and
the corresponding part of the Coherence operational descriptor used to configure the
default implementation is:

```xml
<security-config>
  <enabled system-property="tangosol.coherence.security">true</enabled>
  <login-module-name>Coherence</login-module-name>
  <access-controller>
    <class-name>com.tangosol.net.security.DefaultController</class-name>
    <init-params>
      <init-param id='1'>
        <param-type>java.io.File</param-type>
        <param-value>./keystore.jks</param-value>
      </init-param>
      <init-param id='2'>
        <param-type>java.io.File</param-type>
        <param-value>./permissions.xml</param-value>
      </init-param>
    </init-params>
  </access-controller>
  <callback-handler/>
</security-config>
```

The `login-module-name` element serves as the application name in a login
configuration file (see JAAS Reference Guide1 for complete details). Coherence is
shipped with a Java keystore (JKS) based login module that is contained in the
coherence-login.jar, which depends only on standard Java runtime classes and could
be placed in the JRE's lib/ext (standard extension) directory. The corresponding login
module declaration would look like:

```java
// LoginModule Configuration for Oracle Coherence(TM)
Coherence {
    com.tangosol.security.KeystoreLogin required
    keyStorePath='${user.dir}/${keystore.jks'};
};
```

The `access-controller` element defines the `AccessController` implementation that takes
two parameters to instantiate.

- The first parameter is a path to the same keystore that will be used by both
  controller and login module.
- The second parameter is a path to the access permission file (see discussion
  below).

The `callback-handler` is an optional element that defines a custom implementation of
the `javax.security.auth.callback.CallbackHandler` interface that would be instantiated and
used by Coherence to authenticate the client when all other means are exhausted.

Two more steps have to be performed. To make the default Access Controller
implementation usable in your application, you must perform two additional steps:

1. Create a keystore with necessary principals.
2. Create the permissions file that would declare the access right for the
corresponding principals.

Consider the following example that creates three principals: admin to be used by the
Java Security framework; manager and worker to be used by Coherence:

```
keytool -genkey -v -keystore ./keystore.jks -storepass password -alias admin
-keypass password -dname CN=Administrator,O=MyCompany,L=MyCity,ST=MyState
```
keytool -genkey -v -keystore ./keystore.jks -storepass password -alias manager
-keypass password -dname CN=Manager,OU=MyUnit

keytool -genkey -v -keystore ./keystore.jks -storepass password -alias worker
-keypass password -dname CN=Worker,OU=MyUnit

Consider the following example that assigns all rights to the Manager principal, only join rights to the Worker principal for caches that have names prefixed by common and all rights to the Worker principal for the invocation service named invocation:

```xml
<?xml version='1.0'?>
<permissions>
<grant>
  <principal>
    <class>javax.security.auth.x500.X500Principal</class>
    <name>CN=Manager,OU=MyUnit</name>
  </principal>
  <permission>
    <target>*</target>
    <action>all</action>
  </permission>
</grant>

<grant>
  <principal>
    <class>javax.security.auth.x500.X500Principal</class>
    <name>CN=Worker,OU=MyUnit</name>
  </principal>
  <permission>
    <target>cache=common*</target>
    <action>join</action>
  </permission>
  <permission>
    <target>service=invocation</target>
    <action>all</action>
  </permission>
</grant>
</permissions>
```

**Working in applications with installed security manager**

1. The policy file format is fully described in Java SE Security Guide. Example:

   ```
   grant codeBase "file:${coherence.home}/lib/coherence.jar"
   {
     permission java.security.AllPermission;
   };
   ```

   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.

2. The binaries could be signed using the JDK jarsigner tool, for example:

   ```
   jarsigner -keystore ./keystore.jks -storepass password coherence.jar admin
   ```

   and then additionally protected in the policy file:
grant SignedBy "admin" codeBase "file:${coherence.home}/lib/coherence.jar"
{
    permission java.security.AllPermission;
}

3. All relevant files such as policy format, coherence binaries, and permissions should be protected by operating system mechanisms to prevent malicious modifications.
A filter is a mechanism for plugging into the low-level TCMP stream protocol. Every message that is sent across the network by Coherence is streamed through this protocol. Coherence supports custom filters. By writing a filter, the contents of the network traffic can be modified. The most common examples of modification are encryption and compression.

**Compression Filters**

The compression filter is based on the `java.util.zip` package and compresses message contents thus reducing the network load. This is useful when there is ample CPU available but insufficient network bandwidth. See "Configuring Filters" on page 8-4 for information on enabling this filter.

**Encryption Filters**

Coherence ships with two JCA based encryption filters which can be used to protect the clustered communications for privacy and authenticity.

**Symmetric Encryption Filter**

This filter uses symmetric encryption to protect cluster communications. The encryption key is generated from a shared password known to all cluster members. This filter is suitable for small deployments or where the maintenance and protection of a shared password is feasible.

To enable this filter, specify which services will have their traffic encrypted by using this filter, or to enable it for all cluster traffic you may simply specify it as a filter for the `<outgoing-message-handler>` element.

**Example 8–1  Enabling a Filter for all Network Traffic**

```xml
<outgoing-message-handler>
  <use-filters>
    <filter-name>symmetric-encryption</filter-name>
  </use-filters>
</outgoing-message-handler>
```

The shared password may either be specified in the `<filters>` section of the operational configuration file, or by using the `tangosol.coherence.security.password` system property. See "Symmetric Encryption Filter Parameters" on page 8-2 for additional configuration options.
Symmetric Encryption Filter Parameters

The symmetric encryption filter supports the parameters listed in Table 8–1. See the com.tangosol.net.security.PasswordBasedEncryptionFilter Javadoc for additional configuration details.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>algorithm</td>
<td>Specifies the mechanism to use in deriving a secret key from the above material. Default value is PBEWithMD5AndDES.</td>
</tr>
<tr>
<td>iterations</td>
<td>Specifies the iteration count to use in deriving the key. Default value is 32.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the raw material used to generate the secret key. Preconfigured is tangosol.coherence.security.password. See &quot;Preconfigured Override Values&quot; on page L-2</td>
</tr>
<tr>
<td>salt</td>
<td>Specifies the salt to use in deriving the key. Default value is nosecret.</td>
</tr>
</tbody>
</table>

PKCS Encryption Filter

This filter uses public key cryptography (asymmetric encryption) to protect the cluster join protocol, and then switches over to much faster symmetric encryption for service level data transfers. Unlike the symmetric encryption filter, there is no persisted shared secret. The symmetric encryption key is randomly generated by the cluster's senior member, and is securely transfer to authenticated cluster members as part of the cluster join protocol. This encryption filter is suitable for deployments where maintenance of a shared secret is not feasible.

**Note:** This filter requires the JVM be configured with a JCA public key cryptography provider implementation such as Bouncy Castle, which supports asymmetric block ciphers. See the JCA documentation for details on installing and configuring JCA providers.

In the default setup each cluster node must be configured with a Java Keystore from which it may retrieve its identity Certificate and associated private key, and a set of trusted Certificates for other cluster members. You can construct this keystore as follows:

Create a Java Keystore and the local cluster member's password protected certificate and private key.

```
keytool -genkey -alias local -keypass secret -keyalg rsa -storepass secret -keystore ./keystore.jks
```

Export this public certificate for inclusion in all cluster members keystores.

```
keytool -export -alias local -keypass secret -storepass secret -keystore ./keystore.jks -rfc -file local.cert
```

Import the Certificates of other trusted cluster members. Each certificate must be stored under a unique but otherwise unimportant alias.

```
keytool -import -alias remote_1 -storepass secret -keystore ./keystore.jks -file local_1.cert
keytool -import -alias remote_2 -storepass secret -keystore ./keystore.jks -file local_2.cert
```
local_2.cert
keytool -import -alias remote_3 -storepass secret -keystore ./keystore.jks -file
local_3.cert

At this point you will have one keystore per cluster node, each containing a single private key plus a full set of trusted public certificates. If new nodes are to be added to the cluster the keystores of all existing nodes must be updated with the new node's certificate.

---

**Note:** You may also choose to supply custom key and trust management logic to eliminate the need for a full keystore per node. See the implementation's documentation for details on customization.

---

Then configure the cluster to encrypt all traffic using this filter by specifying it in the `<outgoing-message-handler>`.

```xml
<outgoing-message-handler>
  <use-filters>
    <filter-name>pkcs-encryption</filter-name>
  </use-filters>
</outgoing-message-handler>
```

The keystore and alias password can be specified either in the `<filters>` section of the operational configuration file, or by using the tangosol.coherence.security.password system property. See "PKCS Encryption Filter Parameters" for additional configuration options.

Note unlike the Symmetric Encryption Filter, this filter is not currently supported by Coherence*Extend, or on a service by service level.

### PKCS Encryption Filter Parameters

The PKCS encryption filter supports the following parameters, see "Encryption Filters" on page 8-1 section for examples, or the com.tangosol.net.security.ClusterEncryptionFilter Javadoc for additional configuration details.

#### Table 8–2  PKCS Encryption Filter Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>asymmetricFilterClassName</td>
<td>Specifies the asymmetric filter implementation. Default value is com.tangosol.net.security.AsymmetricEncryptionFilter.</td>
</tr>
<tr>
<td>keyAlias</td>
<td>Specifies the alias to use in reading the key from the keystore.</td>
</tr>
<tr>
<td>keyPassword</td>
<td>Specifies the password to use in reading the key. Preconfigured value is tangosol.coherence.security.password. See &quot;Preconfigured Override Values&quot; on page L-2.</td>
</tr>
<tr>
<td>store</td>
<td>Specifies the path to the KeyStore. Default value is .keystore.</td>
</tr>
<tr>
<td>sharedKeyType</td>
<td>Specifies the type of shared key. Default value is DESede.</td>
</tr>
<tr>
<td>sharedKeySize</td>
<td>Specifies the size of shared key. Default value is 112.</td>
</tr>
<tr>
<td>storePassword</td>
<td>Specifies the password to use to access the store. If unspecified value of keyPassword parameter will be used.</td>
</tr>
<tr>
<td>storeType</td>
<td>Specifies the type of KeyStore. Default value is JKS.</td>
</tr>
</tbody>
</table>
Configuring Filters

There are two steps to configuring a filter.

1. Declare the filter in the `<filters>` XML element of the `tangosol-coherence.xml` file:

   **Example 8–2  Declaring a Filter in the tangosol-coherence.xml File**

   ```xml
   <filter>
     <filter-name>gzip</filter-name>
     <filter-class>com.tangosol.net.CompressionFilter</filter-class>
     <init-params>
       <init-param>
         <param-name>strategy</param-name>
         <param-value>gzip</param-value>
       </init-param>
     </init-params>
   </filter>
   ```

   For more information on the structure of the `<filters>` XML element of the `tangosol-coherence.xml` file, see the documentation in the `coherence.dtd` file, which is also located inside `coherence.jar`.

2. The second step is to attach the filter to one or more specific services, or to make the filter global (for all services). To specify the filter for a specific service, for example the ReplicatedCache service, add a `<filter-name>` element to the `<use-filters>` element of the service declaration in the `tangosol-coherence.xml` file:

   **Example 8–3  Attaching the Filter to a Service**

   ```xml
   <service>
     <service-type>ReplicatedCache</service-type>
     <service-component>ReplicatedCache</service-component>
     <use-filters>
       <filter-name>gzip</filter-name>
     </use-filters>
     <init-params>...
     </init-params>
   </service>
   ```

   To add the filter to all services, do the same under the `<outgoing-message-handler>` XML element instead of under a `<service>` XML element:

   **Example 8–4  Adding the Filter to All Services**

   ```xml
   <outgoing-message-handler>
     <use-daemon>false</use-daemon>
     <use-filters>
       <filter-name>gzip</filter-name>
     </use-filters>
   </outgoing-message-handler>
   ```

### Table 8–2 (Cont.) PKCS Encryption Filter Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transformation</td>
<td>Specifies the transformation to use. Default value is RSA/NONE/PKCS1Padding.</td>
</tr>
</tbody>
</table>
Creating a Custom Filter

To create a new filter, create a Java class that implements the `com.tangosol.io.WrapperStreamFactory` interface and optionally implements the `com.tangosol.run.xml.XmlConfigurable` interface. The `WrapperStreamFactory` interface provides the stream to be wrapped ("filtered") on input (received message) or output (sending message) and expects a stream back that wraps the original stream. These methods are called for each incoming and outgoing message.

If the filter class implements the `XmlConfigurable` interface, then Coherence will configure the filter after instantiating it. Example 8–5 illustrates a filter declaration in the `tangosol-coherence.xml` file. If the filter is associated with a service type, every time a new service is started of that type, Coherence will instantiate the `CompressionFilter` class and will hold it with the service until the service stops. If the filter is associated with all outgoing messages, Coherence will instantiate the filter on startup and will hold it until the cluster stops.

Example 8–5 Configuration for a Custom Filter

```xml
<filter>
  <filter-name>my-gzip-filter</filter-name>
  <filter-class>com.tangosol.net.CompressionFilter</filter-class>
  <init-params>
    <init-param>
      <param-name>strategy</param-name>
      <param-value>gzip</param-value>
    </init-param>
    <init-param>
      <param-name>buffer-length</param-name>
      <param-value>1024</param-value>
    </init-param>
  </init-params>
</filter>
```

After instantiating the filter, Coherence will call the `setConfig` method (if the filter implements `XmlConfigurable`) with the following XML element:

Example 8–6 Configuring a setConfig Call for a Filter

```xml
<config>
  <strategy>gzip</strategy>
  <buffer-length>1024</buffer-length>
</config>
```
Coherence Priority Tasks provide applications that have critical response time requirements better control of the execution of processes within Coherence. Execution and request timeouts can be configured to limit wait time for long running threads. In addition, a custom task API allows applications to control queue processing. Note that these features should be used with extreme caution because they can dramatically effect performance and throughput of the data grid.

**Priority Tasks — Timeouts**

Care should be taken when configuring Coherence Task Execution timeouts, especially for Coherence applications that pre-date this feature and thus do not handle timeout exceptions. If a write-through in a CacheStore is blocked (for example, if a database query is hung) and exceeds the configured timeout value, the Coherence Task Manager will attempt to interrupt the execution of the thread and an exception will be thrown. In a similar fashion, queries or aggregations that exceed configured timeouts will be interrupted and an exception will be thrown. Applications that use this feature should make sure that they handle these exceptions correctly to ensure system integrity. Since this configuration is performed on a service by service basis, changing these settings on existing caches/services not designed with this feature in mind should be done with great care.

**Configuring Execution Timeouts**

When configuring Execution Timeouts these values need to be considered: request-timeout, task-timeout, and the task-hung-threshold (see "Execution Timeout Parameters"). The request-timeout is the amount of time the client will wait a request to return. The task-timeout is the amount of time that the server will allow the thread to execute before interrupting execution. The task-hung-threshold is the amount of time that a thread can execute before the server reports the thread as "hung." "Hung" threads are for reporting purposes only. These timeout settings are in milliseconds and are configured in the coherence-cache-config.xml or by using command line parameters.

**Execution Timeout Parameters**

Table 9–1 describes the execution timeout parameters.
To set the distributed cache thread count to 7 with a task time out of 5000 milliseconds and a task hung threshold of 10000 milliseconds, the following would need to be added to the coherence-cache-config.xml for the node.

Example 9–1 Sample Task Time and Task Hung Configuration

```xml
<coherence-cache-config>
  <caching-schemes>
    <distributed-scheme>
      <scheme-name>example-distributed</scheme-name>
      <service-name>DistributedCache</service-name>
      <thread-count>7</thread-count>
      <task-timeout>5000ms</task-timeout>
      <task-hung-threshold>10000ms</task-hung-threshold>
    </distributed-scheme>
  </caching-schemes>
</coherence-cache-config>
```

Setting the client request timeout to 15 milliseconds

Example 9–2 Sample Client Request Timeout Configuration

```xml
<coherence-cache-config>
  <caching-schemes>
    <distributed-scheme>
      <scheme-name>example-distributed</scheme-name>
      <service-name>DistributedCache</service-name>
      <request-timeout>15000ms</request-timeout>
    </distributed-scheme>
  </caching-schemes>
</coherence-cache-config>
```

Note: The request-timeout should always be longer than the task-hung-threshold or the task-timeout.
Command Line Options

The command line options can be used to set the service type default (such as distributed cache, invocation, proxy, and so on) for the node. Table 9–2 describes the options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tangosol.coherence.replicated.request.timeout</td>
<td>The default client request timeout for the Replicated cache service</td>
</tr>
<tr>
<td>tangosol.coherence.optimistic.request.timeout</td>
<td>The default client request timeout for the Optimistic cache service</td>
</tr>
<tr>
<td>tangosol.coherence.distributed.request.timeout</td>
<td>The default client request timeout for distributed cache services</td>
</tr>
<tr>
<td>tangosol.coherence.distributed.task.timeout</td>
<td>The default server execution timeout for distributed cache services</td>
</tr>
<tr>
<td>tangosol.coherence.distributed.task.hung</td>
<td>The default time before a thread is reported as hung by distributed cache services</td>
</tr>
<tr>
<td>tangosol.coherence.invocation.request.timeout</td>
<td>The default client request timeout for invocation services</td>
</tr>
<tr>
<td>tangosol.coherence.invocation.task.hung</td>
<td>The default time before a thread is reported as hung by invocation services</td>
</tr>
<tr>
<td>tangosol.coherence.invocation.task.timeout</td>
<td>The default server execution timeout invocation services</td>
</tr>
<tr>
<td>tangosol.coherence.proxy.request.timeout</td>
<td>The default client request timeout for proxy services</td>
</tr>
<tr>
<td>tangosol.coherence.proxy.task.timeout</td>
<td>The default server execution timeout proxy services</td>
</tr>
<tr>
<td>tangosol.coherence.proxy.task.hung</td>
<td>The default time before a thread is reported as hung by proxy services</td>
</tr>
</tbody>
</table>

Priority Task Execution — Custom Objects

The PriorityTask interface enables you to control the ordering in which a service schedules tasks for execution using a thread pool and hold their execution time to a specified limit. Instances of PriorityTask typically also implement either the Invocable or Runnable interface. Priority Task Execution is only relevant when a task back log exists.

The API defines the following ways to schedule tasks for execution

- **SCHEDULE_STANDARD**—a task will be scheduled for execution in a natural (based on the request arrival time) order
- **SCHEDULE_FIRST**—a task will be scheduled in front of any equal or lower scheduling priority tasks and executed as soon as any of worker threads become available
- **SCHEDULE_IMMEDIATE**—a task will be immediately executed by any idle worker thread; if all of them are active, a new thread will be created to execute this task

APIs for Creating Priority Task Objects

Coherence provides the following classes to help create priority task objects:
Priority Processor can be extended to create a custom entry processor.

Priority Filter can be extended to create a custom priority filter.

Priority Aggregator can be extended to create a custom aggregation.

Priority Task can be extended to create an priority invocation class.

After extending each of these classes the developer will need to implement several methods. The return values for getRequestTimeoutMillis, getExecutionTimeoutMillis, and getSchedulingPriority should be stored on a class-by-class basis in your application configuration parameters. These methods are described in Table 9–3.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public long getRequestMethodTimeoutMillis()</td>
<td>Obtains the maximum amount of time a calling thread is willing to wait for a result of the request execution. The request time is measured on the client side as the time elapsed from the moment a request is sent for execution to the corresponding server node(s) and includes: the time it takes to deliver the request to the executing node(s); the interval between the time the task is received and placed into a service queue until the execution starts; the task execution time; the time it takes to deliver a result back to the client. The value of TIMEOUT_DEFAULT indicates a default timeout value configured for the corresponding service; the value of TIMEOUT_NONE indicates that the client thread is willing to wait indefinitely until the task execution completes or is canceled by the service due to a task execution timeout specified by the getExecutionTimeoutMillis() value.</td>
</tr>
<tr>
<td>public long getExecutionTimeoutMillis()</td>
<td>Obtains the maximum amount of time this task is allowed to run before the corresponding service will attempt to stop it. The value of TIMEOUT_DEFAULT indicates a default timeout value configured for the corresponding service; the value of TIMEOUT_NONE indicates that this task can execute indefinitely. If, by the time the specified amount of time passed, the task has not finished, the service will attempt to stop the execution by using the Thread.interrupt() method. In the case that interrupting the thread does not result in the task’s termination, the runCanceled method will be called.</td>
</tr>
<tr>
<td>public int getSchedulingPriority()</td>
<td>Obtains this task’s scheduling priority. Valid values are SCHEDULE_STANDARD, SCHEDULE_FIRST, SCHEDULE_IMMEDIATE</td>
</tr>
<tr>
<td>public void runCanceled(boolean fAbandoned)</td>
<td>This method will be called if and only if all attempts to interrupt this task were unsuccessful in stopping the execution or if the execution was canceled before it had a chance to run at all. Since this method is usually called on a service thread, implementors must exercise extreme caution since any delay introduced by the implementation will cause a delay of the corresponding service.</td>
</tr>
</tbody>
</table>

**Errors Thrown by Task Timeouts**

When a task timeout occurs the node will get a RequestTimeoutException. Example 9–3 illustrates an exception that may be thrown.
Example 9–3  Exception Thrown by a TaskTimeout

com.tangosol.net.RequestTimeoutException: Request timed out after 4015 millis
  at com.tangosol.coherence.component.util.daemon.queueProcessor.Service.
    checkRequestTimeout(Service.CDB:8)
    at com.tangosol.coherence.component.util.daemon.queueProcessor.Service.
      poll(Service.CDB:52)
    at com.tangosol.coherence.component.util.daemon.queueProcessor.Service.
      poll(Service.CDB:18)
    at com.tangosol.coherence.component.util.daemon.queueProcessor.service.
      InvocationService.query(InvocationService.CDB:17)
    at com.tangosol.coherence.component.util.safeService.
      SafeInvocationService.query(SafeInvocationService.CDB:1)
Specifying a Custom Eviction Policy

The `LocalCache` class is used for size-limited caches. It is used both for caching on-heap objects (as in a local cache or the front portion of a near cache) and as the backing map for a partitioned cache. Applications can provide custom eviction policies for use with a `LocalCache`.

Note that Coherence’s default eviction policy is very effective for most workloads; the majority of applications will not need to provide a custom policy. Generally, it is best to restrict the use of eviction policies to scenarios where the evicted data is present in a backing system (that is, the back portion of a near cache or a database). Eviction should be treated as a physical operation (freeing memory) and not a logical operation (deleting an entity).

Example 10–1 shows the implementation of a simple custom eviction policy:

**Example 10–1 Implementing a Custom Eviction Policy**

```java
import com.tangosol.net.cache.CacheEvent;
import com.tangosol.net.cache.ConfigurableCacheMap;
import com.tangosol.net.cache.LocalCache;
import com.tangosol.net.cache.LocalCache;
import com.tangosol.util.AbstractMapListener;
import com.tangosol.util.MapEvent;
import java.util.Iterator;

public class MyEvictionPolicy extends AbstractMapListener implements
ConfigurableCacheMap.EvictionPolicy
{
    LocalCache m_cache = null;

    public void entryInserted(MapEvent evt)
    {
        System.out.println("entryInserted:" + isSynthetic(evt) + evt);
        if (m_cache == null)
        {
            m_cache = (LocalCache) evt.getMap();
        }
    }

    public void entryUpdated(MapEvent evt)
    {
        System.out.println("entryUpdated:" + isSynthetic(evt) + evt);
    }

    public void entryDeleted(MapEvent evt)
    {
        System.out.println("entryDeleted:" + isSynthetic(evt) + evt);
    }
}
```
String isSynthetic(MapEvent evt)
{
    // synthetic events are caused by internal processing - eviction or
    // loading
    return ((CacheEvent) evt).isSynthetic() ? " SYNTHETIC " : " ";
}

public void entryTouched(OldCache.Entry entry)
{
    System.out.println("entryTouched:" + entry.getKey());
}

public void requestEviction(int cMaximum)
{
    int cCurrent = m_cache.getUnits();
    System.out.println("requestEviction: current:" + cCurrent + " to:" +
    cMaximum);

    // ...
    // ... eviction policy calculations ...
    //
    for (Iterator iter = m_cache.entrySet().iterator(); iter.hasNext();)
    {
        ConfigurableCacheMap.Entry entry = (ConfigurableCacheMap.Entry)
        iter.next();
        if (m_cache.getUnits() > cMaximum)
        {
            m_cache.evict(entry.getKey());
        }
        else
        {
            break;
        }
    }
}

public MyEvictionPolicy()
{
}

Example 10–2 illustrates a Coherence cache configuration file
(coherence-cache-config.xml) with an eviction policy:

Example 10–2 Custom Eviction Policy in a coherence-cache-config.xml File

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

<!DOCTYPE cache-config SYSTEM "cache-config.dtd">

<cache-config>
    <caching-scheme-mapping>
        <cache-mapping>
            <cache-name>test</cache-name>
            <scheme-name>example-near</scheme-name>
        </cache-mapping>
    </caching-scheme-mapping>
</cache-config>
```
<caching-schemes>
  <distributed-scheme>
    <scheme-name>example-distributed</scheme-name>
    <service-name>DistributedCache</service-name>
    <backing-map-scheme>
      <local-scheme>
        <scheme-ref>example-backing-map</scheme-ref>
      </local-scheme>
    </backing-map-scheme>
    <autostart>true</autostart>
  </distributed-scheme>

  <near-scheme>
    <scheme-name>example-near</scheme-name>
    <front-scheme>
      <local-scheme>
        <eviction-policy>
          <class-scheme>
            <class-name>MyEvictionPolicy</class-name>
          </class-scheme>
        </eviction-policy>
        <high-units>10</high-units>
      </local-scheme>
    </front-scheme>
    <back-scheme>
      <distributed-scheme>
        <scheme-ref>example-distributed</scheme-ref>
      </distributed-scheme>
    </back-scheme>
    <invalidation-strategy>all</invalidation-strategy>
    <autostart>true</autostart>
  </near-scheme>

  <local-scheme>
    <scheme-name>example-backing-map</scheme-name>
    <eviction-policy>HYBRID</eviction-policy>
    <high-units>{back-size-limit 0}</high-units>
    <expiry-delay>{back-expiry 1h}</expiry-delay>
    <flush-delay>1m</flush-delay>
    <cachestore-scheme></cachestore-scheme>
  </local-scheme>
</caching-schemes>
</cache-config>
Oracle Coherence provides explicit support for efficient caching of huge amounts of automatically-expiring data using potentially high-latency storage mechanisms such as disk files. The benefits include supporting much larger data sets than can be managed in memory, while retaining an efficient expiry mechanism for timing out the management (and automatically freeing the resources related to the management) of that data. Optimal usage scenarios include the ability to store many large objects, XML documents or content that will be rarely accessed, or whose accesses will tolerate a higher latency if the cached data has been paged to disk. See “Storage and Backing Map” in the Oracle Coherence Getting Started Guide.

Understanding Serialization Paged Cache

This feature is known as a Serialization Paged Cache:

- **Serialization** implies that objects stored in the cache are serialized and stored in a Binary Store; refer to the existing features Serialization Map and Serialization Cache.

- **Paged** implies that the objects stored in the cache are segmented for efficiency of management.

- **Cache** implies that there can be limits specified to the size of the cache; in this case, the limit is the maximum number of concurrent pages that the cache will manage before expiring pages, starting with the oldest page.

The result is a feature that organizes data in the cache based on the time that the data was placed in the cache, and then is capable of efficiently expiring that data from the cache, an entire page at a time, and typically without having to reload any data from disk.

Configuring Serialization Paged Cache

The primary configuration for the Serialization Paged Cache is composed of two parameters: The number of pages that the cache will manage, and the length of time represented by each page. For example, to cache data for one day, the cache can be configured as 24 pages of one hour each, or 96 pages of 15 minutes each, and so on.

Each page of data in the cache is managed by a separate Binary Store. The cache requires a Binary Store Manager, which provides the means to create and destroy these Binary Stores. Coherence provides Binary Store Managers for all of the built-in Binary Store implementations, including Berkley DB (referred to as "BDB") and the various NIO implementations.
Optimizing a Partitioned Cache Service

Coherence provides an optimization for the partitioned cache service, since when it is used to back a partitioned cache—the data being stored in any of the Serialization Maps and Caches is entirely binary in form. This is called the Binary Map optimization, and when it is enabled, it gives the Serialization Map, the Serialization Cache and the Serialization Paged Cache permission to assume that all data being stored in the cache is binary. The result of this optimization is a lower CPU and memory utilization, and also slightly higher performance. See the `<external-scheme>` and `<paged-external-scheme>` cache configuration elements.

Configuring for High Availability

Explicit support is also provided in the Serialization Paged Cache for the high-availability features of the partitioned cache service, by providing a configuration that can be used for the primary storage of the data and a configuration that is optimized for the backup storage of the data. The configuration for the backup storage is known as a passive model, because it does not actively expire data from its storage, but rather reflects the expiration that is occurring on the primary cache storage. When using the high-availability data feature (a backup count of one or greater; the default is one) for a partitioned cache service, and using the Serialization Paged Cache as the primary backing storage for the service, we strongly suggest that you also use the Serialization Paged Cache as the backup store, and configure the backup with the passive option. See the `<paged-external-scheme>` cache configuration elements.

Configuring Load Balancing and Failover

When used with the distributed cache service, special considerations should be made for load balancing and failover purposes. The partition-count parameter of the distributed cache service should be set higher than normal if the amount of cache data is very large or huge; that will break up the overall cache into smaller chunks for load-balancing and recovery processing due to failover. For example, if the cache is expected to be one terabyte in size, twenty thousand partitions will break the cache up into units averaging about 50MB in size. If a unit (the size of a partition) is too large, it will cause an out-of-memory condition when load-balancing the cache. (Remember to make sure that the partition count is a prime number; see http://primes.utm.edu/lists/small/ for lists of prime numbers that you can use.)

Supporting Huge Caches

To support huge caches (for example, terabytes) of expiring data, the expiration processing is performed concurrently on a daemon thread with no interruption to the cache processing. The result is that many thousands or millions of objects can exist in a single cache page, and they can be expired asynchronously, thus avoiding any interruption of service. The daemon thread is an option that is enabled by default, but it can be disabled. See the `<external-scheme>` and `<paged-external-scheme>` cache configuration elements.

When the cache is used for large amounts of data, the pages will typically be disk-backed. Since the cache eventually expires each page, thus releasing the disk resources, the cache uses a virtual erase optimization by default. This means that data that is explicitly removed or expired from the cache is not actually removed from the underlying Binary Store, but when a page (a Binary Store) is completely emptied, it will be erased in its entirety. This reduces I/O by a considerable margin, particularly during expiry processing and during operations such as load-balancing that have to
redistribute large amounts of data within the cluster. The cost of this optimization is that the disk files (if a disk-based Binary Store option is used) will tend to be larger than the data that they are managing would otherwise imply; since disk space is considered to be inexpensive compared to other factors such as response times, the virtual erase optimization is enabled by default, but it can be disabled. Note that the disk space is typically allocated locally to each server, and thus a terabyte cache partitioned over one hundred servers would only use about 20GB of disk space per server (10GB for the primary store and 10GB for the backup store, assuming one level of backup.)
This section describes different patterns you can use to pre-load the cache. The patterns include bulk loading and distributed loading.

Performing Bulk Loading and Processing

Example 12–5, PagedQuery.java, demonstrates techniques for efficiently bulk loading and processing items in a Coherence Cache.

Bulk Writing to a Cache

A common scenario when using Coherence is to pre-populate a cache before the application uses it. A simple way to do this is illustrated by the Java code in Example 12–1:

Example 12–1  Pre-Loading a Cache

```java
public static void bulkLoad(NamedCache cache, Connection conn)
{
    Statement s;
    ResultSet rs;

    try
    {
        s = conn.createStatement();
        rs = s.executeQuery("select key, value from table");
        while (rs.next())
        {
            Integer key   = new Integer(rs.getInt(1));
            String  value = rs.getString(2);
            cache.put(key, value);
        }
        ...
    }
    catch (SQLException e)
    {
        (...)
    }
}
```

This technique works, but each call to put may result in network traffic, especially for partitioned and replicated caches. Additionally, each call to put will return the object it just replaced in the cache (per the java.util.Map interface) which will add more unnecessary overhead. Loading the cache can be made much more efficient by using the ConcurrentMap.putAll method instead. This is illustrated in Example 12–2:
Example 12–2  Pre-Loading a Cache Using ConcurrentHashMap.putAll

public static void bulkLoad(NamedCache cache, Connection conn) {
    Statement s;
    ResultSet rs;
    Map buffer = new HashMap();

    try {
        int count = 0;
        s = conn.createStatement();
        rs = s.executeQuery("select key, value from table");
        while (rs.next()) {
            Integer key   = new Integer(rs.getInt(1));
            String  value = rs.getString(2);
            buffer.put(key, value);

            // this loads 1000 items at a time into the cache
            if ((count++ % 1000) == 0) {
                cache.putAll(buffer);
                buffer.clear();
            }
        }
        if (!buffer.isEmpty()) {
            cache.putAll(buffer);
        }
    }
    catch (SQLException e) {
    }
}

Efficient processing of filter results

Coherence provides the ability to query caches based on criteria by using the Filter API. Here is an example (given entries with integers as keys and strings as values):

Example 12–3  Using a Filter to Query a Cache

NamedCache c = CacheFactory.getCache("test");

// Search for entries that start with 'c'
Filter query = new LikeFilter(IdentityExtractor.INSTANCE, "c%", '\', true);

// Perform query, return all entries that match
Set results = c.entrySet(query);
for (Iterator i = results.iterator(); i.hasNext();)
    { Map.Entry e = (Map.Entry) i.next();
        out("key: "+e.getKey() + ", value: "+e.getValue());
    }

This example works for small data sets, but it may encounter problems, such as running out of heap space, if the data set is too large. Example 12–4 illustrates a pattern to process query results in batches to avoid this problem:
Example 12–4  Processing Query Results in Batches

```java
public static void performQuery()
{
    NamedCache c = CacheFactory.getCache("test");

    // Search for entries that start with 'c'
    Filter query = new LikeFilter(IdentityExtractor.INSTANCE, "c%", '\\', true);

    // Perform query, return keys of entries that match
    Set keys = c.keySet(query);

    // The amount of objects to process at a time
    final int BUFFER_SIZE = 100;

    // Object buffer
    Set buffer = new HashSet(BUFFER_SIZE);

    for (Iterator i = keys.iterator(); i.hasNext();)
    {
        buffer.add(i.next());

        if (buffer.size() >= BUFFER_SIZE)
        {
            // Bulk load BUFFER_SIZE number of objects from cache
            Map entries = c.getAll(buffer);

            // Process each entry
            process(entries);

            // Done processing these keys, clear buffer
            buffer.clear();
        }
    }

    // Handle the last partial chunk (if any)
    if (!buffer.isEmpty())
    {
        process(c.getAll(buffer));
    }
}

public static void process(Map map)
{
    for (Iterator ie = map.entrySet().iterator(); ie.hasNext();)
    {
        Map.Entry e = (Map.Entry) ie.next();
        out("key: "+e.getKey()+ ", value: "+e.getValue());
    }
}
```

In this example, all keys for entries that match the filter are returned, but only BUFFER_SIZE (in this case, 100) entries are retrieved from the cache at a time.

Note that LimitFilter can be used to process results in parts, similar to the example above. However LimitFilter is meant for scenarios where the results will be paged, such as in a user interface. It is not an efficient means to process all data in a query result.
A Bulk Loading and Processing Example

Example 12–5 illustrates PagedQuery.java, a sample program that demonstrates the concepts described in the previous section.

To run the example, follow these steps:

1. Save the following Java file as com/tangosol/examples/PagedQuery.java
2. Point the classpath to the Coherence libraries and the current directory
3. Compile and run the example

Example 12–5  A Sample Bulk Loading Program

```java
package com.tangosol.examples;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.tangosol.net.cache.NearCache;
import com.tangosol.util.Base;
import com.tangosol.util.Filter;
import com.tangosol.util.filter.LikeFilter;
import java.io.Serializable;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.HashSet;

/**<*
 * This sample application demonstrates the following:
 * <ul>
 * <li>Obtaining a back cache from a near cache for populating a cache.</li>
 * Since the near cache holds a limited subset of the data in a cache it is
 * more efficient to bulk load data directly into the back cache instead of
 * the near cache.
 * </ul>
 * <ul>
 * <li>Populating a cache in bulk using <tt>putAll</tt>.</li>
 * This is more efficient than <tt>put</tt> for a large amount of entries.
 * </ul>
 * <ul>
 * <li>Executing a filter against a cache and processing the results in bulk.</li>
 * This sample issues a query against the cache using a filter. The result is
 * a set of keys that represent the query results. Instead of iterating
 * through the keys and loading each item individually with a <tt>get</tt>,
 * this sample loads entries from the cache in bulk using <tt>getAll</tt> which
 * is more efficient.
 * </ul>
 * /**<*
 * @author cp
 */
public class PagedQuery
    extends Base
    {
        /**<*
public static void main(String[] asArg) {
    NamedCache cacheContacts = CacheFactory.getCache("contacts",
            Contact.class.getClassLoader());
    populateCache(cacheContacts);
    executeFilter(cacheContacts);
    CacheFactory.shutdown();
}

// ----- populate the cache ---------------------------------------------
/**
 * Populate the cache with test data. This example shows how to populate
 * the cache a chunk at a time using {@link NamedCache#putAll} which is more
 * efficient than {@link NamedCache#put}.
 *
 * @param cacheDirect the cache to populate. Note that this should <b>not</b>
 *                     be a near cache since that will thrash the cache
 *                     if the load size exceeds the near cache max size.
 */
public static void populateCache(NamedCache cacheDirect) {
    if (cacheDirect.isEmpty()) {
        Map mapBuffer = new HashMap();
        for (int i = 0; i < 100000; ++i) {
            // make up some fake data
            Contact contact = new Contact();
            contact.setName(getRandomName() + ' ' + getRandomName());
            contact.setPhone(getRandomPhone());
            mapBuffer.put(new Integer(i), contact);
            // this loads 1000 items at a time into the cache
            if ((i % 1000) == 0) {
                out("Adding " + mapBuffer.size() + " entries to cache");
                cacheDirect.putAll(mapBuffer);
                mapBuffer.clear();
            }
        }
        if (!mapBuffer.isEmpty()) {
            cacheDirect.putAll(mapBuffer);
        }
    }
}

/**
 * Creates a random name.
 * @return a random string between 4 to 11 chars long
 */
public static String getRandomName() {

Random rnd = getRandom();
int cch = 4 + rnd.nextInt(7);
char[] ach = new char[cch];
ach[0] = (char) ('A' + rnd.nextInt(26));
for (int of = 1; of < cch; ++of)
    { 
        ach[of] = (char) ('a' + rnd.nextInt(26));
    }
return new String(ach);

/**
* Creates a random phone number
* @return a random string of integers 10 chars long
*/
public static String getRandomPhone()
{
    Random rnd = getRandom();
    return "("
        + toDecString(100 + rnd.nextInt(900), 3)
        + " "
        + toDecString(100 + rnd.nextInt(900), 3)
        + "-
        + toDecString(10000, 4);
}

// ----- process the cache ---------------------------------------------

/**
* Query the cache and process the results in batches. This example
* shows how to load a chunk at a time using (link NamedCache#getAll)
* which is more efficient than (link NamedCache#get).
* @param cacheDirect the cache to issue the query against
*/
private static void executeFilter(NamedCache cacheDirect)
{
    Filter query = new LikeFilter("getName", "C%");

    // Let's say we want to process 100 entries at a time
    final int CHUNK_COUNT = 100;

    // Start by querying for all the keys that match
    Set setKeys = cacheDirect.keySet(query);

    // Create a collection to hold the "current" chunk of keys
    Set setBuffer = new HashSet();

    // Iterate through the keys
    for (Iterator iter = setKeys.iterator(); iter.hasNext(); )
        { 
            // Collect the keys into the current chunk
            setBuffer.add(iter.next());

            // handle the current chunk when it gets big enough
            if (setBuffer.size() >= CHUNK_COUNT)
                { 
                    // Instead of retrieving each object with a get,
                    // retrieve a chunk of objects at a time with a getAll.
processContacts(cacheDirect.getAll(setBuffer));
setBuffer.clear();
}

// Handle the last partial chunk (if any)
if (!setBuffer.isEmpty())
{
    processContacts(cacheDirect.getAll(setBuffer));
}

/**
 * Process the map of contacts. In a real application some sort of
 * processing for each map entry would occur. In this example each
 * entry is logged to output.
 *
 * @param map  the map of contacts to be processed
 */
public static void processContacts(Map map)
{
    out("processing chunk of " + map.size() + " contacts: ");
    for (Iterator iter = map.entrySet().iterator(); iter.hasNext(); )
    {
        Map.Entry entry = (Map.Entry) iter.next();
        out(" " + entry.getKey() + "=" + entry.getValue());
    }
}

// ----- inner classes --------------------------------------------------

/**
 * Sample object used to populate cache
 */
public static class Contact
    extends Base
    implements Serializable
{
    public Contact() {}
    public String getName()
    {
        return m_sName;
    }
    public void setName(String sName)
    {
        m_sName = sName;
    }
    public String getPhone()
    {
        return m_sPhone;
    }
    public void setPhone(String sPhone)
    {
        m_sPhone = sPhone;
    }
    public String toString()
    {

public boolean equals(Object o)
{
    if (o instanceof Contact)
    {
        Contact that = (Contact) o;
        return equals(this.getName(), that.getName())
                && equals(this.getPhone(), that.getPhone());
    }
    return false;
}

public int hashCode()
{
    int result;
    result = (m_sName != null ? m_sName.hashCode() : 0);
    result = 31 * result + (m_sPhone != null ? m_sPhone.hashCode() : 0);
    return result;
}

private String m_sName;
private String m_sPhone;
}

Example 12–6 illustrates the terminal output from Coherence when you compile and run the example:

Example 12–6 Terminal Output from the Bulk Loading Program
$ export COHERENCE_HOME=[**Coherence install directory**]
$ export CLASSPATH=$COHERENCE_HOME/lib/coherence.jar:.
$ javac com/tangosol/examples/PagedQuery.java
$ java com.tangosol.examples.PagedQuery

2008-09-15 12:19:44.156 Oracle Coherence 3.4/405 <Info> (thread=main, member=n/a): Loaded operational configuration from resource "jar:file:/C:/coherence/lib/coherence.jar!/tangosol-coherence.xml"
2008-09-15 12:19:44.171 Oracle Coherence 3.4/405 <Info> (thread=main, member=n/a): Loaded operational overrides from resource "jar:file:/C:/coherence/lib/coherence.jar!/tangosol-coherence-override-dev.xml"
2008-09-15 12:19:44.171 Oracle Coherence 3.4/405 <Info> (thread=main, member=n/a): Optional configuration override "/tangosol-coherence-override.xml" is not specified

Oracle Coherence Version 3.4/405
Grid Edition: Development mode
Copyright (c) 2000-2008 Oracle. All rights reserved.

2008-09-15 12:19:44.812 Oracle Coherence GE 3.4/405 <D5> (thread=Cluster, member=n/a): Service Cluster joined the cluster
Performing Distributed Bulk Loading

When pre-populating a Coherence partitioned cache with a large data set, it may be more efficient to distribute the work to Coherence cluster members. Distributed loading will allow for higher data throughput rates to the cache by leveraging the aggregate network bandwidth and CPU power of the cluster. When performing a distributed load, the application will need to decide on the following:

- which cluster members will perform the load
- how to divide the data set among the members

The application should consider the load that will be placed on the underlying data source (such as a database or file system) when selecting members and dividing work. For example, a single database can easily be overwhelmed if too many members execute queries concurrently.

A Distributed Bulk Loading Example

This section outlines the general steps to perform a simple distributed load. The example assumes that the data is stored in files and will be distributed to all storage-enabled members of a cluster.
Performing Distributed Bulk Loading

1. Retrieve the set of storage-enabled members. For example, the following method uses the `getStorageEnabledMembers` method to retrieve the storage-enabled members of a distributed cache.

   **Example 12–7  Retrieving Storage-Enabled Members of the Cache**

   ```java
   protected Set<String> getStorageMembers(NamedCache cache)
   {
     return ((DistributedCacheService) cache.getCacheService()).
         getStorageEnabledMembers();
   }
   ```

2. Divide the work among the storage enabled cluster members. For example, the following routine returns a map, keyed by member, containing a list of files assigned to that member.

   **Example 12–8  Routine to Get a List of Files Assigned to a Cache Member**

   ```java
   protected Map<Member, List<String>> divideWork(Set members, List<String> fileNames)
   {
     Iterator i = members.iterator();
     Map<Member, List<String>> mapWork = new HashMap(members.size());
     for (String sFileName : fileNames)
     {
       Member member = (Member) i.next();
       List<String> memberFileNames = mapWork.get(member);
       if (memberFileNames == null)
       {
         memberFileNames = new ArrayList();
         mapWork.put(member, memberFileNames);
       }
       memberFileNames.add(sFileName);
       // recycle through the members
       if (!i.hasNext())
       {
         i = members.iterator();
       }
     }
     return mapWork;
   }
   ```

3. Launch a task that will perform the load on each member. For example, use Coherence's `InvocationService` to launch the task. In this case, the implementation of `LoaderInvocable` will need to iterate through `memberFileNames` and process each file, loading its contents into the cache. The cache operations normally performed on the client will need to be executed through the `LoaderInvocable`.

   **Example 12–9  Class to Load Each Member of the Cache**

   ```java
   public void load()
   {
     NamedCache cache = getCache();
     Set members = getStorageMembers(cache);
     List<String> fileNames = getFileNames();
   }
   ```

---

12-10  Oracle Coherence Developer's Guide
Map<Member, List<String>> mapWork = divideWork(members, fileNames);

InvocationService service = (InvocationService) CacheFactory.getService("InvocationService");

for (Map.Entry<Member, List<String>> entry : mapWork.entrySet()) {
    Member member = entry.getKey();
    List<String> memberFileNames = entry.getValue();

    LoaderInvocable task = new LoaderInvocable(memberFileNames, cache.getCacheName());
    service.execute(task, Collections.singleton(member), this);
}

Running a Distributed Bulk Loading Example

The examples in the previous section are taken from DistributedLoader.java, which is included in the attached zip file, coherence-example-distributedload.zip. This sample application uses the InvocationService to distribute the task of loading data into a cache. Each storage-enabled member of the cluster will be responsible for loading a portion of the data. The data in this example will come from several CSV files and the unit of work is one file. All storage-enabled nodes must have access to all of the data files.

To build and run the example, you must have the following software installed:

- J2SE SDK 1.4 or later
- Apache Ant
- Oracle Coherence

Building the Sample Application

1. Extract the contents of coherence-example-distributedload.zip into your file system.
2. Update the bin\set-env.cmd file to reflect your system environment.
3. Open a command prompt and execute the following command in the bin directory to build the samples:
   
   C:\distributedLoad\bin\ant.cmd build

   After running the samples, you can completely remove all build artifacts from your file system by running the clean command:
   
   C:\distributedLoad\bin\ant.cmd clean

Running the Sample Application

1. Start multiple cache servers (from the bin directory):

   C:\distributedLoad\bin\server.cmd

2. Run the client loader (from the bin directory):

   C:\distributedLoad\bin\load.cmd
After entering `load.cmd` on the command line, you will messages indicating that the various members are joining the services. Then you will see messages that indicate that the date is being distributed among the members. In this example, four cache servers were started.

**Example 12–10  Server Response from the Sample Distributed Loading Application**

```plaintext
...  
Member(Id=1, Timestamp=2008-09-15 16:49:04.359, Address=ip_address:8088, MachineId=49690, Location=site:us.oracle.com,machine:machine_name,process:21952, Role=CoherenceServer)  
Member(Id=2, Timestamp=2008-09-15 16:49:50.25, Address=ip_address:8089, MachineId=49690, Location=site:us.oracle.com,machine:machine_name,process:16604, Role=CoherenceServer)  
Member(Id=3, Timestamp=2008-09-15 16:49:54.937, Address=ip_address:8090, MachineId=49690, Location=site:us.oracle.com,machine:machine_name,process:7344, Role=CoherenceServer)  
Member(Id=4, Timestamp=2008-09-15 16:49:58.734, Address=ip_address:8091, MachineId=49690, Location=site:us.oracle.com,machine:machine_name,process:19052, Role=CoherenceServer)  
2008-09-15 16:51:00.593/4.890 Oracle Coherence GE 3.4/405 <D5> (thread=main, member=5): Loading stock file names from '..\data'  
2008-09-15 16:51:00.593/4.890 Oracle Coherence GE 3.4/405 <D5> (thread=main, member=5): Files to load: ['..\data\AAPL.CSV', '..\data\BT.CSV', '..\data\DELL.CSV', '..\data\GOOG.CSV', '..\data\HPQ.CSV', '..\data\JAVA.CSV', '..\data\MSFT.CSV']  
2008-09-15 16:51:00.593/4.890 Oracle Coherence GE 3.4/405 <D5> (thread=main, member=5): Files to load: ['..\data\BT.CSV', '..\data\JAVA.CSV']  
2008-09-15 16:51:00.593/4.890 Oracle Coherence GE 3.4/405 <D5> (thread=main, member=5): Files to load: ['..\data\DELL.CSV', '..\data\MSFT.CSV']  
2008-09-15 16:51:00.593/4.890 Oracle Coherence GE 3.4/405 <D5> (thread=main, member=5): Files to load: ['..\data\GOOG.CSV', '..\data\HPQ.CSV', '..\data\YHOO.CSV']  
2008-09-15 16:51:00.593/4.890 Oracle Coherence GE 3.4/405 <D5> (thread=main, member=5): Files to load: ['..\data\GOOG.CSV', '..\data\MSFT.CSV']  
2008-09-15 16:51:00.593/4.890 Oracle Coherence GE 3.4/405 <D5> (thread=main, member=5): Files to load: ['..\data\AAPL.CSV', '..\data\HPQ.CSV', '..\data\YHOO.CSV']  
2008-09-15 16:51:37.796/42.093 Oracle Coherence GE 3.4/405 <D5>  
```
Performing Distributed Bulk Loading

(thread=main, member=5): Load finished in 37.20 secs

C:\distributedload\bin>
Coherence is architected as a collection of services. Each Coherence service consists of the Coherence code that implements the service, along with an associated configuration. The service runs on an allocated pool of threads with associated queues that receive requests and return responses.

Coherence does not support re-entrant calls. A "re-entrant service call" occurs when a service thread, in the act of processing a request, makes a request to that same service. As all requests to a service are delivered by using the inbound queue, and Coherence uses a thread-per-request model, this means that each reentrant request would consume an additional thread (the calling thread would block while awaiting a response). Note that this is distinct from the similar-sounding concept of recursion.

Re-entrancy, Services, and Service Threads

A service is defined as a unique combination of a service name and a service type (such as Invocation, Replicated, or Distributed). For example, you can call from a distributed service Dist-Customers into a distributed service named Dist-Inventory, or from a distributed service named Dist-Customers into a replicated service named Repl-Catalog. Service names are configured in the cache configuration file using the <service-name> element.

Parent-Child Object Relationships

In the current implementation of Coherence, it is irrelevant whether the "call" is local or remote. This complicates the use of key association to support the efficient assembly of parent-child relationships. If you use key association to co-locate a Parent object with all of its Child objects, then you cannot send an EntryProcessor to the parent object and have that EntryProcessor "grab" the (local) Child objects. This is true even though the Child objects are already in-process.

To access both a parent object and its child objects, you can do any of the following:

- Embed the child objects within the parent object (using an "aggregate" pattern) or,
- Use direct access to the server-side backing map (which requires advanced knowledge to do safely), or
- Run the logic on another service (for example, Invocation targeted by using PartitionedService.getKeyOwner), and have that service access the data by using NamedCache interfaces, or
- Place the child objects on another service which would allow reentrant calls (but incur network access since there is no affinity between partitions in different cache services).
Using the aggregate pattern is probably the best solution for most use cases. However, if this is impractical (due to size restrictions, for example), and there is a need to access both the parent and child objects without using a client/server model, the Invocation service approach is probably the best compromise for most use cases.

**Avoiding Deadlock**

Even when re-entrancy is allowed, one should be very careful to avoid saturating the thread pool and causing catastrophic deadlock. For example, if service A calls service B, and service B calls service A, there is a possibility that a sufficient number of concurrent calls could fill one of the thread pools, which would cause a form of deadlock. As with traditional locking, using ordered access (for example, service A can call service B, but not vice versa) can help.

So:

- Service A calling into service A is never allowed
- Service A calling into service B, and service B calling back into service A is technically allowed but is deadlock-prone and should be avoided if at all possible.
  - Service A calling into service B, and service B calling into service C, and service C calling back into service A is similarly restricted
- Service A calling into service B is allowed
  - Service A calling into service B, and service B calling into service C, and service A calling into service C is similarly allowed

A service thread is defined as any thread involved in fulfilling a Coherence API request. Service threads may invoke any of the following entities:

- Map Listeners
- Membership Listeners
- Network Filters
- Custom Serialization/Deserialization such as `ExternalizableLite` implementations
- Backing Map Listeners
- `CacheLoader/CacheStore` Modules
- Query logic such as `Aggregators`, `Filters`, `ValueExtractors` and `Comparators`
- Entry Processors
- Triggers
- `InvocationService` Invocables

These entities should never make re-entrant calls back into their own services.

**Re-entrancy and Listeners**

Membership listeners can be used to observe the active set of members participating in the cluster or a specific service. Membership listener threading can be complex; thus, re-entrant calls from a member listener to any Coherence service should be avoided.
This section contains the following chapters:

- Chapter 14, "Evaluating Performance and Scalability"
- Chapter 15, "Performing a Multicast Connectivity Test"
- Chapter 16, "Performing a Datagram Test for Network Performance"
- Chapter 17, "Configuring and Using Coherence*Extend"
- Chapter 18, "High Resolution Timesource (Linux)"
- Chapter 19, "Performance Tuning"
- Chapter 20, "Setting Single Server Mode"
The Coherence distributed caches will often be evaluated with respect to pre-existing local caches. The local caches generally take the form of in-process hash maps. While Coherence does include facilities for in-process non-clustered caches, direct performance comparison between local caches and a distributed cache not realistic. By the very nature of being out of process, the distributed cache must perform serialization and network transfers. For this cost you gain cluster wide coherency of the cache data, and data and query scalability beyond what a single JVM or machine is capable of providing. This does not mean that you cannot achieve impressive performance using a distributed cache, but it must be evaluated in the correct context.

**Measuring Latency and Throughput**

When evaluating performance you try to establish two things, latency, and throughput. A simple performance analysis test may simply try performing a series of timed cache accesses in a tight loop. While these tests may accurately measure latency, to measure maximum throughput on a distributed cache a test must make use of multiple threads concurrently accessing the cache, and potentially multiple test clients. In a single threaded test the client thread will naturally spend the majority of the time simply waiting on the network. By running multiple clients/threads, you can more efficiently make use of your available processing power by issuing several requests in parallel. The use of batching operations can also be used to increase the data density of each operation. As you add threads, you should see that the throughput continues to increase until you’ve maxxed-out the CPU or network, while the overall latency remains constant for the same period.

**Demonstrating Scalability**

To show true linear scalability as you increase cluster size, you need to be prepared to be add hardware, and not simply JVMs to the cluster. Adding JVMs to a single machine will scale only up to the point where the CPU or network are fully used.

Plan on testing with clusters with more then just two cache servers (storage enabled nodes). The jump from one to two cache servers will not show the same scalability as from two to four. The reason for this is because by default Coherence will maintain one backup copy of each piece of data written into the cache. The process of maintaining backups only begins when there are two storage-enabled nodes in the cluster (there must be a place to put the backup). Thus when you move from a one to two, the amount of work involved in a mutating operation such as cache.put actually doubles, but beyond that the amount of work stays fixed, and will be evenly distributed across the nodes.
Tuning Your Environment

To get the most out of your cluster it is important that you've tuned of your environment and JVMs. Chapter 19, "Performance Tuning," provides good start to getting the most out of your environment. For example, Coherence includes a registry script for Windows (optimize.reg), which will adjust a few critical settings and allow Windows to achieve much higher data rates.

Measurements on a Large Cluster

The following graphs show the results of scaling out a cluster in an environment of 100 machines. In this particular environment, Coherence was able to scale to the limits of the network's switching infrastructure. Namely, there were 8 sets of ~12 machines, each set having a 4Gbs link to a central switch. Thus for this test Coherence's network throughput scales up to ~32 machines at which point it has maxxed-out the available bandwidth, beyond that it continues to scale in total data capacity.

Figure 14–1  Coherence Throughput versus Number of Machines

This figure is described in the text.

**********************************************************************************************
This figure is described in the text.

Latency for 10MB operations (~300ms) is not included in the graph for display reasons, as the payload is 1000x the next payload size.
Performing a Multicast Connectivity Test

Included with Coherence is a Multicast Test utility, which helps you determine if multicast is enabled between two or more computers. This is a connectivity test, not a load test, each instance will by default only transmit a single multicast packet once every two seconds. For network load testing, see Chapter 16, "Performing a Datagram Test for Network Performance."

Running the Multicast Test Utility

The Multicast Test utility supports a large number of configuration options, though only a few are required for basic operation. To run the Multicast Test utility use the following syntax from the command line:

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

Table 15–1 describes the available command line options for the Multicast Test utility.

<table>
<thead>
<tr>
<th>Command</th>
<th>Required/Optional</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-local</td>
<td>Optional</td>
<td>The address of the NIC to transmit on, specified as an IP address</td>
<td>localhost</td>
</tr>
<tr>
<td>-group</td>
<td>Optional</td>
<td>The multicast address to use, specified as IP:port.</td>
<td>237.0.0.1:9000</td>
</tr>
<tr>
<td>-ttl</td>
<td>Optional</td>
<td>The time to live for multicast packets.</td>
<td>4</td>
</tr>
<tr>
<td>-delay</td>
<td>Optional</td>
<td>The delay between transmitting packets, specified in seconds.</td>
<td>2</td>
</tr>
<tr>
<td>-display</td>
<td>Optional</td>
<td>The number of bytes to display from unexpected packets.</td>
<td>0</td>
</tr>
</tbody>
</table>

Sample Commands

```
java com.tangosol.net.MulticastTest -group 237.0.0.1:9000
```

For ease of use, `multicast-test.sh` and `multicast-test.cmd` scripts are provided in the Coherence bin directory, and can be used to execute this test.

Note: before Coherence 3.1 the following syntax was used, and scripts were not provided:

```
java com.tangosol.net.MulticastTest <ip-addr> <multicast-addr> <port> <ttl> <delay-secs>
```
Multicast Test Example

Presume that you want to test if you can use multicast address 237.0.0.1, port 9000 (the test’s defaults) to send messages between two servers: Server A with IP address 195.0.0.1 and Server B with IP address 195.0.0.2.

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

From a command prompt, enter the following command:

**Example 15–1 Command to Determine a Multicast Address**

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

After pressing ENTER, you should see the Multicast Test utility display how it is sending sequential multicast packets and receiving them. **Example 15–2 illustrates sample output.**

**Example 15–2 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: Received test packet 1 from self.
Tue Mar 17 15:59:53 EST 2008: Received test packet 2 from self.
...
```

When you have seen several these packets sent and received successfully, you can press CTRL-C to stop further testing.

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

You can repeat the same test on Server B to assure that it too has the multicast enabled for it’s port combination.

Now to test multicast communications between Server A and Server B. For this test we will use a nonzero TTL which will allow the packets to leave their respective servers. By default the test will use a TTL of 4, if you believe that there may be more network hops required to route packets between Server A and Server B, you may specify a higher TTL value.

Start the test on Server A and Server B by entering the following command into the command windows and pressing ENTER:

```
multicast-test.sh
```

You should see something like the following on Server A:

**Example 15–3 Sample Multicast Test Results from 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: Received test packet 2 from self.
```
Troubleshooting Multicast Communications

Performing a Multicast Connectivity Test

and something like the following on Server B:

Example 15–4 Sample Multicast Test Results on Server B

Starting test on ip=serverb/195.0.0.2, group=/237.0.0.1:9000, ttl=4
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: 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: 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.

You can see that 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 we start Server B and it receives packet 1 from Server B.

Troubleshooting Multicast Communications

If you are unable to establish bidirectional multicast communication please try the following:

- **Firewalls**—If any of the machines running the multicast test employ firewalls, the firewall may be blocking the traffic. Consult your OS/firewall documentation for details on allowing multicast traffic.
- **Switches**—Ensure that your switches are configured to forward multicast traffic.
- **IPv6**—On OSs which support IPv6 Java may be attempting to route the Multicast traffic over IPv6 rather then IPv4. Try specifying the following Java system property to force IPv4 networking `java.net.preferIPv4Stack=true`.

Troubleshooting Multicast Communications

- 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 will occur if you run the test with the same multicast address as a running Coherence cluster, or any other multicast application.

- Multiple NICs — If your machines have multiple network interfaces you may 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. You may also need to explicitly set your machines routing table to forward multicast traffic through the desired network interface. This can be done by issuing the command in Example 15–5:

**Example 15–5  Command to Set Machine Routing Table**
```
route add -net 224.0.0.0 netmask 240.0.0.0 dev eth1
```

Where eth1 is the device which will be designated to transmit multicast traffic.

- AIX — On AIX systems you may run into the following multicast issues:
  - IPv6 — In addition to specifying java.net.preferIPv4Stack=true you may need to configure the OS to perform IPv4 name resolution by adding hosts=local,bind4 to your /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 "well-known-addresses" on page H-55 for details.
  - MTU — Configure the MTU for the multicast device to 1500 bytes.
  - Cisco Switches — See "Deploying to Cisco Switches" on page M-2 for the list of known issues.
  - Foundry Switches — See "Deploying to Foundry Switches" on page M-5 for the list of known issues.

If multicast is not functioning properly, you will need to consult with your network administrator or sysadmin to determine the cause and to correct the situation.
Performing a Datagram Test for Network Performance

Included with Coherence is a Datagram Test utility which can be used to test and tune network performance between two or more machines. The Datagram test operates in one of three modes, either as a packet publisher, a packet listener, or both. When run a publisher will transmit UDP packets to the listener who will measure the throughput, success rate, and other statistics.

To achieve maximum performance it is suggested that you tune your environment based on the results of these tests. See Chapter 19, "Performance Tuning" for more information.

Running the Datagram Test Utility

The Datagram test supports a large number of configuration options, though only a few are required for basic operation. To run the Datagram Test utility use the following syntax from the command line:

```
java com.tangosol.net.DatagramTest <command value ...> <addr:port ...>
```

Table 16–1 describes the available command line options for the Datagram Test utility.

<table>
<thead>
<tr>
<th>Command</th>
<th>Required/Optional</th>
<th>Applicability</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-local</td>
<td>Optional</td>
<td>Both</td>
<td>The local address to bind to, specified as addr:port</td>
<td>localhost:9999</td>
</tr>
<tr>
<td>-packetSize</td>
<td>Optional</td>
<td>Both</td>
<td>The size of packet to work with, specified in bytes.</td>
<td>1468</td>
</tr>
<tr>
<td>-processBytes</td>
<td>Optional</td>
<td>Both</td>
<td>The number of bytes (in multiples of 4) of each packet to process.</td>
<td>4</td>
</tr>
<tr>
<td>-rxBufferSize</td>
<td>Optional</td>
<td>Listener</td>
<td>The size of the receive buffer, specified in packets.</td>
<td>1428</td>
</tr>
<tr>
<td>-txBufferSize</td>
<td>Optional</td>
<td>Publisher</td>
<td>The size of the transmit buffer, specified in packets.</td>
<td>16</td>
</tr>
<tr>
<td>-txRate</td>
<td>Optional</td>
<td>Publisher</td>
<td>The rate at which to transmit data, specified in megabytes.</td>
<td>unlimited</td>
</tr>
<tr>
<td>-txIterations</td>
<td>Optional</td>
<td>Publisher</td>
<td>Specifies the number of packets to publish before exiting.</td>
<td>unlimited</td>
</tr>
<tr>
<td>-txDurationMs</td>
<td>Optional</td>
<td>Publisher</td>
<td>Specifies how long to publish before exiting.</td>
<td>unlimited</td>
</tr>
</tbody>
</table>
Datagram Test Example

The following command line is for a listener:

```
java -server com.tangosol.net.DatagramTest -local box1:9999 -packetSize 1468
```

The following command line is for a publisher:

```
java -server com.tangosol.net.DatagramTest -local box2:9999 -packetSize 1468
```

For ease of use, `datagram-test.sh` and `datagram-test.cmd` scripts are provided in the Coherence `bin` directory, and can be used to execute this test.

**Sample Commands for a Listener and a Publisher**

Presume that you want 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 will act as a packet publisher and the other as a packet listener, the publisher will transmit packets as fast as possible and the listener will measure and report performance statistics. First start the listener on Server A.

**Example 16–1 Command to Start a Listener**

`datagram-test.sh`

After pressing ENTER, you should see the Datagram Test utility showing you that it is ready to receive packets.

**Example 16–2 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
```

As you can see by default the test will try to allocate a network receive buffer large enough to hold 1428 packets, or about 2 MB. If it is unable to allocate this buffer it will
report an error and exit. You can either decrease the requested buffer size using the 
-rxBufferSize parameter or increase your operating system network buffer settings. For best performance it is recommended that you increase the operating system buffers. See the following forum post for details on tuning your operating system for Coherence.

When the listener process is running you may start the publisher on Server B, directing it to publish to Server A.

**Example 16–3  Command to Start a Publisher**

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

After pressing ENTER, you should see the new Datagram test instance on Server B start both a listener and a publisher. Note in this configuration Server B listener will not be used. The output illustrates in Example 16–4 should appear in the Server B command window.

**Example 16–4  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
```

The series of o and O tick 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 you should see a corresponding set of i and I tick marks, representing network (I)nput. This indicates that the two test instances are communicating.

**Reporting**

Periodically, each side of the test (publisher and listener) will report performance statistics.

**Publisher Statistics**

The publisher simply reports the rate at which it is publishing data on the network. A typical report is as follows:

**Example 16–5  Sample Publisher Report**

```
Tx summary 1 peers:
 life: 97 MB/sec, 69642 packets/sec
```
The report includes both the current transmit rate (since last report) and the lifetime transmit rate.

**Listener Statistics**

Table 16–2 describes the statistics that can be reported by the listener.

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

As with the publisher both current and lifetime statistics are report. **Example 16–6** displays a typical report:

**Example 16–6  Sample Lifetime Statistics**

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. On a 100 Mb network setup you should be able to achieve rates of around 10 MB/sec. On
a 1 Gb network you should be able to achieve rates of around 100 MB/sec. Achieving these rates will likely require some tuning (see below).

**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/sec the test should put on the network.

**Bidirectional Testing**

You may also run the test in a bidirectional mode where both servers act as publishers and listeners. To do this simply restart test instances, supplying the instance on Server A with Server B's address, by running the command in Example 16–7 on Server A.

**Example 16–7  Running Datagram Test in Bi-Directional Mode**

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

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

**Distributed Testing**

You may also use more than two machines in testing, for instance you can setup two publishers to target a single listener. This style testing is far more realistic than simple one-to-one testing, and may identify bottlenecks in your network which you were not otherwise aware of.

Assuming you intend to construct a cluster consisting of four machines, you can run the datagram test among all of them as follows:

On Servera:

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

On Serverb:

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

On Serverc:

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

On Serverd:

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

This test sequence will cause all nodes to send a total of 100MB per second to all other nodes (that is, 33MB/node/sec). On a fully switched network 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 will 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 will cause all other nodes to delay testing until the final node is started.
Coherence*Extend extends the reach of the core Coherence TCMP cluster to a wider range of consumers, including desktops, remote servers and machines located across WAN connections. Typical uses of Coherence*Extend include providing desktop applications with access to Coherence caches (including support for Near Cache and continuous query) and Coherence cluster "bridges" that link together multiple Coherence clusters connected by using a high-latency, unreliable WAN.

Coherence*Extend consists of two basic components: a client and a Coherence*Extend clustered service hosted by one or more DefaultCacheServer processes. The adapter library includes implementations of both the CacheService and InvocationService interfaces that route all requests to a Coherence*Extend clustered service instance running within the Coherence cluster. The Coherence*Extend clustered service in turn responds to client requests by delegating to an actual Coherence clustered service (for example, a Partitioned or Replicated cache service). The client adapter library and Coherence*Extend clustered service use a low-level messaging protocol to communicate with each other. Coherence*Extend includes the Extend-TCP transport binding for this protocol which uses a high performance, scalable TCP/IP-based communication layer to connect to the cluster.

The choice of a transport binding is configuration-driven and is completely transparent to the client application that uses Coherence*Extend. A Coherence*Extend service is retrieved like a Coherence clustered service: using the CacheFactory class. Once obtained, a client uses the Coherence*Extend service in the same way as it would if it were part of the Coherence cluster. The fact that operations are being sent to a remote cluster node is transparent to the client application.

**General Instructions**

Configuring and using Coherence*Extend requires four basic steps:

1. Create a client-side Coherence cache configuration descriptor that includes one or more `<remote-cache-scheme>` and `<remote-invocation-scheme>` configuration elements
2. Create a cluster-side Coherence cache configuration descriptor that includes one or more `<proxy-scheme>` configuration elements
3. Launch one or more DefaultCacheServer processes
4. Create a client application that uses one or more Coherence*Extend services. See "Sample Coherence*Extend Client Application" on page 17-5.

---

Note: Coherence*Extend-JMS support has been deprecated.
5. Launch the client application

Configuring and Using Coherence*Extend-TCP

Client-side Cache Configuration Descriptor

A Coherence*Extend client that uses the Extend-TCP transport binding must define a Coherence cache configuration descriptor which includes a `<remote-cache-scheme>` and/or `<remote-invocation-scheme>` element with a child `<tcp-initiator>` element containing various TCP/IP-specific configuration information. Example 17–1 illustrates a sample descriptor.

*Example 17–1  Coherence*Extend Client Descriptor that uses Extend-TCP*

```xml
<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">
<cache-config>
  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>dist-extend</cache-name>
      <scheme-name>extend-dist</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <near-scheme>
      <scheme-name>extend-near</scheme-name>
      <front-scheme>
        <local-scheme>
          <high-units>1000</high-units>
        </local-scheme>
      </front-scheme>
      <back-scheme>
        <remote-cache-scheme>
          <scheme-ref>extend-dist</scheme-ref>
        </remote-cache-scheme>
      </back-scheme>
      <invalidation-strategy>all</invalidation-strategy>
    </near-scheme>

    <remote-cache-scheme>
      <scheme-name>extend-dist</scheme-name>
      <service-name>ExtendTcpCacheService</service-name>
      <initiator-config>
        <tcp-initiator>
          <remote-addresses>
            <socket-address>
              <address>localhost</address>
              <port>9099</port>
            </socket-address>
          </remote-addresses>
        </tcp-initiator>
      </initiator-config>
    </remote-cache-scheme>
  </caching-schemes>
</cache-config>
```
This cache configuration descriptor defines two caching schemes, one that uses Extend-TCP to connect to a remote Coherence cluster (<remote-cache-scheme>) and one that maintains an in-process size-limited near cache of remote Coherence caches (again, accessed by using Extend-TCP). Additionally, the cache configuration descriptor defines a <remote-invocation-scheme> that allows the client application to execute tasks within the remote Coherence cluster. Both the <remote-cache-scheme> and <remote-invocation-scheme> elements have a <tcp-initiator> child element which includes all TCP/IP-specific information needed to connect the client with the Coherence*Extend clustered service running within the remote Coherence cluster.

When the client application retrieves a NamedCache by using the CacheFactory using, for example, the name dist-extend, the Coherence*Extend adapter library will connect to the Coherence cluster by using TCP/IP (using the address localhost and port 9099) and return a NamedCache implementation that routes requests to the NamedCache with the same name running within the remote cluster. Likewise, when the client application retrieves a InvocationService by calling CacheFactory.getConfigurableCacheFactory().ensureService("Extend TcpInvocationService"), the Coherence*Extend adapter library will connect to the Coherence cluster by using TCP/IP (again, using the address localhost and port 9099) and return an InvocationService implementation that executes synchronous Invocable tasks within the remote clustered JVM to which the client is connected.

Note that the <remote-addresses> configuration element (see <tcp-initiator>) can contain multiple <socket-address> child elements. The Coherence*Extend adapter library will attempt to connect to the addresses in a random order, until either the list is exhausted or a TCP/IP connection is established.
Cluster-side Cache (a.k.a Coherence Extend Proxy) Configuration Descriptor

For a Coherence*Extend-TCP client to connect to a Coherence cluster, one or more DefaultCacheServer processes must be running that use a Coherence cache configuration descriptor. This descriptor must include a `<proxy-scheme>` element with a child `<tcp-acceptor>` element containing various TCP/IP-specific configuration information. Example 17–2 illustrates a sample descriptor.

**Example 17–2  Cluster-Side Cache Configuration Descriptor for Extend-TCP**

```xml
<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">
<cache-config>
  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>dist-*</cache-name>
      <scheme-name>dist-default</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <distributed-scheme>
      <scheme-name>dist-default</scheme-name>
      <lease-granularity>member</lease-granularity>
      <backing-map-scheme>
        <local-scheme/>
      </backing-map-scheme>
      <autostart>true</autostart>
    </distributed-scheme>

    <proxy-scheme>
      <service-name>ExtendTcpProxyService</service-name>
      <thread-count>5</thread-count>
      <acceptor-config>
        <tcp-acceptor>
          <local-address>
            <address=localhost</address>
            <port>9099</port>
          </local-address>
        </tcp-acceptor>
      </acceptor-config>
      <autostart>true</autostart>
    </proxy-scheme>
  </caching-schemes>
</cache-config>
```

This cache configuration descriptor defines two clustered services, one that uses Extend-TCP to allow remote Extend-TCP clients to connect to the Coherence cluster and a standard Partitioned cache service. Since this descriptor is used by a DefaultCacheServer it is important that the `<autostart>` configuration element for each service is set to true so that clustered services are automatically restarted upon termination. The `<proxy-scheme>` element has a `<tcp-acceptor>` child element which includes all TCP/IP-specific information needed to accept client connection requests over TCP/IP.

The Coherence*Extend clustered service will listen to a TCP/IP ServerSocket (bound to address localhost and port 9099) for connection requests. When, for example, a client attempts to connect to a Coherence NamedCache called `dist-extend-direct`, the Coherence*Extend clustered service will proxy
subsequent requests to the NamedCache with the *same name* which, in this case, will be a Partitioned cache.

**Launching an Extend-TCP DefaultCacheServer Process**

To start a DefaultCacheServer that uses the cluster-side Coherence cache configuration described earlier to allow Extend-TCP clients to connect to the Coherence cluster by using TCP/IP, you need to do the following:

- Change the current directory to the Coherence library directory (%COHERENCE_HOME%\lib on Windows and $COHERENCE_HOME/lib on UNIX)
- Make sure that the paths are configured so that the Java command will run
- Start the DefaultCacheServer command line application with the `-Dtangosol.coherence.cacheconfig` system property set to the location of the cluster-side Coherence cache configuration descriptor described earlier

For example (note that the following command is broken up into multiple lines here only for formatting purposes; this is a single command typed on one line):

```java
java -cp coherence.jar:<classpath to client application>
   -Dtangosol.coherence.cacheconfig=file://<path to the server-side cache configuration descriptor>
   com.tangosol.net.DefaultCacheServer
```

**Launching an Extend-TCP Client Application**

To start a client application that uses Extend-TCP to connect to a remote Coherence cluster by using TCP/IP, you need to do the following:

- Change the current directory to the Coherence library directory (%COHERENCE_HOME%\lib on Windows and $COHERENCE_HOME/lib on UNIX)
- Make sure that the paths are configured so that the Java command will run
- Start your client application with the `-Dtangosol.coherence.cacheconfig` system property set to the location of the client-side Coherence cache configuration descriptor described earlier

For example (note that the command in Example 17–3 is broken up into multiple lines here only for formatting purposes; this is a single command typed on one line):

**Example 17–3 Command to Start a Client Application that Uses Extend-TCP**

```java
java -cp coherence.jar:<classpath to client application>
   -Dtangosol.coherence.cacheconfig=file://<path to the client-side cache configuration descriptor>
   <client application Class name>
```

**Sample Coherence*Extend Client Application**

Example 17–4 demonstrates how to retrieve and use a Coherence*Extend CacheService and InvocationService. This example increments an Integer value in a remote Partitioned cache and then retrieves the value by executing an Invocable on the clustered JVM to which the client is attached:

**Example 17–4 Sample Coherence*Extend Application**

```java
public static void main(String[] asArg)
    throws Throwable
```
{ NamedCache cache = CacheFactory.getCache("dist-extend");
    Integer IValue = (Integer) cache.get("key");
    if (IValue == null)
    {
        IValue = new Integer(1);
    }
    else
    {
        IValue = new Integer(IValue.intValue() + 1);
    }
    cache.put("key", IValue);

    InvocationService service = (InvocationService)
        CacheFactory.getConfigurableCacheFactory()
            .ensureService("ExtendTcpInvocationService");

    Map map = service.query(new AbstractInvocable()
    {
        public void run()
        {
            setResult(CacheFactory.getCache("dist-extend").get("key"));
        }
    }, null);

    Integer IValue = (Integer) map.get(service.getCluster().getLocalMember());
}

Note that this example could also be run on a Coherence node (that is, within the cluster) verbatim. The fact that operations are being sent to a remote cluster node over TCP is completely transparent to the client application.

**Coherence*Extend InvocationService**

Since, by definition, a Coherence*Extend client has no direct knowledge of the cluster and the members running within the cluster, the Coherence*Extend InvocationService only allows Invocable tasks to be executed on the JVM to which the client is connected. Therefore, you should always pass a null member set to the query() method. As a consequence of this, the single result of the execution will be keyed by the local Member, which will be null if the client is not part of the cluster. This Member can be retrieved by calling service.getCluster().getLocalMember(). Additionally, the Coherence*Extend InvocationService only supports synchronous task execution (that is, the execute() method is not supported).

**Advanced Configuration**

**Network Filters**

Like Coherence clustered services, Coherence*Extend services support pluggable network filters. Filters can be used to modify the contents of network traffic before it is placed "on the wire". Most standard Coherence network filters are supported, including the compression and symmetric encryption filters. For more information on configuring filters, see Chapter 8, "Network Filters."
To use network filters with Coherence*Extend, a `<use-filters>` element must be added to the `<initiator-config>` element in the client-side cache configuration descriptor and to the `<acceptor-config>` element in the cluster-side cache configuration descriptor.

For example, to encrypt network traffic exchanged between a Coherence*Extend client and the clustered service to which it is connected, configure the client-side `<remote-cache-scheme>` and `<remote-invocation-scheme>` elements as illustrated in Example 17–5 (assuming the symmetric encryption filter has been named symmetric-encryption):

**Example 17–5  Client-Side Configuration to Encrypt Network Traffic**

```xml
<remote-cache-scheme>
  <scheme-name>extend-dist</scheme-name>
  <service-name>ExtendTcpCacheService</service-name>
  <initiator-config>
    <tcp-initiator>
      <remote-addresses>
        <socket-address>
          <address>localhost</address>
          <port>9099</port>
        </socket-address>
        <remote-addresses>
          <connect-timeout>10s</connect-timeout>
        </tcp-initiator>
        <outgoing-message-handler>
          <request-timeout>5s</request-timeout>
        </outgoing-message-handler>
        <use-filters>
          <filter-name>symmetric-encryption</filter-name>
        </use-filters>
      </remote-addresses>
    </initiator-config>
  </remote-cache-scheme>

<remote-invocation-scheme>
  <scheme-name>extend-invocation</scheme-name>
  <service-name>ExtendTcpInvocationService</service-name>
  <initiator-config>
    <tcp-initiator>
      <remote-addresses>
        <socket-address>
          <address>localhost</address>
          <port>9099</port>
        </socket-address>
        <remote-addresses>
          <connect-timeout>10s</connect-timeout>
        </tcp-initiator>
        <outgoing-message-handler>
          <request-timeout>5s</request-timeout>
        </outgoing-message-handler>
        <use-filters>
          <filter-name>symmetric-encryption</filter-name>
        </use-filters>
      </remote-addresses>
    </initiator-config>
  </remote-invocation-scheme>
```

Example 17–6 illustrates the configuration for the cluster-side `<proxy-scheme>` element:
Example 17–6  Cluster-Side Proxy Scheme Configuration

```xml
<proxy-scheme>
  <service-name>ExtendTcpProxyService</service-name>
  <thread-count>5</thread-count>
  <acceptor-config>
    <tcp-acceptor>
      <local-address>
        <address>localhost</address>
        <port>9099</port>
      </local-address>
    </tcp-acceptor>
    <use-filters>
      <filter-name>symmetric-encryption</filter-name>
    </use-filters>
  </acceptor-config>
  <autostart>true</autostart>
</proxy-scheme>
```

**Note:** The contents of the `<use-filters>` element must be the same in the client and cluster-side cache configuration descriptors.

### Connection Error Detection and Failover

When a Coherence*Extend service detects that the connection between the client and cluster has been severed (for example, due to a network, software, or hardware failure), the Coherence*Extend client service implementation (that is, CacheService or InvocationService) will dispatch a `MemberEvent.MEMBER_LEFT` event to all registered MemberListeners and the service will be stopped. If the client application attempts to subsequently use the service, the service will automatically restart itself and attempt to reconnect to the cluster. If the connection is successful, the service will dispatch a `MemberEvent.MEMBER_JOINED` event; otherwise, a fatal exception will be thrown to the client application.

A Coherence*Extend service has several mechanisms for detecting dropped connections. Some are inherent to the underlying TCP/IP protocol, whereas others are implemented by the service itself. The latter mechanisms are configured by using the `<outgoing-message-handler>` configuration element.

The primary configurable mechanism used by a Coherence*Extend client service to detect dropped connections is a request timeout. When the service sends a request to the remote cluster and does not receive a response within the request timeout interval (see the `<request-timeout>` subelement of `<outgoing-message-handler>`), the service assumes that the connection has been dropped. The Coherence*Extend client and clustered services can also be configured to send a periodic heartbeat over the connection (see `<heartbeat-interval>` and `<heartbeat-timeout>` subelements of `<outgoing-message-handler>`). If the service does not receive a response within the configured heartbeat timeout interval, the service assumes that the connection has been dropped.

**WARNING:** If you do not specify a `<request-timeout/>`, a Coherence*Extend service will use an infinite request timeout. In general, this is not a recommended configuration, as it could result in an unresponsive application. For most use cases, you should specify a reasonable finite request timeout.
Read-only NamedCache Access

By default, the Coherence*Extend clustered service allows both read and write access to proxied NamedCache instances. To prohibit Coherence*Extend clients from modifying cached content, use the `<cache-service-proxy>` child configuration element. Example 17–7 illustrates a sample configuration.

**Example 17–7  Client-Side Configuration to Allow Read-only Access to the Cache**

```xml
<proxy-scheme>

...  

<proxy-config>
 <cache-service-proxy>
  <read-only>true</read-only>
 </cache-service-proxy>
</proxy-config>

<autostart>true</autostart>
</proxy-scheme>
```

Client-side NamedCache Locking

By default, the Coherence*Extend clustered service disallows Coherence*Extend clients from acquiring NamedCache locks. To enable client-side locking, use the `<cache-service-proxy>` child configuration element. For example:

**Example 17–8  Client Configuration to Allow NamedCache Locking**

```xml
<proxy-scheme>

...  

<proxy-config>
 <cache-service-proxy>
  <lock-enabled>true</lock-enabled>
 </cache-service-proxy>
</proxy-config>

<autostart>true</autostart>
</proxy-scheme>
```

If you enable client-side locking and your client application uses the `NamedCache.lock()` and `unlock()` methods, it is important that you specify the **member-based** (rather than thread-based) locking strategy for any Partitioned or Replicated cache services defined in your cluster-side Coherence cache configuration descriptor. Because the Coherence*Extend clustered service uses a pool of threads to execute client requests concurrently, it cannot guarantee that the same thread will execute subsequent requests from the same Coherence*Extend client.

To specify the member-based locking strategy for a Partitioned or Replicated cache service, use the `<lease-granularity>` configuration element. Example 17–9 illustrates a sample configuration.

**Example 17–9  Client Configuration to Allow Locking for Partitioned or Replicated Caches**

```xml
<distributed-scheme>

 <scheme-name>dist-default</scheme-name>
 <lease-granularity>member</lease-granularity>
 <backing-map-scheme>
```
Disabling Proxied Services

By default, the Coherence*Extend clustered service exposes two proxied services to clients: a CacheService proxy and an InvocationService proxy. In some cases, it may be desirable to disable one of the two proxies. This is possible by using the <enabled> configuration element in each of the corresponding proxy configuration sections. For example, to disable the InvocationService proxy so that remote clients cannot execute Invocable objects within the cluster, you’d configure the Coherence*Extend clustered service as illustrated in Example 17–10:

Example 17–10  Client Configuration to Disable Proxy Service

```xml
<proxy-scheme>
  ...
  <proxy-config>
    <invocation-service-proxy>
      <enabled>false</enabled>
    </invocation-service-proxy>
  </proxy-config>
  <autostart>true</autostart>
</proxy-scheme>
```

Likewise, to prevent remote clients from accessing caches in the cluster, you would use a configuration similar to the one illustrated in Example 17–11:

Example 17–11  Client Configuration to Prevent Cache Access

```xml
<proxy-scheme>
  ...
  <proxy-config>
    <cache-service-proxy>
      <enabled>false</enabled>
    </cache-service-proxy>
  </proxy-config>
  <autostart>true</autostart>
</proxy-scheme>
```
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 boot 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. 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!
kernl: TSC cannot be used as a timesource.
kernl: Possible reasons for this are:
kernl: You're running with Speedstep,
kernl: You don't have DMA enabled for your hard disk (see hdparm),
kernl: Incorrect TSC synchronization on an SMP system (see dmesg).
kernl: 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 machines. If present it is suggested that you work with your system administrator to identify the cause and allow the TSC timesource to be used.
To achieve maximum performance with Coherence it is suggested that you test and tune your operating environment. Testing is covered in Chapter 16, "Performing a Datagram Test for Network Performance."

Tuning recommendations are available for:

- Operating System Tuning
- Network Tuning
- JVM Tuning
- Coherence Network Tuning

Operating System Tuning

- Socket Buffer Sizes
- High Resolution timesource (Linux)
- Datagram size (Microsoft Windows)
- Thread Scheduling (Microsoft Windows)
- Swapping

Socket Buffer Sizes

To help minimization of packet loss, the operating system socket buffers need to be large enough to handle the incoming network traffic while your Java application is paused during garbage collection. By default Coherence will attempt to allocate a socket buffer of 2MB. If your operating system is not configured to allow for large buffers Coherence will use smaller buffers. Most versions of UNIX have a very low default buffer limit, which should be increased to at least 2MB.

Starting with Coherence 3.1 you will receive the following warning if the operating system failed to allocate the full size buffer.

**Example 19–1 Message Indicating OS Failed to Allocate the Full Buffer Size**

UnicastUdpSocket failed to set receive buffer size to 1428 packets (2096304 bytes); actual size is 89 packets (131071 bytes). Consult your OS documentation regarding increasing the maximum socket buffer size. Proceeding with the actual value may cause sub-optimal performance.

Though it is safe to operate with the smaller buffers it is recommended that you configure your operating system to allow for larger buffers.
On Linux execute (as root):

```bash
sysctl -w net.core.rmem_max=2096304
sysctl -w net.core.wmem_max=2096304
```

On Solaris execute (as root):

```bash
ndd -set /dev/udp udp_max_buf 2096304
```

On AIX execute (as root):

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

On Windows:

Windows does not impose a buffer size restriction by default.

Other:

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

You may configure Coherence to request alternate sized buffers for packet publishers and unicast listeners by using the
`coherence/cluster-config/packet-publisher/packet-buffer/maximum-packets` and
`coherence/cluster-config/unicast-listener/packet-buffer/maximum-packets` elements. For more information, see "packet-buffer" on page H-35.

**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 boot 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. 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.

**Example 19–2 Log Message from a Linux Timesource**

``` plaintext
kernel: Losing too many ticks!
kernl: TSC cannot be used as a timesource.
kernl: Possible reasons for this are:
kernl: You're running with Speedstep,
kernl: You don't have DMA enabled for your hard disk (see hdparm),
kernl: Incorrect TSC synchronization on an SMP system (see dmesg).
kernl: 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 machines. If present it is suggested that you work with your system administrator to identify the cause and allow 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 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. Reboot

Note: Included in Coherence 3.1 and above is an optimize.reg script which will perform this change for you, it can be found in the coherence/bin directory of your installation. After running the script you must reboot your computer for the changes to take effect.

For more details on this parameter see Appendix C of

Thread Scheduling (Microsoft Windows)

Windows (including NT, 2000 and XP) 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.
5. Select the Advanced tab.
6. Under Processor scheduling, choose Background services.

Note: Coherence includes an optimize.reg script which will perform this change for you, it can be found in the coherence/bin directory of your installation.
Swapping

Ensure that you have sufficient memory such that you are not making active use of swap space on your machines. You may monitor the swap rate using tools such as `vmstat` and `top`. If you find that you are actively moving through swap space this will likely have a significant impact on Coherence’s performance. Often this will manifest itself as Coherence nodes being removed from the cluster due to long periods of unresponsiveness caused by them having been "swapped out".

Network Tuning

- Network Interface Settings
- Bus Considerations
- Network Infrastructure Settings
- Ethernet Flow-Control
- Path MTU

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 will display 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, please review Sun issues 102712 and 102741 which relate to packet corruption and multicast disconnections. These will 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.

---

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.
Bus Considerations

For 1Gb and faster PCI network cards the system's bus speed may be the limiting factor for network performance. PCI and PCI-X busses are half-duplex, and all devices will run at the speed of the slowest device on the bus. Standard PCI buses have a maximum throughput of approximately 1Gb/sec and thus are not capable of fully using a full-duplex 1Gb NIC. PCI-X has a much higher maximum throughput (1GB/sec), but can be hobbled by a single slow device on the bus. If you find that you are not able to achieve satisfactory bidirectional data rates it is suggested that you evaluate your machine's bus configuration. For instance simply relocating the NIC to a private bus may improve performance.

Network Infrastructure Settings

If you experience frequent multi-second communication pauses across multiple cluster nodes you may need to increase 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 19−3  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 configuration 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.

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 will then halt transmissions for the interval specified by the PAUSE frame. Note that this pause blocks all traffic from the transmitting side, even traffic destined for machines which are not overloaded. This can induce a head of line blocking condition, where one overloaded machine on a switch effectively slows down all other machines. Most switch vendors will recommend that Ethernet flow-control be disabled for inter switch links, and at most be used on ports which are directly connected to machines. Even in this setup head of line blocking can still occur, and thus it is advisable to disable Ethernet flow-control all together. 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.

For more details on this subject see
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 will result in an under used network. If your equipment uses a different MTU, then configure Coherence by specifying a packet size which is 32 bytes smaller then the network path’s minimal MTU. The packet size may be specified in coherence/cluster-config/packet-publisher/packet-size/maximum-length and preferred-length configuration elements. For more information on these elements, see "packet-size" on page H-41.

If you are unsure of your equipment’s MTU along the full path between nodes you can use either the standard ping or traceroute utility to determine it. To do this, execute a series of ping or traceroute operations between the two machines. With each attempt you will specify a different packet size, starting from a high value and progressively moving downward until the packets start to make it through without fragmentation. You will need to specify a particular packet size, and to not fragment the packets.

On Linux execute:

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

On Solaris execute:

```bash
traceroute -F serverb 1468
```

On Windows execute:

```bash
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 need to use packets smaller then 1468 you may want to contact your network administrator to get the MTU increased to at least 1500.

JVM Tuning

- Server Mode
- Sizing the Heap
- GC Monitoring & Tuning

Server Mode

It is recommended that you run all your Coherence JVMs in server mode, by specifying the `-server` on the JVM command line. This allows for several performance optimizations for long running applications.

Sizing the Heap

It is generally recommended that heap sizes be kept at 1GB or below as larger heaps will have a more significant impact on garbage collection times. On 1.5 and higher JVMs larger heaps are reasonable, but will likely require additional GC tuning. For more information, see “Heap Size Considerations”.
Running with a fixed sized heap will save your JVM from having to grow the heap on demand and will result 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 -Xms1024m -Xmx1024m ...
```

Note that the JVM process will consume more system memory than the specified heap size, for instance a 1GB JVM will consume 1.3GB of memory. This should be taken into consideration when determining the maximum number of JVMs which you will run on a machine. The actual allocated size can be monitored with tools such as top. See "Heap Size Considerations" for additional details on heap size considerations.

**GC Monitoring & Tuning**

Frequent garbage collection pauses which are in the range of 100ms or more are likely to have a noticeable impact on network performance. During these pauses a Java application is unable to send or receive packets, and in the case of receiving, the operating system buffered packets may be discarded and need to be retransmitted.

Specify \(-verbose:gc\) or \(-Xloggc:\) on the JVM command line to monitor the frequency and duration of garbage collection pauses.

See http://java.sun.com/docs/hotspot/gc5.0/gc_tuning_5.html for details on GC tuning.

Starting with Coherence 3.2 log messages will be generated when one cluster node detects that another cluster node has been unresponsive for a period, generally indicating that a target cluster node was in a GC cycle.

**Example 19–4 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 stats were last reset. Nodes reported as unresponsive for more than a few percent of their lifetime may be worth investigating for GC tuning.

**Coherence Network Tuning**

Coherence includes configuration elements for throttling the amount of traffic it will place on the network; see the documentation for \(<traffic-jam>, <flow-control>\) and \(<burst-mode>\), these settings are used to control the rate of packet flow within and between cluster nodes.

**Validation**

To determine how these settings are affecting performance you need to check if your cluster nodes are experiencing packet loss and/or packet duplication. This can be obtained by looking at the following JMX stats on various cluster nodes:

- **ClusterNodeMBean.PublisherSuccessRate**—If less than 1.0, packets are being detected as lost and being resent. Rates below 0.98 may warrant investigation and tuning.

- **ClusterNodeMBean.ReceiverSuccessRate**—If less than 1.0, the same packet is being received multiple times (duplicates), likely due to the publisher being overly aggressive in declaring packets as lost.
ClusterNodeMBean.WeakestChannel—Identifies the remote cluster node which the current node is having most difficulty communicating with.

For information on using JMX to monitor Coherence see Chapter 21, "How to Manage Coherence Using JMX."
If you want to perform unit testing or quick restarts, you might find it more convenient to avoid the network and run in single-server mode. To constrain Coherence to run on a single server, set the multicast packet time-to-live to 0, and set the unicast IP address.

You can configure these properties either by declaring system properties on the command line or by editing the values in the operational configuration descriptor, tangosol-coherence.xml file.

### Setting Single Server Mode in the Operation Configuration Descriptor

In the tangosol-coherence.xml file, the multicast packet time to live value is defined by the `<time-to-live>` subelement of the `<multicast-listener>` element. The `<time-to-live>` value determines the maximum number of "hops" a packet may traverse between network segments. Setting this subelement to 0 keeps the packets from leaving the originating machine.

The unicast IP address is defined by the `<address>` subelement of the `<unicast-listener>` element. This subelement specifies the IP address that a Socket will listen or publish on. Setting this subelement to an IP address that is never used will prevent Coherence from joining the network.

The following XML code fragment illustrates a single server mode configuration in the tangosol-coherence.xml file.

#### Example 20–1  Single Server Mode Configuration

```xml
<coherence>
  <cluster-config>
    ...
    <multicast-listener>
      <time-to-live>0</time-to-live>
    ...
    <multicast-listener>
    ...
    <unicast-listener>
      <address>127.0.0.1</address>
    ...
    </unicast-listener>
    ...
  </cluster-config>
</coherence>
```
Setting Single Server Mode on the Command Line

Coherence defines system properties that allow you to set the multicast packet time-to-live and the unicast IP address for single server mode on the command line. This feature is useful when you need to change the settings for a single JVM, or if you want to start an application with settings that differ from those in the descriptor files.

The following system properties can be used to define single server mode.

- `tangosol.coherence.ttl`—Multicast packet time to live. Set to "0" to keep the packets from leaving the originating machine.
- `tangosol.coherence.localhost`—Unicast IP address.

The sample command line in Example 20–2 illustrates starting coherence in single server mode:

```
Example 20–2  Command to Start Coherence in Single Server Mode
java -Dtangosol.coherence.localhost=127.0.0.1 -Dtangosol.coherence.ttl=0 -jar coherence.jar
```

See Appendix L, "Command Line Overrides" for more information on system properties defined by Coherence.
Part III
Managing and Monitoring Oracle Coherence

This section contains the following chapters:

- Chapter 21, "How to Manage Coherence Using JMX"
- Chapter 22, "JMX Reporter"
- Chapter 23, "How to Create a Custom Report"
- Chapter 24, "How to Modify Report Batch"
- Chapter 25, "Analyzing Reporter Content"
- Chapter 26, "How to Run a Report on Demand"
- Chapter 27, "Configuring Custom MBeans"
- Chapter 28, "How to Manage Custom MBeans Within the Cluster"
Coherence includes facilities for managing and monitoring Coherence resources by using the Java Management Extensions (JMX) API. JMX is a Java standard for managing and monitoring Java applications and services. It defines a management architecture, design patterns, APIs, and services for building general solutions to manage Java-enabled resources. This section assumes familiarity with JMX terminology. If you are new to JMX, you should start with this article: "Getting Started with Java Management Extensions (JMX): Developing Management and Monitoring Solutions".

To manage Coherence using JMX:

- Add JMX libraries to the Coherence classpath (if necessary)
- Configure the Coherence Management Framework
- Access Coherence MBeans to view and manipulate them using a JMX client of your choice

**Note:** JMX support:

Coherence Enterprise Edition and higher support clustered JMX, allowing access to JMX statistics for the entire cluster from any member. Coherence Standard Edition provides only local JMX information.

**Add JMX libraries to the Coherence classpath**

To manage a Coherence cluster using JMX, ensure that you have the necessary JMX 1.0 or later classes (javax.management.*) in the classpath of at least one Coherence cluster node, known as an MBeanServer host. The cluster nodes that are not MBeanServer hosts will be managed by the MBeanServer host(s) by using the Coherence Invocation service.

All compliant Java SE 5.0 JREs and Java EE application servers supply a JMX 1.0 or later implementation; therefore, if the MBeanServer host node is running within a Java SE 5.0 JVM or Java EE application server, no additional actions are necessary. For standalone applications running within a pre-Java SE 5.0 JVM, you can download the necessary JMX libraries from the JMX download Web site and add them to the classpath.
Configure the Coherence Management Framework

In most cases, you can enable JMX management simply by setting the `tangosol.coherence.management` Java system property on all Coherence cluster nodes that are acting as MBeanServer hosts and the `tangosol.coherence.management.remote` Java system property on all cluster nodes:

- `Dtangosol.coherence.management=all`
- `Dtangosol.coherence.management.remote=true`

The use of dedicated JMX cluster members is a common pattern. This approach avoids loading JMX software into every single cluster member, while still providing fault-tolerance should a single JMX member run into issues.

In general, the Coherence Management Framework is configured by the `management-configuration` element in a Coherence Operational Override Configuration deployment descriptor (`tangosol-coherence-override.xml`). The following subelements control the behavior of the Management Framework:

<table>
<thead>
<tr>
<th>Table 21-1</th>
<th>Elements that Control the Behavior of the Management Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>allow-remote-management</td>
<td>Specifies whether this cluster node will register its MBeans in a remote MBeanServer(s).</td>
</tr>
<tr>
<td>domain-name</td>
<td>Specifies the name of the JMX domain used to register MBeans exposed by the Coherence Management Framework.</td>
</tr>
<tr>
<td>managed-nodes</td>
<td>Specifies whether a cluster node's JVM has an in-process MBeanServer and if so, whether the node allows management of other nodes' managed objects. Valid values are none, local-only, remote-only and all. For example, if a node has an in-process MBeanServer and you would like this node to manage other nodes' MBeans, then set this attribute to all.</td>
</tr>
<tr>
<td>read-only</td>
<td>Specifies whether the MBeans exposed by this cluster node allow operations that modify run-time attributes.</td>
</tr>
</tbody>
</table>

For additional information on each of these attributes, see Appendix H, "Operational Configuration Elements."

**Note:** Coherence monitors Java platform MBeans to display JVM statistics from across the cluster alongside the other statistics tracked by Coherence. Platform MBeans are a feature of 1.5 JVMs. When running on a 1.4 JVM, the following exception occurs:

```
java.lang.NoClassDefFoundError:
javax.management.MalformedObjectNameException
```

This message can either be safely ignored, or you can disable the feature using the following system property:

- `Dtangosol.coherence.management.jvm.all=false`
Access Coherence MBeans

After configuring the Coherence Management Framework and launching one or more Coherence cluster nodes (at least one being an MBeanServer host) you can view and manipulate the Coherence MBeans registered by all cluster nodes using standard JMX API calls. See the Javadoc for the `com.tangosol.net.management.Registry` class for details on the various MBean types registered by Coherence clustered services.

Coherence ships with two examples that demonstrate accessing Coherence MBeans by using JMX. The first uses the `HttpAdapter`, shipped as part of the JMX reference implementation (jmxtools.jar). To run the example on a pre-Java SE 5.0 JVM, start the Coherence command line application using the following command on Windows (note that it is broken up into multiple lines here only for formatting purposes; this is a single command entered on one line):

```
java -cp jmxri.jar;jmxtools.jar;coherence.jar
   -Dtangosol.coherence.management=all
   -Dtangosol.coherence.management.remote=true
   com.tangosol.net.CacheFactory
```

On UNIX:

```
java -cp jmxri.jar;jmxtools.jar;coherence.jar
   -Dtangosol.coherence.management=all
   -Dtangosol.coherence.management.remote=true
   com.tangosol.net.CacheFactory
```

When the Coherence command line application has started, enter `jmx 8082` and press `return`. This starts the `HttpAdapter` on `http://localhost:8082` in the cluster node's JVM and makes the cluster node an MBeanServer host. You can now use the `HttpAdapter` Web application to view and manipulate Coherence MBeans registered by all cluster nodes:
Alternatively, you can run this example with the Sun Java SE 5.0 JVM and use the JConsole utility included with the Sun Java SE 5.0 JDK to view and manipulate Coherence MBeans. To do so, start the Coherence command line application using the following command (note that it is broken up into multiple lines here only for formatting purposes; this is a single command entered on one line):

```
java -Dcom.sun.management.jmxremote  
-Dtangosol.coherence.management=all  
-Dtangosol.coherence.management.remote=true  
-jar coherence.jar
```

When the Coherence command line application has started, launch the JConsole utility (located in the bin directory of the Sun Java SE 5.0 JDK distribution) and open a new connection to the JVM running the Coherence command line application:
Using Coherence MBeanConnector to Access MBeans

The second example is a JSP page (JmxCacheExplorer.jsp) that displays basic information on each running Coherence cache using JMX API calls. You can find this example in the examples/jsp/explore directory under the root of your Coherence installation.

Additional JMX examples may be found on the Coherence Forums.

Using Coherence MBeanConnector to Access MBeans

Coherence ships with a program to launch a cluster node as a dedicated MBeanServer host. This program provides access to Coherence MBeans by using the JMX Remote API using RMI or the HTTP server provided by Sun's JMX RI. The RMI and HTTP ports are user-configurable, allowing for access through a firewall. The server is started using the following command (note that it is broken up into multiple lines here only for formatting purposes; this is a single command entered on one line):

```
java -Dtangosol.coherence.management=all 
   -cp coherence.jar com.tangosol.net.management.MBeanConnector [-http -rmi]
```
To allow access by using JMX RMI, include the `-rmi` flag. To allow access by using HTTP and a web browser, include the `-http` flag. Both flags may be included; however at least one must present for the node to start.

Table 21–2 describes optional properties that can be used for JMX RMI configuration:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tangosol.coherence.management.remote.host</td>
<td>The host that the JMX server will bind to. Default is localhost. (NOTE: on Redhat Linux this may have to be changed to the host name or IP address)</td>
</tr>
<tr>
<td>tangosol.coherence.management.remote.registryport</td>
<td>The port used for the JMX RMI registry. Default is 9000.</td>
</tr>
<tr>
<td>tangosol.coherence.management.remote.connectionport</td>
<td>The port used for the JMX RMI connection. Default is 3000.</td>
</tr>
</tbody>
</table>

Table 21–3 describes optional properties that can be used for HTTP configuration. (NOTE: This flag requires Sun’s JMX RI in the classpath):

Table 21–3 Optional Properties that can be used for Http Configuration

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tangosol.coherence.management.remote.httpport</td>
<td>The port used for the HTTP connection. Default is 8888.</td>
</tr>
</tbody>
</table>

To connect by using JConsole with default settings use the following command:

```
```

To connect by using HTTP with default settings use the following URL:

```
http://localhost:8888
```

**Configuring Management Refresh Methodology**

The current release of Coherence offers several ways to reduce the latency of management information. Refresh policy was introduced in Coherence 3.3 to allow for optimization of the retrieval of information from remotely managed nodes. Two new settings were added to help integrators and administrators configure the refresh policy.

The `tangosol.coherence.management.refresh.expire` property specifies the minimum time interval between the remote retrieval of management information from remote nodes.

```
-Dtangosol.coherence.management.refresh.expire
```

The value of this element must be in the following format:

```
[\d]+(\.[\d]+)?[MS|ms|S|s|M|m|H|h|D|d]?
```

where the first non-digits (from left to right) indicate the unit of time duration:

- MS or ms (milliseconds)
- S or s (seconds)
- M or m (minutes)
- H or h (hours)
- D or d (days)

If the value does not contain a unit, a unit of milliseconds is assumed.

The `tangosol.coherence.management.refresh.policy` property defines the refresh policy for the MBean.

`-Dtangosol.coherence.management.refresh.policy`

Table 21–4 describes valid values for this property.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>refresh-ahead (default)</td>
<td>MBeans are refreshed before they are requested based on prior usage patterns after the expiry delay has passed. This setting can reduce latency of the management information with a minor increase in network consumption. This setting is best when MBeans are accessed in a repetitive/programmatic pattern.</td>
</tr>
<tr>
<td>refresh-behind</td>
<td>Each MBean will be refreshed after the data is accessed. This method ensures optimal response time. However, the information returned will be offset by the last refresh time.</td>
</tr>
<tr>
<td>refresh-expired</td>
<td>This setting has the same functionality as in pre-3.4 Coherence releases. Each MBean will be refreshed from the remote node when it is accessed and the expiry delay has passed from the last refresh. This setting is best used when MBeans are accessed in a random pattern.</td>
</tr>
</tbody>
</table>
Coherence provides a JMX reporting capability (the Reporter). The Reporter provides out-of-the-box reports that help administrators and developers manage capacity and trouble shoot problems. Custom reports can also be created.

**Note:** Plan for archiving and removing. Due to the volume of the information created by the Reporter, you must have a plan for archiving and/or removing the results BEFORE starting the Reporter.

### Basic Configuration

Enabling the Reporter with basic content requires setting the system properties:

Example 22–1 illustrates the properties on the “management” node.

**Example 22–1  System Properties for Reporter on the “Management” Node**

- `Dtaangosol.coherence.management.report.autostart=true`
- `Dtaangosol.coherence.management=all`
- `Dcom.sun.management.jmxremote`

Example 22–2 illustrates the properties on the “managed” node.

**Example 22–2  System Properties for Reporter on the “Managed” Node**

- `Dtaangosol.coherence.management.remote=true`

Basic configuration will create a single Reporter node that will log the JMX statistics for all nodes in the cluster. The log files will be placed in the working directory of the application.

### Administration

The JMX Reporter is managed through an MBean under the Coherence Domain. The Reporter MBean provides information related to the status and performance of the Reporter. The MBean also provides the capability to start and stop the service and run a report on demand.

**Figure 22–1** illustrates the attributes available to the Reporter MBean. The JConsole is being used to view the MBean.
Figure 22–1  Reporter Attributes in JConsole

This figure is described in the text.

***********************************************************************************************

Figure 22–2 illustrates the operations available to the Reporter MBean. For a full description of the Reporter Attributes see the Reporter section of the javadoc.
Figure 22–2  Reporter Operations in JConsole

This figure is described in the text.

Data Analysis

Seven files are created each hour by the Reporter. Each file is prefixed with the date and hour the report was executed in a YYYYMMDDHH format. This allows for easy location and purging of unwanted information. The files generated are described in Table 22–1:

Table 22–1  File Names Generated by Reporter

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYYYMMDDHH-memory-status.txt</td>
<td>Contains memory and garbage collection information about each node.</td>
</tr>
<tr>
<td>YYYYMMDDHH-network-health.txt</td>
<td>Contains the publisher success rates and receiver success rates for the entire grid</td>
</tr>
<tr>
<td>YYYYMMDDHH-network-health-detail.txt</td>
<td>Contains the publisher success rates and receiver success rates for each node</td>
</tr>
<tr>
<td>YYYYMMDDHH-node.txt</td>
<td>Contains the list of nodes that were members of the grid</td>
</tr>
<tr>
<td>YYYYMMDDHH-service.txt</td>
<td>Contains Request and Task information for each service.</td>
</tr>
<tr>
<td>YYYYMMDDHH-proxy.txt</td>
<td>Contains utilization information about each proxy node in the grid</td>
</tr>
</tbody>
</table>
Creating Custom Reports

1. Create the custom report configuration file. See Chapter 23, "How to Create a Custom Report."

2. Update report batch to execute the report. See Chapter 24, "How to Modify Report Batch."

3. Run on demand. See Chapter 26, "How to Run a Report on Demand."

Running Reporter in a Distributed Configuration

A distributed configuration is only recommended in situations where grid stability is an issue. In this configuration, the distributed reporters will run independently, and the execution times will not align. Therefore, grid level analysis is extremely difficult but node level analysis during periods when nodes may be leaving or joining the grid will still be available.

When running in distributed mode, each node logs local JMX statistics while allowing for centralized management of the Reporters. To enable this configuration set the following system properties

On the "managing" node:

Example 22–3 System Properties for Reporter in Distributed Mode on the "Managing" Node

-Dtangosol.coherence.management.report.autostart=false
-Dtangosol.coherence.management.report.distributed=true
-Dtangosol.coherence.management=all
-Dcom.sun.management.jmxremote

On the "managed" node:

Example 22–4 System Properties for Reporter in Distributed Mode on the "Managed" Node

-Dtangosol.coherence.management.report.autostart=true
-Dtangosol.coherence.management.report.distributed=true
-Dtangosol.coherence.management=local-only
-Dtangosol.coherence.management.remote=true

Table 22–1 (Cont.) File Names Generated by Reporter

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYYYMMDDHH-cache-usage.txt</td>
<td>Contains cache utilization (put, get, and so on) statistic for each cache</td>
</tr>
</tbody>
</table>

See Chapter 25, "Analyzing Reporter Content" for a complete description of the data contained in each file.
The Coherence reporting feature provides a capable query definition that allows for any information residing in the Coherence JMX data source to be logged to a text file. After a custom report has been created, it can be included in a report batch and executed on a specified time interval by the `ReportControl MBean`. For a complete description of the report configuration XML file see the `report-config.dtd` which is packaged in the `coherence.jar` file.

### Configuring a Report File

To correctly generate the report file, several elements must be configured. These elements are described in **Table 23–1**.

<table>
<thead>
<tr>
<th>Element</th>
<th>Optional/Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;file-name&gt;</code></td>
<td>Required</td>
<td>The file name to create or update when the report is executed. For more information, see &quot;file-name Element&quot;.</td>
</tr>
<tr>
<td><code>&lt;delim&gt;</code></td>
<td>Optional</td>
<td>The column delimiter for the report. Valid values are <code>{tab}</code>, <code>{space}</code> or a printable character. The default value is <code>{tab}</code>. If a string longer than one character is entered, the first character in the string is used.</td>
</tr>
<tr>
<td><code>&lt;hide-headers&gt;</code></td>
<td>Optional</td>
<td>A boolean element to determine if the headers are to be included in the report. If true, the column headers and the report description are not included in the file. The default value is false.</td>
</tr>
</tbody>
</table>

### file-name Element

The value of this element will have the output path from the `<report-path>` element pre-pended to it and the report will be generated in this location. If the Coherence node cannot access this path, then the file will not be created.

### file-name Macros

There are pre-defined macros that you can use with the file-name element. These macros can add a node name, a batch number, or a date to the generated file name.
Specifying Data Columns

Data columns can be sourced from JMX Attributes, ObjectName key part, JMX composite attributes, JMX joined attributes, Report macros, and Report Constants.

How to Include an Attribute

To include data from MBeans returned from the query-pattern, the report must have a column with an attribute source. This is the most common item that will be included in the report.

Example 23–1 illustrates how to include the RoleName attribute from the query pattern Coherence:type=Node,*.

Example 23–1  Including an Attribute Obtained from a Query Pattern

```xml
<column id = "RoleName">
  <type>attribute</type>
  <name>RoleName</name>
  <header>Role Name</header>
</column>
```

Table 23–2  Macros that can be Used with the file-name Element

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch</td>
<td>Will include a batch Identifier into the filename of the report. If the information is kept for a short amount of time or is frequently uploaded into and RDBMS.</td>
</tr>
<tr>
<td>date</td>
<td>Will include the date (with the format YYYYMMDD), into the file name of the report. This is used mostly when the data will only be kept for a certain period and then will be discarded.</td>
</tr>
<tr>
<td>node</td>
<td>Will include the node ID into the file name string. This configuration setting is helpful when many nodes are executing the same report and the output files will be integrated for the analysis.</td>
</tr>
</tbody>
</table>

file-name Macro Examples

The following example will create a file 20090101_network_status.txt on January 1, 2009. The filename will change with the system time on the node executing the report.

```xml
<file-name>{{date}}_network_status.txt</file-name>
```

The following example will create a file 00012_network_status.txt when the report is executed on node 12. Note that due to the volatile nature of the Node Id, long term storage in this manner is not recommended.

```xml
<file-name>{node}_network_status.txt</file-name>
```

The following example will create a file 0000000021_network_status.txt on the 21st execution of the report. Note that due to the volatile nature of the batch, long term storage in this manner is not recommended.

```xml
<file-name>{batch}_network_status.txt</file-name>
```
How to Include Part of the Key

A value that is present in an ObjectName key can be obtained from the ObjectNames returned from the query-pattern. This value can subsequently be included in the report.

Example 23–2 illustrates how to include the nodeId key part from the query pattern Coherence:type=Node,*.

Example 23–2 Including Part of an ObjectName Key in a Report

```xml
<column id ="NodeId">
  <type>key</type>
  <name>nodeId</name>
  <header>Node Id</header>
</column>
```

How to Include Information from Composite Attributes

JMX composite values can be used to include part of a composite data attribute in a report.

Example 23–9 illustrates how to include the startTime of the LastGCInfo attribute from the query pattern java.lang:type=GarbageCollector,*.

Example 23–3 Including Information from a Composite Attribute in a Report

```xml
<column id="LastGCStart">
  <type>attribute</type>
  <name>LastGcInfo/startTime</name>
  <header>LastGC Start Time</header>
</column>
```

How to Include Information from Multiple MBeans

A JMX join attribute is required when a report requires information from multiple MBeans. The major considerations when creating a join is to determine both the primary query, the join query and the foreign key. The primary query should be the query that returns the appropriate number of rows for the report. The join query pattern must reference a single MBean and can not contain a wild card (*). The foreign key is determined by what attributes from the primary query that are required to complete the join query string.

The reporter feature that enables joins between MBeans is a column substitution macro. The column substitution allows for the resulting value from a column to be included as part of a string. A column substitution macro is a column ID attribute surrounded by curly braces "{}". The reporter does not check for cyclical references and will fail during execution if a cycle is configured.

Including Multiple MBean Information Example

You can draw information from more than one MBean and include it in a report. This requires a join between the MBeans.

Note: The major limitation of join attributes is that the result of the join must have only one value.
For example, if a report requires the TotalGets from the Cache MBean (Coherence:type=cache, *) and RoleName from the Node MBean (Coherence:type=Node, *), then a join attribute must be used.

Since a greater number of MBeans will come from the Cache MBean, Coherence:type=Cache, * would be the primary query and the RoleName would be the join attribute. The foreign key for this join is the nodeId key part from the Cache MBean and it must be included in the report. The configuration for this scenario is illustrated in Example 23–4.

**Example 23–4 Including Information from Multiple MBeans in a Report**

```xml
<column id="RoleName">
  <type>attribute</type>
  <name>RoleName</name>
  <header>Role Name</header>
  <query>
    <pattern>Coherence:type=Node,nodeId={NodeFK}</pattern>
  </query>
</column>

<column id="NodeFK">
  <type>key</type>
  <name>nodeId</name>
  <header>Node Id</header>
</column>
```

**How to Use Report Macros**

There are three report macros that can be included in a report:

- Report Time (report-time)—is the time and date that the report was executed. This information is useful for time series analysis.
- Report Batch/Count (report-count)—is a long identifier that can be used to correlate information from different reports executed at the same time.
- Reporting Node (report-node)—is used when integrating information from the same report executed on different nodes or excluding the executing node information from the report.

To include the execution time into the report:

**Example 23–5 Including Execution Time in a Report**

```xml
<column id="ReportTime">
  <type>global</type>
  <name>{report-time}</name>
  <header>Report Time</header>
</column>
```

To include the Report Batch/Count:

**Example 23–6 Including the Report Batch/Count in a Report**

```xml
<column id="ReportBatch">
  <type>global</type>
  <name>{report-count}</name>
  <header>batch</header>
</column>
```
To include the execution node:

**Example 23–7 Including the Execution Node**

```xml
<column id='ReportNode'>
    <type>global</type>
    <name>{report-node}</name>
    <header>ExecNode</header>
    <hidden>true</hidden>
</column>
```

### How to Include Constant Values

Report constants can be used to either static values or report parameters. These constants can be either double or string values. Often, these are used in filters to limit the results to a particular data set or in calculations.

**Example 23–8** illustrates how to include a constant double of 1.0 in a report:

**Example 23–8 Including a Constant Numeric Value in a Report**

```xml
<column id='One'>
    <type>constant</type>
    <header>Constant1</header>
    <data-type>double</data-type>
    <value>1.0</value>
    <hidden>true</hidden>
</column>
```

**Example 23–9** illustrates how to include the constant string `dist-Employee` in a report:

**Example 23–9 Including a Constant String in a Report**

```xml
<column id='EmployeeCacheName'>
    <type>constant</type>
    <header>Employee Cache Name</header>
    <data-type>string</data-type>
    <value>dist-Employee</value>
    <hidden>true</hidden>
</column>
```

### Including Queries in a Report

The query is the foundation of the information included in a report. Each query includes a query pattern, column references, and an optional filter reference. The query pattern is a string that is a JMX `ObjectName` query string. This string can return one or more MBeans. The column references must be defined in the `<columns>` section of the report definition file. The filter reference must be defined in the `<filters>` section of the report section.

**Example 23–10** illustrates how to include the list all the Node IDs and RoleNames in the cluster where the RoleName equals CoherenceServer.

**Example 23–10 Including a List of the Cluster’s NodeIDs and RoleNames in a Report**

```xml
<filters>
    <filter id='equalsRef'>
        <type>equals</type>
        <params>
            <param name='roleName'>CoherenceServer</param>
        </params>
    </filter>
</filters>
```
Using Filters to Construct Reports

Filters limit the data returned in the Report. Filters are either comparison filters or composite filters. Comparison Filters evaluate the results of two columns while composite filters evaluate the boolean results from one or two filters. Comparison filters are equals, greater, and less.

Composite Filter types are and, or, and not. Each composite filter evaluates the filter parameters first to last and apply standard boolean logic. Composite filter evaluation uses standard short circuit logic. Cyclic references checks are not performed during execution. If a cyclic reference occurs, it will create a runtime error.

Example 23–11 illustrates how to define an equals filter where RoleRef and StringRef are defined columns.

**Example 23–11 Using an Equals Filter for a Report**

```xml
<filters>
  <filter id="equals">
    <type>equals</type>
    <params>
      <column-ref>RoleRef</column-ref>
      <column-ref>StringRef</column-ref>
    </params>
  </filter>
</filters>
```
Example 23–12 illustrates how to define a filter where the number of PacketsResent are greater than PacketsSent (assuming PacketsResent and PacketsSent are valid column references).

**Example 23–12  Defining a "Greater Than" Filter for a Report**

```xml
<filters>
  <filter id="greaterRef">
    <type>greater</type>
    <params>
      <column-ref>PacketsResent</column-ref>
      <column-ref>PacketsSent</column-ref>
    </params>
  </filter>
</filters>
```

Example 23–13 illustrates how to define a filter where the number of PacketsResent are less than PacketsSent (assuming PacketsResent and PacketsSent are valid column references).

**Example 23–13  Defining a "Less Than" Filter for a Report**

```xml
<filters>
  <filter id="greaterRef">
    <type>less</type>
    <params>
      <column-ref>PacketsResent</column-ref>
      <column-ref>PacketsSent</column-ref>
    </params>
  </filter>
</filters>
```

Example 23–14 illustrates how to define an and filter (assuming all column-ref values are valid).

**Example 23–14  Defining an "And" Filter for a Report**

```xml
<filters>
  <filter id="equalsRef">
    <type>equals</type>
    <params>
      <column-ref>RoleRef</column-ref>
      <column-ref>StringRef</column-ref>
    </params>
  </filter>

  <filter id="greaterRef">
    <type>greater</type>
    <params>
      <column-ref>PacketsResent</column-ref>
      <column-ref>PacketsSent</column-ref>
    </params>
  </filter>

  <filter>
    <type>and</type>
    <params>
      <filter-ref>greaterRef</filter-ref>
      <filter-ref>equalsRef</filter-ref>
    </params>
  </filter>
</filters>
```
Example 23–15 illustrates how to define an or filter (assuming all column-ref values are valid).

Example 23–15  Defining an “Or” Filter for a Report

<filters>
  <filter id="equalsRef">
    <type>equals</type>
    <params>
      <column-ref>RoleRef</column-ref>
      <column-ref>StringRef</column-ref>
    </params>
  </filter>
  <filter id="greaterRef">
    <type>greater</type>
    <params>
      <column-ref>PacketsResent</column-ref>
      <column-ref>PacketsSent</column-ref>
    </params>
  </filter>
  <filter>
    <type>or</type>
    <params>
      <filter-ref>greaterRef</filter-ref>
      <filter-ref>equalsRef</filter-ref>
    </params>
  </filter>
</filters>

Example 23–16 illustrates how to define a not equals filter, where RoleRef and StringRef are defined columns.

Example 23–16  Defining a "Not Equals" Filter for a Report

<filters>
  <filter id="equals">
    <type>equals</type>
    <params>
      <column-ref>RoleRef</column-ref>
      <column-ref>StringRef</column-ref>
    </params>
  </filter>
  <filter id="Not">
    <type>not</type>
    <params>
      <filter-ref>equals</filter-ref>
    </params>
  </filter>
</filters>
Using Functions to Construct a Report

Reporter functions allow mathematical calculations to be performed on data elements within the same row of the report. The supported functions are Add, Subtract, Multiply, and Divide. Function columns can then be included as parameters into other function columns.

Function Examples

Example 23–17 illustrates how to add columns (Attribute1 and Attribute2) and place the results into a third column (Addition).

```
Example 23–17 Adding Column Values and Including Results in a Different Column

<column id="AttributeID1">
  <name>Attribute1</name>
</column>

<column id="AttributeID2">
  <name>Attribute2</name>
</column>

<column id="Addition">
  <type>function</type>
  <name>Add2Columns</name>
  <header>Adding Columns</header>
  <function-name>add</function-name>
  <params>
    <column-ref>AttributeID1</column-ref>
    <column-ref>AttributeID2</column-ref>
  </params>
</column>
```

Example 23–18 illustrates how to subtract one column value (Attribute2) from another (Attribute1) and place the results into a third column (Subtraction).

```
Example 23–18 Subtracting Column Values and Including Results in a Different Column

<column id="AttributeID1">
  <name>Attribute1</name>
</column>

<column id="AttributeID2">
  <name>Attribute2</name>
</column>

<column id="Subtraction">
  <type>function</type>
  <name>Subtract2Columns</name>
  <header>Difference</header>
  <function-name>subtract</function-name>
  <params>
    <column-ref>AttributeID1</column-ref>
    <column-ref>AttributeID2</column-ref>
  </params>
</column>
```

Example 23–19 illustrates how to multiply column values (Attribute1 and Attribute2) place the results into a third column (Multiplication).

```
Example 23–19 Multiplying Column Values and Including Results in a Different Column

<column id="AttributeID1">
  <name>Attribute1</name>
</column>

<column id="AttributeID2">
  <name>Attribute2</name>
</column>

<column id="Multiplication">
  <type>function</type>
  <name>Multiply2Columns</name>
  <header>Multiplying Columns</header>
  <function-name>multiply</function-name>
  <params>
    <column-ref>AttributeID1</column-ref>
    <column-ref>AttributeID2</column-ref>
  </params>
</column>
```
Example 23–19  Multiplying Column Values and Including Results in a Different Column

```xml
<column id="AttributeID1">
  <name>Attribute1</name>
</column>

<column id="AttributeID2">
  <name>Attribute2</name>
</column>

<column id="Multiplication">
  <type>function</type>
  <name>Multiply2Columns</name>
  <header>Multiply Columns</header>
  <function-name>multiply</function-name>
  <params>
    <column-ref>AttributeID1</column-ref>
    <column-ref>AttributeID2</column-ref>
  </params>
</column>
```

Example 23–20 illustrates how to divide one column (Attribute1) by another (Attribute2) into a third column (Division). The result of all division is a Double data type.

Example 23–20  Dividing Column Values and Including Results in a Different Column

```xml
<column id="AttributeID1">
  <name>Attribute1</name>
</column>

<column id="AttributeID2">
  <name>Attribute2</name>
</column>

<column id="Division">
  <type>function</type>
  <name>Dividing2Columns</name>
  <header>Division</header>
  <function-name>Divide</function-name>
  <params>
    <column-ref>AttributeID1</column-ref>
    <column-ref>AttributeID2</column-ref>
  </params>
</column>
```

Using Aggregates to Construct a Report

Reporter aggregates allow for multiple rows to be aggregated into a single value or row. Table 23–3 describes the available aggregate types.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg</td>
<td>Calculate the mean value for all values in the column.</td>
</tr>
<tr>
<td>max</td>
<td>Return the maximum value for all values in the column.</td>
</tr>
<tr>
<td>min</td>
<td>Return the minimum value for all values in the column.</td>
</tr>
<tr>
<td>sum</td>
<td>Add all the values from a column.</td>
</tr>
</tbody>
</table>
Aggregate Examples

Sum the values in the size column

**Example 23–21  Adding the Values in a Column**

```xml
<column id ="SumRef">
  <type>function</type>
  <function-name>sum</function-name>
  <column-ref>size</column-ref>
  <header>Sum</header>
</column>
```

Average the values in the size column

**Example 23–22  Calculating the Average of Values in a Column**

```xml
<column id ="AverageRef">
  <type>function</type>
  <header>Average</header>
  <function-name>avg</function-name>
  <column-ref>size</column-ref>
</column>
```

Find the maximum the value in the size column

**Example 23–23  Finding the Maximum Value in a Column**

```xml
<column id ="MaximumRef">
  <type>function</type>
  <header>Maximum</header>
  <function-name>max</function-name>
  <column-ref>size</column-ref>
</column>
```

Find the minimum the value in the size column

**Example 23–24  Finding the Minimum Value in a Column**

```xml
<column id ="MinimumRef">
  <type>function</type>
  <header>Minimum</header>
  <function-name>min</function-name>
  <column-ref>size</column-ref>
</column>
```

Constructing Delta Functions

Many numeric attributes in the Coherence report are cumulative. These values are reset only when the `resetStatistics` operation is executed on the MBean. To determine the state of the system without resetting the statistics, the Reporter uses a delta function. The delta function subtracts the prior value of a column from the current value of a column and returns the difference.

The prior values for a report are stored in a map on the Reporter client. This map is keyed by the 'delta key'. By default, the delta key is the MBean name for the attribute. However, when one-to-one relationship does not exist between the MBean and the rows in the report, or the MBean name is subject to change between executions of the report, the delta key will be calculated using the columns provided in the `<params>` section.
Note: Accuracy of Delta Functions: delta functions are only correct when the report is running as part of a report batch.

Delta Function Examples

Example 23–25 illustrates how to include a delta calculation of an attribute. (Assume PacketsSent is a defined column)

Example 23–25  Delta Calculation for an Attribute
<column id="DeltaPacketsSent">
  <type>function</type>
  <name>PacketsSent</name>
  <header>Delta Sent</header>
  <function-name>delta</function-name>
  <column-ref>PacketsSent</column-ref>
</column>

Example 23–26 illustrates how to include a delta calculation of an attribute with an alternate delta key. (Assume PacketsSent, NodeID and TimeStamp are defined columns)

Example 23–26  Delta Calculation for an Attribute with an Alternate Delta Key
<column id="DeltaPacketsSent">
  <type>function</type>
  <name>PacketsSent</name>
  <header>Delta Sent</header>
  <function-name>delta</function-name>
  <column-ref>PacketsSent</column-ref>
  <params>
    <column-ref>NodeID</column-ref>
    <column-ref>TimeStamp</column-ref>
  </params>
</column>
How to Modify Report Batch

Configuring a report batch is one of the steps in creating a custom report. You typically configure it after creating report configuration files. This configuration file determines what reports the reporter executes, how often the reports get executed, and where the reports are saved. If a single report can be used with different parameters, these parameters are also configured in the report batch. For more information on report configuration files, see Chapter 23, "How to Create a Custom Report".

Report Batch Deployment Descriptor

Use the report batch deployment descriptor to specify the various options for creating custom reports.

Document Location

The name and location of the descriptor defaults to `report-group.xml`. The default descriptor (packaged in `coherence.jar`) will be used unless a custom file is found in the application's classpath.

Document Root

The root element of the POF user type descriptor is `report-group`. This is where you may begin specifying the format of the custom report.

System Properties

Table 24–1 describes the system properties that can be used to control report batch from the command line.

<table>
<thead>
<tr>
<th>Property</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tangosol.coherence.management. report.configuration</td>
<td>reports/report-group.xml</td>
<td>The XML file containing the Reporter configuration settings, such as the list of reports, the report frequency, and so on.</td>
</tr>
<tr>
<td>tangosol.coherence.management. report.autostart</td>
<td>false</td>
<td>Flag to automatically start the reporter when the node is started.</td>
</tr>
<tr>
<td>tangosol.coherence.management. report.distributed</td>
<td>false</td>
<td>Determines if the reporter is running in a central model (false) or on every node in the cluster (true).</td>
</tr>
</tbody>
</table>
Document Format

The report batch descriptor should begin with the following DOCTYPE declaration:

```xml
<!DOCTYPE report-group SYSTEM "report-group.dtd">
```

Example 24–1 illustrates the nesting of elements in a report batch document.

**Example 24–1  Format of a Report Batch Configuration File (report-group.xml)**

```xml
<report-group>
  <frequency/>
  <output-directory/>
  <report-list>
    <location/>
    <report-config>
      <init-params>
        <init-param/>
      </init-params>
      <report-config/>
    </report-config>
  </report-list>
</report-group>
```
## Report Batch Element Index

Table 24–2 describes the relationship between the report batch elements.

### Table 24–2 Report Batch Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Used in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>frequency</td>
<td>report-group</td>
</tr>
<tr>
<td>location</td>
<td>report-list</td>
</tr>
<tr>
<td>init-param</td>
<td>init-params</td>
</tr>
<tr>
<td>init-params</td>
<td>report-config</td>
</tr>
<tr>
<td>output-directory</td>
<td>report-group</td>
</tr>
<tr>
<td>param-name</td>
<td>init-param</td>
</tr>
<tr>
<td>param-type</td>
<td>init-param</td>
</tr>
<tr>
<td>param-value</td>
<td>init-param</td>
</tr>
<tr>
<td>report-config</td>
<td>report-group</td>
</tr>
<tr>
<td>report-group</td>
<td>root element</td>
</tr>
<tr>
<td>report-list</td>
<td>report-group</td>
</tr>
</tbody>
</table>
**frequency**

Used in: [report-group](#)

**Description**

Required. A string containing the number of seconds, minutes between each execution of the report batch. 10s will run the report every 10 seconds. 5m will run the report every 5 minutes. Selecting an appropriate frequency is critical. If the frequency is too short, the reporter can generate a large amount of data and consume significant disk space. If the frequency is too long, the information will not be useful. It is recommended that a process for purging and archiving historical information is in place before configuring the reporter.
location

Used in: report-list

Description

Required. The path to the report configuration file. For more information on this file, see Chapter 23, "How to Create a Custom Report".
init-param

Used in: init-params

Description

The init-param element contains an initialization parameter for a report. The parameter consists of either a parameter name or type, and its value.
init-params

Used in: report-config

Description

Optional. The init-params element contains a list of initialization parameters.
output-directory

Used in: report-group

Description

Optional. The directory path to prepend to the output file names from the report configuration files. The username which the node is executing must have read write access to this path.
param-name

Used in: init-param

Description

The `param-name` element specifies the name of the initialization parameter.
param-type

Used in: init-param

Description

The param-type element specifies the Java type of the initialization parameter. Supported types are:

- string—indicates that the value is a java.lang.String
- long—indicates that the value is a java.lang.Long
- double—indicates that the value is a java.lang.Double
param-value

Used in: init-param

Description

The `param-value` element specifies a value of the initialization parameter. The value is in a format specific to the type of the parameter.
report-config

Used in: report-group

Description

The report-config contains the configuration file name and the initialization parameters for the report.
report-group

Used in: root element

Description

Describes the report list, the frequency, the report parameters, and the output directory for the batch.
report-list

Used in: report-group

Description

Required. The list of reports to include in the batch. This element contains the <report-config> subelement.
Coherence provides out of the box information that helps administrators and developers better analyze usage and configuration issues that may occur.

Network Health

The Network Health report contains the primary aggregates for determining the health of the network communications. The network health file is a tab delimited file that is prefixed with the date in YYYYMMDDHH format and post fixed with -network-health.txt. For example 2009013113-network-health.txt would be created on January 31, 2009 at 1:00 PM. Table 25–1 describes the content of the Network Health report.

<table>
<thead>
<tr>
<th>Table 25–1</th>
<th>Contents of the Network Health Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>Type</td>
</tr>
<tr>
<td>Batch Counter</td>
<td>Long</td>
</tr>
<tr>
<td>Report Time</td>
<td>Date</td>
</tr>
<tr>
<td>Min Node Rx Success</td>
<td>Double</td>
</tr>
<tr>
<td>Grid Rx Success</td>
<td>Double</td>
</tr>
<tr>
<td>Min Node Tx Success</td>
<td>Double</td>
</tr>
<tr>
<td>Grid TX Success</td>
<td>Double</td>
</tr>
</tbody>
</table>

Network Health Detail

The Network Health report supporting node level details for determining the health of the network communications. The network health detail file is a tab delimited file that is prefixed with the date in YYYYMMDDHH format and post fixed with -network-health-detail.txt. For example
Table 25–2  Contents of the Network Health Detail Report

<table>
<thead>
<tr>
<th>Column</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Counter</td>
<td>Long</td>
<td>A sequential counter to help integrate information between related files. This value does reset when the reporter restarts and is not consistent across nodes. However, it is helpful when trying to integrate files.</td>
</tr>
<tr>
<td>Report Time</td>
<td>Date</td>
<td>The system time when the report executed.</td>
</tr>
<tr>
<td>Node Id</td>
<td>Long</td>
<td>The node for the network statistics.</td>
</tr>
<tr>
<td>Tx Success</td>
<td>Double</td>
<td>The publisher success rate for the node. If this value is within 2%-3% of the &quot;Min Node Tx Success&quot; and more than 10% less than the &quot;Grid Tx Success&quot; for the batch in the Network Health File, the corresponding node may be having difficulty communicating with the cluster. Constrained CPU, constrained network bandwidth or high network latency could cause this to occur.</td>
</tr>
<tr>
<td>RX Success</td>
<td>Double</td>
<td>The receiver success rate for the node. If this value is within 2%-3% of the &quot;Min Node Rx Success&quot; and more than 10% less than the &quot;Grid Tx Success&quot; for the batch in the Network Health File, the corresponding node may be having difficulty communicating with the cluster. Constrained CPU, constrained network bandwidth or high network latency could cause this to occur.</td>
</tr>
<tr>
<td>PacketsSent</td>
<td>Double</td>
<td>The total number of network packets sent by the node.</td>
</tr>
<tr>
<td>Current Packets Sent</td>
<td>Long</td>
<td>The number of packets sent by the node since the prior execution of the report.</td>
</tr>
<tr>
<td>PacketsResent</td>
<td>Long</td>
<td>The total number of network packets resent by the node. Packets will be resent when the receiver of the packet receives and invalid packet or when an acknowledge packet is not sent within the appropriate amount of time.</td>
</tr>
<tr>
<td>Current Packet Resent</td>
<td>Long</td>
<td>The number of network packets resent by the node since the prior execution of the report.</td>
</tr>
<tr>
<td>PacketsRepeated</td>
<td>Long</td>
<td>The total number of packets received more than once.</td>
</tr>
<tr>
<td>Current Packets Received</td>
<td>Long</td>
<td>The number of packets received since the last execution of the report.</td>
</tr>
<tr>
<td>PacketsReceived</td>
<td>Long</td>
<td>The total number of packets received by the node.</td>
</tr>
<tr>
<td>Current Packets Received</td>
<td>Long</td>
<td>The total number of packets received by the node since the last execution of the report.</td>
</tr>
</tbody>
</table>

Memory Status

The Memory Status report must be run as part of a report batch. The values are helpful in understanding memory consumption on each node and across the grid. For data to be included nodes must be configured to publish platform MBean information. The memory status file is a tab delimited file that is prefixed with the date in YYYYMMDDHH format and post fixed with -memory-status.txt. For example
Cache Size

The cache size report can be executed either on demand or it can be added as part of the report batch and the Caches should have the `<unit-calculator>` subelement of `<local-scheme>` set to `BINARY`. The cache size file is a tab delimited file that is prefixed with the date in `YYYYMMDDHH` format and post fixed with `-cache-size.txt`. For example `2009013101-cache-size.txt` would be created on January 31, 2009 at 1:00 AM. Table 25–4 describes the content of the Cache Size report.

Table 25–4 Contents of the Cache Size Report

<table>
<thead>
<tr>
<th>Column</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Counter</td>
<td>Long</td>
<td>A sequential counter to help integrate information between related files. This value does reset when the reporter restarts and is not consistent across nodes. However, it is helpful when trying to integrate files.</td>
</tr>
<tr>
<td>Report Time</td>
<td>Date</td>
<td>The system time when the report executed.</td>
</tr>
<tr>
<td>Cache Name</td>
<td>String</td>
<td>The name of the cache.</td>
</tr>
</tbody>
</table>

Analyzing Reporter Content 25-3
The service report provides information to the requests processed, request failures, and request backlog, tasks processed, task failures and task backlog. Request Count and Task Count are useful to determine performance and throughput of the service. RequestPendingCount and Task Backlog are useful in determining capacity issues or blocked processes. Task Hung Count, Task Timeout Count, Thread Abandoned Count, Request Timeout Count are the number of unsuccessful executions that have occurred in the system. Table 25–5 describes the contents of the Service report.

<table>
<thead>
<tr>
<th>Column</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemoryMB</td>
<td>Double</td>
<td>The MB consumed by the objects in the cache. This does not include indexes or over head.</td>
</tr>
<tr>
<td>Avg Object Size</td>
<td>Double</td>
<td>The Average memory consumed by each object.</td>
</tr>
<tr>
<td>Cache Size</td>
<td>Double</td>
<td>The number of objects in the cache.</td>
</tr>
<tr>
<td>Memory Bytes</td>
<td>Double</td>
<td>The number of bytes consumed by the objects in the cache. This does not include indexes or over head.</td>
</tr>
</tbody>
</table>

**Service Report**

The service report provides information to the requests processed, request failures, and request backlog, tasks processed, task failures and task backlog. Request Count and Task Count are useful to determine performance and throughput of the service. RequestPendingCount and Task Backlog are useful in determining capacity issues or blocked processes. Task Hung Count, Task Timeout Count, Thread Abandoned Count, Request Timeout Count are the number of unsuccessful executions that have occurred in the system. Table 25–5 describes the contents of the Service report.

**Table 25–5 Contents of the Service Report**

<table>
<thead>
<tr>
<th>Column</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Counter</td>
<td>Long</td>
<td>A sequential counter to help integrate information between related files. This value does reset when the reporter restarts and is not consistent across nodes. However, it is helpful when trying to integrate files.</td>
</tr>
<tr>
<td>Report Time</td>
<td>Date</td>
<td>The system time when the report executed.</td>
</tr>
<tr>
<td>Service</td>
<td>String</td>
<td>The service name.</td>
</tr>
<tr>
<td>Node Id</td>
<td>String</td>
<td>The numeric node identifier.</td>
</tr>
<tr>
<td>Refresh Time</td>
<td>Date</td>
<td>The system time when the service information was updated from a remote node.</td>
</tr>
<tr>
<td>Request Count</td>
<td>Long</td>
<td>The number of requests since the last report execution.</td>
</tr>
<tr>
<td>RequestPendingCount</td>
<td>Long</td>
<td>The number of pending requests at the time of the report.</td>
</tr>
<tr>
<td>RequestPendingDuration</td>
<td>Long</td>
<td>The duration for the pending requests at the time of the report.</td>
</tr>
<tr>
<td>Request Timeout Count</td>
<td>Long</td>
<td>The number of request timeouts since the last report execution.</td>
</tr>
<tr>
<td>Task Count</td>
<td>Long</td>
<td>The number of tasks executed since the last report execution.</td>
</tr>
<tr>
<td>Task Backlog</td>
<td>Long</td>
<td>The task backlog at the time of the report execution.</td>
</tr>
<tr>
<td>Task Timeout Count</td>
<td>Long</td>
<td>The number of task timeouts since the last report execution.</td>
</tr>
<tr>
<td>Task Hung Count</td>
<td>Long</td>
<td>The number of tasks that hung since the last report execution.</td>
</tr>
<tr>
<td>Thread Abandoned Count</td>
<td>Long</td>
<td>The number of threads abandoned since the last report execution.</td>
</tr>
</tbody>
</table>
**Node List**

Due to the transient nature of the node identifier (nodeId), the reporter logs out a list of nodes and the user defined `<member-identity>` information. The node list file is a tab delimited file that is prefixed with the date in YYYYMMDDHH format and post fixed with -nodes.txt. For example 2009013101-nodes.txt would be created on January 31, 2009 at 1:00 AM. Table 25–6 describes the content of the Node List report.

<table>
<thead>
<tr>
<th>Column</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Counter</td>
<td>Long</td>
<td>A sequential counter to help integrate information between related files. This value does reset when the reporter restarts and is not consistent across nodes. However, it is helpful when trying to integrate files.</td>
</tr>
<tr>
<td>Report Time</td>
<td>Date</td>
<td>The system time when the report executed.</td>
</tr>
<tr>
<td>Node Id</td>
<td>String</td>
<td>The numeric node identifier.</td>
</tr>
<tr>
<td>UnicastAddress</td>
<td>String</td>
<td>The Unicast address for the node.</td>
</tr>
<tr>
<td>MemberName</td>
<td>String</td>
<td>The member name for the node.</td>
</tr>
<tr>
<td>ProcessName</td>
<td>String</td>
<td>The process name for the node.</td>
</tr>
<tr>
<td>RoleName</td>
<td>String</td>
<td>The role name for the node.</td>
</tr>
<tr>
<td>MachineName</td>
<td>String</td>
<td>The machine name for the node.</td>
</tr>
<tr>
<td>RackName</td>
<td>String</td>
<td>The rack name for the node.</td>
</tr>
<tr>
<td>SiteName</td>
<td>String</td>
<td>The site name for the node.</td>
</tr>
<tr>
<td>Refresh Time</td>
<td>Date/Time</td>
<td>The time which the information was refreshed from a remote node. If the time is not the same as the refresh time on other rows in the batch, the node did not respond in a timely matter. This is often caused by a node preforming a garbage collection. Any information regarding a node with an &quot;old&quot; refresh date is questionable.</td>
</tr>
</tbody>
</table>

**Proxy Report**

The proxy file provides information about proxy servers and the information being transferred to clients. The Proxy file is a tab delimited file that is prefixed with the date in YYYYMMDDHH format and post fixed with -report-proxy.txt. For example 2009013101-report-proxy.txt would be created on January 31, 2009 at 1:00 AM. Table 25–7 describes the content of the Proxy report.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Counter</td>
<td>Long</td>
<td>A sequential counter to help integrate information between related files. This value does reset when the reporter restarts and is not consistent across nodes. However, it is helpful when trying to integrate files.</td>
</tr>
<tr>
<td>Report Time</td>
<td>Date</td>
<td>The system time when the report executed.</td>
</tr>
<tr>
<td>Node Id</td>
<td>String</td>
<td>The numeric node identifier.</td>
</tr>
<tr>
<td>Service Name</td>
<td>String</td>
<td>The name of the proxy service.</td>
</tr>
<tr>
<td>HostIp</td>
<td>String</td>
<td>The IP Address and Port of the proxy service.</td>
</tr>
</tbody>
</table>
### Table 25–7  (Cont.) Contents of the Proxy Report

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionCount</td>
<td>Long</td>
<td>The current number of connections to the proxy service.</td>
</tr>
<tr>
<td>OutgoingByteBacklog</td>
<td>Long</td>
<td>The number of bytes queued to be sent by the proxy service.</td>
</tr>
<tr>
<td>OutgoingMessageBacklog</td>
<td>Long</td>
<td>The number of messages queued by the proxy service.</td>
</tr>
<tr>
<td>Bytes Sent</td>
<td>Long</td>
<td>The number of bytes sent by the proxy service since the last execution of the report.</td>
</tr>
<tr>
<td>Bytes Received</td>
<td>Long</td>
<td>The number of bytes received by the proxy service since the last execution of the report.</td>
</tr>
<tr>
<td>Messages Sent</td>
<td>Long</td>
<td>The number of messages sent by the proxy service since the last execution of the report.</td>
</tr>
<tr>
<td>Messages Received</td>
<td>Long</td>
<td>The number of messages received by the proxy service since the last execution of the report.</td>
</tr>
</tbody>
</table>
A report can be run on demand by using either JConsole or the JMX HTTP Adapter. The Reporter MBean operations contain a `runReport(String sReportPath)` method. The report path can either be a resource in `coherence.jar` or a file URL.

Figure 26–1 illustrates the reporter operations in JConsole.
How to Run ReportControl MBean at Node Startup

When set to true, the `tangosol.coherence.management.report.autostart` system property allows the ReportControl MBean to start execution when the node is started. This property must be used with the `tangosol.coherence.management.report.group` system property and the configuration of the custom MBean XML file.

In Example 26–1, the `tangosol.coherence.management.report.autostart` system property is set to true.

Example 26–1  `tangosol.coherence.management.report.autostart` System Property
- `Dtangosol.coherence.management.report.autostart=true`

How to Configure the ReportControl MBean

The report group system property, `tangosol.coherence.management.report.group` configures the ReportControl MBean with the specified configuration file. This property must be used in correlation with the `tangosol.coherence.management.report.autostart` property and the configuration of the custom MBean XML file.

In Example 26–2, the `tangosol.coherence.management.report.group` property points to the custom MBean XML file `report-batch.xml`.

Example 26–2  `tangosol.coherence.management.report.group` System Property
- `Dtangosol.coherence.management.report.group=./report-batch.xml`
Creating an MBean XML Configuration File

Custom MBeans are configured in an XML configuration file. The elements in the file describe the MBean type, MBean implementation, and the target MBean `ObjectName`. The current release of Coherence supports these types of custom MBeans:

- Standard MBeans
- MXBeans
- JMX MBeans

See Appendix K, "MBean Configuration Elements" for a complete descriptions of the elements used in this chapter.

Configuring Standard MBeans

The configuration in Example 27–1 will create a `Coherence:type=Query,nodeId=<nodeId>` using the standard MBean `com.oracle.customMBeans.Query` class for the node. This example specifies an MBean class (`mbean-class`), an MBean name (`mbean-name`), and whether it is registered (`enabled`) in the instance.

**Example 27–1 Using an MBean to Create a Query Node**

```xml
<mbeans>
  <mbean id="100">
    <mbean-class>com.oracle.customMBeans.Query</mbean-class>
    <mbean-name>type=Query</mbean-name>
    <enabled>true</enabled>
  </mbean>
</mbeans>
```

Configuring MXBeans

The configuration in Example 27–2 will execute the standard Java method `getMemoryMXBean` in the `java.lang.management.ManagementFactory` class and use the result to create a `Coherence:type=java,SubSystem=Memory,nodeId=<nodeId>` for the node. The example specifies an MBean factory (`mbean-factory`), an accessor method name on the factory (`mbean-accessor`), an MBean name (`mbean-name`), and whether it is registered (`enabled`) in the instance.

This chapter provides information on configuring standard, MX, and JMX MBeans.
Example 27–2  Getting an MBean for the Memory System of a Java Virtual Machine

```xml
<mbeans>
  <mbean id="2">
    <mbean-factory>java.lang.management.ManagementFactory</mbean-factory>
    <mbean-accessor>getMemoryMXBean</mbean-accessor>
    <mbean-name>type=java,SubSystem=Memory</mbean-name>
    <enabled>true</enabled>
  </mbean>
</mbeans>
```

Configuring JMX MBeans

JMX MBeans are MBeans that exist in a local MBean server that need to be added to the Coherence Management structure. This allows consolidation of MBeanServer information into a single source. The configuration in Example 27–3 executes the JMX query, `java.lang:*`, on the node's local MBean server and uses the results to create corresponding MBeans on the centralized Coherence MBean server. The example specifies a JMX MBean query (`mbean-query`), an MBean name (`mbean-name`), and whether it is registered (`enabled`) in the instance.

Example 27–3  Executing a JMX Query and Creating an MBean on the MBean Server

```xml
<mbeans>
  <mbean id="1">
    <mbean-query>java.lang:*</mbean-query>
    <mbean-name>type=Platform</mbean-name>
    <enabled>true</enabled>
  </mbean>
</mbeans>
```

Figure 27–1 illustrates the results on the query in JConsole.
Enabling a Custom MBean Configuration File

You can enable the custom MBean configuration file by setting a system property or by including a specially named file in the class path.

Setting a System Property

Coherence provides the following system property to specify the name and location of a custom MBean configuration file. Setting this system property will cause the Coherence node to load the MBeans defined in the file represented by filename.

Example 27–4 System Property to Load an MBean

-Dtangosol.coherence.mbeans=<filename>

Adding a Custom MBean Configuration File to the Class Path

By convention, Coherence recognizes the configuration file named custom-mbeans.xml as containing a custom MBean configuration. If you name your custom MBean configuration file custom-mbeans.xml and include it in the class path, then the Coherence node will load the configured MBeans.
In addition to managing Coherence with JMX, Coherence provides the ability to manage and monitor "custom MBeans" (that is, application-level MBeans) within the Coherence JMX Management and Monitoring framework. This enables you to manage or monitor any application-level MBean from any JVM, node, or end-point within the cluster.

In addition to the standard Coherence managed object types, any dynamic or standard MBean type may be registered using the `com.tangosol.net.management.Registry` interface.

### Custom MBean Configuration

Coherence 3.4 can be configured to load platform and standard MBeans on connection to the cluster. This allows administrators and support personnel to update and view system and application information from all nodes in a cluster from a single location. This feature also eliminates the need for JMX programs to connect to multiple sources to gather information.

### How to Add a Standard MBean to Coherence

The following instructions describe how to add a standard MBean to Coherence:

1. Create a standard MBean.
2. Add a standard MBean Class or JAR to the Coherence classpath (including central management node).
3. Create a custom MBean XML configuration file (see "Creating an MBean XML Configuration File" on page 27-1).
4. Modify node startup scripts to reference `custom-mbean.xml` (see "Enabling a Custom MBean Configuration File" on page 27-3).

### How to Programatically Add a Standard MBean to Coherence

Example 28–1 illustrates sample code that programmatically adds a standard MBean to Coherence.

```java
Registry registry = CacheFactory.ensureCluster().getManagement();
Custom bean = new Custom();
String sName = registry.ensureGlobalName("type=Custom");
```
registry.register(sName, bean);

**How to Add the Results of a JMX Query to Coherence**

The following instructions describe how to add the results of a JMX query to Coherence.

1. Create a custom MBean XML file (see "Creating an MBean XML Configuration File" on page 27-1).
2. Configure node startup script to include JMX MBean Server
3. Configure a node startup script to reference custom-mbean.xml (see "Enabling a Custom MBean Configuration File" on page 27-3).

Figure 28–1 illustrates an example of running a JMX Query in JConsole.

**Figure 28–1 JMX Query Run in JConsole**

![JMX Query Run in JConsole](image)

This figure is described in the text.

*****************************************************************************
Coherence tends to be so simple to use in development that developers do not take the necessary planning steps and precautions when moving an application using Coherence into production. This article is intended to accomplish the following:

■ Create a healthy appreciation for the complexities of deploying production software, particularly large-scale infrastructure software and enterprise applications;
■ Enumerate areas that require planning when deploying Coherence;
■ Define why production awareness should exist for each of those areas;
■ Suggest or require specific approaches and solutions for each of those areas; and
■ Provide a check-list to minimize risk when deploying to production.

Deployment recommendations are available for:

■ Network
■ Hardware
■ Operating System
■ JVM
■ Java Security Manager
■ Application Instrumentation
■ Coherence Editions and Modes
■ Coherence Operational Configuration
■ Coherence Cache Configuration
■ Large Cluster Configuration
■ Death Detection
During development, a Coherence-enabled application on a developer’s local machine can accidentally form a cluster with the application running on other developers’ machines.

Developers often use and test Coherence locally on their workstations. There are several ways in which they may accomplish this, including:

- Setting the multicast TTL to zero,
- Using a “loopback”, or
- By each developer using a different multi-cast address and port from all other developers.

If one of these approaches is not used, then multiple developers on the same network will find that Coherence has clustered across different developers’ locally running instances of the application; in fact, this happens relatively often and causes confusion when it is not understood by the developers.

Setting the TTL to zero on the command line is very simple: Add the following to the JVM startup parameters:

-Dtangosol.coherence.ttl=0

Starting with Coherence version 3.2, setting the TTL to zero for all developers is also very simple. Edit the tangosol-coherence-override-dev.xml in the coherence.jar file, changing the TTL setting as follows:

<time-to-live system-property="tangosol.coherence.ttl">0</time-to-live>

On some UNIX operating systems, including some versions of Linux and Mac OS X, setting the TTL to zero may not be enough to isolate a cluster to a single machine. To be safe, assign a different cluster name for each developer, for example using the developer’s email address as the cluster name. If the cluster communication does go across the network to other developer machines, then the different cluster name will cause an error on the node that is attempting to start up.

To ensure that the clusters are completely isolated, select a different multicast IP address and port for each developer. In some organizations, a simple approach is to use the developer’s phone extension number as part of the multicast address and as the port number (or some part of it). For information on configuring the multicast IP address and port, see "multicast-listener” on page H-29.

During development, clustered functionality is often not being tested.

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

Make sure that the application is being tested in a clustered configuration as development proceeds. 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 machine). This works well with the TTL=0 setting, since clustering on a single machine 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.

**What is the type and speed of the production network?**

Most production networks are based on gigabit Ethernet, with a few still built on slower 100Mb Ethernet or faster ten-gigabit Ethernet. It is important to understand the topology of the production network, and what the full set of devices that will connect all of the servers that will be running 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 more than one switch 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 will result 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 will have full “mesh” bandwidth up to a limit of four servers on each switch (e.g all four servers on one switch can communicate at full speed with the four servers on the other switch). However, adding additional servers will 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 will be 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 actually needs to 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 can and do 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.

Oracle strongly suggests GigE or faster: Gigabit Ethernet is supported by most servers built since 2004, and Gigabit switches are economical, available and widely deployed.

Before deploying an application, you must run the Datagram Test to test the actual network speed and determine its capability for pushing large amounts of data. Furthermore, the Datagram test 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, such as occurs with the default configuration of Cisco 6500 series switches. See "Deploying to Cisco Switches" on page M-2 for more information.
**Will the production deployment use 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. Oracle suggests the use of multicast when possible because it is an efficient option for many servers to communicate. However, there are several common reasons why multicast cannot be used:

- 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.

First determine if multicast will be used. In other words, determine if the desired deployment configuration is to use multicast.

Before deploying an application that will use multicast, you must run the Multicast Test to verify that multicast is working and to determine the correct (the minimum) TTL value for the production environment. See Chapter 15, "Performing a Multicast Connectivity Test" for more information.

Applications that cannot use multicast for deployment must use the WKA configuration. See "well-known-addresses" on page H-55 and "Network Protocols" for more information.

**Are your network devices configured optimally?**

If the above datagram and/or multicast tests have failed or returned poor results, it is possible that there are configuration problems with the network devices in use. Even if the tests passed without incident and the results were perfect, it is still possible that there are lurking issues with the configuration of the network devices.

Review the suggestions in "Network Tuning" on page 19-4.

**How will the cluster handle a sustained network outage?**

The Coherence cluster protocol is capable of detecting and handling 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" on page A-14 for more details. See also:

- "Deploying to Cisco Switches" on page M-2
- "Deploying to Foundry Switches" on page M-5

**Hardware**

During development, developers can form unrealistic performance expectations. Most developers have relatively fast workstations. Combined with test cases that are typically non-clustered and tend to represent single-user access (that is, only the developer), the application may seem extraordinarily responsive.

Include as a requirement that realistic load tests be built that can be run with simulated concurrent user load.

Test routinely in a clustered configuration with simulated concurrent user load.
During development, developer productivity can be adversely affected by inadequate hardware resources, and certain types of quality can also be affected negatively.

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. Desktop systems and workstations can often be configured with 4GB for minimal additional cost.

Developer systems should have two CPU cores or more. Although this will have the likely side-effect of making developers happier, the actual purpose is to increase the quality of code related to multi-threading, since many bugs related to concurrent execution of multiple threads will only show up on multi-CPU systems (systems that contain multiple processor sockets and/or CPU cores).

What are the supported and suggested server hardware platforms for deploying Coherence on?

The short answer is that 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 Sun Sparc (including Niagra) 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.
- Oracle conducts internal Coherence certification on all IBM server platforms at least once a year.
- Oracle and Azul test Coherence regularly on Azul appliances, including the newly-announced 48-core "Vega 2" chip.

If the server hardware purchase is still in the future, the following are suggested for Coherence (as of December 2006):

The most cost-effective server hardware platform is "commodity x86", either Intel or AMD, with one to two processor sockets and two to four CPU cores per processor socket. If selecting an AMD Opteron system, it is strongly recommended that it be a two processor socket system, since memory capacity is usually halved in a single socket system. Intel "Woodcrest" and "Clovertown" Xeons are strongly recommended over the previous Intel Xeon CPUs due to significantly improved 64-bit support, much lower power consumption, much lower heat emission and far better performance. These new Xeons are currently the fastest commodity x86 CPUs, and can support a large memory capacity per server regardless of the processor socket count by using fully buffered memory called "FB-DIMMs".

It is strongly recommended that servers be configured with a minimum of 4GB of RAM. For applications that plan to store massive amounts of data in memory - tens or hundreds of gigabytes, or more - it is recommended to evaluate the cost-effectiveness of 16GB or even 32GB of RAM per server. As of December, 2006, commodity x86 server RAM is readily available in a density of 2GB per DIMM, with higher densities available from only a few vendors and carrying a large price premium; this means that a server with 8 memory slots will only support 16GB in a cost-effective manner. Also
note that a server with a very large amount of RAM will likely need to run more Coherence nodes (JVMs) per server to use that much memory, so having a larger number of CPU cores will help. Applications that are "data heavy" will require a higher ratio of RAM to CPU, while applications that are "processing heavy" will require a lower ratio. For example, it may be sufficient to have two dual-core Xeon CPUs in a 32GB server running 15 Coherence "Cache Server" nodes performing mostly identity-based operations (cache accesses and updates), but if an application makes frequent use of Coherence features such as indexing, parallel queries, entry processors and parallel aggregation, then it will be more effective to have two quad-core Xeon CPUs in a 16GB server - a 4:1 increase in the CPU:RAM 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. In the case of PCI-X having the NIC on an isolated or otherwise lightly loaded 133MHz bus may significantly improve performance.

How many servers are optimal?
Coherence is primarily a scale-out technology. While Coherence can effectively scale-up on large servers by using multiple JVMs per server, the natural mode of operation is to span several small servers (for example, 2-socket or 4-socket commodity servers). Specifically, failover and failback are more efficient in larger configurations. And the impact of a server failure is lessened. As a rule of thumb, a cluster should contain at least four physical servers. In most WAN configurations, each data center will have independent clusters (usually interconnected by Extend-TCP). This will increase the total number of discrete servers (four servers per data center, multiplied by the number of data centers).

Coherence is quite 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. As discussed in the network section of this document, 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.

Also note that 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. Please note that disabling UDP multicast (by using WKA) or running on slower networks (for example, 100Mbps Ethernet) will reduce network efficiency and make scaling more difficult.

Does it matter how JVMs are distributed among servers?
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.

1. 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 is not the same as the number of JVMs on the other server, so even if we start with two servers with equal number of JVMs, losing one JVM will force the grid out of machine-safe state. Four or more machines present the most stable topology, but deploying on just three servers would work if the other rules are adhered to.

2. 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.
3. 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.

4. The margin of safety improves as the number of JVMs tends toward equality on all machines in the cluster; this is more of a "rule of thumb" than the preceding "hard" rules.

See also:
- "Deploying to IBM BladeCenters" on page M-5
- "Deploying to Virtual Machines" on page M-9

**Operating System**

During development, developers typically use a different operating system than the one that the application will be deployed to.

The top three operating systems for application development using Coherence are, in this order: Windows 2000/XP (~85%), Mac OS X (~10%) and Linux (~5%). The top four operating systems for production deployment are, in this order: Linux, Solaris, AIX and Windows. Thus, it is relatively unlikely that the development and deployment operating system will be the same.

Make sure that regular testing is occurring on the target operating system.

**What are the supported and suggested server operating systems for deploying Coherence on?**

Oracle tests on and supports various Linux distributions (including customers that have custom Linux builds), Sun Solaris, IBM AIX, Windows Vista/2003/2000/XP, Apple Mac OS X, OS/400 and z/OS. Additionally, Oracle supports customers running HP-UX and various BSD UNIX distributions.

If the server operating system decision is still in the future, the following are suggested for Coherence (as of December 2006):

For commodity x86 servers, Linux distributions based on the Linux 2.6 kernel are recommended. While it is expected that most 2.6-based Linux distributions will provide a good environment for running Coherence, the following are recommended by Oracle: RedHat Enterprise Linux (version 4 or later) and Suse Linux Enterprise (version 10 or later). Oracle also routinely tests using distributions such as RedHat Fedora Core 5 and even Knoppix "Live CD".

Review and follow the instructions in Appendix M, "Platform-Specific Deployment Considerations" for the operating system that Coherence will be deployed on.

**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. With modern commodity hardware and a modern JVM, a Java process with a reasonable heap size (512MB-2GB) will typically perform a full garbage collection in a few seconds if all of the process memory is in RAM. However, this may grow to many minutes if the JVM is partially resident on disk. During garbage collection, the node will appear 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 mark the unresponsive node as "failed" and perform failover processing. Neither of these is a good option, and so it is important to avoid excessive pauses due to garbage collection. JVMs should be pinned into physical RAM, or at least configured so that the JVM will not be paged to disk.

Note that periodic processes (such as daily backup programs) may cause memory usage spikes that could cause Coherence JVMs to be paged to disk.

See also:
- "Deploying to AIX" on page M-1
- "Deploying to Linux" on page M-6
- "Deploying to OS X" on page M-7
- "Deploying to Solaris" on page M-8
- "Deploying to Windows" on page M-10
- "Deploying to z OS" on page M-10

JVM

During development, developers typically use the latest Sun 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 GC configuration and approaches to GC tuning;
- Differing behaviors in thread scheduling, garbage collection behavior and performance, and the performance of running code.

Make sure that regular testing is occurring on the JVM that will be used in production.

**Which JVM configuration options should be used?**

JVM configuration options vary over versions and between vendors, but the following are generally suggested:
- Using the `-server` option will result in substantially better performance.
- Using identical heap size values for both `-Xms` and `-Xmx` will yield substantially better performance on Sun and JRockit JVMs, and "fail fast" memory allocation. See the specific Deployment Considerations for various JVMs below.
- For naive tuning, a heap size of 1GB is a good compromise that balances per-JVM overhead and garbage collection performance.
  - Larger heap sizes are allowed and commonly used, but may require tuning to keep garbage collection pauses manageable.
- 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. See the specific Deployment Considerations below for instructions on configuring this on common JVMs.
What are the supported and suggested JVMs for deploying Coherence on?

In terms of Oracle Coherence versions:

- Coherence 3.x versions are supported on the Sun JDK versions 1.4 and 1.5, and JVMs corresponding to those versions of the Sun JDK. Starting with Coherence 3.3 the 1.6 JVMs are also supported.

- Coherence version 2.x (currently at the 2.5.1 release level) is supported on the Sun JDK versions 1.2, 1.3, 1.4 and 1.5, and JVMs corresponding to those versions of the Sun JDK.

Often the choice of JVM is 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 Sun Solaris/Sparc, IBM builds a JVM using the Sun JVM source code instead of its own.

- Oracle WebLogic typically includes a JVM which is intended to be used with it. On some platforms, this is the Oracle WebLogic JRockit JVM.

- 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, the Sun JVM is recommended. Generally speaking, the recent update versions are recommended. For example:

---

**Note:** Oracle recommends testing and deploying using the latest supported Sun JVM based on your platform and Coherence version. It has been observed that running Coherence on 1.5 and later JVMs exhibits significant performance improvements as compared to running on older JVMs.

---

Basically, at some point 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 Appendix M, "Platform-Specific Deployment Considerations" for the JVM that Coherence will be deployed on.

Must all nodes run the same JVM vendor and version?

No. 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 and/or version) 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 JRockit JVMs" on page M-2
Java Security Manager

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.

Application Instrumentation

Be cautious when using instrumented management and monitoring solutions. Some Java-based management and monitoring solutions use instrumentation (for example, bytecode-manipulation and ClassLoader substitution). While there are no known open issues with the latest versions of the primary vendors, Oracle has observed issues in the past.

Coherence Editions and Modes

During development, use the development mode. The Coherence download includes a fully functional Coherence product supporting all editions and modes. The default configuration is for Grid Edition in Development mode.

Coherence may be configured to operate in either 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.

It is recommended to use the development mode for all pre-production activities, such as development and testing. This is an important safety feature, because Coherence automatically prevents these nodes from joining a production cluster. The production mode must be explicitly specified when using Coherence in a production environment.

Coherence may be configured to support a limited feature set, based on the customer license agreement.

Only the edition and the number of licensed CPUs specified within the customer license agreement can be used in a production environment.

When operating outside of the production environment it is allowable to run any Coherence edition. However, it is recommended that only the edition specified within the customer license agreement be used. This will protect the application from unknowingly making use of unlicensed features.

All nodes within a cluster must use the same license edition and mode.

Starting with Oracle Coherence 3.4, customer-specific license keys are no longer part of product deployment.

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.

**Example A–1  Verifying Hardware Configuration**

```java
java -cp tangosol.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 to the "support" email address at Oracle.

**How are the edition and mode configured?**

There is a `<license-config>` configuration section in `tangosol-coherence.xml` (located in `coherence.jar`) for edition and mode related information.

**Example A–2  Sample Coherence License Configuration**

```xml
<license-config>
  <edition-name system-property="tangosol.coherence.edition">GE</edition-name>
  <license-mode system-property="tangosol.coherence.mode">dev</license-mode>
</license-config>
```

In addition to preventing mixed mode clustering, the `license-mode` also dictates the operational override file which will be used. When in `dev` mode the `tangosol-coherence-override-dev.xml` file will be used, whereas the `tangosol-coherence-override-prod.xml` file will be used when the `prod` mode is specified. As the mode controls which override file is used, the `<license-mode>` configuration element is only usable in the base `tangosol-coherence.xml` file and not within the override files.

These elements are defined by the corresponding `coherence.dtd` in `coherence.jar`. It is possible to specify this edition on the command line using the command line override:

```bash
-Dtangosol.coherence.edition=RTC
```

Valid values are listed in Table A–1:

<table>
<thead>
<tr>
<th>Value</th>
<th>Coherence Edition</th>
<th>Compatible Editions</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>Grid Edition</td>
<td>RTC, DC</td>
</tr>
<tr>
<td>EE</td>
<td>Enterprise Edition</td>
<td>DC</td>
</tr>
<tr>
<td>SE</td>
<td>Standard Edition</td>
<td>DC</td>
</tr>
<tr>
<td>RTC</td>
<td>Real-Time Client</td>
<td>GE</td>
</tr>
<tr>
<td>DC</td>
<td>Data Client</td>
<td>GE, EE, SE</td>
</tr>
</tbody>
</table>

Note: clusters running different editions may connect by using Coherence*Extend as a Data Client.

For more information on overrides, see Appendix L, "Command Line Overrides".

**Ensuring that RTC nodes don't use Coherence TCMP**

The RTC nodes can connect to clusters using either Coherence TCMP or Coherence Extend. If the intention is to connect over Extend it is advisable to disable TCMP on
that node to ensure that it only connects by using Extend. TCMP may be disabled using the system property `tangosol.coherence.tcmp.enabled`. See the `<enabled>` subelement of "packet-publisher" on page H-39.

---

**Coherence Operational Configuration**

Operational configuration relates to the configuration of Coherence at the cluster level including such things as:

- Cluster and member descriptors
- Network settings
- Security
  - Membership restrictions
  - Access Control
  - Encryption

The operational aspects are normally configured by using the `tangosol-coherence-override.xml` file. See "Operational Configuration Deployment Descriptors" on page H-1 for more information on this file.

The contents of this file will likely differ 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 not be the responsibility of the application developers, instead it should fall under the jurisdiction of the systems administrators who are far more familiar with the workings of the production systems.

All cluster nodes should use the same operational configuration descriptor. A centralized configuration file may be maintained and accessed by specifying the file's location as a URL using the tangosol.coherence.override system property. Any node specific values may be specified by using system properties. See Appendix L, "Command Line Overrides" for more information on the properties.

The override file should contain only the subset of configuration elements which you want to customize. This will not only make your configuration more readable, but will allow you to take advantage of updated defaults in future Coherence releases. All override elements should be copied exactly from the original tangosol-coherence.xml, including the id attribute of the element.

Member descriptors may be used to provide detailed identity information that is useful for defining the location and role of the cluster member. Specifying these items will aid in the management of large clusters by making it easier to identify the role of a remote nodes if issues arise.

---

**Coherence Cache Configuration**

Cache configuration relates to the configuration of Coherence at a per-cache level including such things as:

- Cache topology (`<distributed-scheme>`, `<replicated-scheme>`, `<near-scheme>`, and so on)
- Cache capacities (see `<high-units>` subelement of `<local-scheme>`)
The cache configuration aspects are normally configured by using the `coherence-cache-config.xml` file. See "Cache Configuration Deployment Descriptor" on page D-1 for more information this file.

The default `coherence-cache-config.xml` file included within `coherence.jar` is intended only as an example and is not suitable for production use. It is suggested that you produce your own cache configuration file with definitions tailored to your application needs.

All cluster nodes should use the same cache configuration descriptor. A centralized configuration file may be maintained and accessed by specifying the file's location as a URL using the `tangosol.coherence.cacheconfig` system property.

Choose the cache topology which is most appropriate for each cache's usage scenario.

It is important to size limit your caches based on the allocated JVM heap size. Even if you never expect to fully load the cache, having the limits in place will help protect your application from `OutOfMemoryExceptions` if your expectations are later negated.

For a 1GB heap that at most ¾ of the heap be allocated for cache storage. With the default one level of data redundancy this implies a per server cache limit of 375MB for primary data, and 375MB for backup data. The amount of memory allocated to cache storage should fit within the tenured heap space for the JVM. See Sun's GC tuning guide for details.

It is important to note that when multiple cache schemes are defined for the same cache service name the first to be loaded will dictate 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.

For multiple caches which use the same cache service it is recommended that the service related elements be defined only once, and that they be inherited by the various cache-schemes which will use them.

If you want different values for these items on a cache by cache basis then multiple services may be configured.

For partitioned caches Coherence will evenly distribute 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 machines with additional resources multiple cache servers may be used to effectively make use of the machine's resources.

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

For caches which are backed by a cache store it is recommended that the parent service be configured with a thread pool as requests to the cache store may block on I/O. The pool is enabled by using the `<thread-count>` subelement of `<distributed-scheme>` element. For non-CacheStore-based caches more threads are unlikely to improve performance and should left disabled.

Unless explicitly specified all cluster nodes will be storage enabled, that is, will act as cache servers. It is important to control which nodes in your production environment will be storage enabled and storage disabled. The `tangosol.coherence.distributed.localstorage` system property may be used to control this, setting it to either `true` or `false`. Generally, only dedicated cache
servers, all other cluster nodes should be configured as storage disabled. This is especially important for short lived processes which may join the cluster perform some work, and exit the cluster, having these nodes as storage enabled will introduce unneeded re-partitioning. See the <local-storage> subelement of <distributed-scheme> for more information about the system property.

Large Cluster Configuration

Are there special considerations for large clusters?

- The general recommendation for the <partition-count> subelement of <distributed-scheme> is to be a prime number close to the square of the number of storage enabled nodes. While is a good suggestion for small to medium sized clusters, for large clusters it can add too much overhead. For clusters exceeding 128 storage enabled JVMs, the partition count should be fixed, at roughly 16,381.

- Coherence clusters which consist of over 400 TCMP nodes need to increase the default maximum packet size Coherence will use. 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%. The maximum packet size is configured as part of the coherence operational configuration file, see "packet-size" on page H-41 for details on changing this setting.

- For large clusters which have hundreds of JVMs it is also recommended that <multicast-listener> be enabled, as it will allow for more efficient cluster wide transmissions. These cluster wide transmissions are rare, but when they do occur multicast can provide noticeable benefits in large clusters.

Death Detection

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 will consult 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 will assume 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 will restart it's clustered services and attempt 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 when connectivity is restored the previously isolated cluster nodes will rejoin the running cluster. 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 had already taken on ownership of that data (restoring from backups).

Without connectivity it is obviously not possible for a node to identify the state of other nodes. This means that from the point of view of a single node, local network adapter failure and network wide switch failure look identical, and are thus handled in the same way, as described above. The important difference is that in the case of 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.
Obviously dropping all data is not desirable, and thus if you want to avoid this as part of a sustained switch failure you must take additional precautions. Options include:

- Extend allowable outage duration: The maximum time a node(s) may be unresponsive before being removed from the cluster is configured by using the `<timeout-milliseconds>` subelement of `<packet-delivery>`, and defaults to one minute for production configurations. Increasing this value will allow the cluster to wait longer for connectivity to return. The downside of increasing this value it may also take longer to handle the case where just a single node has lost connectivity.

- 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 in the event of a switch failure. 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.

- Delay node restart: The cluster death detection action can be reconfigured to delay the node restart until connectivity is restored. By delaying the restart until connectivity is restored an isolated node is allowed to continue running with whatever data it had available at the time of disconnect. When connectivity is restored the nodes will detect each other and form a new cluster. In forming a new cluster all but the most senior node will be required to restart. This results in behavior which is nearly identical to the default behavior because the majority of the nodes will restart, and drop their data. It may be beneficial for cases in which replicated caches are in use as the senior most node's copy of the data will survive the restart. To enable the delayed restart the `tangosol.coherence.departure.threshold` system property must be set to a value that is greater then the size of the cluster.

---

**Note:** When running on Microsoft Windows it is also necessary to ensure the Windows does not disable the network adapter when it is disconnected. To do this, add the following Windows registry DWORD, setting it to `1:HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\DisableDHCPMediaSense`. 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 machine, 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, see Linux bonding, Solaris trunking and Windows teaming for further details.

---

**tangosol-license.xml Deprecated**

As of Coherence 3.4, the `tangosol-license.xml` file is no longer used.
This appendix provides an overview and comparison of the types of caches offered by Coherence.

**Distributed Cache**

A distributed, or partitioned, cache is a clustered, fault-tolerant cache that has linear scalability. Data is partitioned among all the machines of the cluster. For fault-tolerance, partitioned caches can be configured to keep each piece of data on one or more unique machines within a cluster. Distributed caches are the most commonly used caches in Coherence.

**Replicated Cache**

A replicated cache is a clustered, fault tolerant cache where data is fully replicated to every member in the cluster. This cache offers the fastest read performance with linear performance scalability for reads but poor scalability for writes (as writes must be processed by every member in the cluster). Because data is replicated to all machines, adding servers does not increase aggregate cache capacity.

**Optimistic Cache**

An optimistic cache is a clustered cache implementation similar to the replicated cache implementation but without any concurrency control. This implementation offers higher write throughput than a replicated cache. It also allows an alternative underlying store for the cached data (for example, a MRU/MFU-based cache). However, if two cluster members are independently pruning or purging the underlying local stores, it is possible that a cluster member may have a different store content than that held by another cluster member.

**Near Cache**

A near cache is a hybrid cache; it typically fronts a distributed cache or a remote cache with a local cache. Near cache invalidates front cache entries, using configurable invalidation strategy, and provides excellent performance and synchronization. Near cache backed by a partitioned cache offers zero-millisecond local access for repeat data access, while enabling concurrency and ensuring coherency and fail-over, effectively combining the best attributes of replicated and partitioned caches.
### Local Cache

A local cache is a cache that is local to (completely contained within) a particular cluster node. While it is not a clustered service, the Coherence local cache implementation is often used in combination with various clustered cache services.

### Remote Cache

A remote cache describes any out of process cache accessed by a Coherence*Extend client. All cache requests are sent to a Coherence proxy where they are delegated to one of the other Coherence cache types (Replicated, Optimistic, Partitioned).

### Summary of Cache Types

#### Numerical Terms:
- **JVMs** = number of JVMs
- **DataSize** = total size of cached data (measured without redundancy)
- **Redundancy** = number of copies of data maintained
- **LocalCache** = size of local cache (for near caches)

#### Table B–1 Summary of Cache Types and Characteristics

<table>
<thead>
<tr>
<th>Topology</th>
<th>Replicated Cache</th>
<th>Optimistic Cache</th>
<th>Partitioned Cache</th>
<th>Near Cache backed by partitioned cache</th>
<th>LocalCache not clustered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replicated</td>
<td>Replicated</td>
<td>Replicated</td>
<td>Partitioned Cache</td>
<td>Local Caches + Partitioned Cache</td>
<td>Local Cache</td>
</tr>
<tr>
<td>Read Performance</td>
<td>Instant 5</td>
<td>Instant 5</td>
<td>Locally cached: instant 5</td>
<td>Locally cached: instant 5</td>
<td>Instant 5</td>
</tr>
<tr>
<td>Performance</td>
<td>Instant 5</td>
<td>Instant 5</td>
<td>Remote: network speed 1</td>
<td>Remote: network speed 1</td>
<td></td>
</tr>
<tr>
<td>Fault Tolerance</td>
<td>Extremely High</td>
<td>Extremely High</td>
<td>Configurable 4</td>
<td>Configurable 4</td>
<td>Zero</td>
</tr>
<tr>
<td>Write Performance</td>
<td>Fast 2</td>
<td>Fast 2</td>
<td>Extremely fast 3</td>
<td>Extremely fast 3</td>
<td>Instant 5</td>
</tr>
<tr>
<td>Memory Usage (Per JVM)</td>
<td>DataSize</td>
<td>DataSize</td>
<td>DataSize/JVMs x Redundancy</td>
<td>LocalCache + [DataSize / JVMs]</td>
<td>DataSize</td>
</tr>
<tr>
<td>Coherency</td>
<td>fully coherent</td>
<td>fully coherent</td>
<td>fully coherent</td>
<td>fully coherent 6</td>
<td>n/a</td>
</tr>
<tr>
<td>Memory Usage (Total)</td>
<td>JVMs x DataSize</td>
<td>JVMs x DataSize</td>
<td>Redundancy x DataSize</td>
<td>[Redundancy x DataSize] + [JVMs x LocalCache]</td>
<td>n/a</td>
</tr>
<tr>
<td>Locking</td>
<td>fully transactional</td>
<td>none</td>
<td>fully transactional</td>
<td>fully transactional</td>
<td>fully transactional</td>
</tr>
<tr>
<td>Typical Uses</td>
<td>Metadata</td>
<td>n/a (see Near Cache)</td>
<td>Read-write caches</td>
<td>Read-heavy caches w/ access affinity</td>
<td>Local data</td>
</tr>
</tbody>
</table>
Notes:

1. As a rough estimate, with 100mbit Ethernet, network reads typically require ~20ms for a 100KB object. With gigabit Ethernet, network reads for 1KB objects are typically sub-millisecond.

2. Requires UDP multicast or a few UDP unicast operations, depending on JVM count.

3. Requires a few UDP unicast operations, depending on level of redundancy.

4. Partitioned caches can be configured with as many levels of backup as desired, or zero if desired. Most installations use one backup copy (two copies total).

5. Limited by local CPU/memory performance, with negligible processing required (typically sub-millisecond performance).

6. Listener-based Near caches are coherent; expiry-based near caches are partially coherent for non-transactional reads and coherent for transactional access.
Summary of Cache Types
Use Coherence caches to cache value objects. These objects may represent data from any source, either internal (such as session data, transient data, and so on) or external (such as a database, mainframe, and so on).

Objects placed in the cache must be serializable. Because serialization is often the most expensive part of clustered data management, Coherence provides the following options for serializing/deserializing data:

- com.tangosol.io.pof.PofSerializer – The Portable Object Format (also referred to as POF) is a language agnostic binary format. POF was designed to be incredibly efficient in both space and time and has become the recommended serialization option in Coherence. See "The Portable Object Format" in the Oracle Coherence Getting Started Guide.

- java.io.Serializable – The simplest, but slowest option.

- java.io.Externalizable – This requires developers to implement serialization manually, but can provide significant performance benefits. Compared to java.io.Serializable, this can cut serialized data size by a factor of two or more (especially helpful with Distributed caches, as they generally cache data in serialized form). Most importantly, CPU usage is dramatically reduced.

- com.tangosol.io.ExternalizableLite – This is very similar to java.io.Externalizable, but offers better performance and less memory usage by using a more efficient I/O stream implementation.

- com.tangosol.run.xml.XmlBean – A default implementation of ExternalizableLite (For more details, see the API Javadoc for XmlBean).

**Note:** Remember, when serializing an object, Java serialization automatically crawls every visible object (by using object references, including collections like Map and List). As a result, cached objects should not refer to their parent objects directly (holding onto an identifying value like an integer is OK). Objects that implement their own serialization routines are not affected.
This appendix provides a listing of the elements that can be used in a cache configuration. In addition, it describes the deployment descriptor file in which they appear.

**Cache Configuration Deployment Descriptor**

Use the cache configuration deployment descriptor to specify the various types of caches which can be used within a cluster. For information on configuring cluster communication and services see Appendix H, "Operational Configuration Elements."

**Document Location**

The name and location of the descriptor is specified in the operational deployment descriptor and defaults to `coherence-cache-config.xml`. The default configuration descriptor (packaged in `coherence.jar`) will be used unless a custom one is found within the application's classpath. It is recommended that all nodes within a cluster use identical cache configuration descriptors.

**Document Root**

The root element of the configuration descriptor is `<cache-config>`. This is where you may begin configuring your caches.

**Document Format**

The Cache Configuration descriptor should begin with the following DOCTYPE declaration:

```
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">
```

---

**Note:** When deploying Coherence into environments where the default character set is EBCDIC rather than ASCII, make sure that this descriptor file is in ASCII format and is deployed into its runtime environment in the binary format.

---

**Command Line Override**

Oracle Coherence provides a powerful command line override feature which allows any element defined in this descriptor to be overridden from the Java command line if
it has a system-property attribute defined in the descriptor. For more information on this feature, see Appendix L, "Command Line Overrides".

Examples

See Appendix F, "Sample Cache Configurations" for usage examples.
The following table lists all non-terminal elements which may be used from within a cache configuration.

<table>
<thead>
<tr>
<th>Element</th>
<th>Used In:</th>
</tr>
</thead>
<tbody>
<tr>
<td>acceptor-config</td>
<td>proxy-scheme</td>
</tr>
<tr>
<td>async-store-manager</td>
<td>external-scheme, paged-external-scheme</td>
</tr>
<tr>
<td>authorized-hosts</td>
<td>tcp-acceptor</td>
</tr>
<tr>
<td>backing-map-scheme</td>
<td>distributed-scheme, optimistic-scheme, replicated-scheme</td>
</tr>
<tr>
<td>backup-storage</td>
<td>distributed-scheme</td>
</tr>
<tr>
<td>bdb-store-manager</td>
<td>external-scheme, paged-external-scheme, async-store-manager</td>
</tr>
<tr>
<td>cache-config</td>
<td>root element</td>
</tr>
<tr>
<td>cache-mapping</td>
<td>caching-scheme-mapping</td>
</tr>
<tr>
<td>cache-service-proxy</td>
<td>proxy-config</td>
</tr>
<tr>
<td>caching-scheme-mapping</td>
<td>cache-config</td>
</tr>
<tr>
<td>caching-schemes</td>
<td>cache-config</td>
</tr>
<tr>
<td>class-scheme</td>
<td>caching-schemes, local-scheme, distributed-scheme, replicated-scheme, optimistic-scheme, near-scheme, overflow-scheme, read-write-backing-map-scheme, cachestore-scheme, listener</td>
</tr>
<tr>
<td>cachestore-scheme</td>
<td>local-scheme, read-write-backing-map-scheme</td>
</tr>
<tr>
<td>custom-store-manager</td>
<td>external-scheme, paged-external-scheme, async-store-manager</td>
</tr>
<tr>
<td>disk-scheme</td>
<td>caching-schemes</td>
</tr>
<tr>
<td>distributed-scheme</td>
<td>caching-schemes, near-scheme, overflow-scheme</td>
</tr>
<tr>
<td>external-scheme</td>
<td>caching-schemes, distributed-scheme, replicated-scheme, optimistic-scheme, near-scheme, overflow-scheme, read-write-backing-map-scheme</td>
</tr>
<tr>
<td>init-param</td>
<td>init-params</td>
</tr>
<tr>
<td>init-params</td>
<td>class-scheme</td>
</tr>
<tr>
<td>initiator-config</td>
<td>remote-cache-scheme, remote-invocation-scheme</td>
</tr>
<tr>
<td>invocation-scheme</td>
<td>caching-schemes</td>
</tr>
<tr>
<td>jms-acceptor</td>
<td>acceptor-config</td>
</tr>
<tr>
<td>jms-initiator</td>
<td>initiator-config</td>
</tr>
<tr>
<td>key-associator</td>
<td>distributed-scheme</td>
</tr>
<tr>
<td>key-partitioning</td>
<td>distributed-scheme</td>
</tr>
<tr>
<td>lh-file-manager</td>
<td>external-scheme, paged-external-scheme, async-store-manager</td>
</tr>
</tbody>
</table>
### Table D–1 (Cont.) Cache Configuration Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Used In:</th>
</tr>
</thead>
<tbody>
<tr>
<td>listener</td>
<td>disk-scheme, local-scheme, external-scheme, paged-external-scheme, paged-external-scheme, distributed-scheme, replicated-scheme, optimistic-scheme, near-scheme, overflow-scheme, read-write-backing-map-scheme</td>
</tr>
<tr>
<td>local-scheme</td>
<td>caching-schemes, distributed-scheme, replicated-scheme, optimistic-scheme, near-scheme, overflow-scheme, read-write-backing-map-scheme</td>
</tr>
<tr>
<td>near-scheme</td>
<td>caching-schemes</td>
</tr>
<tr>
<td>nio-file-manager</td>
<td>external-scheme, paged-external-scheme, async-store-manager</td>
</tr>
<tr>
<td>nio-memory-manager</td>
<td>external-scheme, paged-external-scheme, async-store-manager</td>
</tr>
<tr>
<td>operation-bundling</td>
<td>cachestore-scheme, distributed-scheme, remote-cache-scheme</td>
</tr>
<tr>
<td>optimistic-scheme</td>
<td>caching-schemes, near-scheme, overflow-scheme</td>
</tr>
<tr>
<td>outgoing-message-handler</td>
<td>acceptor-config, initiator-config</td>
</tr>
<tr>
<td>overflow-scheme</td>
<td>caching-schemes, distributed-scheme, replicated-scheme, optimistic-scheme, read-write-backing-map-scheme</td>
</tr>
<tr>
<td>paged-external-scheme</td>
<td>caching-schemes, distributed-scheme, replicated-scheme, optimistic-scheme, near-scheme, overflow-scheme, read-write-backing-map-scheme</td>
</tr>
<tr>
<td>partitioned</td>
<td>backing-map-scheme</td>
</tr>
<tr>
<td>proxy-config</td>
<td>proxy-scheme</td>
</tr>
<tr>
<td>proxy-scheme</td>
<td>caching-schemes</td>
</tr>
<tr>
<td>read-write-backing-map-scheme</td>
<td>caching-schemes, distributed-scheme, replicated-scheme, optimistic-scheme</td>
</tr>
<tr>
<td>remote-cache-scheme</td>
<td>cachestore-scheme, caching-schemes, near-scheme</td>
</tr>
<tr>
<td>remote-invocation-scheme</td>
<td>caching-schemes</td>
</tr>
<tr>
<td>replicated-scheme</td>
<td>caching-schemes, near-scheme, overflow-scheme</td>
</tr>
<tr>
<td>tcp-acceptor</td>
<td>acceptor-config</td>
</tr>
<tr>
<td>tcp-initiator</td>
<td>initiator-config</td>
</tr>
</tbody>
</table>
acceptor-config

Used in: proxy-scheme

Description

The acceptor-config element specifies the configuration information for a protocol-specific connection acceptor. The connection acceptor is used by a proxy service to enable Coherence*Extend clients to connect to the cluster and use the services offered by the cluster without having to join the cluster.

The acceptor-config element must contain exactly one protocol-specific connection acceptor configuration element (either jms-acceptor or tcp-acceptor).

Elements

Table D–2 describes the elements you can define within the acceptor-config element.

Table D–2 acceptor-config Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;connection-limit&gt;</td>
<td>Optional</td>
<td>The maximum number of simultaneous connections allowed by this connection acceptor. Valid values are positive integers and zero. A value of zero implies no limit. Default value is zero.</td>
</tr>
<tr>
<td>&lt;jms-acceptor&gt;</td>
<td>Optional</td>
<td>Specifies the configuration info for a connection acceptor that enables Coherence*Extend clients to connect to the cluster over JMS.</td>
</tr>
<tr>
<td>&lt;outgoing-message-handler&gt;</td>
<td>Optional</td>
<td>Specifies the configuration info used by the connection acceptor to detect dropped client-to-cluster connections.</td>
</tr>
</tbody>
</table>
Table D–2  (Cont.) acceptor-config Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| <serializer>    | Optional          | Specifies the class configuration info for a com.tangosol.io.Serializer implementation used by the connection acceptor to serialize and deserialize user types. For example, the following configures a ConfigurablePofContext that uses the my-pof-types.xml POF type configuration file to deserialize user types to and from a POF stream:  
  <serializer>  
    <class-name>com.tangosol.io.pof.ConfigurablePofContext</class-name>  
    <init-params>  
      <init-param>  
        <param-type>string</param-type>  
        <param-value>my-pof-types.xml</param-value>  
      </init-param>  
    </init-params>  
  </serializer> |
| <tcp-acceptor>  | Optional          | Specifies the configuration info for a connection acceptor that enables Coherence Extend clients to connect to the cluster over TCP/IP.          |
| <use-filters>   | Optional          | Contains the list of <filters> names to be used by this connection acceptor. For example, specifying use-filter as follows will activate gzip compression for all network messages, which can help substantially with WAN and low-bandwidth networks.  
  <use-filters>  
    <filter-name>gzip</filter-name>  
  </use-filters> |
**address-provider**

*Used in:* **tcp-initiator**

**Description**

Contains the configuration info for an address factory that implements the `com.tangosol.net.AddressProvider` interface.

**Elements**

*Table D–3* describes the subelements you can define within the `address-provider` element.

**Table D–3  address-provider Subelements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;class-factory-name&gt;</code></td>
<td>Optional</td>
<td>Specifies a fully specified name of a Java class that will be used as a factory for object instantiation.</td>
</tr>
<tr>
<td><code>&lt;class-name&gt;</code></td>
<td>Required</td>
<td>The name of a class that implements the <code>com.tangosol.net.AddressProvider</code> interface.</td>
</tr>
<tr>
<td><code>&lt;init-params&gt;</code></td>
<td>Optional</td>
<td>Specifies initialization parameters which are accessible by implementations which support the <code>com.tangosol.run.xml.XmlConfigurable</code> interface, or which include a public constructor with a matching signature.</td>
</tr>
<tr>
<td><code>&lt;method-name&gt;</code></td>
<td>Optional</td>
<td>Specifies the name of a static factory method on the factory class which will perform object instantiation.</td>
</tr>
</tbody>
</table>
async-store-manager

Used in: external-scheme, paged-external-scheme.

Description

The async-store-manager element adds asynchronous write capabilities to other store manager implementations. Supported store managers include:

- **custom-store-manager**—allows definition of custom implementations of store managers
- **bdb-store-manager**—uses Berkeley Database JE to implement an on disk cache
- **lh-file-manager**—uses a Coherence LH on disk database cache
- **nio-file-manager**—uses NIO to implement memory-mapped file based cache
- **nio-memory-manager**—uses NIO to implement an off JVM heap, in-memory cache

Implementation

This store manager is implemented by the com.tangosol.io.AsyncBinaryStoreManager class.

Elements

Table D–4 describes the subelements you can define within the async-store-manager element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| <async-limit>    | Optional          | Specifies the maximum number of bytes that will be queued to be written asynchronously. Setting the value to zero does not disable the asynchronous writes; instead, it indicates that the implementation default for the maximum number of bytes are necessary. The value of this element must be in the following format: `[\d]+[.][\d]+?[K|M][B]`? where the first non-digit (from left to right) indicates the factor with which the preceding decimal value should be multiplied:  
  - K (kilo, 210)  
  - M (mega, 220)  
  If the value does not contain a factor, a factor of one is assumed. Valid values are any positive memory sizes and zero. Default value is 4MB. |
<p>| &lt;bdb-store-manager&gt; | Optional          | Configures the external cache to use Berkeley Database JE on disk databases for cache storage. |
| &lt;class-name&gt;     | Optional          | Specifies a custom implementation of the async-store-manager. Any custom implementation must extend the com.tangosol.io.AsyncBinaryStoreManager class and declare the exact same set of public constructors. |
| &lt;custom-store-manager&gt; | Optional          | Configures the external cache to use a custom storage manager implementation. |</p>
<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;init-params&gt;</code></td>
<td>Optional</td>
<td>Specifies initialization parameters, for use in custom async-store-manager implementations which implement the com.tangosol.run.xml.XmlConfigurable interface.</td>
</tr>
<tr>
<td><code>&lt;lh-file-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the external cache to use a Coherence LH on disk database for cache storage.</td>
</tr>
<tr>
<td><code>&lt;nio-file-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the external cache to use a memory-mapped file for cache storage.</td>
</tr>
<tr>
<td><code>&lt;nio-memory-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the external cache to use an off JVM heap, memory region for cache storage.</td>
</tr>
</tbody>
</table>
backing-map-scheme

Used in: distributed-scheme, optimistic-scheme, replicated-scheme

Description

Specifies what type of cache will be used within the cache server to store the entries.

When using an overflow-based backing map it is important that the corresponding backup-storage be configured for overflow (potentially using the same scheme as the backing-map). See "Partitioned Cache with Overflow" on page F-6 for an example configuration.

Note: The partitioned subelement is used if and only if the parent element is the distributed-scheme.

Elements

Table D–5 describes the subelements you can define within the backing-map-scheme element:

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/ Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-scheme&gt;</td>
<td>Optional</td>
<td>Class schemes provide a mechanism for instantiating an arbitrary Java object for use by other schemes. The scheme which contains this element will dictate what class or interface(s) must be extended.</td>
</tr>
<tr>
<td>&lt;external-scheme&gt;</td>
<td>Optional</td>
<td>External schemes define caches which are not JVM heap based, allowing for greater storage capacity.</td>
</tr>
<tr>
<td>&lt;local-scheme&gt;</td>
<td>Optional</td>
<td>Local cache schemes define in-memory &quot;local&quot; caches. Local caches are generally nested within other cache schemes, for instance as the front-tier of a near scheme.</td>
</tr>
<tr>
<td>&lt;paged-external-scheme&gt;</td>
<td>Optional</td>
<td>As with external-scheme, paged-external-schemes define caches which are not JVM heap based, allowing for greater storage capacity.</td>
</tr>
<tr>
<td>&lt;partitioned&gt;</td>
<td>Optional</td>
<td>The overflow-scheme defines a two-tier cache consisting of a fast, size limited front-tier, and slower but much higher capacity back-tier cache.</td>
</tr>
<tr>
<td>&lt;overflow-scheme&gt;</td>
<td>Optional</td>
<td>The read-write-backing-map-scheme defines a backing map which provides a size limited cache of a persistent store.</td>
</tr>
<tr>
<td>&lt;read-write-backing-map-scheme&gt;</td>
<td>Optional</td>
<td>The versioned-backing-map-scheme is an extension of a read-write-backing-map-scheme, defining a size limited cache of a persistent store. It uses object versioning to determine what updates need to be written to the persistent store.</td>
</tr>
</tbody>
</table>
backup-storage

Used in: distributed-scheme.

Description

The backup-storage element specifies the type and configuration of backup storage for a partitioned cache.

Elements

The following table describes the elements you can define within the backup-storage element.

<table>
<thead>
<tr>
<th>Table D–6 backup-storage Subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>&lt;class-name&gt;</td>
</tr>
<tr>
<td>&lt;directory&gt;</td>
</tr>
</tbody>
</table>
| <initial-size> | Optional | Only applicable with the off-heap and file-mapped types. Specifies the initial buffer size in bytes. The value of this element must be in the following format: 

```
ex*{d}+[.|{d}]?([K|k]|M|m|G|g)|B|b]?
```

where the first non-digit (from left to right) indicates the factor with which the preceding decimal value should be multiplied:

- K or k (kilo, 2^10)
- M or m (mega, 2^20)
- G or g (giga, 2^30)

If the value does not contain a factor, a factor of mega is assumed. Legal values are positive integers between 1 and Integer.MAX_VALUE - 1023 (that is, 2,147,482,624 bytes). Default value is the backup-storage/initial-size value specified in the tangosol-coherence.xml descriptor. See "DistributedCache Service Parameters" on page I-3 for more information. |
Table D–6  (Cont.) backup-storage Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| <maximum-size>  | Optional          | Only applicable with the off-heap and file-mapped types. Specifies the initial buffer size in bytes. The value of this element must be in the following format:

\[d+\{.\}d]\?K|k|M|m|G|g]\?B|b]?

where the first non-digit (from left to right) indicates the factor with which the preceding decimal value should be multiplied:

- K or k (kilo, 210)
- M or m (mega, 220)
- G or g (giga, 230)

If the value does not contain a factor, a factor of mega is assumed. Legal values are positive integers between 1 and Integer.MAX_VALUE - 1023 (that is, 2,147,482,624 bytes). Default value is the backup-storage/maximum-size value specified in the tangosol-coherence.xml descriptor. See “DistributedCache Service Parameters” on page I-3 for more information. |
| <scheme-name>   | Optional          | Only applicable with the scheme type. Specifies a scheme name for the ConfigurableCacheFactory. Default value is the backup-storage/scheme-name value specified in the tangosol-coherence.xml descriptor. See “DistributedCache Service Parameters” on page I-3 for more information. |
| <type>          | Required          | Specifies the type of the storage used to hold the backup data. Legal values are:

- on-heap—The corresponding implementations class is java.util.HashMap.
- off-heap—The corresponding implementations class is com.tangosol.io.nio.BinaryMap using the com.tangosol.io.nio.DirectBufferManager.
- file-mapped—The corresponding implementations class is com.tangosol.io.nio.BinaryMap using the com.tangosol.io.nio.MappedBufferManager.
- custom—The corresponding implementations class is the class specified by the class-name element.
- scheme—The corresponding implementations class is specified as a caching-scheme by the scheme-name element.

Default value is the value specified in the tangosol-coherence.xml descriptor. For more information, see the <backup-storage/type> parameter in “DistributedCache Service Parameters” on page I-3. |
**bdb-store-manager**


---

**Note:** Berkeley Database JE Java class libraries are required to use a `bdb-store-manager`, see the Berkeley Database JE product page for additional information.

---

**Description**

The BDB store manager is used to define external caches which will use Berkeley Database JE on disk embedded databases for storage. See the examples of Berkeley-based store configurations in "Persistent Cache on Disk" on page F-3 and "In-memory Cache with Disk Based Overflow" on page F-4.

**Implementation**

This store manager is implemented by the `com.tangosol.io.bdb.BerkeleyDBBinaryStoreManager` class, and produces `BinaryStore` objects implemented by the `com.tangosol.io.bdb.BerkeleyDBBinaryStore` class.

**Elements**

Table D–7 describes the elements you can define within the `bdb-store-manager` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;class-name&gt;</code></td>
<td>Optional</td>
<td>Specifies a custom implementation of the Berkeley Database BinaryStoreManager. Any custom implementation must extend the <code>com.tangosol.io.bdb.BerkeleyDBBinaryStoreManager</code> class and declare the exact same set of public constructors.</td>
</tr>
<tr>
<td><code>&lt;directory&gt;</code></td>
<td>Optional</td>
<td>Specifies the path name to the root directory where the Berkeley Database JE store manager will store files. If not specified or specified with a non-existent directory, a temporary directory in the default location will be used.</td>
</tr>
<tr>
<td><code>&lt;init-params&gt;</code></td>
<td>Optional</td>
<td>Specifies additional Berkeley DB configuration settings. See Berkeley DB Configuration. Also used to specify initialization parameters, for use in custom implementations which implement the <code>com.tangosol.run.xml.XmlConfigurable</code> interface.</td>
</tr>
<tr>
<td><code>&lt;store-name&gt;</code></td>
<td>Optional</td>
<td>Specifies the name for a database table that the Berkeley Database JE store manager will use to store data in. Specifying this parameter will cause the <code>bdb-store-manager</code> to use non-temporary (persistent) database instances. This is intended only for local caches that are backed by a cache loader from a non-temporary store, so that the local cache can be pre-populated from the disk on startup. When specified, it is recommended that it use the <code>{cache-name}</code> macro. Normally this parameter should be left unspecified, indicating that temporary storage is to be used. See Appendix E, &quot;Cache Configuration Parameter Macros&quot; for more information on the <code>{cache-name}</code> macro.</td>
</tr>
</tbody>
</table>
bundle-config

Used in: operation-bundling.

Description

The bundle-config element specifies the bundling strategy configuration for one or more bundle-able operations.

Elements

Table D–8 describes the subelements you can define within the bundle-config element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;auto-adjust&gt;</td>
<td>Optional</td>
<td>Specifies whether the auto adjustment of the preferred-size value (based on the run-time statistics) is allowed. Valid values are true or false. Default value is false.</td>
</tr>
<tr>
<td>&lt;delay-millis&gt;</td>
<td>Optional</td>
<td>Specifies the maximum amount of time in milliseconds that individual execution requests are allowed to be deferred for a purpose of “bundling” them together and passing into a corresponding bulk operation. If the preferred-size threshold is reached before the specified delay, the bundle is processed immediately. Valid values are positive numbers. Default value is 1.</td>
</tr>
</tbody>
</table>
| <operation-name>| Required          | Specifies the operation name for which calls performed concurrently on multiple threads will be “bundled” into a functionally analogous “bulk” operation that takes a collection of arguments instead of a single one. Valid values depend on the bundle configuration context. For the <cachestore-scheme> the valid operations are:  
  - load"  
  - store  
  - erase  
  For the <distributed-scheme> and <remote-cache-scheme> the valid operations are:  
  - get  
  - put  
  - remove  
  In all cases there is a pseudo operation named all, referring to all valid operations. Default value is all. |
| <preferred-size>| Optional          | Specifies the bundle size threshold. When a bundle size reaches this value, the corresponding “bulk” operation will be invoked immediately. This value is measured in context-specific units. Valid values are zero (disabled bundling) or positive values. Default value is zero. |
| <thread-threshold>| Optional          | Specifies the minimum number of threads that must be concurrently executing individual (non-bundled) requests for the bundler to switch from a pass-through to a bundling mode. Valid values are positive numbers. Default value is 4. |
cache-config

Root Element

Description

The cache-config element is the root element of the cache configuration descriptor, coherence-cache-config.xml. For more information on this document, see "Cache Configuration Deployment Descriptor" on page D-1.

At a high level, a cache configuration consists of cache schemes and cache scheme mappings. Cache schemes describe a type of cache, for instance a database backed, distributed cache. Cache mappings define what scheme to use for a given cache name.

Elements

Table D–9 describes the subelements you can define within the cache-config element.

<table>
<thead>
<tr>
<th>Table D–9  cache-config Subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
</tr>
<tr>
<td>&lt;caching-scheme-mapping&gt;</td>
</tr>
<tr>
<td>&lt;caching-schemes&gt;</td>
</tr>
</tbody>
</table>
cache-mapping

Used in: caching-scheme-mapping

Description

Each cache-mapping element specifies the caching-schemes which are to be used for a given cache name or pattern.

Elements

Table D–10 describes the subelements you can define within the cache-mapping element.

Table D–10 cache-mapping Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| <cache-name>  | Required          | Specifies a cache name or name pattern. The name is unique within a cache factory. The following cache name patterns are supported:  
|               |                   | ■ exact match, for example, MyCache  
|               |                   | ■ prefix match, for example, My* that matches to any cache name starting with My  
|               |                   | ■ any match "*", that matches to any cache name  
|               |                   | The patterns get matched in the order of specificity (more specific definition is selected whenever possible). For example, if both MyCache and My* mappings are specified, the scheme from the MyCache mapping will be used to configure a cache named MyCache. |
| <scheme-name> | Required          | Contains the caching scheme name. The name is unique within a configuration file. Caching schemes are configured in the caching-schemes element. |
| <init-params> | Optional          | Allows specifying replaceable cache scheme parameters. During cache scheme parsing, any occurrence of any replaceable parameter in format param-name is replaced with the corresponding parameter value. Consider the following cache mapping example:  
|               |                   | <cache-mapping>  
|               |                   |     <cache-name>My</cache-name>  
|               |                   |     <scheme-name>my-scheme</scheme-name>  
|               |                   |     <init-params>  
|               |                   |         <init-param>  
|               |                   |         <param-name>cache-loader</param-name>  
|               |                   |         <param-value>com.acme.MyCacheLoader</param-value>  
|               |                   |         </init-param>  
|               |                   |         <init-param>  
|               |                   |         <param-name>size-limit</param-name>  
|               |                   |         <param-value>1000</param-value>  
|               |                   |         </init-param>  
|               |                   |     </init-params>  
|               |                   | </cache-mapping>  
|               |                   | For any cache name match My*, any occurrence of the literal cache-loader in any part of the corresponding cache-scheme element will be replaced with the string com.acme.MyCacheLoader and any occurrence of the literal size-limit will be replaced with the value of 1000. Since Coherence 3.0 |
cache-service-proxy

Used in: proxy-config

Description

The cache-service-proxy element contains the configuration info for a cache service proxy managed by a proxy service.

Elements

Table D–11 describes the elements you can define within the cache-service-proxy element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;enabled&gt;</td>
<td>Optional</td>
<td>Specifies whether the cache service proxy is enabled. If disabled, clients will not be able to access any proxied caches. Legal values are true or false. Default value is true.</td>
</tr>
<tr>
<td>&lt;lock-enabled&gt;</td>
<td>Optional</td>
<td>Specifies whether lock requests from remote clients are permitted on a proxied cache. Legal values are true or false. Default value is false.</td>
</tr>
<tr>
<td>&lt;read-only&gt;</td>
<td>Optional</td>
<td>Specifies whether requests from remote clients that update a cache are prohibited on a proxied cache. Legal values are true or false. Default value is false.</td>
</tr>
</tbody>
</table>
**cachestore-scheme**

Used in: local-scheme, read-write-backing-map-scheme, versioned-backing-map-scheme.

**Description**

Cache store schemes define a mechanism for connecting a cache to a back-end data store. The cache store scheme may use any class implementing either the `com.tangosol.net.cache.CacheStore` or `com.tangosol.net.cache.CacheLoader` interfaces, where the former offers read-write capabilities, where the latter is read-only. Custom implementations of these interfaces may be produced to connect Coherence to various data stores. See "Cache of a Database" on page F-4 for an example of using a cachestore-scheme.

**Elements**

Table D–12 describes the elements you can define within the cachestore-scheme element.

<table>
<thead>
<tr>
<th>Table D–12 cachestore-scheme Subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>&lt;scheme-name&gt;</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
</tr>
<tr>
<td>&lt;class-scheme&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>&lt;remote-cache-scheme&gt;</td>
</tr>
<tr>
<td>&lt;operation-bundling&gt;</td>
</tr>
</tbody>
</table>
caching-scheme-mapping

Used in: cache-config

Description

Defines mappings between cache names, or name patterns, and caching-schemes. For instance you may define that caches whose names start with accounts- will use a distributed (distributed-scheme) caching scheme, while caches starting with the name rates- will use a replicated-scheme caching scheme.

Elements

Table D–13 describes the subelement you can define within the caching-scheme-mapping element.

Table D–13 caching-scheme-mapping Subelement

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;cache-mapping&gt;</td>
<td>Optional</td>
<td>Contains a single binding between a cache name and the caching scheme this cache will use.</td>
</tr>
</tbody>
</table>
caching-schemes

Used in: cache-config

Description
The `caching-schemes` element defines a series of cache scheme elements. Each cache scheme defines a type of cache, for instance a database backed partitioned cache, or a local cache with an LRU eviction policy. Scheme types are bound to actual caches using `caching-scheme-mappings`.

Scheme Types and Names
Each of the cache scheme element types is used to describe a different type of cache, for instance distributed, versus replicated. Multiple instances of the same type may be defined so long as each has a unique scheme-name.

Example D–1 illustrates the configuration of two different distributed schemes

**Example D–1 Configuring Two Different Distributed Schemes**

```xml
<distributed-scheme>
  <scheme-name>DistributedInMemoryCache</scheme-name>
  <service-name>DistributedCache</service-name>
  <backing-map-scheme>
    <local-scheme/>
  </backing-map-scheme>
</distributed-scheme>

<distributed-scheme>
  <scheme-name>DistributedOnDiskCache</scheme-name>
  <service-name>DistributedCache</service-name>
  <backing-map-scheme>
    <external-scheme>
      <nio-file-manager>
        <initial-size>8MB</initial-size>
        <maximum-size>512MB</maximum-size>
        <directory></directory>
      </nio-file-manager>
    </external-scheme>
  </backing-map-scheme>
</distributed-scheme>
```

Nested Schemes
Some caching scheme types contain nested scheme definitions. For instance in the above example the distributed schemes include a nested scheme definition describing their backing map.

Scheme Inheritance
Caching schemes can be defined by specifying all the elements required for a given scheme type, or by inheriting from another named scheme of the same type, and selectively overriding specific values. Scheme inheritance is accomplished by including a `<scheme-ref>` element in the inheriting scheme containing the `scheme-name` of the scheme to inherit from.
For example, the two configurations in Example D-2 will produce equivalent 
DistributedInMemoryCache scheme definitions:

### Example D-2 Configuring Equivalent Scheme Definitions

```xml
<distributed-scheme>
  <scheme-name>DistributedInMemoryCache</scheme-name>
  <service-name>DistributedCache</service-name>
  <backing-map-scheme>
    <local-scheme>
      <eviction-policy>LRU</eviction-policy>
      <high-units>1000</high-units>
      <expiry-delay>1h</expiry-delay>
    </local-scheme>
  </backing-map-scheme>
</distributed-scheme>

<distributed-scheme>
  <scheme-name>DistributedInMemoryCache</scheme-name>
  <service-name>DistributedCache</service-name>
  <backing-map-scheme>
    <local-scheme>
      <scheme-ref>LocalSizeLimited</scheme-ref>
    </local-scheme>
  </backing-map-scheme>
</distributed-scheme>

<local-scheme>
  <scheme-name>LocalSizeLimited</scheme-name>
  <eviction-policy>LRU</eviction-policy>
  <high-units>1000</high-units>
  <expiry-delay>1h</expiry-delay>
</local-scheme>
```

Note that while the first is somewhat more compact, the second offers the ability to 
easily reuse the LocalSizeLimited scheme within multiple schemes. Example D-3 
demonstrates multiple schemes reusing the same LocalSizeLimited base 
definition, but the second imposes a different expiry-delay.

### Example D-3 Multiple Schemes Reusing the Same Base Definition

```xml
<distributed-scheme>
  <scheme-name>DistributedInMemoryCache</scheme-name>
  <service-name>DistributedCache</service-name>
  <backing-map-scheme>
    <local-scheme>
      <scheme-ref>LocalSizeLimited</scheme-ref>
    </local-scheme>
  </backing-map-scheme>
</distributed-scheme>

<replicated-scheme>
  <scheme-name>ReplicatedInMemoryCache</scheme-name>
  <service-name>ReplicatedCache</service-name>
  <backing-map-scheme>
    <local-scheme>
      <scheme-ref>LocalSizeLimited</scheme-ref>
      <expiry-delay>10m</expiry-delay>
    </local-scheme>
  </backing-map-scheme>
</replicated-scheme>
```
Table D–14 caching-schemes Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;local-scheme&gt;</td>
<td>Optional</td>
<td>Defines a cache scheme which provides on-heap cache storage.</td>
</tr>
<tr>
<td>&lt;external-scheme&gt;</td>
<td>Optional</td>
<td>Defines a cache scheme which provides off-heap cache storage, for instance on disk.</td>
</tr>
<tr>
<td>&lt;paged-external-scheme&gt;</td>
<td>Optional</td>
<td>Defines a cache scheme which provides off-heap cache storage, that is size-limited by using time based paging.</td>
</tr>
<tr>
<td>&lt;distributed-scheme&gt;</td>
<td>Optional</td>
<td>Defines a cache scheme where storage of cache entries is partitioned across the cluster nodes.</td>
</tr>
<tr>
<td>&lt;replicated-scheme&gt;</td>
<td>Optional</td>
<td>Defines a cache scheme where each cache entry is stored on all cluster nodes.</td>
</tr>
<tr>
<td>&lt;optimistic-scheme&gt;</td>
<td>Optional</td>
<td>Defines a replicated cache scheme which uses optimistic rather then pessimistic locking.</td>
</tr>
<tr>
<td>&lt;near-scheme&gt;</td>
<td>Optional</td>
<td>Defines a two tier cache scheme which consists of a fast local front-tier cache of a much larger back-tier cache.</td>
</tr>
<tr>
<td>&lt;versioned-near-scheme&gt;</td>
<td>Optional</td>
<td>Defines a near-scheme which uses object versioning to ensure coherence between the front and back tiers.</td>
</tr>
<tr>
<td>&lt;overflow-scheme&gt;</td>
<td>Optional</td>
<td>Defines a two tier cache scheme where entries evicted from a size-limited front-tier overflow and are stored in a much larger back-tier cache.</td>
</tr>
<tr>
<td>&lt;invocation-scheme&gt;</td>
<td>Optional</td>
<td>Defines an invocation service which can be used for performing custom operations in parallel across cluster nodes.</td>
</tr>
<tr>
<td>&lt;read-write-backing-map-scheme&gt;</td>
<td>Optional</td>
<td>Defines a backing map scheme which provides a cache of a persistent store.</td>
</tr>
<tr>
<td>&lt;versioned-backing-map-scheme&gt;</td>
<td>Optional</td>
<td>Defines a backing map scheme which uses object versioning to determine what updates need to be written to the persistent store.</td>
</tr>
</tbody>
</table>
### Table D–14 (Cont.) caching-schemes Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;remote-cache-scheme&gt;</code></td>
<td>Optional</td>
<td>Defines a cache scheme that enables caches to be accessed from outside a Coherence cluster by using Coherence*Extend.</td>
</tr>
<tr>
<td><code>&lt;class-scheme&gt;</code></td>
<td>Optional</td>
<td>Defines a cache scheme using a custom cache implementation. Any custom implementation must implement the <code>java.util.Map</code> interface, and include a zero-parameter public constructor. Additionally if the contents of the Map can be modified by anything other than the <code>CacheService</code> itself (for example, if the Map automatically expires its entries periodically or size-limits its contents), then the returned object must implement the <code>com.tangosol.util.ObservableMap</code> interface.</td>
</tr>
<tr>
<td><code>&lt;disk-scheme&gt;</code></td>
<td>Optional</td>
<td><strong>Note:</strong> As of Coherence 3.0, the disk-scheme configuration element has been deprecated and replaced by the <code>external-scheme</code> and <code>paged-external-scheme</code> configuration elements.</td>
</tr>
</tbody>
</table>
**class-scheme**

Used in: caching-schemes, local-scheme, distributed-scheme, replicated-scheme, optimistic-scheme, near-scheme, versioned-near-scheme, overflow-scheme, read-write-backing-map-scheme, versioned-backing-map-scheme, cachestore-scheme, listener

**Description**

Class schemes provide a mechanism for instantiating an arbitrary Java object for use by other schemes. The scheme which contains this element will dictate what class or interface(s) must be extended. See "Cache of a Database" on page F-4 for an example of using a class-scheme.

The class-scheme may be configured to either instantiate objects directly by using their class-name, or indirectly by using a class-factory-name and method-name. The class-scheme must be configured with either a class-name or class-factory-name and method-name.

**Elements**

Table D-15 describes the elements you can define within the class-scheme element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;scheme-name&gt;</td>
<td>Optional</td>
<td>Specifies the scheme's name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See &quot;Scheme Inheritance&quot; on page D-20 for more information.</td>
</tr>
<tr>
<td>&lt;class-name&gt;</td>
<td>Optional</td>
<td>Contains a fully specified Java class name to instantiate. This class must extend an appropriate implementation class as dictated by the containing scheme and must declare the exact same set of public constructors as the superclass.</td>
</tr>
<tr>
<td>&lt;class-factory-name&gt;</td>
<td>Optional</td>
<td>Specifies a fully specified name of a Java class that will be used as a factory for object instantiation.</td>
</tr>
<tr>
<td>&lt;method-name&gt;</td>
<td>Optional</td>
<td>Specifies the name of a static factory method on the factory class which will perform object instantiation.</td>
</tr>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Specifies initialization parameters which are accessible by implementations which support the com.tangosol.run.xml.XmlConfigurable interface, or which include a public constructor with a matching signature.</td>
</tr>
</tbody>
</table>
custom-store-manager


Description

Used to create and configure custom implementations of a store manager for use in external caches.

Elements

Table D–16 describes the elements you can define within the custom-store-manager element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/ Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-name&gt;</td>
<td>Required</td>
<td>Specifies the implementation of the store manager. The specified class must implement the com.tangosol.io.BinaryStoreManager interface.</td>
</tr>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Specifies initialization parameters, for use in custom store manager implementations which implement the com.tangosol.run.xml.XmlConfigurable interface.</td>
</tr>
</tbody>
</table>
Note: As of Coherence 3.0, the disk-scheme configuration element has been deprecated and replaced with the `<external-scheme>` and `<paged-external-scheme>` configuration elements.
**distributed-scheme**

Used in: caching-schemes, near-scheme, versioned-near-scheme, overflow-scheme, versioned-backing-map-scheme

**Description**

The distributed-scheme defines caches where the storage for entries is partitioned across cluster nodes. See "Partitioned Cache Service" for a more detailed description of partitioned caches. See “Partitioned Cache” on page F-6 examples of various distributed-scheme configurations.

**Clustered Concurrency Control**

Partitioned caches support cluster wide key-based locking so that data can be modified in a cluster without encountering the classic missing update problem. Note that any operation made without holding an explicit lock is still atomic but there is no guarantee that the value stored in the cache does not change between atomic operations.

**Cache Clients**

The partitioned cache service supports the concept of cluster nodes which do not contribute to the overall storage of the cluster. Nodes which are not storage enabled (see <local-storage> subelement) are considered "cache clients".

**Cache Partitions**

The cache entries are evenly segmented into several logical partitions (see <partition-count> subelement), and each storage enabled (see <local-storage> subelement) cluster node running the specified partitioned service (see <service-name> subelement) will be responsible for maintain a fair-share of these partitions.

**Key Association**

By default the specific set of entries assigned to each partition is transparent to the application. In some cases it may be advantageous to keep certain related entries within the same cluster node. A key-associator (see <key-associator> subelement) may be used to indicate related entries, the partitioned cache service will ensure that associated entries reside on the same partition, and thus on the same cluster node. Alternatively, key association may be specified from within the application code by using keys which implement the \texttt{com.tangosol.net.cache.KeyAssociation} interface.

**Cache Storage (Backing Map)**

Storage for the cache is specified by using the backing-map-scheme (see <backing-map-scheme> subelement). For instance a partitioned cache which uses a local-scheme for its backing map will result in cache entries being stored in-memory on the storage enabled cluster nodes.

**Failover**

For the purposes of failover a configurable number of backups (see <backup-count> subelement) of the cache may be maintained in backup-storage (see <backup-storage> subelement) across the cluster nodes. Each backup is also divided into
partitions, and when possible a backup partition will not reside on the same physical machine as the primary partition. If a cluster node abruptly leaves the cluster, responsibility for its partitions will automatically be reassigned to the existing backups, and new backups of those partitions will be created (on remote nodes) to maintain the configured backup count.

Partition Redistribution

When a node joins or leaves the cluster, a background redistribution of partitions occurs to ensure that all cluster nodes manage a fair-share of the total number of partitions. The amount of bandwidth consumed by the background transfer of partitions is governed by the transfer-threshold (see \texttt{<transfer-threshold>} subelement).

Elements

Table D–17 describes the elements you can define within the \texttt{distributed-scheme} element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{&lt;scheme-name&gt;}</td>
<td>Optional</td>
<td>Specifies the scheme’s name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td>\texttt{&lt;scheme-ref&gt;}</td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See “Scheme Inheritance” on page D-20 for more information.</td>
</tr>
<tr>
<td>\texttt{&lt;service-name&gt;}</td>
<td>Optional</td>
<td>Specifies the name of the service which will manage caches created from this scheme. Services are configured in the \texttt{&lt;services&gt;} element in the \texttt{tangosol-coherence.xml} descriptor. See Appendix H, “Operational Configuration Elements” for more information.</td>
</tr>
<tr>
<td>\texttt{&lt;serializer&gt;}</td>
<td>Optional</td>
<td>Specifies the class configuration info for a \texttt{com.tangosol.io.Serializer} implementation used by the Partitioned service to serialize and deserialize user types. For example, the following configures a \texttt{ConfigurablePofContext} that uses the default \texttt{coherence-pof-config.xml} configuration file to write objects to and read from a stream:</td>
</tr>
<tr>
<td>\texttt{&lt;listener&gt;}</td>
<td>Optional</td>
<td>Specifies an implementation of a \texttt{com.tangosol.MapListener} which will be notified of events occurring on the cache.</td>
</tr>
</tbody>
</table>
### Table D–17  (Cont.) distributed-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;backing-map-scheme&gt;</td>
<td>Optional</td>
<td>Specifies what type of cache will be used within the cache server to store the entries. The content of the backing map scheme could start with an optional value: &lt;partitioned&gt; – This optional value specifies whether the backing map itself is partitioned. It is respected only within a distributed-scheme. See “Storage” in the Oracle Coherence Getting Started Guide. Legal child schemes are: ■ local-scheme ■ external-scheme ■ paged-external-scheme ■ class-scheme ■ overflow-scheme ■ read-write-backing-map-scheme ■ versioned-backing-map-scheme Note that when using an overflow-based backing map it is important that the corresponding &lt;backup-storage&gt; be configured for overflow (potentially using the same scheme as the backing-map). See &quot;Partitioned Cache with Overflow&quot; on page F-6 for an example configuration.</td>
</tr>
<tr>
<td>&lt;partition-count&gt;</td>
<td>Optional</td>
<td>Specifies the number of partitions that a partitioned (distributed) cache will be “chopped up” into. Each member running the partitioned cache service that has the local-storage (&lt;local-storage&gt; subelement) option set to true will manage a &quot;fair&quot; (balanced) number of partitions. The number of partitions should be a prime number and sufficiently large such that a given partition is expected to be no larger than 50MB in size. The following are good defaults for sample service storage sizes: service storage partition-count 100M 257 1G 509 10G 2039 50G 4093 100G 8191 A list of first 1,000 primes can be found at <a href="http://primes.utm.edu/lists/">http://primes.utm.edu/lists/</a> Valid values are positive integers. Default value is the value specified in the tangosol-coherence.xml descriptor. See the partition-count parameter &quot;DistributedCache Service Parameters&quot; on page I-3 for more information.</td>
</tr>
<tr>
<td>&lt;key-associator&gt;</td>
<td>Optional</td>
<td>Specifies a class that will be responsible for providing associations between keys and allowing associated keys to reside on the same partition. This implementation must have a zero-parameter public constructor.</td>
</tr>
</tbody>
</table>
Optional Specifies a class that implements the com.tangosol.net.partition.KeyPartitioningStrategy interface, which will be responsible for assigning keys to partitions. This implementation must have a zero-parameter public constructor. If unspecified, the default key partitioning algorithm will be used, which ensures that keys are evenly segmented across partitions.

Optional Specifies a class that implements the com.tangosol.net.partition.PartitionListener interface.

Optional Specifies the number of members of the partitioned cache service that hold the backup data for each unit of storage in the cache. Value of 0 means that in the case of abnormal termination, some portion of the data in the cache will be lost. Value of N means that if up to N cluster nodes terminate immediately, the cache data will be preserved. To maintain the partitioned cache of size M, the total memory usage in the cluster does not depend on the number of cluster nodes and will be in the order of M*(N+1). Recommended values are 0 or 1. Default value is the backup-count value specified in the tangosol-coherence.xml descriptor. See the backup-count parameter in value specified in the tangosol-coherence.xml descriptor. See “DistributedCache Service Parameters” on page I-3 for more information.

Optional Specifies the number of members of the partitioned cache service that will hold the backup data for each unit of storage in the cache that does not require write-behind, that is, data that is not vulnerable to being lost even if the entire cluster were shut down. Specifically, if a unit of storage is marked as requiring write-behind, then it will be backed up on the number of members specified by the <backup-count> subelement, and if the unit of storage is not marked as requiring write-behind, then it will be backed up by the number of members specified by the <backup-count-after-writebehind> element. This value should be set to 0 or this setting should not be specified at all. The rationale is that since this data is being backed up to another data store, no in-memory backup is required, other than the data temporarily queued on the write-behind queue to be written. (Note that the setting also applies to write-through data, or any data that can be re-loaded from another data store by a CacheLoader or CacheStore.) The value of 0 means that when write-behind has occurred, the backup copies of that data will be discarded. However, until write-behind occurs, the data will be backed up in accordance with the <backup-count> setting. Recommended value is 0 or this element should be omitted altogether.

Optional Specifies the type and configuration for the partitioned cache backup storage.

Optional Specifies the number of daemon threads used by the partitioned cache service. If zero, all relevant tasks are performed on the service thread. Legal values are positive integers or zero. Default value is the thread-count value specified in the tangosol-coherence.xml descriptor. See the lthread-count parameter in “DistributedCache Service Parameters” on page I-3 for more information.
Cache Configuration Elements

Table D–17 (Cont.) distributed-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;lease-granularity&gt;</td>
<td>Optional</td>
<td>Specifies the lease ownership granularity. Available since release 2.3. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>thread</td>
</tr>
<tr>
<td></td>
<td></td>
<td>member</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A value of thread means that locks are held by a thread that obtained them and can only be released by that thread. A value of member means that locks are held by a cluster node and any thread running on the cluster node that obtained the lock can release it. Default value is the lease-granularity value specified in the tangosol-coherence.xml descriptor. See the lease-granularity parameter in “DistributedCache Service Parameters” on page I-3 for more information.</td>
</tr>
<tr>
<td>&lt;transfer-threshold&gt;</td>
<td>Optional</td>
<td>Specifies the threshold for the primary buckets distribution in kilo-bytes. When a new node joins the partitioned cache service or when a member of the service leaves, the remaining nodes perform a task of bucket ownership re-distribution. During this process, the existing data gets re-balanced along with the ownership information. This parameter indicates a preferred message size for data transfer communications. Setting this value lower will make the distribution process take longer, but will reduce network bandwidth utilization during this activity. Legal values are integers greater than zero. Default value is the transfer-threshold value specified in the tangosol-coherence.xml descriptor. See the transfer-threshold parameter in “DistributedCache Service Parameters” on page I-3 for more information.</td>
</tr>
<tr>
<td>&lt;local-storage&gt;</td>
<td>Optional</td>
<td>Specifies whether a cluster node will contribute storage to the cluster, that is, maintain partitions. When disabled the node is considered a cache client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Normally this value should be left unspecified within the configuration file, and instead set on a per-process basis using the tangosol.coherence.distributed.localstorage system property. This allows cache clients and servers to use the same configuration descriptor. Legal values are true or false. Default value is the local-storage value specified in the tangosol-coherence.xml descriptor. See the local-storage parameter in “DistributedCache Service Parameters” on page I-3 for more information.</td>
</tr>
<tr>
<td>&lt;autostart&gt;</td>
<td>Optional</td>
<td>The autostart element is intended to be used by cache servers (that is, com.tangosol.net.DefaultCacheServer). It specifies whether the cache services associated with this cache scheme should be automatically started at a cluster node. Legal values are true or false. Default value is false.</td>
</tr>
<tr>
<td>&lt;task-hung-threshold&gt;</td>
<td>Optional</td>
<td>Specifies the amount of time in milliseconds that a task can execute before it is considered &quot;hung&quot;. Note: a posted task that has not yet started is never considered as hung. This attribute is applied only if the Thread pool is used (the thread-count value is positive). Legal values are positive integers or zero (indicating no default timeout). Default value is the task-hung-threshold value specified in the tangosol-coherence.xml descriptor. See the task-hung-threshold parameter in “DistributedCache Service Parameters” on page I-3 for more information.</td>
</tr>
</tbody>
</table>
Table D–17  (Cont.)  distributed-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;task-timeout&gt;</td>
<td>Optional</td>
<td>Specifies the timeout value in milliseconds for requests executing on the service worker threads. This attribute is applied only if the thread pool is used (the thread-count value is positive). If zero is specified, the default service-guardian &lt;timeout-milliseconds&gt; value is used. Legal values are non-negative integers. Default value is the value specified in the tangosol-coherence.xml descriptor. See the task-timeout parameter in &quot;DistributedCache Service Parameters&quot; on page I-3.</td>
</tr>
</tbody>
</table>
| <request-timeout> | Optional   | Specifies the maximum amount of time a client will wait for a response before abandoning the original request. The request time is measured on the client side as the time elapsed from the moment a request is sent for execution to the corresponding server node(s) and includes the following:  
  - the time it takes to deliver the request to an executing node (server)  
  - the interval between the time the task is received and placed into a service queue until the execution starts  
  - the task execution time  
  - the time it takes to deliver a result back to the client  
  Legal values are positive integers or zero (indicating no default timeout). Default value is the value specified in the tangosol-coherence.xml descriptor. See the request-timeout parameter in "DistributedCache Service Parameters" on page I-3 for more information. |
| <operation-bundling> | Optional | Specifies the configuration info for a bundling strategy. |
**external-scheme**

Used in: caching-schemes, distributed-scheme, replicated-scheme, optimistic-scheme, near-scheme, versioned-near-scheme, overflow-scheme, read-write-backing-map-scheme, versioned-backing-map-scheme

**Description**

External schemes define caches which are not JVM heap based, allowing for greater storage capacity. See “Local Caches (accessible from a single JVM)” on page F-2 for examples of various external cache configurations.

**Implementation**

This scheme is implemented by:

- com.tangosol.net.cache.SerializationMap—for unlimited size caches
- com.tangosol.net.cache.SerializationCache—for size limited caches

The implementation type is chosen based on the following rule:

- if the `<high-units>` subelement is specified and not zero then SerializationCache is used;
- otherwise SerializationMap is used.

**Pluggable Storage Manager**

External schemes use a pluggable store manager to store and retrieve binary key value pairs. Supported store managers include:

- a wrapper providing asynchronous write capabilities for other store manager implementations
- allows definition of custom implementations of store managers
- uses Berkeley Database JE to implement an on disk cache
- uses a Coherence LH on disk database cache
- uses NIO to implement memory-mapped file based cache
- uses NIO to implement an off JVM heap, in-memory cache

**Size Limited Cache**

The cache may be configured as size-limited, which means that when it reaches its maximum allowable size (that is, the `<high-units>` subelement) it prunes itself.

---

**Note:** Eviction against disk-based caches can be expensive, consider using a paged-external-scheme for such cases.

**Entry Expiration**

External schemes support automatic expiration of entries based on the age of the value, as configured by the `<expiry-delay>` subelement.
Persistence (long-term storage)

External caches are generally used for temporary storage of large data sets, for example as the back-tier of an overflow-scheme. Certain implementations do however support persistence for non-clustered caches, see the <store-name> subelement of bdb-store-manager and the <manager-filename> subelement of lh-file-manager for details. Clustered persistence should be configured by using a read-write-backing-map-scheme on a distributed-scheme.

Elements

Table D-18 describes the elements you can define within the external-scheme element.

Table D-18 external-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;scheme-name&gt;</td>
<td>Optional</td>
<td>Specifies the scheme's name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See &quot;Scheme Inheritance&quot; on page D-20 for more information</td>
</tr>
</tbody>
</table>
| <class-name>      | Optional          | Specifies a custom implementation of the external cache. Any custom implementation must extend one of the following classes:
  - com.tangosol.net.cache.SerializationCache—for size limited caches
  - com.tangosol.net.cache.SerializationMap—for unlimited size caches
  - com.tangosol.net.cache.SimpleSerializationMap—for unlimited size caches
  and declare the exact same set of public constructors as the superclass. |
| <init-params>     | Optional          | Specifies initialization parameters, for use in custom external cache implementations which implement the com.tangosol.run.xml.XmlConfigurable interface. |
| <listener>        | Optional          | Specifies an implementation of a com.tangosol.util.MapListener which will be notified of events occurring on the cache. |
| <high-units>      | Optional          | Used to limit the size of the cache. Contains the maximum number of units that can be placed in the cache before pruning occurs. An entry is the unit of measurement. When this limit is exceeded, the cache will begin the pruning process, evicting the least recently used entries until the number of units is brought below this limit. The scheme's class-name element may be used to provide custom extensions to SerializationCache, which implement alternative eviction policies. Legal values are positive integers or zero. Zero implies no limit. Default value is zero. |
<unit-calculator> Optional Specifies the type of unit calculator to use. A unit calculator is used to determine the cost (in "units") of a given object. Legal values are:

- FIXED—A unit calculator that assigns an equal weight of 1 to all cached objects.
- BINARY—A unit calculator that assigns an object a weight equal to the number of bytes of memory required to cache the object. This requires that the objects are com.tangosol.util.Binary instances, as in a Partitioned cache. See com.tangosol.net.cache.BinaryMemoryCalculator for additional details.
- <class-scheme>— A custom unit calculator, specified as a class-scheme. The class specified within this scheme must implement the com/tangosol/net/cache/ConfigurableCacheMap.UnitCalculator interface.

This element is used only if the high-units element is set to a positive number. Default value is FIXED.

<unit-factor> Optional The unit-factor element specifies the factor by which the units, low-units and high-units properties are adjusted. Using a BINARY unit calculator, for example, the factor of 1048576 could be used to count megabytes instead of bytes.

Using a BINARY unit calculator, for example, the factor of 1048576 could be used to count megabytes instead of bytes.

Note: This element was introduced only to avoid changing the type of the units, low units and high units properties from 32-bit values to 64-bit values and is used only if the high-units element is set to a positive number. It is expected that this element will be dropped in Coherence 4.

Valid values are positive integer numbers. Default value is 1.

<expiry-delay> Optional Specifies the amount of time from last update that entries will be kept by the cache before being expired. Entries that are expired will not be accessible and will be evicted. The value of this element must be in the following format:

\[
[d]+[.][d]+\text{[MS|ms|S|s|M|m|H|h|D|d]}?
\]

where the first non-digits (from left to right) indicate the unit of time duration:

- MS or ms (milliseconds)
- S or s (seconds)
- M or m (minutes)
- H or h (hours)
- D or d (days)

If the value does not contain a unit, a unit of seconds is assumed. A value of zero implies no expiry. Default value is zero.

<async-store-manager> Optional Configures the external cache to use an asynchronous storage manager wrapper for any other storage manager. See "Pluggable Storage Manager" on page D-33.

<custom-store-manager> Optional Configures the external cache to use a custom storage manager implementation.
Table D–18 (Cont.) external-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;bdb-store-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the external cache to use Berkeley Database JE on disk databases for cache storage.</td>
</tr>
<tr>
<td><code>&lt;lh-file-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the external cache to use a Coherence LH on disk database for cache storage.</td>
</tr>
<tr>
<td><code>&lt;nio-file-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the external cache to use a memory-mapped file for cache storage.</td>
</tr>
<tr>
<td><code>&lt;nio-memory-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the external cache to use an off JVM heap, memory region for cache storage.</td>
</tr>
</tbody>
</table>
initiator-config

Used in: remote-cache-scheme, remote-invocation-scheme.

Description

The initiator-config element specifies the configuration info for a protocol-specific connection initiator. A connection initiator allows a Coherence*Extend client to connect to a cluster (by using a connection acceptor) and use the clustered services offered by the cluster without having to first join the cluster.

The initiator-config element must contain exactly one protocol-specific connection initiator configuration element (either jms-initiator or tcp-initiator).

Elements

Table D–19 describes the elements you can define within the initiator-config element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;jms-initiator&gt;</code></td>
<td>Optional</td>
<td>Specifies the configuration info for a connection initiator that connects to the cluster over JMS.</td>
</tr>
<tr>
<td><code>&lt;outgoing-message-handler&gt;</code></td>
<td>Optional</td>
<td>Specifies the configuration info used by the connection initiator to detect dropped client-to-cluster connections.</td>
</tr>
</tbody>
</table>
| `<serializer>`       | Optional          | Specifies the class configuration info for a Serializer implementation used by the connection initiator to serialize and deserialize user types. For example, the following configures a ConfigurablePofContext that uses the my-pof-types.xml POF type configuration file to deserialize user types to and from a POF stream:

```
<serializer>
  <class-name>com.tangosol.io.pof.ConfigurablePofContext</class-name>
  <init-params>
    <init-param>
      <param-type>string</param-type>
      <param-value>my-pof-types.xml</param-value>
    </init-param>
  </init-params>
</serializer>
```

| `<tcp-initiator>`    | Optional          | Specifies the configuration info for a connection initiator that connects to the cluster over TCP/IP. |
| `<use-filters>`      | Optional          | Contains the list of `<filters>` names to be used by this connection initiator. In the following example, specifying use-filter will activate gzip compression for all network messages, which can help substantially with WAN and low-bandwidth networks. |

```
<use-filters>
  <filter-name>gzip</filter-name>
</use-filters>
```
init-param

Used in: init-params.
Defines an individual initialization parameter.

Elements

Table D-20 describes the elements you can define within the init-param element.
### Table D–20  init-param Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;param-name&gt;</td>
<td>Optional</td>
<td>Contains the name of the initialization parameter. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;class-name&gt;com.mycompany.cache.CustomCacheLoader&lt;/class-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;init-params&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;init-param&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;param-name&gt;sTableName&lt;/param-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;param-value&gt;EmployeeTable&lt;/param-value&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;/init-param&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;init-param&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;param-name&gt;iMaxSize&lt;/param-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;param-value&gt;2000&lt;/param-value&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;/init-param&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;/init-params&gt;</code></td>
</tr>
<tr>
<td>&lt;param-type&gt;</td>
<td>Optional</td>
<td>Contains the Java type of the initialization parameter. The following standard types are supported:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ java.lang.String (a.k.a. string)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ java.lang.Boolean (a.k.a. boolean)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ java.lang.Integer (a.k.a. int)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ java.lang.Long (a.k.a. long)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ java.lang.Double (a.k.a. double)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ java.math.BigDecimal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ java.io.File</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ java.sql.Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ java.sql.Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ java.sql.Timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;class-name&gt;com.mycompany.cache.CustomCacheLoader&lt;/class-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;init-params&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;init-param&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;param-type&gt;java.lang.String&lt;/param-type&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;param-value&gt;EmployeeTable&lt;/param-value&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;/init-param&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;init-param&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;param-type&gt;int&lt;/param-type&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;param-value&gt;2000&lt;/param-value&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;/init-param&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;/init-params&gt;</code></td>
</tr>
<tr>
<td>&lt;param-value&gt;</td>
<td>Optional</td>
<td>Contains the value of the initialization parameter. The value is in the format specific to the Java type of the parameter. See Appendix E, “Cache Configuration Parameter Macros” for the list of available macros.</td>
</tr>
</tbody>
</table>
init-params

Used in: class-scheme, cache-mapping.

Description

Defines a series of initialization parameters as name-value pairs. See "Cache of a Database" on page F-4 for an example of using init-params.

Elements

Table D–21 describes the elements you can define within the init-params element.

| Table D–21  | init-params Subelements
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Required/</td>
</tr>
<tr>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>&lt;init-param&gt;</td>
<td>Optional</td>
</tr>
</tbody>
</table>
invocation-scheme

Used in: caching-schemes.

Description

Defines an Invocation Service. The invocation service may be used to perform custom operations in parallel on any number of cluster nodes. See the com.tangosol.net.InvocationService API for additional details.

Elements

The following table describes the elements you can define within the invocation-scheme element.

<table>
<thead>
<tr>
<th>Table D–22 invocation-scheme Subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
</tr>
<tr>
<td>&lt;scheme-name&gt;</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
</tr>
<tr>
<td>&lt;service-name&gt;</td>
</tr>
</tbody>
</table>
| <serializer> | Optional | Specifies the class configuration info for a com.tangosol.io.Serializer implementation used by the Invocation service to serialize and deserialize user types. For example, the following configures a ConfigurablePofContext that uses the default coherence-pof-types.xml configuration file to write objects to and read from a stream:  
  <serializer>  
    <class-name>com.tangosol.io.pof.ConfigurablePofContext</class-name>  
  </serializer>  
  Services are configured from within the <services> element in the tangosol-coherence.xml descriptor. See Appendix H, “Operational Configuration Elements” for more information. |
| <thread-count> | Optional | Specifies the number of daemon threads used by the invocation service. If zero, all relevant tasks are performed on the service thread. Legal values are positive integers or zero. Default value is the thread-count value specified in the tangosol-coherence.xml descriptor. See the thread-count parameter in “InvocationService Parameters” on page I-8. |
| <autostart> | Optional | The autostart element is intended to be used by cache servers (that is, com.tangosol.net.DefaultCacheServer). It specifies whether this service should be automatically started at a cluster node. Legal values are true or false. Default value is false. |
### Table D–22 (Cont.) invocation-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/ Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;task-hung-threshold&gt;</td>
<td>Optional</td>
<td>Specifies the amount of time in milliseconds that a task can execute before it is considered &quot;hung&quot;. Note: a posted task that has not yet started is never considered as hung. This attribute is applied only if the Thread pool is used (the thread-count value is positive). Legal values are positive integers or zero (indicating no default timeout). Default value is the task-hung-threshold value specified in the tangosol-coherence.xml descriptor. See the task-hung-threshold parameter in &quot;InvocationService Parameters&quot; on page I-8.</td>
</tr>
<tr>
<td>&lt;task-timeout&gt;</td>
<td>Optional</td>
<td>Specifies the default timeout value in milliseconds for tasks that can be timed-out (for example, implement the com.tangosol.net.PriorityTask interface), but don't explicitly specify the task execution timeout value. The task execution time is measured on the server side and does not include the time spent waiting in a service backlog queue before being started. This attribute is applied only if the thread pool is used (the thread-count value is positive). If zero is specified, the default service-guardian &lt;timeout-milliseconds&gt; value is used. Legal values are non-negative integers. Default value is the task-timeout value specified in the tangosol-coherence.xml descriptor. See the task-timeout parameter in &quot;InvocationService Parameters&quot; on page I-8.</td>
</tr>
<tr>
<td>&lt;request-timeout&gt;</td>
<td>Optional</td>
<td>Specifies the default timeout value in milliseconds for requests that can time-out (for example, implement the com.tangosol.net.PriorityTask interface), but don't explicitly specify the request timeout value. The request time is measured on the client side as the time elapsed from the moment a request is sent for execution to the corresponding server node(s) and includes the following:&lt;br&gt; (1) the time it takes to deliver the request to an executing node (server); (2) the interval between the time the task is received and placed into a service queue until the execution starts; (3) the task execution time; (4) the time it takes to deliver a result back to the client.&lt;br&gt; Legal values are positive integers or zero (indicating no default timeout). Default value is the request-timeout value specified in the tangosol-coherence.xml descriptor. See the request-timeout parameter in &quot;InvocationService Parameters&quot; on page I-8.</td>
</tr>
</tbody>
</table>
invocation-service-proxy

Used in: proxy-config

Description

The `invocation-service-proxy` element contains the configuration info for an invocation service proxy managed by a proxy service.

Elements

Table D–23 describes the elements you can define within the `invocation-service-proxy` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;enabled&gt;</td>
<td>Optional</td>
<td>Specifies whether the invocation service proxy is enabled. If disabled, clients will not be able to execute Invocable objects on the proxy service JVM. Legal values are <code>true</code> or <code>false</code>. Default value is <code>true</code>.</td>
</tr>
</tbody>
</table>
The **jms-acceptor** element specifies the configuration info for a connection acceptor that accepts connections from Coherence*Extend clients over JMS. For additional details and example configurations see Chapter 17, "Configuring and Using Coherence*Extend."

### Elements

Table D–24 describes the elements you can define within the **jms-acceptor** element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;queue-connection-factory-name&gt;</td>
<td>Required</td>
<td>Specifies the JNDI name of the JMS QueueConnectionFactory used by the connection acceptor.</td>
</tr>
<tr>
<td>&lt;queue-name&gt;</td>
<td>Required</td>
<td>Specifies the JNDI name of the JMS Queue used by the connection acceptor.</td>
</tr>
</tbody>
</table>
The `jms-initiator` element specifies the configuration info for a connection initiator that enables Coherence*Extend clients to connect to a remote cluster by using JMS. For additional details and example configurations see Chapter 17, "Configuring and Using Coherence*Extend."

Elements

The following table describes the elements you can define within the `jms-initiator` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/ Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;queue-connection-factory-name&gt;</code></td>
<td>Required</td>
<td>Specifies the JNDI name of the JMS QueueConnectionFactory used by the connection initiator.</td>
</tr>
<tr>
<td><code>&lt;queue-name&gt;</code></td>
<td>Required</td>
<td>Specifies the JNDI name of the JMS Queue used by the connection initiator.</td>
</tr>
<tr>
<td><code>&lt;connect-timeout&gt;</code></td>
<td>Optional</td>
<td>Specifies the maximum amount of time to wait while establishing a connection with a connection acceptor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The value of this element must be in the following format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>[^d]+([^.)][^d]+)?[^MSmsSsMmHhDd]?</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>where the first non-digits (from left to right) indicate the unit of time duration:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- MS or ms (milliseconds)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- S or s (seconds)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- M or m (minutes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- H or h (hours)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- D or d (days)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the value does not contain a unit, a unit of milliseconds is assumed. Default value is an infinite timeout.</td>
</tr>
</tbody>
</table>

**Note:** Coherence*Extend-JMS support has been deprecated.
key-associator

Used in: distributed-scheme

Description

Specifies an implementation of a com.tangosol.net.partition.KeyAssociator which will be used to determine associations between keys, allowing related keys to reside on the same partition.

Alternatively the cache's keys may manage the association by implementing the com.tangosol.net.cache.KeyAssociation interface.

Elements

Table D–26 describes the elements you can define within the key-associator element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-name&gt;</td>
<td>Required</td>
<td>The name of a class that implements the com.tangosol.net.partition.KeyAssociator interface. This implementation must have a zero-parameter public constructor. Default value is the value of the key-associator/class-name parameter specified in the tangosol.coherence.xml descriptor. See &quot;DistributedCache Service Parameters&quot; on page I-3 for more information.</td>
</tr>
</tbody>
</table>
key-partitioning

Used in: distributed-scheme

Description

Specifies an implementation of a `com.tangosol.net.partition.KeyPartitioningStrategy` which will be used to determine the partition in which a key will reside.

Elements

Table D–27 describes the elements you can define within the key-partitioning element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;class-name&gt;</code></td>
<td>Required</td>
<td>The name of a class that implements the <code>com.tangosol.net.partition.KeyPartitioningStrategy</code> interface. This implementation must have a zero-parameter public constructor. Default value is the value of the key-partitioning/class-name parameter specified in the <code>tangosol-coherence.xml</code> descriptor. See “DistributedCache Service Parameters” on page I-3 for more information.</td>
</tr>
</tbody>
</table>
**lh-file-manager**


**Description**

Configures a store manager which will use a Coherence LH on disk embedded database for storage. See "Persistent Cache on Disk" on page F-3 and "In-memory Cache with Disk Based Overflow" on page F-4 for examples of LH-based store configurations.

**Implementation**

Implemented by the `com.tangosol.io.lh.LHBinaryStoreManager` class. The `BinaryStore` objects created by this class are instances of `com.tangosol.io.lh.LHBinaryStore`.

**Elements**

Table D–28 describes the elements you can define within the `lh-file-manager` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;class-name&gt;</code></td>
<td>Optional</td>
<td>Specifies a custom implementation of the LH BinaryStoreManager. Any custom implementation must extend the <code>com.tangosol.io.lh.LHBinaryStoreManager</code> class and declare the exact same set of public constructors.</td>
</tr>
<tr>
<td><code>&lt;init-params&gt;</code></td>
<td>Optional</td>
<td>Specifies initialization parameters, for use in custom LH file manager implementations which implement the <code>com.tangosol.run.xml.XmlConfigurable</code> interface.</td>
</tr>
<tr>
<td><code>&lt;directory&gt;</code></td>
<td>Optional</td>
<td>Specifies the path name for the root directory that the LH file manager will use to store files in. If not specified or specifies a non-existent directory, a temporary file in the default location will be used.</td>
</tr>
<tr>
<td><code>&lt;file-name&gt;</code></td>
<td>Optional</td>
<td>Specifies the name for a non-temporary (persistent) file that the LH file manager will use to store data in. Specifying this parameter will cause the lh-file-manager to use non-temporary database instances. Use this parameter only for local caches that are backed by a cache loader from a non-temporary file: this allows the local cache to be pre-populated from the disk file on startup. When specified it is recommended that it use the <code>{cache-name}</code> macro described in Appendix E, “Cache Configuration Parameter Macros” macro. Normally this parameter should be left unspecified, indicating that temporary storage is to be used.</td>
</tr>
</tbody>
</table>
listener


Description

The Listener element specifies an implementation of a com.tangosol.util.MapListener which will be notified of events occurring on a cache.

Elements

The following table describes the elements you can define within the listener element.

Table D–29 listener Subelement

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-scheme&gt;</td>
<td>Required</td>
<td>Specifies the full class name of the listener implementation to use. The specified class must implement the com.tangosol.util.MapListener interface.</td>
</tr>
</tbody>
</table>
local-scheme

Used in: caching-schemes, distributed-scheme, replicated-scheme, optimistic-scheme, near-scheme, versioned-near-scheme, overflow-scheme, read-write-backing-map-scheme, versioned-backing-map-scheme

Description

Local cache schemes define in-memory "local" caches. Local caches are generally nested within other cache schemes, for instance as the front-tier of a near-scheme. See "Local Cache of a Partitioned Cache (Near cache)" on page F-7 for examples of various local cache configurations.

Implementation

Local caches are implemented by the com.tangosol.net.cache.LocalCache class.

Cache of an External Store

A local cache may be backed by an external cache store (see "cachestore-scheme" on page D-18). Cache misses will read-through to the back end store to retrieve the data. If a writable store is provided, cache writes will propagate to the cache store as well. For optimizing read/write access against a cache store, see the "read-write-backing-map-scheme" on page D-76.

Size Limited Cache

The cache may be configured as size-limited, which means that when it reaches its maximum allowable size (see <allowable-size> subelement) it prunes itself back to a specified smaller size (see <low-units> subelement), choosing which entries to evict according to its eviction-policy (see <eviction-policy> subelement). The entries and size limitations are measured in terms of units as calculated by the scheme's unit-calculator (see <unit-calculator> subelement).

Entry Expiration

The local cache supports automatic expiration of entries based on the age of the value, as configured by the expiry-delay (see <expiry-delay> subelement).

Elements

Table D–30 describes the elements you can define within the local-scheme element.

Table D–30   local-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;scheme-name&gt;</td>
<td>Optional</td>
<td>Specifies the scheme’s name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See &quot;Scheme Inheritance&quot; on page D-20 for more information.</td>
</tr>
<tr>
<td>&lt;service-name&gt;</td>
<td>Optional</td>
<td>Specifies the name of the service which will manage caches created from this scheme. Services are configured from within the &lt;services&gt; element in the tangosol-coherence.xml descriptor. See Appendix H, &quot;Operational Configuration Elements&quot; for more information.</td>
</tr>
</tbody>
</table>
### Table D–30  (Cont.) local-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-name&gt;</td>
<td>Optional</td>
<td>Specifies a custom implementation of the local cache. Any custom implementation must extend the <code>com.tangosol.net.cache.LocalCache</code> class and declare the exact same set of public constructors.</td>
</tr>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Specifies initialization parameters, for use in custom local cache implementations which implement the <code>com.tangosol.run.xml.XmlConfigurable</code> interface.</td>
</tr>
<tr>
<td>&lt;listener&gt;</td>
<td>Optional</td>
<td>Specifies an implementation of a <code>com.tangosol.util.MapListener</code> which will be notified of events occurring on the cache.</td>
</tr>
<tr>
<td>&lt;cachestore-scheme&gt;</td>
<td>Optional</td>
<td>Specifies the store which is being cached. If unspecified the cached data will only reside in memory, and only reflect operations performed on the cache itself.</td>
</tr>
<tr>
<td>&lt;eviction-policy&gt;</td>
<td>Optional</td>
<td>Specifies the type of eviction policy to use. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ LRU - Least Recently Used eviction policy chooses which entries to evict based on how recently they were last accessed, evicting those that were not accessed the for the longest period first.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ LFU - Least Frequently Used eviction policy chooses which entries to evict based on how often they are being accessed, evicting those that are accessed least frequently first.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ HYBRID - Hybrid eviction policy chooses which entries to evict based the combination (weighted score) of how often and recently they were accessed, evicting those that are accessed least frequently and were not accessed for the longest period first.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &lt;class-scheme&gt; - A custom eviction policy, specified as a class-scheme. The class specified within this scheme must implement the <code>com/tangosol/net/cache/ConfigurableCacheMap.EvictionPolicy</code> interface.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value is HYBRID.</td>
</tr>
<tr>
<td>&lt;high-units&gt;</td>
<td>Optional</td>
<td>Used to limit the size of the cache. Contains the maximum number of units that can be placed in the cache before pruning occurs. An entry is the unit of measurement, unless it is overridden by an alternate unit-calculator (see &lt;unit-calculator&gt; subelement). When this limit is exceeded, the cache will begin the pruning process, evicting entries according to the eviction policy until the low-units (see &lt;low-units&gt; subelement) size is reached. Legal values are positive integers or zero. Zero implies no limit. Default value is 0.</td>
</tr>
<tr>
<td>&lt;low-units&gt;</td>
<td>Optional</td>
<td>Contains the number of units that the cache will be pruned down to when pruning takes place. An entry is the unit of measurement, unless it is overridden by an alternate unit-calculator (see &lt;unit-calculator&gt; subelement). When pruning occurs entries will continue to be evicted according to the eviction policy until this size. Legal values are positive integers or zero. Zero implies the default. Default value is 75% of the high-units setting (that is, for a high-units setting of 1000 the default low-units will be 750).</td>
</tr>
</tbody>
</table>
Optional Specifies the type of unit calculator to use. A unit calculator is used to determine the cost (in "units") of a given object. Legal values are:

- **FIXED**—A unit calculator that assigns an equal weight of 1 to all cached objects.
- **BINARY**—A unit calculator that assigns an object a weight equal to the number of bytes of memory required to cache the object. This requires that the objects are `com.tangosol.util.Binary` instances, as in a Partitioned cache. See `com.tangosol.net.cache.BinaryMemoryCalculator` for additional details.
- **<class-scheme>**—A custom unit calculator, specified as a class-scheme. The class specified within this scheme must implement the `com/tangosol/net/cache/ConfigurableCacheMap.UnitCalculator` interface.

This element is used only if the `high-units` element is set to a positive number. Default value is `FIXED`.

Optional The unit-factor element specifies the factor by which the units, low-units and high-units properties are adjusted. Using a **BINARY** unit calculator, for example, the factor of 1048576 could be used to count megabytes instead of bytes.

Using a **BINARY** unit calculator, for example, the factor of 1048576 could be used to count megabytes instead of bytes.

**Note:** This element was introduced only to avoid changing the type of the units, low units and high units properties from 32-bit values to 64-bit values and is used only if the high-units element is set to a positive number. It is expected that this element will be dropped in Coherence 4.

Valid values are positive integer numbers. Default value is 1.
The unit-factor element specifies the factor by which the units, low-units and high-units properties are adjusted. Using a BINARY unit calculator, for example, the factor of 1048576 could be used to count megabytes instead of bytes.

Valid values are positive integer numbers. Default value is 1.

Note: This element was introduced only to avoid changing the type of the units, low units and high units properties from 32-bit values to 64-bit values.

Optional The expiry-delay element specifies the amount of time from last update that entries will be kept by the cache before being marked as expired. Any attempt to read an expired entry will result in a reloading of the entry from the configured cache store (see <cachestore-scheme>). Expired values are periodically discarded from the cache based on the flush-delay (see <flush-delay> subelement). The value of this element must be in the following format:

\[ [\d]+\[.\] [\d]+? [MS|ms|S|s|M|m|H|h|D|d] \]

where the first non-digits (from left to right) indicate the unit of time duration:

- MS or ms (milliseconds)
- S or s (seconds)
- M or m (minutes)
- H or h (hours)
- D or d (days)

If the value does not contain a unit, a unit of seconds is assumed. A value of zero implies no expiry. Default value is zero.

Optional The flush-delay element specifies the time interval between periodic cache flushes, which will discard expired entries from the cache, thus freeing resources. The value of this element must be in the following format:

\[ [\d]+\[.\] [\d]+? [MS|ms|S|s|M|m|H|h|D|d] \]

where the first non-digits (from left to right) indicate the unit of time duration:

- MS or ms (milliseconds)
- S or s (seconds)
- M or m (minutes)
- H or h (hours)
- D or d (days)

If the value does not contain a unit, a unit of seconds is assumed. If <expiry-delay> is enabled, the default flush-delay is 1m, otherwise a default of zero is used and automatic flushes are disabled.
near-scheme

Used in: caching-schemes.

Description

The near-scheme defines a two-tier cache consisting of a front-tier (see <front-scheme> subelement) which caches a subset of a back-tier cache (see <back-scheme> subelement). The front-tier is generally a fast, size limited cache, while the back-tier is slower, but much higher capacity. A typical deployment might use a local-scheme for the front-tier, and a distributed-scheme for the back-tier. The result is that a portion of a large partitioned cache will be cached locally in-memory allowing for very fast read access. See Appendix B, “Types of Caches in Coherence,” for a more detailed description of near caches, and “Local Cache of a Partitioned Cache (Near cache)” on page F-7 for an example of near cache configurations.

Implementation

The near scheme is implemented by the com.tangosol.net.cache.NearCache class.

Front-tier Invalidation

Specifying an invalidation-strategy (see <invalidation-strategy> subelement) defines a strategy that is used to keep the front tier of the near cache in sync with the back tier. Depending on that strategy a near cache is configured to listen to certain events occurring on the back tier and automatically update (or invalidate) the front portion of the near cache.

Elements

Table D–31 describes the elements you can define within the near-scheme element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;scheme-name&gt;</td>
<td>Optional</td>
<td>Specifies the scheme's name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See &quot;Scheme Inheritance&quot; on page D-20 for more information</td>
</tr>
<tr>
<td>&lt;class-name&gt;</td>
<td>Optional</td>
<td>Specifies a custom implementation of the near cache. Any custom implementation must extend the com.tangosol.net.cache.NearCache class and declare the exact same set of public constructors.</td>
</tr>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Specifies initialization parameters for custom near cache implementations which implement the com.tangosol.run.xml.XmlConfigurable interface.</td>
</tr>
<tr>
<td>&lt;listener&gt;</td>
<td>Optional</td>
<td>Specifies an implementation of a com.tangosol.util.MapListener which will be notified of events occurring on the cache.</td>
</tr>
</tbody>
</table>
### Table D–31  (Cont.) near-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;front-scheme&gt;</td>
<td>Required</td>
<td>Specifies the cache-scheme to use in creating the front-tier cache. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- local-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- paged-external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- class-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The eviction policy of the front-scheme defines which entries will be cached locally. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;front-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;local-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;eviction-policy&gt;HYBRID&lt;/eviction-policy&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;high-units&gt;1000&lt;/high-units&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/local-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/front-scheme&gt;</td>
</tr>
</tbody>
</table>
Table D–31 (Cont.) near-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;back-scheme&gt;</td>
<td>Required</td>
<td>Specifies the cache-scheme to use in creating the back-tier cache. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ distributed-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ replicated-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ optimistic-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ local-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ paged-external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ class-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ remote-cache-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;back-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;distributed-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;scheme-ref&gt;default-distributed&lt;/scheme-ref&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/distributed-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/back-scheme&gt;</td>
</tr>
<tr>
<td>&lt;invalidation-strategy&gt;</td>
<td>Optional</td>
<td>Specifies the strategy used keep the front-tier in-sync with the back-tier. Please see com.tangosol.net.cache.NearCache for more details. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ none - instructs the cache not to listen for invalidation events at all. This is the best choice for raw performance and scalability when business requirements permit the use of data which might not be absolutely current. Freshness of data can be guaranteed by use of a sufficiently brief eviction policy. The worst case performance is identical to a standard Distributed cache.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ present - instructs the near cache to listen to the back map events related only to the items currently present in the front map. This strategy works best when cluster nodes have sticky data access patterns (for example, HTTP session management with a sticky load balancer).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ all - instructs the near cache to listen to all back map events. This strategy is optimal for read-heavy access patterns where there is significant overlap between the front caches on each cluster member.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ auto - instructs the near cache to switch between present and all strategies automatically based on the cache statistics. Default value is auto.</td>
</tr>
<tr>
<td>&lt;autostart&gt;</td>
<td>Optional</td>
<td>The autostart element is intended to be used by cache servers (that is, com.tangosol.net.DefaultCacheServer). It specifies whether the cache services associated with this cache scheme should be automatically started at a cluster node. Legal values are true or false. Default value is false.</td>
</tr>
</tbody>
</table>
nio-file-manager


**Description**

Configures an external store which uses memory-mapped file for storage.

**Implementation**

This store manager is implemented by the com.tangosol.io.nio.MappedStoreManager class. The BinaryStore objects created by this class are instances of the com.tangosol.io.nio.BinaryMapStore.

**Elements**

Table D–32 describes the elements you can define within the nio-file-manager element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-name&gt;</td>
<td>Optional</td>
<td>Specifies a custom implementation of the local cache. Any custom implementation must extend the com.tangosol.io.nio.MappedStoreManager class and declare the exact same set of public constructors.</td>
</tr>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Specifies initialization parameters, for use in custom nio-file-manager implementations which implement the com.tangosol.run.xml.XmlConfigurable interface.</td>
</tr>
<tr>
<td>&lt;initial-size&gt;</td>
<td>Optional</td>
<td>Specifies the initial buffer size in megabytes. The value of this element must be in the following format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[d]+(.[d]+)?(K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>where the first non-digit (from left to right) indicates the factor with which the preceding decimal value should be multiplied:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ K or k (kilo, 210)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ M or m (mega, 220)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ G or g (giga, 230)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the value does not contain a factor, a factor of mega is assumed. Legal values are positive integers between 1 and Integer.MAX_VALUE - 1023 (that is, 2,147,482,624 bytes). Default value is 1MB.</td>
</tr>
<tr>
<td>&lt;maximum-size&gt;</td>
<td>Optional</td>
<td>Specifies the maximum buffer size in bytes. The value of this element must be in the following format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[d]+(.[d]+)?(K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>where the first non-digit (from left to right) indicates the factor with which the preceding decimal value should be multiplied:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ K or k (kilo, 210)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ M or m (mega, 220)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ G or g (giga, 230)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the value does not contain a factor, a factor of mega is assumed. Legal values are positive integers between 1 and Integer.MAX_VALUE - 1023 (that is, 2,147,482,624 bytes). Default value is 1024MB.</td>
</tr>
<tr>
<td>&lt;directory&gt;</td>
<td>Optional</td>
<td>Specifies the path name for the root directory that the manager will use to store files in. If not specified or specifies a non-existent directory, a temporary file in the default location will be used.</td>
</tr>
</tbody>
</table>
nio-memory-manager


Description

Configures a store-manager which uses an off JVM heap, memory region for storage, which means that it does not affect the Java heap size and the related JVM garbage-collection performance that can be responsible for application pauses. See "NIO In-memory Cache" on page F-2 for an example of an NIO cache configuration.

Note: Some JVMs (starting with 1.4) require the use of a command line parameter if the total NIO buffers will be greater than 64MB. For example: -XX:MaxDirectMemorySize=512M

Implementation

Implemented by the com.tangosol.io.nio.DirectStoreManager class. The BinaryStore objects created by this class are instances of the com.tangosol.io.nio.BinaryMapStore.

Elements

Table D–33 describes the elements you can define within the nio-memory-manager element.

Table D–33  nio-memory-manager Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-name&gt;</td>
<td>Optional</td>
<td>Specifies a custom implementation of the local cache. Any custom implementation must extend the com.tangosol.io.nio.DirectStoreManager class and declare the exact same set of public constructors.</td>
</tr>
</tbody>
</table>
Table D–33  (Cont.) nio-memory-manager Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Specifies initialization parameters, for use in custom nio-memory-manager implementations which implement the com.tangosol.run.xml.XmlConfigurable interface.</td>
</tr>
<tr>
<td>&lt;initial-size&gt;</td>
<td>Optional</td>
<td>Specifies the initial buffer size in bytes. The value of this element must be in the following format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[\d]+[.][\d]+?[K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the preceding decimal value should be multiplied:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ K or k (kilo, 210)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ M or m (mega, 220)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ G or g (giga, 230)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the value does not contain a factor, a factor of mega is assumed. Legal values are positive integers between 1 and Integer.MAX_VALUE - 1023 (that is, 2,147,482,624 bytes). Default value is 1MB.</td>
</tr>
<tr>
<td>&lt;maximum-size&gt;</td>
<td>Optional</td>
<td>Specifies the maximum buffer size in bytes. The value of this element must be in the following format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[\d]+[.][\d]+?[K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the preceding decimal value should be multiplied:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ K or k (kilo, 210)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ M or m (mega, 220)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ G or g (giga, 230)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the value does not contain a factor, a factor of mega is assumed. Legal values are positive integers between 1 and Integer.MAX_VALUE - 1023 (that is, 2,147,482,624 bytes). Default value is 1024MB.</td>
</tr>
</tbody>
</table>
operation-bundling

Used in: cachestore-scheme, distributed-scheme, remote-cache-scheme.

Description

The operation-bundling element specifies the configuration info for a particular bundling strategy.

Bundling is a process of coalescing multiple individual operations into "bundles". It could be beneficial when

- there is a continuous stream of operations on multiple threads in parallel;
- individual operations have relatively high latency (network or database-related); and
- there are functionally analogous "bulk" operations that take a collection of arguments instead of a single one without causing the latency to grow linearly (as a function of the collection size).

**Note:** As with any bundling algorithm, there is a natural trade-off between the resource utilization and average request latency. Depending on a particular application usage pattern, enabling this feature may either help or hurt the overall application performance.

See com.tangosol.net.cache.AbstractBundler for additional implementation details.

Elements

Table D–34 describes the subelement for the operation-bundling element.

<table>
<thead>
<tr>
<th>Table D–34</th>
<th>operation-bundling Subelement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Required/Optional</td>
</tr>
<tr>
<td>&lt;bundle-config&gt;</td>
<td>Required</td>
</tr>
</tbody>
</table>
optimistic-scheme

Used in: caching-schemes, near-scheme, versioned-near-scheme, overflow-scheme

The optimistic scheme defines a cache which fully replicates all of its data to all cluster nodes that run the service (see <service-name> subelement). See Appendix B, "Types of Caches in Coherence" for a more detailed description of optimistic caches.

Optimistic Locking

Unlike the replicated-scheme and distributed-scheme caches, optimistic caches do not support concurrency control (locking). Individual operations against entries are atomic but there is no guarantee that the value stored in the cache does not change between atomic operations. The lack of concurrency control allows optimistic caches to support very fast write operations.

Cache Storage (Backing Map)

Storage for the cache is specified by using the backing-map-scheme (see <backing-map-scheme> subelement). For instance an optimistic cache which uses a local-scheme for its backing map will result in cache entries being stored in-memory.

Elements

Table D–35 describes the elements you can define within the optimistic-scheme element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;scheme-name&gt;</td>
<td>Optional</td>
<td>Specifies the scheme’s name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See &quot;Scheme Inheritance&quot; on page D-20 for more information.</td>
</tr>
<tr>
<td>&lt;service-name&gt;</td>
<td>Optional</td>
<td>Specifies the name of the service which will manage caches created from this scheme. Services are configured from within the &lt;services&gt; parameter in tangosol-coherence.xml. See Appendix H, &quot;Operational Configuration Elements&quot; for more information.</td>
</tr>
</tbody>
</table>
### Table D–35 (Cont.) optimistic-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;listener&gt;</code></td>
<td>Optional</td>
<td>Specifies an implementation of a <code>com.tangosol.util.MapListener</code> which will be notified of events occurring on the cache.</td>
</tr>
<tr>
<td><code>&lt;backing-map-scheme&gt;</code></td>
<td>Optional</td>
<td>Specifies what type of cache will be used within the cache server to store the entries. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ local-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ paged-external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ overflow-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ class-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To ensure cache coherence, the backing-map of an optimistic cache must not use a read-through pattern to load cache entries. Either use a cache-aside pattern from outside the cache service, or switch to the distributed-scheme, which supports read-through clustered caching.</td>
</tr>
<tr>
<td><code>&lt;autostart&gt;</code></td>
<td>Optional</td>
<td>The autostart element is intended to be used by cache servers (that is, <code>com.tangosol.net.DefaultCacheServer</code>). It specifies whether the cache services associated with this cache scheme should be automatically started at a cluster node. Legal values are true or false. Default value is false.</td>
</tr>
</tbody>
</table>
**outgoing-message-handler**

**Used in:** acceptor-config, initiator-config.

**Description**

The outgoing-message-handler specifies the configuration info used to detect dropped client-to-cluster connections. For connection initiators and acceptors that use connectionless protocols, this information is necessary to proactively detect and release resources allocated to dropped connections. Connection-oriented initiators and acceptors can also use this information as an additional mechanism to detect dropped connections.

**Elements**

Table D–36 describes the elements you can define within the outgoing-message-handler element.
### Table D–36 outgoing-message-handler Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<heartbeat-interval>`  | Optional          | Specifies the interval between ping requests. A ping request is used to ensure the integrity of a connection. The value of this element must be in the following format: $[\d]+[\d.]+[MS|ms|S|s|M|m|H|h|D|d]?$ where the first non-digits (from left to right) indicate the unit of time duration:  
  - MS or ms (milliseconds)  
  - S or s (seconds)  
  - M or m (minutes)  
  - H or h (hours)  
  - D or d (days)  
  If the value does not contain a unit, a unit of milliseconds is assumed. A value of zero disables ping requests. The default value is zero. |
| `<heartbeat-timeout>`   | Optional          | Specifies the maximum amount of time to wait for a response to a ping request before declaring the underlying connection unusable. The value of this element must be in the following format: $[\d]+[\d.]+[MS|ms|S|s|M|m|H|h|D|d]?$ where the first non-digits (from left to right) indicate the unit of time duration:  
  - MS or ms (milliseconds)  
  - S or s (seconds)  
  - M or m (minutes)  
  - H or h (hours)  
  - D or d (days)  
  If the value does not contain a unit, a unit of milliseconds is assumed. The default value is the value of the request-timeout element. |
| `<request-timeout>`     | Optional          | Specifies the maximum amount of time to wait for a response message before declaring the underlying connection unusable. The value of this element must be in the following format: $[\d]+[\d.]+[MS|ms|S|s|M|m|H|h|D|d]?$ where the first non-digits (from left to right) indicate the unit of time duration:  
  - MS or ms (milliseconds)  
  - S or s (seconds)  
  - M or m (minutes)  
  - H or h (hours)  
  - D or d (days)  
  If the value does not contain a unit, a unit of milliseconds is assumed. The default value is an infinite timeout. |
overflow-scheme

Used in: caching-schemes, distributed-scheme, replicated-scheme, optimistic-scheme, read-write-backing-map-scheme, versioned-backing-map-scheme.

Description

The overflow-scheme defines a two-tier cache consisting of a fast, size limited front-tier, and slower but much higher capacity back-tier cache. When the size limited front fills up, evicted entries are transparently moved to the back. In the event of a cache miss, entries may move from the back to the front. A typical deployment might use a local-scheme for the front-tier, and an external-scheme for the back-tier, allowing for fast local caches with capacities larger the JVM heap would allow. In such a deployment the local-scheme element’s high-units and eviction-policy will control the transfer (eviction) of entries from the front to back caches.

---

**Note:** Relying on overflow for normal cache storage is not recommended. It should only be used to help avoid eviction-related data loss in the case where the storage requirements temporarily exceed the configured capacity. In general, the overflow’s on disk storage should remain empty.

Implementation

Implemented by either com.tangosol.net.cache.OverflowMap or com.tangosol.net.cache.SimpleOverflowMap, see expiry-enabled for details.

Entry Expiration

Overflow supports automatic expiration of entries based on the age of the value, as configured by the expiry-delay (see <expiry-delay> subelement).

Elements

Table D–37 describes the elements you can define within the overflow-scheme element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;scheme-name&gt;</td>
<td>Optional</td>
<td>Specifies the scheme’s name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See &quot;Scheme Inheritance&quot; on page D-20 for more information.</td>
</tr>
<tr>
<td>&lt;class-name&gt;</td>
<td>Optional</td>
<td>Specifies a custom implementation of the overflow cache. Any custom implementation must extend either the com.tangosol.net.cache.OverflowMap or com.tangosol.net.cache.SimpleOverflowMap class, and declare the exact same set of public constructors.</td>
</tr>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Specifies initialization parameters, for use in custom overflow cache implementations which implement the com.tangosol.run.xml.XmlConfigurable interface.</td>
</tr>
<tr>
<td>&lt;listener&gt;</td>
<td>Optional</td>
<td>Specifies an implementation of a com.tangosol.util.MapListener which will be notified of events occurring on the cache.</td>
</tr>
</tbody>
</table>
### Table D–37  (Cont.) overflow-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;front-scheme&gt;</td>
<td>Required</td>
<td>Specifies the cache-scheme to use in creating the front-tier cache. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-  local-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-  external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-  paged-external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-  class-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The eviction policy of the front-scheme defines which entries which items are in the front versus back tiers. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;front-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;local-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;eviction-policy&gt;HYBRID&lt;/eviction-policy&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;high-units&gt;1000&lt;/high-units&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/local-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/front-scheme&gt;</td>
</tr>
<tr>
<td>&lt;back-scheme&gt;</td>
<td>Required</td>
<td>Specifies the cache-scheme to use in creating the back-tier cache. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-  local-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-  external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-  paged-external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-  class-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For Example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;back-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;external-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;lh-file-manager/&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/external-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/back-scheme&gt;</td>
</tr>
<tr>
<td>&lt;miss-cache-scheme&gt;</td>
<td>Optional</td>
<td>Specifies a cache-scheme for maintaining information on cache misses. For caches which are not expiry-enabled (see &lt;expiry-enabled&gt; subelement), the miss-cache is used track keys which resulted in both a front and back tier cache miss. The knowledge that a key is not in either tier allows some operations to perform faster, as they can avoid querying the potentially slow back-tier. A size limited scheme may be used to control how many misses are tracked. If unspecified no cache-miss data will be maintained. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-  local-scheme</td>
</tr>
</tbody>
</table>
### Table D–37 (Cont.) overflow-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;expiry-enabled&gt;</code></td>
<td>Optional</td>
<td>Turns on support for automatically-expiring data, as provided by the com.tangosol.net.cache.CacheMap API. When enabled the overflow-scheme will be implemented using com.tangosol.net.cache.OverflowMap, rather than com.tangosol.net.cache.SimpleOverflowMap. Legal values are true or false. Default value is false.</td>
</tr>
</tbody>
</table>
| `<expiry-delay>`  | Optional          | Specifies the amount of time from last update that entries will be kept by the cache before being expired. Entries that are expired will not be accessible and will be evicted. The value of this element must be in the following format: 

\[ \text{duration} = \text{days}\cdot\text{days} + \text{seconds}\cdot\text{seconds} + \text{minutes}\cdot\text{minutes} \] 

where the first non-digits (from left to right) indicate the unit of time duration:

- MS or ms (milliseconds)
- S or s (seconds)
- M or m (minutes)
- H or h (hours)
- D or d (days)

If the value does not contain a unit, a unit of seconds is assumed. A value of zero implies no expiry. Default value is zero.                                                   |
| `<autostart>`     | Optional          | The autostart element is intended to be used by cache servers (that is, com.tangosol.net.DefaultCacheServer). It specifies whether the cache services associated with this cache scheme should be automatically started at a cluster node. Legal values are true or false. Default value is false. |
paged-external-scheme

Used in: caching-schemes, distributed-scheme, replicated-scheme, optimistic-scheme, near-scheme, versioned-near-scheme, overflow-scheme, read-write-backing-map-scheme, versioned-backing-map-scheme

Description

As with external-scheme, paged-external-schemes define caches which are not JVM heap based, allowing for greater storage capacity. The paged-external-scheme optimizes LRU eviction by using a paging approach (see <paging> subelement). See Chapter 11, "Serialization Paged Cache," for a detailed description of the paged cache functionality.

Implementation

This scheme is implemented by the com.tangosol.net.cache.SerializationPagedCache class.

Paging

Cache entries are maintained over a series of pages, where each page is a separate com.tangosol.io.BinaryStore, obtained from the configured storage manager (see 'Pluggable Storage Manager'). When a page is created it is considered to be the "current" page, and all write operations are performed against this page. On a configurable interval (see <page-duration> subelement) the current page is closed and a new current page is created. Read operations for a given key are performed against the last page in which the key was stored. When the number of pages exceeds a configured maximum (see <page-limit> subelement), the oldest page is destroyed and those items which were not updated since the page was closed are be evicted. For example configuring a cache with a duration of ten minutes per page, and a maximum of six pages, will result in entries being cached for at most an hour. Paging improves performance by avoiding individual delete operations against the storage manager as cache entries are removed or evicted. Instead the cache simply releases its references to those entries, and relies on the eventual destruction of an entire page to free the associated storage of all page entries in a single stroke.

Pluggable Storage Manager

External schemes use a pluggable store manager to create and destroy pages, and to access entries within those pages. Supported store-managers include:

- async-store-manager—a wrapper providing asynchronous write capabilities for of other store-manager implementations
- custom-store-manager—allows definition of custom implementations of store-managers
- bdb-store-manager—uses Berkeley Database JE to implement an on disk cache
- lh-file-manager—uses a Coherence LH on disk database cache
- nio-file-manager—uses NIO to implement memory-mapped file based cache
- nio-memory-manager—uses NIO to implement an off JVM heap, in-memory cache
Persistence (long-term storage)

Paged external caches are used for temporary storage of large data sets, for example as the back-tier of an overflow-scheme. These caches are not usable as for long-term storage (persistence), and will not survive beyond the life of the JVM. Clustered persistence should be configured by using a read-write-backing-map-scheme on a distributed-scheme. If a non-clustered persistent cache is what is needed, refer to "Persistence (long-term storage)" on page D-34.

Elements

Table D–38 describes the elements you can define within the paged-external-scheme element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;scheme-name&gt;</code></td>
<td>Optional</td>
<td>Specifies the scheme’s name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td><code>&lt;scheme-ref&gt;</code></td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See &quot;Scheme Inheritance&quot; on page D-20 for more information.</td>
</tr>
<tr>
<td><code>&lt;class-name&gt;</code></td>
<td>Optional</td>
<td>Specifies a custom implementation of the external paged cache. Any custom implementation must extend the com.tangosol.net.cache.SerializationPagedCache class and declare the exact same set of public constructors.</td>
</tr>
<tr>
<td><code>&lt;init-params&gt;</code></td>
<td>Optional</td>
<td>Specifies initialization parameters, for use in custom external paged cache implementations which implement the com.tangosol.run.xml.XmlConfigurable interface.</td>
</tr>
<tr>
<td><code>&lt;listener&gt;</code></td>
<td>Optional</td>
<td>Specifies an implementation of a com.tangosol.util.MapListener which will be notified of events occurring on the cache.</td>
</tr>
<tr>
<td><code>&lt;page-limit&gt;</code></td>
<td>Required</td>
<td>Specifies the maximum number of active pages for the paged cache. Legal values are positive integers between 2 and 3600.</td>
</tr>
<tr>
<td><code>&lt;page-duration&gt;</code></td>
<td>Optional</td>
<td>Specifies the length of time, in seconds, that a page in the paged cache is current. The value of this element must be in the following format: [d]+{.}[d]+?{MS</td>
</tr>
<tr>
<td><code>&lt;async-store-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the paged external cache to use an asynchronous storage manager wrapper for any other storage manager. See &quot;Pluggable Storage Manager&quot; on page D-33 for more information.</td>
</tr>
<tr>
<td><code>&lt;custom-store-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the paged external cache to use a custom storage manager implementation.</td>
</tr>
</tbody>
</table>
### Table D–38  (Cont.) paged-external-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;bdb-store-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the paged external cache to use Berkeley Database JE on disk databases for cache storage.</td>
</tr>
<tr>
<td><code>&lt;lh-file-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the paged external cache to use a Coherence LH on disk database for cache storage.</td>
</tr>
<tr>
<td><code>&lt;nio-file-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the paged external cache to use a memory-mapped file for cache storage.</td>
</tr>
<tr>
<td><code>&lt;nio-memory-manager&gt;</code></td>
<td>Optional</td>
<td>Configures the paged external cache to use an off JVM heap, memory region for cache storage.</td>
</tr>
</tbody>
</table>
partition-listener

Used in: distributed-scheme

Description

Specifies an implementation of a com.tangosol.net.partition.PartitionListener interface, which allows receiving partition distribution events.

Elements

Table D–39 describes the elements you can define within the partition-listener element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-name&gt;</td>
<td>Required</td>
<td>The name of a class that implements the PartitionListener interface. This implementation must have a zero-parameter public constructor. Default value is the value specified in the partition-listener/class-name parameter in the tangosol-coherence.xml descriptor. See “DistributedCache Service Parameters” on page I-3 for more information.</td>
</tr>
</tbody>
</table>
partitioned

Used in: backing-map-scheme (within a distributed-scheme only)

Description

The partitioned element specifies whether the enclosed backing map is a PartitionAwareBackingMap. (This element is respected only for backing-map-scheme that is a child of a distributed-scheme.) If set to true, the specific scheme contained in the backing-map-scheme element will be used to configure backing maps for each individual partition of the PartitionAwareBackingMap; otherwise it is used for the entire backing map itself.

The concrete implementations of the PartitionAwareBackingMap interface are:

- com.tangosol.net.partition.ObservableSplittingBackingCache
- com.tangosol.net.partition.PartitionSplittingBackingCache
- com.tangosol.net.partition.ReadWriteSplittingBackingMap

Valid values are true or false. Default value is false.
**proxy-config**

Used in: *proxy-scheme*.

**Description**

The `proxy-config` element specifies the configuration info for the clustered service proxies managed by a proxy service. A service proxy is an intermediary between a remote client (connected to the cluster by using a connection acceptor) and a clustered service used by the remote client.

**Elements**

Table D–40 describes the elements you can define within the `proxy-config` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;cache-service-proxy&gt;</code></td>
<td>Optional</td>
<td>Specifies the configuration info for a cache service proxy managed by the proxy service.</td>
</tr>
<tr>
<td><code>&lt;invocation-service-proxy&gt;</code></td>
<td>Optional</td>
<td>Specifies the configuration info for an invocation service proxy managed by the proxy service.</td>
</tr>
</tbody>
</table>
proxy-scheme

Used in: caching-schemes.

Description

The `proxy-scheme` element contains the configuration info for a clustered service that allows Coherence*Extend clients to connect to the cluster and use clustered services without having to join the cluster.

Elements

Table D–41 describes the subelements you can define within the `proxy-scheme` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;scheme-name&gt;</code></td>
<td>Optional</td>
<td>Specifies the scheme’s name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td><code>&lt;scheme-ref&gt;</code></td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See “Scheme Inheritance” on page D-20 for more information.</td>
</tr>
<tr>
<td><code>&lt;service-name&gt;</code></td>
<td>Optional</td>
<td>Specifies the name of the service.</td>
</tr>
</tbody>
</table>
| `<serializer>`           | Optional          | Specifies the class configuration info for a `com.tangosol.io.Serializer` implementation used by the Proxy service to serialize and deserialize user types. For example, the following configures a `ConfigurablePofContext` that uses the default `coherence-pof-types.xml` configuration file to write objects to and read from a stream: `<serializer>`  
  `<class-name>com.tangosol.io.pof.ConfigurablePofContext</class-name>`  
  `</serializer>` |
| `<task-hung-threshold>`  | Optional          | Specifies the amount of time in milliseconds that a task can execute before it is considered “hung”. Note: a posted task that has not yet started is never considered as hung. This attribute is applied only if the Thread pool is used (the `thread-count` value is positive). Legal values are positive integers or zero (indicating no default timeout). Default value is the value specified in the `tangosol-coherence.xml` descriptor. See the task-hung-threshold parameter in "ProxyService Parameters" on page I-9 for more information. |
| `<task-timeout>`         | Optional          | Specifies the timeout value in milliseconds for requests executing on the service worker threads. This attribute is applied only if the thread pool is used (the `thread-count` value is positive). If zero is specified, the default `service-guardian <timeout-milliseconds>` value is used. Legal values are non-negative integers. Default value is the value specified in the `tangosol-coherence.xml` descriptor. See the task-timeout parameter in "ProxyService Parameters" on page I-9. |
### Table D-41 (Cont.) proxy-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| <request-timeout>    | Optional          | Specifies the maximum amount of time a client will wait for a response before abandoning the original request. The request time is measured on the client side as the time elapsed from the moment a request is sent for execution to the corresponding server node(s) and includes the following:  
  ■ the time it takes to deliver the request to an executing node (server)  
  ■ the interval between the time the task is received and placed into a service queue until the execution starts  
  ■ the task execution time  
  ■ the time it takes to deliver a result back to the client  
  Legal values are positive integers or zero (indicating no default timeout). Default value is the value specified in the `tangosol-coherence.xml` descriptor. See the request-timeout parameter in "DistributedCache Service Parameters" on page I-3 for more information. |
| <thread-count>  | Optional          | Specifies the number of daemon threads used by the service. If zero, all relevant tasks are performed on the service thread. Legal values are positive integers or zero. Default value is the value specified in the `thread-count` parameter of the `tangosol-coherence.xml` descriptor. See "ProxyService Parameters" on page I-9 for more information. |
| <acceptor-config> | Required          | Contains the configuration of the connection acceptor used by the service to accept connections from Coherence*Extend clients and to allow them to use the services offered by the cluster without having to join the cluster.                                                                                                                       |
| <proxy-config>  | Optional          | Contains the configuration of the clustered service proxies managed by this service.                                                                                                                                                                                                                                                                                   |
| <autostart>    | Optional          | The autostart element is intended to be used by cache servers (that is, `com.tangosol.net.DefaultCacheServer`). It specifies whether this service should be automatically started at a cluster node. Legal values are `true` or `false`. Default value is `false`.                                                   |
The read-write-backing-map-scheme defines a backing map which provides a size limited cache of a persistent store. See "Read-Through, Write-Through, Write-Behind, and Refresh-Ahead Caching" in the Oracle Coherence Getting Started Guide for more details.

The read-write-backing-map-scheme is implemented by the com.tangosol.net.cacheReadWriteBackingMap class.

A read write backing map maintains a cache backed by an external persistent cache store (see <cachestore-scheme> subelement), cache misses will read-through to the back-end store to retrieve the data. If a writable store is provided, cache writes will propagate to the cache store as well.

When enabled (see <refreshahead-factor> subelement) the cache will watch for recently accessed entries which are about to expire, and asynchronously reload them from the cache store. This insulates the application from potentially slow reads against the cache store, as items periodically expire.

When enabled (see <write-delay> subelement) the cache will delay writes to the back-end cache store. This allows for the writes to be batched (see <write-batch-factor> subelement) into more efficient update blocks, which occur asynchronously from the client thread.

The following table describes the elements you can define within the read-write-backing-map-scheme element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;scheme-name&gt;</td>
<td>Optional</td>
<td>Specifies the scheme's name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See &quot;Scheme Inheritance&quot; on page D-20 for more information.</td>
</tr>
<tr>
<td>&lt;class-name&gt;</td>
<td>Optional</td>
<td>Specifies a custom implementation of the read write backing map. Any custom implementation must extend the com.tangosol.net.cache.ReadWriteBackingMap class and declare the exact same set of public constructors.</td>
</tr>
</tbody>
</table>
### Table D–42 (Cont.) read-write-backing-map-schema Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Specifies initialization parameters, for use in custom read write backing map implementations which implement the <code>com.tangosol.run.xml.XmlConfigurable</code> interface.</td>
</tr>
<tr>
<td>&lt;listener&gt;</td>
<td>Optional</td>
<td>Specifies an implementation of a <code>com.tangosol.util.MapListener</code> which will be notified of events occurring on the cache.</td>
</tr>
<tr>
<td>&lt;cachestore-scheme&gt;</td>
<td>Optional</td>
<td>Specifies the store to cache. If unspecified the cached data will only reside within the internal cache (see <code>&lt;internal-cache-scheme&gt;</code> subelement), and only reflect operations performed on the cache itself.</td>
</tr>
<tr>
<td>&lt;cachestore-timeout&gt;</td>
<td>Optional</td>
<td>Specifies the timeout interval in milliseconds to use for CacheStore read and write operations. If a CacheStore operation times out, the executing thread is interrupted and may ultimately lead to the termination of the cache service. If <code>0</code> is specified, the default <code>service-guardian &lt;timeout-milliseconds&gt;</code> value is used. Legal values are non-negative integers.</td>
</tr>
<tr>
<td>&lt;internal-cache-scheme&gt;</td>
<td>Required</td>
<td>Specifies a cache-scheme which will be used to cache entries. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ local-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ disk-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ paged-external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ overflow-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ class-scheme</td>
</tr>
<tr>
<td>&lt;miss-cache-scheme&gt;</td>
<td>Optional</td>
<td>Specifies a cache-scheme for maintaining information on cache misses. The miss-cache is used track keys which were not found in the cache store. The knowledge that a key is not in the cache store allows some operations to perform faster, as they can avoid querying the potentially slow cache store. A size-limited scheme may be used to control how many misses are cached. If unspecified no cache-miss data will be maintained. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ local-scheme</td>
</tr>
<tr>
<td>&lt;read-only&gt;</td>
<td>Optional</td>
<td>Specifies if the cache is read only. If <code>true</code> the cache will load data from cachestore for read operations and will not perform any writing to the cachestore when the cache is updated. Legal values are <code>true</code> or <code>false</code>. Default value is <code>false</code>.</td>
</tr>
</tbody>
</table>
**Table D–42 (Cont.) read-write-backing-map-scheme Subelements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<write-delay>`     | Optional          | Specifies the time interval for a write-behind queue to defer asynchronous writes to the cachestore by. The value of this element must be in the following format: `\d+[,.]\d+)?[MS|ms|s|s|M|m|H|h|D|d]?`  
  where the first non-digits (from left to right) indicate the unit of time duration:  
  - MS or ms (milliseconds)  
  - S or s (seconds)  
  - M or m (minutes)  
  - H or h (hours)  
  - D or d (days)  
  If the value does not contain a unit, a unit of seconds is assumed. If zero, synchronous writes to the cachestore (without queuing) will take place, otherwise the writes will be asynchronous and deferred by specified time interval after the last update to the value in the cache. Default is zero. |
| `<write-batch-factor>` | Optional          | The `write-batch-factor` element is used to calculate the "soft-ripe" time for write-behind queue entries. A queue entry is considered to be "ripe" for a write operation if it has been in the write-behind queue for no less than the write-delay interval. The "soft-ripe" time is the point in time before the actual ripe time after which an entry will be included in a batched asynchronous write operation to the CacheStore (along with all other ripe and soft-ripe entries). In other words, a soft-ripe entry is an entry that has been in the write-behind queue for at least the following duration:  
  \[D' = (1.0 - F)*D\]  
  where `D` = write-delay interval  
  `F` = write-batch-factor  
  Conceptually, the write-behind thread uses the following logic when performing a batched update:  
  1. The thread waits for a queued entry to become ripe.  
  2. When an entry becomes ripe, the thread dequeues all ripe and soft-ripe entries in the queue.  
  3. The thread then writes all ripe and soft-ripe entries either by using `store()` (if there is only the single ripe entry) or `storeAll()` (if there are multiple ripe/soft-ripe entries).  
  4. The thread then repeats (1).  
  This element is only applicable if asynchronous writes are enabled (that is, the value of the write-delay element is greater than zero) and the CacheStore implements the `storeAll()` method. The value of the element is expressed as a percentage of the write-delay interval. Legal values are nonnegative doubles less than or equal to 1.0. Default is zero. |
**Table D–42 (Cont.) read-write-backing-map-scheme Subelements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;write-requeue-threshold&gt;</code></td>
<td>Optional</td>
<td>Specifies the maximum size of the write-behind queue for which failed cachestore write operations are requeued. The purpose of this setting is to prevent flooding of the write-behind queue with failed cachestore operations. This can happen in situations where a large number of successive write operations fail. If zero, write-behind requeuing is disabled. Legal values are positive integers or zero. Default is zero.</td>
</tr>
<tr>
<td><code>&lt;refresh-ahead-factor&gt;</code></td>
<td>Optional</td>
<td>The refresh-ahead-factor element is used to calculate the &quot;soft-expiration&quot; time for cache entries. Soft-expiration is the point in time before the actual expiration after which any access request for an entry will schedule an asynchronous load request for the entry. This attribute is only applicable if the internal cache is a <code>local-scheme</code>, configured with the <code>&lt;expiry-delay&gt;</code> subelement. The value is expressed as a percentage of the internal LocalCache expiration interval. If zero, refresh-ahead scheduling will be disabled. If 1.0, then any get operation will immediately trigger an asynchronous reload. Legal values are nonnegative doubles less than or equal to 1.0. Default value is zero.</td>
</tr>
<tr>
<td><code>&lt;rollback-cachestore-failures&gt;</code></td>
<td>Optional</td>
<td>Specifies whether exceptions caught during synchronous cachestore operations are rethrown to the calling thread (possibly over the network to a remote member). If the value of this element is false, an exception caught during a synchronous cachestore operation is logged locally and the internal cache is updated. If the value is true, the exception is rethrown to the calling thread and the internal cache is not changed. If the operation was called within a transactional context, this would have the effect of rolling back the current transaction. Legal values are <code>true</code> or <code>false</code>. Default value is <code>false</code>.</td>
</tr>
</tbody>
</table>
remote-cache-scheme

Used in: cachestore-scheme, caching-schemes, near-scheme.

Description

The remote-cache-scheme element contains the configuration info necessary to use a clustered cache from outside the cluster by using Coherence*Extend.

Elements

The following table describes the elements you can define within the remote-cache-scheme element.

<table>
<thead>
<tr>
<th>Table D–43 remote-cache-scheme Subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>&lt;scheme-name&gt;</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
</tr>
<tr>
<td>&lt;service-name&gt;</td>
</tr>
<tr>
<td>&lt;operation-bundling&gt;</td>
</tr>
<tr>
<td>&lt;initiator-config&gt;</td>
</tr>
</tbody>
</table>
**remote-invocation-scheme**

Used in: caching-schemes

**Description**

The `remote-invocation-scheme` element contains the configuration info necessary to execute tasks within the context of a cluster without having to first join the cluster. This scheme uses Coherence*Extend to connect to the cluster.

**Elements**

Table D–44 describes the elements you can define within the `remote-invocation-scheme` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;scheme-name&gt;</code></td>
<td>Optional</td>
<td>Specifies the scheme's name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td><code>&lt;scheme-ref&gt;</code></td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See &quot;Scheme Inheritance&quot; on page D-20 for more information.</td>
</tr>
<tr>
<td><code>&lt;service-name&gt;</code></td>
<td>Optional</td>
<td>Specifies the name of the service.</td>
</tr>
<tr>
<td><code>&lt;initiator-config&gt;</code></td>
<td>Required</td>
<td>Contains the configuration of the connection initiator used by the service to establish a connection with the cluster.</td>
</tr>
</tbody>
</table>
replicated-scheme

Used in: caching-schemes, near-scheme, versioned-near-scheme, overflow-scheme, versioned-backing-map-scheme

Description

The replicated scheme defines caches which fully replicate all their cache entries on each cluster nodes running the specified service. See "Replicated Cache Service" for a more detailed description of replicated caches.

Clustered Concurrency Control

Replicated caches support cluster wide key-based locking so that data can be modified in a cluster without encountering the classic missing update problem. Note that any operation made without holding an explicit lock is still atomic but there is no guarantee that the value stored in the cache does not change between atomic operations.

Cache Storage (Backning Map)

Storage for the cache is specified by using the backing-map scheme (see <backing-map> subelement). For instance, a replicated cache which uses a local-scheme for its backing map will result in cache entries being stored in-memory.

Elements

Table D–45 describes the elements you can define within the replicated-scheme element.

<table>
<thead>
<tr>
<th>Table D–45</th>
<th>replicated-scheme Subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
<td><strong>Required/Optional</strong></td>
</tr>
<tr>
<td>&lt;scheme-name&gt;</td>
<td>Optional</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
<td>Optional</td>
</tr>
<tr>
<td>&lt;service-name&gt;</td>
<td>Optional</td>
</tr>
<tr>
<td>&lt;listener&gt;</td>
<td>Optional</td>
</tr>
</tbody>
</table>
Table D–45 (Cont.) replicated-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;backing-map-scheme&gt;</td>
<td>Optional</td>
<td>Specifies what type of cache will be used within the cache server to store the entries. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- local-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- paged-external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- overflow-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- class-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To ensure cache coherence, the backing-map of an replicated cache must not use the read-through pattern to load cache entries. Either use a cache-aside pattern from outside the cache service, or switch to the distributed-scheme, which supports read-through clustered caching.</td>
</tr>
<tr>
<td>&lt;standard-lease-milliseconds&gt;</td>
<td>Optional</td>
<td>Specifies the duration of the standard lease in milliseconds. When a lease has aged past this number of milliseconds, the lock will automatically be released. Set this value to zero to specify a lease that never expires. The purpose of this setting is to avoid deadlocks or blocks caused by stuck threads; the value should be set higher than the longest expected lock duration (for example, higher than a transaction timeout). It's also recommended to set this value higher than packet-delivery/timeout-milliseconds value. Legal values are from positive long numbers or zero. Default value is the value specified for packet-delivery/timeout-milliseconds in the tangosol-coherence.xml descriptor. See &quot;ReplicatedCache Service Parameters&quot; on page I-7 for more information.</td>
</tr>
<tr>
<td>&lt;lease-granularity&gt;</td>
<td>Optional</td>
<td>Specifies the lease ownership granularity. Available since release 2.3. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- thread</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- member</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A value of thread means that locks are held by a thread that obtained them and can only be released by that thread. A value of member means that locks are held by a cluster node and any thread running on the cluster node that obtained the lock can release it. Default value is the lease-granularity value specified in the tangosol-coherence.xml descriptor. See &quot;ReplicatedCache Service Parameters&quot; on page I-7 for more information.</td>
</tr>
<tr>
<td>&lt;mobile-issues&gt;</td>
<td>Optional</td>
<td>Specifies whether the lease issues should be transferred to the most recent lock holders. Legal values are true or false. Default value is the mobile-issue value specified in the tangosol-coherence.xml descriptor. See &quot;ReplicatedCache Service Parameters&quot; on page I-7 for more information.</td>
</tr>
<tr>
<td>&lt;autostart&gt;</td>
<td>Optional</td>
<td>The autostart element is intended to be used by cache servers (that is, com.tangosol.net.DefaultCacheServer). It specifies whether the cache services associated with this cache scheme should be automatically started at a cluster node. Legal values are true or false. Default value is false.</td>
</tr>
</tbody>
</table>
tcp-acceptor

Used in: acceptor-config.

Description

The **tcp-acceptor** element specifies the configuration info for a connection acceptor that accepts connections from Coherence*Extend clients over TCP/IP. For additional details and example configurations see Chapter 17, "Configuring and Using Coherence*Extend."

Elements

Table D–46 describes the elements you can define within the **tcp-acceptor** element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<local-address>`        | Required          | Specifies the local address (IP or DNS name) and port that the TCP/IP ServerSocket opened by the connection acceptor will listen on. For example, the following will instruct the connection acceptor to bind the TCP/IP ServerSocket to 192.168.0.2:9099: `<local-address>`  
  `<address>192.168.0.2</address>`  
  `<port>9099</port>`  
  `<reusable>true</reusable>`  
  `</local-address>`  
  The `<reusable>` child element specifies whether a TCP/IP socket can be bound to an address if a previous connection is in a timeout state. When a TCP/IP connection is closed the connection may remain in a timeout state for a period after the connection is closed (typically known as the TIME_WAIT state or 2MSL wait state). For applications using a well known socket address or port it may not be possible to bind a socket to a required address if there is a connection in the timeout state involving the socket address or port. |
| `<keep-alive-enabled>`   | Optional          | Indicates whether keep alive (SO_KEEPALIVE) is enabled on a TCP/IP socket. Valid values are true and false. Keep alive is enabled by default.         |
| `<tcp-delay-enabled>`    | Optional          | Indicates whether TCP delay (Nagle’s algorithm) is enabled on a TCP/IP socket. Valid values are true and false. TCP delay is disabled by default.       |
| `<receive-buffer-size>`  | Optional          | Configures the size of the underlying TCP/IP socket network receive buffer. Increasing the receive buffer size can increase the performance of network I/O for high-volume connections, while decreasing it can help reduce the backlog of incoming data. The value of this element must be in the following format:  
  
  ```markdown
  \[d]+[.]\[d]+?K|k|M|m|G|g|B|b?
  ```  
  
  where the first non-digit (from left to right) indicates the factor with which the preceding decimal value should be multiplied:  
  - K or k (kilo, 210)  
  - M or m (mega, 220)  
  - G or g (giga, 230)  
  If the value does not contain a factor, a factor of one is assumed. Default value is O/S dependent. |
**Table D–46 (Cont.) tcp-acceptor Subelements**

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| <send-buffer-size> | Optional          | Configures the size of the underlying TCP/IP socket network send buffer. The value of this element must be in the following format:  

```
[\d]+\[.\][\d]+?[K|M|G|B]?  
```

where the first non-digit (from left to right) indicates the factor with which the preceding decimal value should be multiplied:
- K or k (kilo, 210)
- M or m (mega, 220)
- G or g (giga, 230)

If the value does not contain a factor, a factor of one is assumed. Default value is O/S dependent.

<table>
<thead>
<tr>
<th>&lt;listen-backlog&gt;</th>
<th>Optional</th>
<th>Configures the size of the TCP/IP server socket backlog queue. Valid values are positive integers. Default value is O/S dependent.</th>
</tr>
</thead>
</table>

| <linger-timeout>   | Optional          | Enables SO_LINGER on a TCP/IP socket with the specified linger time. The value of this element must be in the following format:  

```
[\d]+\[.\][\d]+?[M|S|M|H|D]?  
```

where the first non-digits (from left to right) indicate the unit of time duration:
- MS or ms (milliseconds)
- S or s (seconds)
- M or m (minutes)
- H or h (hours)
- D or d (days)

If the value does not contain a unit, a unit of milliseconds is assumed. Linger is disabled by default.

<table>
<thead>
<tr>
<th>&lt;authorized-hosts&gt;</th>
<th>Optional</th>
<th>A collection of IP addresses of TCP/IP initiator hosts that are allowed to connect to this TCP/IP acceptor.</th>
</tr>
</thead>
</table>
tcp-initiator

Used in: initiator-config.

Description

The tcp-initiator element specifies the configuration info for a connection initiator that enables Coherence*Extend clients to connect to a remote cluster by using TCP/IP. For additional details and example configurations see Chapter 17, "Configuring and Using Coherence*Extend."

Elements

Table D–47 describes the elements you can define within the tcp-initiator element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;local-address&gt;</td>
<td>Optional</td>
<td>Specifies the local address (IP or DNS name) that the TCP/IP socket opened by the connection initiator will be bound to. For example, the following will instruct the connection initiator to bind the TCP/IP socket to the IP address 192.168.0.1: &lt;local-address&gt; &lt;address&gt;192.168.0.1&lt;/address&gt; &lt;/local-address&gt;</td>
</tr>
<tr>
<td>&lt;remote-addresses&gt;</td>
<td>Required</td>
<td>Contains the &lt;socket-address&gt; of one or more TCP/IP connection acceptors. The TCP/IP connection initiator uses this information to establish a TCP/IP connection with a remote cluster. The TCP/IP connection initiator will attempt to connect to the addresses in a random order, until either the list is exhausted or a TCP/IP connection is established. For example, the following will instruct the connection initiator to attempt to connect to 192.168.0.2:9099 and 192.168.0.3:9099 in a random order: &lt;remote-addresses&gt; &lt;socket-address&gt; &lt;address&gt;192.168.0.2&lt;/address&gt; &lt;port&gt;9099&lt;/port&gt; &lt;/socket-address&gt; &lt;socket-address&gt; &lt;address&gt;192.168.0.3&lt;/address&gt; &lt;port&gt;9099&lt;/port&gt; &lt;/socket-address&gt; &lt;/remote-addresses&gt; Alternatively, the set of remote addresses may be specified using a &lt;address-provider&gt; element instead of the list of &lt;socket-address&gt; elements. This approach may be used to implement custom load balancing algorithms and/or dynamic discovery of TCP/IP connection acceptors.</td>
</tr>
<tr>
<td>&lt;keep-alive-enabled&gt;</td>
<td>Optional</td>
<td>Indicates whether keep alive (SO_KEEPALIVE) is enabled on a TCP/IP socket. Valid values are true and false. Keep alive is enabled by default.</td>
</tr>
<tr>
<td>&lt;tcp-delay-enabled&gt;</td>
<td>Optional</td>
<td>Indicates whether TCP delay (Nagle’s algorithm) is enabled on a TCP/IP socket. Valid values are true and false. TCP delay is disabled by default.</td>
</tr>
</tbody>
</table>
optional Configures the size of the underlying TCP/IP socket network receive buffer. Increasing the receive buffer size can increase the performance of network I/O for high-volume connections, while decreasing it can help reduce the backlog of incoming data. The value of this element must be in the following format:

```plaintext
[\d]+[ .][\d]+?\[K|M|G]\[B|b]\?
```

where the first non-digit (from left to right) indicates the factor with which the preceding decimal value should be multiplied:

- K or k (kilo, 210)
- M or m (mega, 220)
- G or g (giga, 230)

If the value does not contain a factor, a factor of one is assumed. Default value is O/S dependent.
<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;send-buffer-size&gt;</td>
<td>Optional</td>
<td>Configures the size of the underlying TCP/IP socket network send buffer. The value of this element must be in the following format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|                       |                   | \[d]+\[.\]d+?\[k|K|m|M|g|G|b|B\]?
|                       |                   | where the first non-digit (from left to right) indicates the factor with which the preceding decimal value should be multiplied:               |
|                       |                   | ■ K or k (kilo, 210)                                                                                                                        |
|                       |                   | ■ M or m (mega, 220)                                                                                                                         |
|                       |                   | ■ G or g (giga, 230)                                                                                                                         |
|                       |                   | If the value does not contain a factor, a factor of one is assumed. Default value is O/S dependent.                                            |
| <connect-timeout>     | Optional          | Specifies the maximum amount of time to wait while establishing a connection with a connection acceptor. The value of this element must be in the |
|                       |                   | following format:                                                                     |
|                       |                   | \[d]+\[.\]d+?\[ms|Ms|s|S|m|M|h|H|d|D\]?
|                       |                   | where the first non-digits (from left to right) indicate the unit of time duration:                                                         |
|                       |                   | ■ MS or ms (milliseconds)                                                               |
|                       |                   | ■ S or s (seconds)                                                                     |
|                       |                   | ■ M or m (minutes)                                                                     |
|                       |                   | ■ H or h (hours)                                                                       |
|                       |                   | ■ D or d (days)                                                                        |
|                       |                   | If the value does not contain a unit, a unit of milliseconds is assumed. Default value is an infinite timeout.                                |
| <linger-timeout>      | Optional          | Enables \texttt{SO\_LINGER} on a TCP/IP socket with the specified linger time. The value of this element must be in the following format:      |
|                       |                   | \[d]+\[.\]d+?\[ms|Ms|s|S|m|M|h|H|d|D\]?
|                       |                   | where the first non-digits (from left to right) indicate the unit of time duration:                                                         |
|                       |                   | ■ MS or ms (milliseconds)                                                               |
|                       |                   | ■ S or s (seconds)                                                                     |
|                       |                   | ■ M or m (minutes)                                                                     |
|                       |                   | ■ H or h (hours)                                                                       |
|                       |                   | ■ D or d (days)                                                                        |
|                       |                   | If the value does not contain a unit, a unit of milliseconds is assumed. Linger is disabled by default.                                    |
version-persistent-scheme

Used in: versioned-backing-map-scheme.

Description

The version-persistent-scheme defines a cache for storing object versioning information in a clustered cache. Specifying a size limit on the specified scheme's backing-map allows control over how many version identifiers are tracked.

Elements

Table D–48 describes the elements you can define within the version-persistent-scheme element.

<table>
<thead>
<tr>
<th>Table D–48 persistent-scheme Subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>&lt;cache-name-suffix&gt;</td>
</tr>
<tr>
<td>&lt;replicated-scheme&gt; or &lt;distributed-scheme&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
version-transient-scheme

Used in: versioned-near-scheme, versioned-backing-map-scheme.

Description

The version-transient-scheme defines a cache for storing object versioning information for use in versioned near-caches. Specifying a size limit on the specified scheme's backing-map allows control over how many version identifiers are tracked.

Elements

The following table describes the elements you can define within the version-transient-scheme element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;cache-name-suffix&gt;</td>
<td>Optional</td>
<td>Specifies the name modifier that is used to create a cache of version objects associated with a given cache. The value of this element is appended to the base cache name. Legal value is a string. Default value is &quot;-version&quot;. For example, if the base case is named Sessions and this name modifier is set to -version, the associated version cache will be named Sessions-version.</td>
</tr>
</tbody>
</table>
| <replicated-scheme> or <distributed-scheme> | Required          | Specifies the scheme for the cache used to maintain the versioning information. Legal values are:  
  * replicated-scheme
  * distributed-scheme |
**versioned-backing-map-scheme**

Used in: `caching-schemes`, `distributed-scheme`, `replicated-scheme`, `optimistic-scheme`.

**Description**

The `versioned-backing-map-scheme` is an extension of a `read-write-backing-map-scheme`, defining a size limited cache of a persistent store. It uses object versioning to determine what updates need to be written to the persistent store. See "Versioning" for more information.

**Implementation**

The `versioned-backing-map-scheme` scheme is implemented by the `com.tangosol.net.cache.VersionedBackingMap` class.

**Cache of an External Store**

As with the `read-write-backing-map-scheme`, a versioned backing map maintains a cache backed by an external persistent cache store (see `<cachestore-scheme>` subelement), cache misses will read-through to the back-end store to retrieve the data. Cache stores may also support updates to the back-end data store.

**Refresh-Ahead and Write-Behind Caching**

As with the `read-write-backing-map-scheme` both the refresh-ahead (see `<refresh-ahead>` subelement) and write-behind (see `<write-behind>` subelement) caching optimizations are supported. See "Read-Through, Write-Through, Write-Behind, and Refresh-Ahead Caching" in the Oracle Coherence Getting Started Guide for more details.

**Versioning**

For entries whose values implement the `com.tangosol.util.Versionable` interface, the versioned backing map will use the version identifier to determine if an update must be written to the persistent store. The primary benefit of this feature is that in the event of cluster node failover, the backup node can determine if the most recent version of an entry has already been written to the persistent store, and if so it can avoid an extraneous write.

**Elements**

Table D–50 describes the elements you can define within the `versioned-backing-map-scheme` element.
### Table D–50  versioned-backing-map-scheme Subelement

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;scheme-name&gt;</td>
<td>Optional</td>
<td>Specifies the scheme's name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See &quot;Scheme Inheritance&quot; on page D-20 for more information.</td>
</tr>
<tr>
<td>&lt;class-name&gt;</td>
<td>Optional</td>
<td>Specifies a custom implementation of the versioned backing map. Any custom implementation must extend the <code>com.tangosol.net.cache.VersionedBackingMap</code> class and declare the exact same set of public constructors.</td>
</tr>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Specifies initialization parameters, for use in custom versioned backing map implementations which implement the <code>com.tangosol.run.xml.XmlConfigurable</code> interface.</td>
</tr>
<tr>
<td>&lt;listener&gt;</td>
<td>Optional</td>
<td>Specifies an implementation of a <code>com.tangosol.util.MapListener</code> which will be notified of events occurring on the cache.</td>
</tr>
<tr>
<td>&lt;cachestore-scheme&gt;</td>
<td>Optional</td>
<td>Specifies the store to cache. If unspecified the cached data will only reside within the (see &lt;internal-cache-scheme&gt; subelement), and only reflect operations performed on the cache itself.</td>
</tr>
<tr>
<td>&lt;internal-cache-scheme&gt;</td>
<td>Required</td>
<td>Specifies a cache-scheme which will be used to cache entries. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- local-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- paged-external-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- overflow-scheme</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- class-scheme</td>
</tr>
<tr>
<td>&lt;miss-cache-scheme&gt;</td>
<td>Optional</td>
<td>Specifies a cache-scheme for maintaining information on cache misses. The miss-cache is used track keys which were not found in the cache store. The knowledge that a key is not in the cache store allows some operations to perform faster, as they can avoid querying the potentially slow cache store. A size-limited scheme may be used to control how many misses are cached. If unspecified no cache-miss data will be maintained. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- local-scheme</td>
</tr>
<tr>
<td>&lt;read-only&gt;</td>
<td>Optional</td>
<td>Specifies if the cache is read only. If true the cache will load data from cachestore for read operations and will not perform any writing to the cachestore when the cache is updated. Legal values are <code>true</code> or <code>false</code>. Default value is <code>false</code>.</td>
</tr>
</tbody>
</table>
Table D–50 (Cont.) versioned-backing-map-scheme Subelement

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| <write-delay>                | Optional          | Specifies the time interval for a write-behind queue to defer asynchronous writes to the cachestore. The value of this element must be in the following format: \([\d]+\[\.[\d]+\]? \([\text{MS|ms|S|s|M|m|H|h|D|d]}\) \?
where the first non-digits (from left to right) indicate the unit of time duration:
  - MS or ms (milliseconds)
  - S or s (seconds)
  - M or m (minutes)
  - H or h (hours)
  - D or d (days)
If the value does not contain a unit, a unit of seconds is assumed. If zero, synchronous writes to the cachestore (without queuing) will take place, otherwise the writes will be asynchronous and deferred by the number of seconds after the last update to the value in the cache. Default is zero.

| <write-batch-factor>        | Optional          | The write-batch-factor element is used to calculate the "soft-ripe" time for write-behind queue entries. A queue entry is considered to be "ripe" for a write operation if it has been in the write-behind queue for no less than the write-delay interval. The "soft-ripe" time is the point in time before the actual "ripe" time after which an entry will be included in a batched asynchronous write operation to the CacheStore (along with all other "ripe" and "soft-ripe" entries). This element is only applicable if asynchronous writes are enabled (that is, the value of the write-delay element is greater than zero) and the CacheStore implements the storeAll() method. The value of the element is expressed as a percentage of the write-delay interval. For example, if the value is zero, only "ripe" entries from the write-behind queue will be batched. On the other hand, if the value is 1.0, all currently queued entries will be batched and the value of the write-delay element will be effectively ignored. Legal values are nonnegative doubles less than or equal to 1.0. Default is zero.

| <write-requeue-threshold>   | Optional          | Specifies the maximum size of the write-behind queue for which failed cachestore write operations are requeued. The purpose of this setting is to prevent flooding of the write-behind queue with failed cachestore operations. This can happen in situations where a large number of successive write operations fail. If zero, write-behind requeuing is disabled. Legal values are positive integers or zero. Default is zero.

| <refresh-ahead-factor>      | Optional          | The refresh-ahead-factor element is used to calculate the "soft-expiration" time for cache entries. Soft-expiration is the point in time before the actual expiration after which any access request for an entry will schedule an asynchronous load request for the entry. This attribute is only applicable if the internal cache (see <internal-cache-scheme> subelement) is a local-scheme, configured with the <location> subelement. The value is expressed as a percentage of the internal LocalCache expiration interval. If zero, refresh-ahead scheduling will be disabled. If 1.0, then any get operation will immediately trigger an asynchronous reload. Legal values are nonnegative doubles less than or equal to 1.0. Default value is zero.
### Table D–50 (Cont.) versioned-backing-map-scheme Subelement

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;rollback-cachestore-failures&gt;</td>
<td>Optional</td>
<td>Specifies whether exceptions caught during synchronous cachestore operations are rethrown to the calling thread (possibly over the network to a remote member). If the value of this element is false, an exception caught during a synchronous cachestore operation is logged locally and the internal cache is updated. If the value is true, the exception is rethrown to the calling thread and the internal cache is not changed. If the operation was called within a transactional context, this would have the effect of rolling back the current transaction. Legal values are true or false. Default value is false.</td>
</tr>
<tr>
<td>&lt;version-persistent-scheme&gt;</td>
<td>Optional</td>
<td>Specifies a cache-scheme for tracking the version identifier for entries in the persistent cachestore (see cachestore-scheme).</td>
</tr>
<tr>
<td>&lt;version-transient-scheme&gt;</td>
<td>Optional</td>
<td>Specifies a cache-scheme for tracking the version identifier for entries in the transient internal cache (see &lt;internal-cache-scheme&gt; subelement).</td>
</tr>
<tr>
<td>&lt;manage-transient&gt;</td>
<td>Optional</td>
<td>Specifies if the backing map is responsible for keeping the transient version cache up to date. If disabled the backing map manages the transient version cache only for operations for which no other party is aware (such as entry expiry). This is used when there is already a transient version cache of the same name being maintained at a higher level, for instance within a versioned-near-scheme. Legal values are true or false. Default value is false.</td>
</tr>
</tbody>
</table>
versioned-near-scheme

Used in: caching-schemes.

**Note:** As of Coherence release 2.3, it is suggested that a near-scheme be used instead of versioned-near-scheme. Legacy Coherence applications use versioned-near-scheme to ensure Coherence through object versioning. As of Coherence 2.3 the near-scheme includes a better alternative, in the form of reliable and efficient front cache invalidation.

**Description**

As with the near-scheme, the versioned-near-scheme defines a two tier cache consisting of a small and fast front-end, and higher-capacity but slower back-end cache. The front-end (see <front-end> subelement) and back-end (see <back-end> subelement) are expressed as normal cache-schemes. A typical deployment might use a local-scheme for the front-end, and a distributed-scheme for the back-end. See Appendix B, "Types of Caches in Coherence" for a more detailed description of versioned near caches.

**Implementation**

The versioned near scheme is implemented by the `com.tangosol.net.cache.VersionedNearCache` class.

**Versioning**

Object versioning is used to ensure coherence between the front and back tiers. See the <version-transient-scheme> subelement for more information

**Elements**

Table D–51 describes the elements you can define within the near-scheme element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;scheme-name&gt;</td>
<td>Optional</td>
<td>Specifies the scheme's name. The name must be unique within a configuration file.</td>
</tr>
<tr>
<td>&lt;scheme-ref&gt;</td>
<td>Optional</td>
<td>Specifies the name of another scheme to inherit from. See &quot;Scheme Inheritance&quot; on page D-20 for more information.</td>
</tr>
<tr>
<td>&lt;class-name&gt;</td>
<td>Optional</td>
<td>Specifies a custom implementation of the versioned near cache. The specified class must extend the <code>com.tangosol.net.cache.VersionedNearCache</code> class and declare the exact same set of public constructors.</td>
</tr>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Specifies initialization parameters, for use in custom versioned near cache implementations which implement the <code>com.tangosol.run.xml.XmlConfigurable</code> interface.</td>
</tr>
<tr>
<td>&lt;listener&gt;</td>
<td>Optional</td>
<td>Specifies an implementation of a <code>com.tangosol.util.MapListener</code> which will be notified of events occurring on the cache.</td>
</tr>
</tbody>
</table>
Table D–51  (Cont.) near-scheme Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| <front-scheme>        | Required          | Specifies the cache-scheme to use in creating the front-tier cache. Legal values are:  
                       |                   |  ■ local-scheme  
                       |                   |  ■ external-scheme  
                       |                   |  ■ paged-external-scheme  
                       |                   |  ■ class-scheme  
                       |                   | For example:  
                       |                   |   <front-scheme>  
                       |                   |     <local-scheme>  
                       |                   |       <scheme-ref>default-eviction</scheme-ref>  
                       |                   |     </local-scheme>  
                       |                   |     </front-scheme>  
                       |                   | or  
                       |                   |   <front-scheme>  
                       |                   |     <class-scheme>  
                       |                   |       <class-name>com.tangosol.util.SafeHashMap</class-name>  
                       |                   |       <init-params></init-params>  
                       |                   |     </class-scheme>  
                       |                   |     </front-scheme>  
| <back-scheme>         | Required          | Specifies the cache-scheme to use in creating the back-tier cache. Legal values are:  
                       |                   |  ■ distributed-scheme  
                       |                   |  ■ replicated-scheme  
                       |                   |  ■ optimistic-scheme  
                       |                   |  ■ local-scheme  
                       |                   |  ■ external-scheme  
                       |                   |  ■ paged-external-scheme  
                       |                   |  ■ class-scheme  
                       |                   | For example:  
                       |                   |   <back-scheme>  
                       |                   |     <distributed-scheme>  
                       |                   |       <scheme-ref>default-distributed</scheme-ref>  
                       |                   |     </distributed-scheme>  
                       |                   |     </back-scheme>  
| <version-transient-scheme> | Optional          | Specifies a scheme for versioning cache entries, which ensures coherence between the front and back tiers.  
| <autostart>           | Optional          | The autostart element is intended to be used by cache servers (that is, com.tangosol.net.DefaultCacheServer). It specifies whether the cache services associated with this cache scheme should be automatically started at a cluster node. Legal values are true or false. Default value is false.
Cache Configuration Parameter Macros

The Cache Configuration Deployment Descriptor (coherence-cache-config.xml) supports parameter macros to minimize custom coding and enable specification of commonly used attributes when configuring class constructor parameters. The macros should be entered enclosed in curly braces as shown below, without any quotes or spaces.

Table E–1 describes the parameter macros that may be specified:

<table>
<thead>
<tr>
<th><code>&lt;param-type&gt;</code></th>
<th><code>&lt;param-value&gt;</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang.String</td>
<td>{cache-name}</td>
<td>Used to pass the current cache name as a constructor parameter. For example:</td>
</tr>
<tr>
<td>(cache-name)</td>
<td></td>
<td>`&lt;class-name&gt;com.mycompany.cache.CustomCacheLoader&lt;/class-name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>`&lt;init-params&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>`&lt;init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>`&lt;param-type&gt;java.lang.String&lt;/param-type&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;param-value&gt;{cache-name}&lt;/param-value&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-params&gt;</td>
</tr>
<tr>
<td>java.lang.ClassLoader</td>
<td>{class-loader}</td>
<td>Used to pass the current classloader as a constructor parameter. For example:</td>
</tr>
<tr>
<td>(class-loader)</td>
<td></td>
<td>`&lt;class-name&gt;com.mycompany.cache.CustomCacheLoader&lt;/class-name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>`&lt;init-params&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>`&lt;init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>`&lt;param-type&gt;java.lang.ClassLoader&lt;/param-type&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;param-value&gt;{class-loader}&lt;/param-value&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-params&gt;</td>
</tr>
</tbody>
</table>
Table E–1  (Cont.) Parameter Macros for Cache Configuration

<table>
<thead>
<tr>
<th>&lt;param-type&gt;</th>
<th>&lt;param-value&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.tangosol.net. BackingMapManager Context</td>
<td>{manager-context}</td>
<td>Used to pass the current BackingMapManagerContext object as a constructor parameter. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;class-name&gt;com.mycompany.cache.CustomCacheLoader&lt;/class-name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;init-params&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-type&gt;com.tangosol.net.BackingMapManagerContext&lt;/param-type&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-value&gt;{manager-context}&lt;/param-value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-params&gt;</td>
</tr>
<tr>
<td>(scheme-ref)</td>
<td>local-scheme</td>
<td>Instantiates an object defined by the &lt;class-scheme&gt;, &lt;local-scheme&gt; or &lt;file-scheme&gt; with the specified &lt;scheme-name&gt; value and uses it as a constructor parameter. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;class-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;scheme-name&gt;dbconnection&lt;/scheme-name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;class-name&gt;com.mycompany.dbConnection&lt;/class-name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;init-params&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-name&gt;driver&lt;/param-name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-type&gt;String&lt;/param-type&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-value&gt;org.gjt.mm.mysql.Driver&lt;/param-value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-name&gt;url&lt;/param-name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-type&gt;String&lt;/param-type&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-value&gt;jdbc:mysql://dbserver:3306/companydb&lt;/param-value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-name&gt;user&lt;/param-name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-type&gt;String&lt;/param-type&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-value&gt;default&lt;/param-value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-name&gt;password&lt;/param-name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-type&gt;String&lt;/param-type&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-value&gt;default&lt;/param-value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-params&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/class-scheme&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;class-name&gt;com.mycompany.cache.CustomCacheLoader&lt;/class-name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;init-params&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-type&gt;{scheme-ref}&lt;/param-type&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;param-value&gt;dbconnection&lt;/param-value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-param&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/init-params&gt;</td>
</tr>
</tbody>
</table>
Consider the following configuration example:

```xml
<cache-config>
  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>boston-*</cache-name>
      <scheme-name>wrapper</scheme-name>
      <init-params>
        <init-param>
          <param-name>delegate-cache-name</param-name>
          <param-value>london-*</param-value>
        </init-param>
      </init-params>
    </cache-mapping>
    <cache-mapping>
      <cache-name>london-*</cache-name>
      <scheme-name>partitioned</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>
  <caching-schemes>
    <class-scheme>
      <scheme-name>wrapper</scheme-name>
      <class-name>com.tangosol.net.cache.WrapperNamedCache</class-name>
      <init-params>
        <init-param>
          <param-type>cache-ref</param-type>
          <param-value>delegate-cache-name</param-value>
        </init-param>
        <init-param>
          <param-type>string</param-type>
          <param-value>cache-name</param-value>
        </init-param>
      </init-params>
    </class-scheme>
    <distributed-scheme>
      <scheme-name>partitioned</scheme-name>
      <service-name>partitioned</service-name>
      <backing-map-scheme>
        <local-scheme>
          <unit-calculator>BINARY</unit-calculator>
        </local-scheme>
      </backing-map-scheme>
      <autostart>true</autostart>
    </distributed-scheme>
  </caching-schemes>
</cache-config>
```

The `CacheFactory.getCache("london-test")` call would result in a standard partitioned cache reference. Conversely, the `CacheFactory.getCache("boston-test")` call would resolve the value of the `delegate-cache-name` parameter to `london-test` and would construct an instance of the `WrapperNamedCache` delegating to the `NamedCache` returned by the `CacheFactory.getCache("london-test")` call.

<table>
<thead>
<tr>
<th>Parameter Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(cache-ref)</td>
<td>cache name</td>
</tr>
</tbody>
</table>

Table E–1 (Cont.) Parameter Macros for Cache Configuration
Sample Cache Configurations

This section provides a series of simple cache scheme configurations. The samples build upon one another and will often use a scheme-ref element to reuse other samples as nested schemes. See "Scheme Inheritance" on page D-20 for an example of <scheme-ref>.

Cache schemes are specified in the caching-schemes element of the cache configuration descriptor coherence-cache-config.xml which is described in Appendix D, "Cache Configuration Elements". These samples only specify a minimum number of settings, follow the embedded links to the scheme’s documentation to see the full set of options.

This section describes configurations for the following caching scenarios:

- **Local Caches (accessible from a single JVM)**
  - In-memory Cache
  - NIO In-memory Cache
  - Size Limited In-memory Cache
  - In-memory Cache with Expiring Entries
  - Cache on Disk
  - Size Limited Cache on Disk
  - Persistent Cache on Disk
  - In-memory Cache with Disk Based Overflow
  - Cache of a Database

- **Clustered Caches (accessible from multiple JVMs)**
  - Replicated Cache
  - Replicated Cache with Overflow
  - Partitioned Cache
  - Partitioned Cache with Overflow
  - Partitioned Cache of a Database
  - Partitioned Cache with a Serializer
  - Local Cache of a Partitioned Cache (Near cache)
Local Caches (accessible from a single JVM)

This section defines a series of local cache schemes. In this context "local" means that the cache is only directly accessible by a single JVM. Later in this document local caches will be used as building blocks for clustered caches. See "Clustered Caches (accessible from multiple JVMs)" on page F-5.

In-memory Cache

Example F–1 uses a local-scheme to define an in-memory cache. The cache will store as much as the JVM heap will allow.

Example F–1 Configuration for a Local, In-memory Cache

```xml
<local-scheme>
  <scheme-name>SampleMemoryScheme</scheme-name>
</local-scheme>
```

NIO In-memory Cache

Example F–2 uses an external-scheme to define an in-memory cache using an nio-memory-manager. The advantage of an NIO memory based cache is that it allows for large in-memory cache storage while not negatively impacting the JVM's GC times. The size of the cache is limited by the maximum size of the NIO memory region. See the <maximum-size> subelement of nio-memory-manager.

Example F–2 Configuration for a NIO In-memory Cache

```xml
<external-scheme>
  <scheme-name>SampleNioMemoryScheme</scheme-name>
  <nio-memory-manager/>
</external-scheme>
```

Size Limited In-memory Cache

Adding a <high-units> sub element to <local-scheme> limits the size of the cache. Here the cache is size limited to one thousand entries. When the limit is exceeded, the scheme's <eviction-policy> will determine which elements to evict from the cache.

Example F–3 Configuration for a Size Limited, In-memory, Local Cache

```xml
<local-scheme>
  <scheme-name>SampleMemoryLimitedScheme</scheme-name>
  <high-units>1000</high-units>
</local-scheme>
```

In-memory Cache with Expiring Entries

Adding an <expiry-delay> subelement to <local-scheme> will cause cache entries to automatically expire if they are not updated for a given time interval. When expired the cache will invalidate the entry, and remove it from the cache.

Example F–4 Configuration for an In-memory Cache with Expiring Entries

```xml
<local-scheme>
  <scheme-name>SampleMemoryExpirationScheme</scheme-name>
  <expiry-delay>5m</expiry-delay>
</local-scheme>
```
Local Caches (accessible from a single JVM)

**Sample Cache Configurations**

### Cache on Disk

Example F–5 uses an `external-scheme` to define an on disk cache. The cache will store as much as the file system will allow.

```
<external-scheme>
  <scheme-name>SampleDiskScheme</scheme-name>
  <lh-file-manager/>
</external-scheme>
```

**Note:** This example uses the `lh-file-manager` for its on disk storage implementation. See "external-scheme" on page D-33 for additional external storage options.

#### Example F–5  Configuration to Define a Cache on Disk

```
<external-scheme>
  <scheme-name>SampleDiskScheme</scheme-name>
  <lh-file-manager/>
</external-scheme>
```

### Size Limited Cache on Disk

Adding a `<high-units>` sub-element to `external-scheme` limits the size of the cache. The cache is size limited to one million entries. When the limit is exceeded, LRU eviction is used determine which elements to evict from the cache. Refer to "paged-external-scheme" on page D-68 for an alternate size limited external caching approach.

#### Example F–6  Configuration for a Size Limited Cache on Disk

```
<external-scheme>
  <scheme-name>SampleDiskLimitedScheme</scheme-name>
  <lh-file-manager/>
  <high-units>1000000</high-units>
</external-scheme>
```

### Persistent Cache on Disk

Example F–7 uses an `external-scheme` to implement a cache suitable for use as long-term storage for a single JVM.

External caches are generally used for temporary storage of large data sets, and are automatically deleted on JVM shutdown. An external-cache can be used for long term storage (see "Persistence (long-term storage)" on page D-34) in non-clustered caches when using either the `lh-file-manager` or `bdb-store-manager` storage managers. For clustered persistence see the "Partitioned Cache of a Database" on page F-6 sample.

The `{cache-name}` macro is used to specify the name of the file the data will be stored in. See Appendix E, "Cache Configuration Parameter Macros" for more information on this macro.

#### Example F–7  Configuration for Persistent cache on disk

```
<external-scheme>
  <scheme-name>SampleDiskPersistentScheme</scheme-name>
  <lh-file-manager>
    <directory>/my/storage/directory</directory>
    <file-name>{cache-name}.store</file-name>
  </lh-file-manager>
</external-scheme>
```

Example F–8 illustrates using Berkeley DB rather then LH.
In-memory Cache with Disk Based Overflow

Example F–9 uses an overflow-scheme to define a size limited in-memory cache, when the in-memory (<front-scheme>) size limit is reached, a portion of the cache contents will be moved to the on disk (<back-scheme>). The front-scheme's <eviction-policy> will determine which elements to move from the front to the back.

Note that this example reuses the examples in "Size Limited Cache on Disk" and "Cache on Disk" on page F-3. to implement the front and back of the cache.

Example F–9  Configuration for In-memory Cache with Disk Based Overflow

```
<overflow-scheme>
  <scheme-name>SampleOverflowScheme</scheme-name>
  <front-scheme>
    <local-scheme>
      <scheme-ref>SampleMemoryLimitedScheme</scheme-ref>
    </local-scheme>
  </front-scheme>
  <back-scheme>
    <external-scheme>
      <scheme-ref>SampleDiskScheme</scheme-ref>
    </external-scheme>
  </back-scheme>
</overflow-scheme>
```

Cache of a Database

Example F–10 uses a read-write-backing-map-scheme to define a cache of a database. This scheme maintains local cache of a portion of the database contents. Cache misses will read-through to the database, and cache writes will be written back to the database.

The cachestore-scheme element is configured with a custom class implementing either the com.tangosol.net.cache.CacheLoader or com.tangosol.net.cache.CacheStore interface. This class is responsible for all operations against the database, such as reading and writing cache entries. See Appendix G, "Sample CacheStores" implementations for examples of writing a cache store.

The {cache-name} macro is used to inform the cache store implementation of the name of the cache it will back. See Appendix E, "Cache Configuration Parameter Macros" for more information on this macro.

Example F–10  Configuration for the Cache of a Database

```
<read-write-backing-map-scheme>
  <scheme-name>SampleDatabaseScheme</scheme-name>
  <internal-cache-scheme>
    <local-scheme>
      <scheme-ref>SampleMemoryScheme</scheme-ref>
    </local-scheme>
  </internal-cache-scheme>
</read-write-backing-map-scheme>
```
Clustered Caches (accessible from multiple JVMs)

This section defines a series of clustered cache examples. Clustered caches are accessible from multiple JVMs (any cluster node running the same cache service). The internal cache storage (backing-map) on each cluster node is defined using local caches (see "Local Caches (accessible from a single JVM)" on page F-2). The cache service provides the capability to access local caches from other cluster nodes.

Replicated Cache

Example F–11 uses the replicated-scheme element to define a clustered cache in which a copy of each cache entry will be stored on all cluster nodes.

The sample in "In-memory Cache" on page F-2 is used to define the cache storage on each cluster node. The size of the cache is only limited by the cluster node with the smallest JVM heap.

Example F–11 Configuration for a Replicated Cache

```xml
<replicated-scheme>
  <scheme-name>SampleReplicatedScheme</scheme-name>
  <backing-map-scheme>
    <local-scheme>
      <scheme-ref>SampleMemoryScheme</scheme-ref>
    </local-scheme>
  </backing-map-scheme>
</replicated-scheme>
```

Replicated Cache with Overflow

The backing-map-scheme element could just as easily specify any of the other local cache samples. For instance, if it had used the "In-memory Cache with Disk Based Overflow" on page F-4, each cluster node would have a local overflow cache allowing for much greater storage capacity.

Example F–12 Configuration for a Replicated Cache with Overflow

```xml
<replicated-scheme>
  <scheme-name>SampleReplicatedOverflowScheme</scheme-name>
  <backing-map-scheme>
    <overflow-scheme>
      <scheme-ref>SampleOverflowScheme</scheme-ref>
    </overflow-scheme>
  </backing-map-scheme>
</replicated-scheme>
```
Partitioned Cache

Example F–13 uses the `distributed-scheme` to define a clustered cache in which cache storage is partitioned across all cluster nodes.

The "In-memory Cache" on page F-2 is used to define the cache storage on each cluster node. The total storage capacity of the cache is the sum of all storage enabled cluster nodes running the partitioned cache service. See the `<local-storage>` subelement of "distributed-scheme" on page D-27.

Example F–13 Configuration for a Partitioned Cache

```xml
<distributed-scheme>
  <scheme-name>SamplePartitionedScheme</scheme-name>
  <backing-map-scheme>
    <local-scheme>
      <scheme-ref>SampleMemoryScheme</scheme-ref>
    </local-scheme>
  </backing-map-scheme>
</distributed-scheme>
```

Partitioned Cache with Overflow

The `backing-map-scheme` element could just as easily specify any of the other local cache samples. For instance if it had used the "In-memory Cache with Disk Based Overflow" on page F-4, each storage-enabled cluster node would have a local overflow cache allowing for much greater storage capacity. Note that the cache's backup storage also uses the same overflow scheme which allows for backup data to be overflowed to disk.

Example F–14 Configuration for a Partitioned Cache with Overflow

```xml
<distributed-scheme>
  <scheme-name>SamplePartitionedOverflowScheme</scheme-name>
  <backing-map-scheme>
    <overflow-scheme>
      <scheme-ref>SampleOverflowScheme</scheme-ref>
    </overflow-scheme>
  </backing-map-scheme>
  <backup-storage>
    <type>scheme</type>
    <scheme-name>SampleOverflowScheme</scheme-name>
  </backup-storage>
</distributed-scheme>
```

Partitioned Cache of a Database

Switching the `backing-map-scheme` element to use a `read-write-backing-map-scheme` allows the cache to load and store entries against an external source such as a database.

Example F–15 reuses the "Cache of a Database" on page F-4 to define the database access.

Example F–15 Configuration for a Partitioned Cache of a Database

```xml
<distributed-scheme>
  <scheme-name>SamplePartitionedDatabaseScheme</scheme-name>
```
Partitioned Cache with a Serializer

Example F–16 uses the serializer element in distributed-scheme to define a serializer that will be used to serialize and deserialize user types. In this case, the partitioned cache will use POF (ConfigurablePofContext) as its serialization format. Note that if you use POF and your application uses any custom user type classes, then you must also define a custom POF configuration for them. See Appendix J, "POF User Type Configuration Elements" for more information on POF elements.

Example F–16  Configuration for a Partitioned Cache with a Serializer
<distributed-scheme>
  <scheme-name>SamplePartitionedPofScheme</scheme-name>
  <service-name>PartitionedPofCache</service-name>
  <serializer>
    <class-name>com.tangosol.io.pof.ConfigurablePofContext</class-name>
  </serializer>
  <backing-map-scheme>
    <local-scheme/>
  </backing-map-scheme>
  <autostart>true</autostart>
</distributed-scheme>

Local Cache of a Partitioned Cache (Near cache)

Example F–17 uses the near-scheme to define a local in-memory cache of a subset of a partitioned cache. The result is that any cluster node accessing the partitioned cache will maintain a local copy of the elements it frequently accesses. This offers read performance close to the replicated-scheme-based caches, while offering the high scalability of a distributed-scheme-based cache.

The "Size Limited In-memory Cache" on page F-2 sample is reused to define the "near" (<front-scheme>) cache, while the "Partitioned Cache" on page F-6 sample is reused to define the near-scheme.

Note that the size limited configuration of the front-scheme specifies the limit on how much of the back-scheme cache is locally cached.

Example F–17  Configuration for a Local Cache of a Partitioned Cache
<near-scheme>
  <scheme-name>SampleNearScheme</scheme-name>
  <front-scheme>
    <local-scheme>
      <scheme-ref>SampleLimitedMemoryScheme</scheme-ref>
    </local-scheme>
  </front-scheme>
  <back-scheme>  
    <distributed-scheme>
      <scheme-ref>SamplePartitionedScheme</scheme-ref>
    </distributed-scheme>
  </back-scheme>
</near-scheme>
Clustered Caches (accessible from multiple JVMs)

</near-scheme>
Cache stores are used by caches to read and write cache entries to external stores such as a database. The examples on this page illustrate different ways in which you can interact with a cache store.

**Note:** Save processing effort by bulk loading the cache. The following examples use the `put` method to write values to the cache store. Often, performing bulk loads with the `putAll` method will result in a savings in processing effort and network traffic. For more information on bulk loading, see Chapter 12, "Pre-Loading the Cache."

---

### Sample CacheStore

This section provides a very basic implementation of the `com.tangosol.net.cache.CacheStore` interface. The implementation in **Example G–1** uses a single database connection by using JDBC, and does not use bulk operations. A complete implementation would use a connection pool, and, if write-behind is used, implement `CacheStore.storeAll()` for bulk JDBC inserts and updates. "Cache of a Database" on page F-4 provides an example of a database cache configuration.

**Example G–1 Implementation of the CacheStore Interface**

```java
package com.tangosol.examples.coherence;

import com.tangosol.net.cache.CacheStore;
import com.tangosol.util.Base;
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Collection;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

/**
 * An example implementation of CacheStore
 */
```
public class DBCacheStore
extends Base
implements CacheStore
{
  // ----- constructors ---------------------------------------------------
  /**
   * Constructs DBCacheStore for a given database table.
   *
   * @param sTableName the db table name
   */
  public DBCacheStore(String sTableName)
  {
    m_sTableName = sTableName;
    configureConnection();
  }

  /**
   * Set up the DB connection.
   */
  protected void configureConnection()
  {
    try
    {
      Class.forName("org.gjt.mm.mysql.Driver");
      m_con = DriverManager.getConnection(DB_URL, DB_USERNAME,
      DB_PASSWORD);
      m_con.setAutoCommit(true);
    }
    catch (Exception e)
    {
      throw ensureRuntimeException(e, "Connection failed");
    }
  }

  // ---- accessors -------------------------------------------------------
  /**
   * Obtain the name of the table this CacheStore is persisting to.
   *
   * @return the name of the table this CacheStore is persisting to
   */
  public String getTableName()
  {
    return m_sTableName;
  }

  /**
   * Obtain the connection being used to connect to the database.
   *
   * @return the connection used to connect to the database
   */
  public Connection getConnection()
  {
    return m_con;
  }
// ----- CacheStore Interface --------------------------------------------

/**
 * Return the value associated with the specified key, or null if the
 * key does not have an associated value in the underlying store.
 *
 * @param oKey  key whose associated value is to be returned
 *
 * @return the value associated with the specified key, or
 *         <tt>null</tt> if no value is available for that key
 */
public Object load(Object oKey) {
    Object oValue = null;
    Connection con = getConnection();
    String sSQL = "SELECT id, value FROM " + getTableName()
        + " WHERE id = ";
    try {
        PreparedStatement stmt = con.prepareStatement(sSQL);
        stmt.setString(1, String.valueOf(oKey));
        ResultSet rslt = stmt.executeQuery();
        if (rslt.next()) {
            oValue = rslt.getString(2);
        }
        stmt.close();
    } catch (SQLException e) {
        throw ensureRuntimeException(e, "Load failed: key=", oKey);
    }
    return oValue;
}

/**
 * Store the specified value under the specific key in the underlying
 * store. This method is intended to support both key/value creation
 * and value update for a specific key.
 *
 * @param oKey    key to store the value under
 * @param oValue  value to be stored
 *
 * @throws UnsupportedOperationException  if this implementation or the
 *         underlying store is read-only
 */
public void store(Object oKey, Object oValue) {
    Connection con = getConnection();
    String sTable = getTableName();
    String sSQL;

// the following is very inefficient; it is recommended to use DB
// specific functionality that is, REPLACE for MySQL or MERGE for Oracle
if (load(oKey) != null)
{
    // key exists - update
    sSQL = "UPDATE " + sTable + " SET value = ? where id = ?";
}
else
{
    // new key - insert
    sSQL = "INSERT INTO " + sTable + " (value, id) VALUES (?,?)";
}
try
{
    PreparedStatement stmt = con.prepareStatement(sSQL);
    int i = 0;
    stmt.setString(++i, String.valueOf(oValue));
    stmt.setString(++i, String.valueOf(oKey));
    stmt.executeUpdate();
    stmt.close();
}
catch (SQLException e)
{
    throw ensureRuntimeException(e, "Store failed: key=" + oKey);
}
/**
 * Remove the specified key from the underlying store if present.
 * @param oKey key whose mapping is to be removed from the map
 * @throws UnsupportedOperationException if this implementation or the
 * underlying store is read-only
 */
public void erase(Object oKey)
{
    Connection con = getConnection();
    String sSQL = "DELETE FROM " + getTableName() + " WHERE id=?";
    try
    {
        PreparedStatement stmt = con.prepareStatement(sSQL);
        stmt.setString(1, String.valueOf(oKey));
        stmt.executeUpdate();
        stmt.close();
    }
catch (SQLException e)
    {
        throw ensureRuntimeException(e, "Erase failed: key=" + oKey);
    }
}
/**
 * Remove the specified keys from the underlying store if present.
 * @param colKeys keys whose mappings are being removed from the cache
 * @throws UnsupportedOperationException if this implementation or the
 * underlying store is read-only
 */
public void eraseAll(Collection colKeys)
{
    throw new UnsupportedOperationException();
}

/**
* Return the values associated with each the specified keys in the
* passed collection. If a key does not have an associated value in
* the underlying store, then the return map will not have an entry
* for that key.
*
* @param colKeys  a collection of keys to load
*
* @return a Map of keys to associated values for the specified keys
*/
public Map loadAll(Collection colKeys)
{
    throw new UnsupportedOperationException();
}

/**
* Store the specified values under the specified keys in the underlying
* store. This method is intended to support both key/value creation
* and value update for the specified keys.
*
* @param mapEntries   a Map of any number of keys and values to store
*
* @throws UnsupportedOperationException  if this implementation or the
*         underlying store is read-only
*/
public void storeAll(Map mapEntries)
{
    throw new UnsupportedOperationException();
}

/**
* Iterate all keys in the underlying store.
*
* @return a read-only iterator of the keys in the underlying store
*/
public Iterator keys()
{
    Connection con  = getConnection();
    String     sSQL = "SELECT id FROM " + getTableName();
    List       list = new LinkedList();
    try
    {
        PreparedStatement stmt = con.prepareStatement(sSQL);
        ResultSet         rslt = stmt.executeQuery();
        while (rslt.next())
        {
            Object oKey = rslt.getString(1);
            list.add(oKey);
        }
        stmt.close();
    }
    catch (SQLException e)
throw ensureRuntimeException(e, "Iterator failed");
}
return list.iterator();

// ----- data members ---------------------------------------------------------------------------

/**
 * The connection.
 */
protected Connection m_con;

/**
 * The db table name.
 */
protected String m_sTableName;

/**
 * Driver class name.
 */
private static final String DB_DRIVER = "org.gjt.mm.mysql.Driver";

/**
 * Connection URL.
 */
private static final String DB_URL = "jdbc:mysql://localhost:3306/CacheStore";

/**
 * User name.
 */
private static final String DB_USERNAME = "root";

/**
 * Password.
 */
private static final String DB_PASSWORD = null;

Sample Controllable CacheStore

This section illustrates the implementation of a controllable cache store. In this scenario, the application can control when updated values in the cache are written to the data store. The most common use case for this scenario is during the initial population of the cache from the data store at startup. At startup, there is no need to write values in the cache back to the data store. Any attempt to do so would be a waste of resources.

The Main.java file in Example G–2 illustrates two different approaches to interacting with a controllable cache store:

- Use a controllable cache (note that it must be on a different service) to enable or disable the cache store. This is illustrated by the ControllableCacheStore1 class.
- Use the CacheStoreAware interface to indicate that objects added to the cache do not need to be stored. This is illustrated by the ControllableCacheStore2 class.
Both ControllableCacheStore1 and ControllableCacheStore2 extend the com.tangosol.net.cache.AbstractCacheStore class. This helper class provides unoptimized implementations of the storeAll and eraseAll operations.

The CacheStoreAware.java file is an interface which can be used to indicate that an object added to the cache should not be stored in the database.

See "Cache of a Database" on page F-4 for a sample cache configurations.

Example G–2 provides a listing of the Main.java interface.

Example G–2 Main.java - Interacting with a Controllable CacheStore

```java
import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.tangosol.net.cache.AbstractCacheStore;
import com.tangosol.util.Base;
import java.io.Serializable;
import java.util.Date;

public class Main extends Base
{
    /**
     * CacheStore implementation which is controlled by a control cache
     */
    public static class ControllableCacheStore1 extends AbstractCacheStore
    {
        public static final String CONTROL_CACHE = "cachestorecontrol";
        String m_sName;

        public static void enable(String sName)
        {
            CacheFactory.getCache(CONTROL_CACHE).put(sName, Boolean.TRUE);
        }

        public static void disable(String sName)
        {
            CacheFactory.getCache(CONTROL_CACHE).put(sName, Boolean.FALSE);
        }

        public void store(Object oKey, Object oValue)
        {
            Boolean isEnabled = (Boolean) CacheFactory.getCache(CONTROL_CACHE).get(m_sName);
            if (isEnabled != null && isEnabled.booleanValue())
            {
                log("controllablecachestore1: enabled " + oKey + " = " + oValue);
            }
            else
            {
                log("controllablecachestore1: disabled " + oKey + " = " + oValue);
            }
        }

        public Object load(Object oKey)
        {
            log("controllablecachestore1: load:" + oKey);
            return new MyValue1(oKey);
        }
    }
}
```
public ControllableCacheStore1(String sName) {
    m_sName = sName;
}

/**
 * CacheStore implementation which is controlled by values
 * implementing the CacheStoreAware interface
 */
public static class ControllableCacheStore2 extends AbstractCacheStore {

    public void store(Object oKey, Object oValue) {
        boolean isEnabled = oValue instanceof CacheStoreAware ? !((CacheStoreAware) oValue).isSkipStore() : true;
        if (isEnabled) {
            log("controllablecachestore2: enabled " + oKey + " = " + oValue);
        } else {
            log("controllablecachestore2: disabled " + oKey + " = " + oValue);
        }
    }

    public Object load(Object oKey) {
        log("controllablecachestore2: load:" + oKey);
        return new MyValue2(oKey);
    }

    public static class MyValue1 implements Serializable {
        String m_sValue;

        public String getValue() {
            return m_sValue;
        }

        public String toString() {
            return "MyValue1\[" + getValue() + "]";
        }

        public MyValue1(Object obj) {
            m_sValue = "value:" + obj;
        }
    }

    public static class MyValue2 extends MyValue1 implements CacheStoreAware {

    }
boolean m_isSkipStore = false;

public boolean isSkipStore()
{
    return m_isSkipStore;
}

public void skipStore()
{
    m_isSkipStore = true;
}

public String toString()
{
    return "MyValue2[" + getValue() + "]";
}

public MyValue2(Object obj)
{
    super(obj);
}

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

        // example 1
        NamedCache cache1 = CacheFactory.getCache("cache1");

        // disable cachestore
        ControllableCacheStore1.disable("cache1");
        for(int i = 0; i < 5; i++)
        {
            cache1.put(new Integer(i), new MyValue1(new Date()));
        }

        // enable cachestore
        ControllableCacheStore1.enable("cache1");
        for(int i = 0; i < 5; i++)
        {
            cache1.put(new Integer(i), new MyValue1(new Date()));
        }

        // example 2
        NamedCache cache2 = CacheFactory.getCache("cache2");

        // add some values with cachestore disabled
        for(int i = 0; i < 5; i++)
        {
            MyValue2 value = new MyValue2(new Date());
            value.skipStore();
            cache2.put(new Integer(i), value);
        }

        // add some values with cachestore enabled
for(int i = 0; i < 5; i++)
{
    cache2.put(new Integer(i), new MyValue2(new Date()));
}

} catch(Throwable oops)
{
    err(oops);
}
finally
{
    CacheFactory.shutdown();
}

Example G–3 provides a listing of the CacheStoreAware.java interface.

**Example G–3  CacheStoreAware.java interface**

```java
public interface CacheStoreAware
{
    public boolean isSkipStore();
}
```
This section describes the elements that control the operational and runtime settings used by Oracle Coherence. These settings are used to create, configure and maintain Coherence clustering, communication, and data management services. This section also describes the deployment descriptor files in which these elements can appear.

**Operational Configuration Deployment Descriptors**

The elements that control the operational and runtime settings to create and configure clustering, communication, and data management services can be specified in either of two deployment descriptors.

The `tangosol-coherence.xml` descriptor is where you specify the operational and runtime elements that control clustering, communication, and data management services. The optional `tangosol-coherence-override.xml` override file is where you specify only the subset of the operational descriptor which you want to adjust. See “Operational Override File (tangosol-coherence-override.xml)” on page H-2 for more information.

For information on configuring caches see Appendix D, ”Cache Configuration Elements.”

**Document Location**

When deploying Coherence, it is important to make sure that the `tangosol-coherence.xml` descriptor is present and situated in the application classpath (like with any other resource, Coherence will use the first one it finds in the classpath). By default (as Oracle ships the software) `tangosol-coherence.xml` is packaged into in the `coherence.jar`.

**Document Root**

The root element of the operational descriptor is `<coherence>`, this is where you may begin configuring your cluster and services.

**Document Format**

Coherence Operational Configuration deployment descriptor should begin with the following DOCTYPE declaration:

**Example H–1  Operational Configuration Deployment Descriptor DOCTYPE Declaration**

```xml
<!DOCTYPE coherence PUBLIC "-//Oracle, Inc.//DTD Oracle Coherence 3.4//EN" "http://www.tangosol.com/dtd/coherence_3_3.dtd">```
**Operational Override File (tangosol-coherence-override.xml)**

Though it is acceptable to supply an alternate definition of the default tangosol-coherence.xml file, the preferred approach to operational configuration is to specify an override file. The override file contains only the subset of the operational descriptor which you want to adjust. The default name for the override file is tangosol-coherence-override.xml, and the first instance found in the classpath will be used. The format of the override file is the same as for the operational descriptor, except that all elements are optional, any missing element will simply be loaded from the operational descriptor.

Multiple levels of override files may also be configured, allowing for additional fine tuning between similar deployment environments such as staging and production. For example Coherence 3.2 and above use this feature to provide alternate configurations such as the logging verbosity based on the deployment type (evaluation, development, production). For more information on logging verbosity, see the `<severity-level>` subelement in "logging-config" on page H-21. See also the tangosol-coherence-override-eval.xml, tangosol-coherence-override-dev.xml, and tangosol-coherence-override-prod.xml files, within coherence.jar for the specific customizations.

**Note:** It is recommended that you supply an override file rather than a custom operational descriptor, thus specifying only the settings you want to adjust.

---

**Command Line Override**

Oracle Coherence provides a very powerful command line override feature which allows for any element defined in this descriptor to be overridden from the Java command line if it has a system-property attribute defined in the descriptor. This feature enables you to use the same operational descriptor (and override file) across all cluster nodes, and provide per-node customizations as system properties. See Appendix L, "Command Line Overrides" for more information on this feature.
Table H–1 lists all non-terminal elements which may be used from within the operational configuration.

<table>
<thead>
<tr>
<th>Element</th>
<th>Used in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>access-controller</td>
<td>security-config</td>
</tr>
<tr>
<td>address-provider</td>
<td>well-known-addresses</td>
</tr>
<tr>
<td>authorized-hosts</td>
<td>cluster-config</td>
</tr>
<tr>
<td>burst-mode</td>
<td>packet-publisher</td>
</tr>
<tr>
<td>callback-handler</td>
<td>security-config</td>
</tr>
<tr>
<td>class-name</td>
<td>access-controller, address-provider, callback-handler, configurable-cache-factory-config, service-failure-policy (see service-guardian)</td>
</tr>
<tr>
<td>cluster-config</td>
<td>coherence</td>
</tr>
<tr>
<td>coherence</td>
<td>root element</td>
</tr>
<tr>
<td>configurable-cache-factory-config</td>
<td>coherence</td>
</tr>
<tr>
<td>filters</td>
<td>cluster-config</td>
</tr>
<tr>
<td>flow-control</td>
<td>packet-delivery</td>
</tr>
<tr>
<td>host-range</td>
<td>authorized-hosts</td>
</tr>
<tr>
<td>incoming-message-handler</td>
<td>cluster-config</td>
</tr>
<tr>
<td>init-param</td>
<td>init-params</td>
</tr>
<tr>
<td>init-params</td>
<td>access-controller, address-provider, callback-handler, configurable-cache-factory-config, filters, services</td>
</tr>
<tr>
<td>license-config</td>
<td>coherence</td>
</tr>
<tr>
<td>logging-config</td>
<td>coherence</td>
</tr>
<tr>
<td>management-config</td>
<td>coherence</td>
</tr>
<tr>
<td>member-identity</td>
<td>cluster-config</td>
</tr>
<tr>
<td>multicast-listener</td>
<td>cluster-config</td>
</tr>
<tr>
<td>notification-queueing</td>
<td>packet-publisher</td>
</tr>
<tr>
<td>outgoing-message-handler</td>
<td>cluster-config</td>
</tr>
<tr>
<td>outstanding-packets</td>
<td>flow-control</td>
</tr>
<tr>
<td>packet-buffer</td>
<td>multicast-listener, packet-publisher, unicast-listener</td>
</tr>
<tr>
<td>packet-bundling</td>
<td>packet-delivery</td>
</tr>
<tr>
<td>packet-delivery</td>
<td>packet-publisher</td>
</tr>
<tr>
<td>packet-pool</td>
<td>incoming-message-handler, packet-publisher</td>
</tr>
<tr>
<td>packet-publisher</td>
<td>cluster-config</td>
</tr>
<tr>
<td>packet-size</td>
<td>packet-publisher</td>
</tr>
<tr>
<td>packet-speaker</td>
<td>cluster-config</td>
</tr>
</tbody>
</table>
### Table H–1 (Cont.) Non-Terminal Operational Configuration Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Used in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pause-detection</td>
<td>flow-control</td>
</tr>
<tr>
<td>security-config</td>
<td>coherence</td>
</tr>
<tr>
<td>service-guardian</td>
<td>cluster-config</td>
</tr>
<tr>
<td>services</td>
<td>cluster-config</td>
</tr>
<tr>
<td>shutdown-listener</td>
<td>cluster-config</td>
</tr>
<tr>
<td>socket-address</td>
<td>well-known-addresses</td>
</tr>
<tr>
<td>tcp-ring-listener</td>
<td>cluster-config</td>
</tr>
<tr>
<td>traffic-jam</td>
<td>packet-publisher</td>
</tr>
<tr>
<td>unicast-listener</td>
<td>cluster-config</td>
</tr>
<tr>
<td>volume-threshold</td>
<td>packet-speaker</td>
</tr>
<tr>
<td>well-known-addresses</td>
<td>unicast-listener</td>
</tr>
</tbody>
</table>
access-controller

Used in: security-config.

Table H–2 describes the subelements you can define within the access-controller element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-name&gt;</td>
<td>Required</td>
<td>Specifies the name of a Java class that implements com.tangosol.net.security.AccessController interface, which will be used by the security framework to check access rights for clustered resources and encrypt/decrypt node-to-node communications regarding those rights. See Chapter 7, &quot;Security Framework&quot; for more information. Default value is com.tangosol.net.security.DefaultController.</td>
</tr>
</tbody>
</table>
| <init-params> | Optional          | Contains one or more initialization parameter(s) for a class that implements the AccessController interface. For the default AccessController implementation the parameters are the paths to the key store file and permissions description file, specified as follows:

```xml
<init-params>
  <init-param id="1">
    <param-type>java.io.File</param-type>
    <param-value system-property="tangosol.coherence.security.keystore"></param-value>
  </init-param>
  <init-param id="2">
    <param-type>java.io.File</param-type>
    <param-value system-property="tangosol.coherence.security.permissions"></param-value>
  </init-param>
</init-params>
```

Preconfigured value based on the default AccessController implementation and the default parameters as specified above are tangosol.coherence.security.keystore and tangosol.coherence.security.permissions. For more information on preconfigured overrides, see Appendix L, "Command Line Overrides." For more information on the elements you can define within the init-param element, see "init-param" on page H-18.
address-provider

Used in: well-known-addresses

Description

The address-provider element specifies the configuration info for an address factory that implements the AddressProvider interface.

Elements

Table H–3 describes the subelements you can define within the address-provider element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-name&gt;</td>
<td>Optional</td>
<td>Specifies the name of a Java class that implements the <code>com.tangosol.util.AddressProvider</code> interface.</td>
</tr>
<tr>
<td>&lt;class-factory-name&gt;</td>
<td>Optional</td>
<td>Specifies the name of a Java class that is a factory for an AddressProvider object.</td>
</tr>
<tr>
<td>&lt;method-name&gt;</td>
<td>Optional</td>
<td>Specifies the name of the factory method to call on the factory specified by the &lt;class-factory-name&gt; configuration element.</td>
</tr>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Specifies the parameters used to initialize the AddressProvider instance.</td>
</tr>
</tbody>
</table>
authorized-hosts

Used in: cluster-config.

Description

If specified, restricts cluster membership to the cluster nodes specified in the collection of unicast addresses, or address range. The unicast address is the address value from the authorized cluster nodes' unicast-listener element. Any number of host-address and host-range elements may be specified.

Elements

Table H–4 describes the subelements you can define within the authorized-hosts element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;host-address&gt;</td>
<td>Optional</td>
<td>Specifies an IP address or hostname. If any are specified, only hosts with specified host-addresses or within the specified host-ranges will be allowed to join the cluster. The content override attributes id can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document.</td>
</tr>
<tr>
<td>&lt;host-range&gt;</td>
<td>Optional</td>
<td>Specifies a range of IP addresses. If any are specified, only hosts with specified host-addresses or within the specified host-ranges will be allowed to join the cluster. The content override attributes id can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document.</td>
</tr>
</tbody>
</table>

The content override attributes xml-override and id can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57.
burst-mode

Used in: packet-publisher.

Description

The burst-mode element is used to control the rate at which packets will transmitted on the network, by specifying the maximum number of packets to transmit without pausing. By default this feature is disabled and is typically only needed when flow-control is disabled, or when operating with heavy loads on a half-duplex network link. This setting only effects packets which are sent by the packet-speaker.

Elements

Table H–5 describes the subelements you can define within the burst-mode element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;maximum-packets&gt;</td>
<td>Required</td>
<td>Specifies the maximum number of packets that will be sent in a row without pausing. Zero indicates no limit. By setting this value relatively low, Coherence is forced to hold back when sending a large number of packets, which may reduce collisions in some instances or allow incoming traffic to be more quickly processed. Default value is 0.</td>
</tr>
<tr>
<td>&lt;pause-milliseconds&gt;</td>
<td>Required</td>
<td>Specifies the minimum number of milliseconds to delay between long bursts of packets. By increasing this value, Coherence is forced to hold back when sending a large number of packets, which may reduce collisions in some instances or allow incoming traffic to be more quickly processed. Default value is 10.</td>
</tr>
</tbody>
</table>
callback-handler

Used in: security-config.

Table H–6 describes the elements you can define within the callback-handler element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-name&gt;</td>
<td>Required</td>
<td>Specifies the name of a Java class that provides the implementation for the javax.security.auth.callback.CallbackHandler interface.</td>
</tr>
<tr>
<td>&lt;init-params&gt;</td>
<td>Optional</td>
<td>Contains one or more initialization parameter(s) for a CallbackHandler implementation. For more information on the elements you can define within the init-param element, refer to &quot;init-param&quot; on page H-18.</td>
</tr>
</tbody>
</table>
class-name

Used in: access-controller, address-provider, callback-handler, configurable-cache-factory-config, service-failure-policy in service-guardian

Description

Specifies the name of a Java class.
cluster-config

Used in: `<coherence>`

Description

Contains the cluster configuration information, including communication and service parameters.

Elements

Table H–7 describes the subelements you can define within the `cluster-config` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;authorized-hosts&gt;</code></td>
<td>Optional</td>
<td>Specifies the hosts which are allowed to join the cluster.</td>
</tr>
<tr>
<td><code>&lt;filters&gt;</code></td>
<td>Optional</td>
<td>Specifies data transformation filters, which can be used to perform custom transformations on data being transferred between cluster nodes.</td>
</tr>
<tr>
<td><code>&lt;incoming-message-handler&gt;</code></td>
<td>Required</td>
<td>Specifies configuration information for the Incoming message handler, used for dispatching incoming cluster communications.</td>
</tr>
<tr>
<td><code>&lt;member-identity&gt;</code></td>
<td>Optional</td>
<td>Specifies detailed identity information that is useful for defining the location and role of the cluster member.</td>
</tr>
<tr>
<td><code>&lt;multicast-listener&gt;</code></td>
<td>Required</td>
<td>Specifies the configuration information for the Multicast listener, used for receiving point-to-multipoint network communications.</td>
</tr>
<tr>
<td><code>&lt;outgoing-message-handler&gt;</code></td>
<td>Required</td>
<td>Specifies configuration information for the Outgoing message handler, used for dispatching outgoing cluster communications.</td>
</tr>
<tr>
<td><code>&lt;packet-publisher&gt;</code></td>
<td>Required</td>
<td>Specifies configuration information for the Packet publisher, used for managing network data transmission.</td>
</tr>
<tr>
<td><code>&lt;packet-speaker&gt;</code></td>
<td>Required</td>
<td>Specifies configuration information for the Packet speaker, used for network data transmission.</td>
</tr>
<tr>
<td><code>&lt;services&gt;</code></td>
<td>Required</td>
<td>Specifies the declarative data for all available Coherence services.</td>
</tr>
<tr>
<td><code>&lt;service-guardian&gt;</code></td>
<td>Required</td>
<td>Specifies the configuration information for the service guardians, used for detecting and resolving service deadlock.</td>
</tr>
<tr>
<td><code>&lt;shutdown-listener&gt;</code></td>
<td>Required</td>
<td>Specifies the action to take upon receiving an external shutdown request.</td>
</tr>
<tr>
<td><code>&lt;tcp-ring-listener&gt;</code></td>
<td>Required</td>
<td>Specifies configuration information for the TCP Ring listener, used to death detection.</td>
</tr>
<tr>
<td><code>&lt;unicast-listener&gt;</code></td>
<td>Required</td>
<td>Specifies the configuration information for the Unicast listener, used for receiving point-to-point network communications.</td>
</tr>
</tbody>
</table>

The content override attribute `xml-override` can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See “Element Attributes” on page H-57 for more information on this attribute.
coherence

root element

Description

The coherence element is the root element of the operational deployment descriptor tangosol-coherence.xml.

Elements

Table H–8 describes the elements you can define within the coherence element.

Table H–8 coherence Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;cluster-config&gt;</code></td>
<td>Required</td>
<td>Contains the cluster configuration information. This element is where most communication and service parameters are defined.</td>
</tr>
<tr>
<td><code>&lt;logging-config&gt;</code></td>
<td>Required</td>
<td>Contains the configuration information for the logging facility.</td>
</tr>
<tr>
<td><code>&lt;configurable-cache-factory-config&gt;</code></td>
<td>Required</td>
<td>Contains configuration information for the configurable cache factory. It controls where, from, and how the cache configuration settings are loaded.</td>
</tr>
<tr>
<td><code>&lt;management-config&gt;</code></td>
<td>Required</td>
<td>Contains the configuration information for the coherence Management Framework. See Chapter 21, &quot;How to Manage Coherence Using JMX&quot; for more information.</td>
</tr>
<tr>
<td><code>&lt;security-config&gt;</code></td>
<td>Optional</td>
<td>Contains the configuration information for the Coherence Security Framework.</td>
</tr>
<tr>
<td><code>&lt;license-config&gt;</code></td>
<td>Optional</td>
<td>Contains the edition and operational mode configuration.</td>
</tr>
</tbody>
</table>

The content override attribute xml-override can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information on this attribute.
configurable-cache-factory-config

Used in: coherence.

Elements

Table H–9 describes the elements you can define within the configurable-cache-factory-config element.

Table H–9 configurable-cache-factory-config Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;class-name&gt;</td>
<td>Required</td>
<td>Specifies the name of a Java class that provides the cache configuration factory. Default value is com.tangosol.net.DefaultConfigurableCacheFactory.</td>
</tr>
</tbody>
</table>
| <init-params>   | Optional          | Contains one or more initialization parameter(s) for a cache configuration factory class which implements the com.tangosol.run.xml.XmlConfigurable interface. For the default cache configuration factory class (DefaultConfigurableCacheFactory) the parameters are specified as follows:  
|                 |                   | <init-param>                                                                                                                                                                                                |
|                 |                   |     <param-type>java.lang.String</param-type>                                                                                                                                                                |
|                 |                   |     <param-value system-property="tangosol.coherence.cacheconfig"> coherence-cache-config.xml                                                        |
|                 |                   |     </param-value>                                                                                                                                                                                          |
|                 |                   |     </init-param>                                                                                                                                                                                            |

Preconfigured is tangosol.coherence.cacheconfig. Unless an absolute or relative path is specified, such as with ./path/to/config.xml, the application's classpath will be used to find the specified descriptor. See Appendix L, "Command Line Overrides" for more information on overrides.

The content override attribute xml-override can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information on this attribute.
filters

Used in: `cluster-config`.

Description

Data transformation filters can be used by `services` to apply a custom transformation on data being transferred between cluster nodes. This can be used for instance to compress or encrypt Coherence network traffic. See the `<filter-class>` element for more information.

Implementation

Data transformation filters are implementations of the `com.tangosol.util.WrapperStreamFactory` interface.

---

**Note:** Data transformation filters are not related to `com.tangosol.util.Filter`, which is part of the Coherence API for querying caches.

Elements

Table H–10 describes the elements you can define within each `filters` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;filter-name&gt;</code></td>
<td>Required</td>
<td>Specifies the canonical name of the filter. This name is unique within the cluster. For example: <code>gzip</code>. The content override attribute <code>id</code> can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document.</td>
</tr>
<tr>
<td><code>&lt;filter-class&gt;</code></td>
<td>Required</td>
<td>Specifies the class name of the filter implementation. This class must have a zero-parameter public constructor and must implement the <code>com.tangosol.util.WrapperStreamFactory</code> interface.</td>
</tr>
</tbody>
</table>
| `<init-params>` | Optional | Specifies initialization parameters, for configuring filters which implement the `com.tangosol.run.xml.XmlConfigurable` interface. For example when using a `com.tangosol.net.CompressionFilter` the parameters are specified as follows:
```xml
<init-param>
  <param-name>strategy</param-name>
  <param-value>gzip</param-value>
</init-param>
<init-param>
  <param-name>level</param-name>
  <param-value>default</param-value>
</init-param>
```
For more information on the parameter values for the standard filters refer to, refer to Chapter 8, "Network Filters."

The content override attributes `id` and `xml-override` can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information on these attributes.
flow-control

Used in: packet-delivery.

Description

The flow-control element contains configuration information related to packet throttling and remote GC detection.

Remote GC Detection

Remote Pause detection allows Coherence to detect and react to a cluster node becoming unresponsive (likely due to a long GC). When a node is marked as paused, packets addressed to it will be sent at a lower rate until the node resumes responding. This remote GC detection is used to avoid flooding a node while it is incapable of responding.

Packet Throttling

Flow control allows Coherence to dynamically adjust the rate at which packets are transmitted to a given cluster node based on point to point transmission statistics.

Elements

Table H–11 describes the elements you can define within the flow-control subelement.

Table H–11  flow-control Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;enabled&gt;</td>
<td>Optional</td>
<td>Specifies if flow control is enabled. Default is true</td>
</tr>
<tr>
<td>&lt;pause-detection&gt;</td>
<td>Optional</td>
<td>Defines the number of packets that will be resent to an unresponsive cluster node before assuming that the node is paused.</td>
</tr>
<tr>
<td>&lt;outstanding-packets&gt;</td>
<td>Optional</td>
<td>Defines the number of unconfirmed packets that will be sent to a cluster node before packets addressed to that node will be deferred.</td>
</tr>
</tbody>
</table>
host-range

Used in: authorized-hosts.

Description

Specifies a range of unicast addresses of nodes which are allowed to join the cluster.

Elements

Table H–12 describes the elements you can define within each host-range element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;from-address&gt;</td>
<td>Required</td>
<td>Specifies the starting IP address for a range of host addresses. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>198.168.1.1.</td>
</tr>
<tr>
<td>&lt;to-address&gt;</td>
<td>Required</td>
<td>Specifies to-address element specifies the ending IP address (inclusive) for a range of hosts. For example: 198.168.2.255.</td>
</tr>
</tbody>
</table>

The content override attribute id can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information on this attribute.
incoming-message-handler

Used in: cluster-config.

Description

The incoming-message-handler assembles UDP packets into logical messages and dispatches them to the appropriate Coherence service for processing.

Elements

Table H–13 describes the subelements you can define within the incoming-message-handler element.

### Table H–13 incoming-message-handler Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;maximum-time-</td>
<td>Required</td>
<td>Specifies the maximum time variance between sending and receiving broadcast Messages when trying to determine the difference between a new cluster Member’s system time and the cluster time. The smaller the variance, the more certain one can be that the cluster time will be closer between multiple systems running in the cluster; however, the process of joining the cluster will be extended until an exchange of Messages can occur within the specified variance. Normally, a value as small as 20 milliseconds is sufficient, but with heavily loaded clusters and multiple network hops it is possible that a larger value would be necessary. Default value is 16.</td>
</tr>
<tr>
<td>variance&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;use-nack-packets&gt;</td>
<td>Required</td>
<td>Specifies whether the packet receiver will use negative acknowledgments (packet requests) to pro-actively respond to known missing packets. See ‘notification-queueing’ on page H-31 for additional details and configuration. Legal values are true or false. Default value is true.</td>
</tr>
<tr>
<td>&lt;priority&gt;</td>
<td>Required</td>
<td>Specifies a priority of the incoming message handler execution thread. Legal values are from 1 to 10. Default value is 7.</td>
</tr>
<tr>
<td>&lt;packet-pool&gt;</td>
<td>Required</td>
<td>Specifies how many incoming packets Coherence will buffer before blocking.</td>
</tr>
</tbody>
</table>

The content override attribute xml-override can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See “Element Attributes” on page H-57 for more information on this attribute.
init-param

Used in: init-params.

Description

Defines an individual initialization parameter.

Elements

Table H–14 describes the elements you can define within the init-param element.

<table>
<thead>
<tr>
<th>Table H–14</th>
<th>init-param Subelement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Required/Optional</td>
</tr>
<tr>
<td>&lt;param-name&gt;</td>
<td>Optional</td>
</tr>
<tr>
<td>&lt;param-type&gt;</td>
<td>Optional</td>
</tr>
<tr>
<td>&lt;param-value&gt;</td>
<td>Required</td>
</tr>
</tbody>
</table>

The content override attribute id can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information on this attribute.
init-params

Used in: address-provider, filters, services, configurable-cache-factory-config, access-controller, and callback-handler.

Description

Defines a series of initialization parameters.

Elements

Table H–15 describes the elements you can define within the init-params element.

Table H–15  init-params Subelement

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;init-param&gt;</td>
<td>Optional</td>
<td>Defines an individual initialization parameter.</td>
</tr>
</tbody>
</table>
license-config

Used in: coherence.

Table H–16 describes the elements you can define within the license-config element.

Table H–16  license-config Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;edition-name&gt;</td>
<td>Optional</td>
<td>Specifies the product edition that the member will use. This allows multiple product editions to be used within the same cluster, with each member specifying the edition that it will be using. Valid values are: GE (Grid Edition), EE (Enterprise Edition), SE (Standard Edition), RTC (Real-Time Client), DC (Data Client). Default value is GE.</td>
</tr>
<tr>
<td>&lt;license-mode&gt;</td>
<td>Optional</td>
<td>Specifies whether the product is being used in an development or production mode. Valid values are prod (Production), and dev (Development). <strong>Note:</strong> This value cannot be overridden in tangosol-coherence-override.xml. It must be specified in tangosol-coherence.xml or (preferably) supplied as system property tangosol.coherence.mode on the Java command line. Default value is dev.</td>
</tr>
</tbody>
</table>
logging-config

Used in: coherence.

Elements

The following table describes the elements you can define within the logging-config element.
### Table H–17  logging-config Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;destination&gt;</td>
<td>Required</td>
<td>Specifies the output device used by the logging system. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- stdout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- stderr (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- jdk</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- log4j</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- a file name</td>
</tr>
</tbody>
</table>

If jdk is specified as the destination, Coherence must be run using JDK 1.4 or later; likewise, if log4j is specified, the Log4j libraries must be in the classpath. In both cases, the appropriate logging configuration mechanism (system properties, property files, and so on) are necessary to configure the JDK/Log4j logging libraries. Preconfigured value is tangosol.coherence.log. See Appendix L, "Command Line Overrides" for more information.
### Table H–17  (Cont.) logging-config Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;severity-level&gt;</td>
<td>Required</td>
<td>Specifies which logged messages will be output to the log destination. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0—only output without a logging severity level specified will be logged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1—all the above plus errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2—all the above plus warnings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3—all the above plus informational messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 4-9—all the above plus internal debugging messages (the higher the number, the more the messages)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- -1—no messages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value is 3. Preconfigured value is tangosol.coherence.log.level. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td>&lt;message-format&gt;</td>
<td>Required</td>
<td>Specifies how messages that have a logging level specified will be formatted before passing them to the log destination. The value of the message-format element is static text with the following replaceable parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- {date}—the date/time format (to a millisecond) at which the message was logged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- {version}—the Oracle Coherence exact version and build details</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- {level}—the logging severity level of the message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- {thread}—the thread name that logged the message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- {member}—the cluster member id (if the cluster is currently running)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- {location}—the fully qualified cluster member id: cluster-name, site-name, rack-name, machine-name, process-name and member-name (if the cluster is currently running)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- {role}—the specified role of the cluster member</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- {text}—the text of the message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value is: {date} Oracle Coherence {version} &lt;{level}&gt; (thread={thread}, member={member}): {text}</td>
</tr>
</tbody>
</table>
The content override attribute `xml-override` can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information on this attribute.
management-config

Used in: coherence.

Elements

Table H–18 describes the elements you can define within the management-config element.

Table H–18 management-config Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Optional/Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;default-domain-name&gt;</code></td>
<td>Required</td>
<td>Specifies the name of the JMX domain used to register MBeans exposed by the Coherence Management Framework. See Chapter 21, &quot;How to Manage Coherence Using JMX&quot; for more information.</td>
</tr>
<tr>
<td><code>&lt;managed-nodes&gt;</code></td>
<td>Required</td>
<td>Specifies whether a cluster node's JVM has an [in-process] MBeanServer and if so, whether this node allows management of other nodes' managed objects. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ none—No MBeanServer is instantiated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ local-only—Manage only MBeans which are local to the cluster node (that is, within the same JVM).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ remote-only—Manage MBeans on other remotely manageable cluster nodes. See <code>&lt;allowed-remote-management&gt;</code> subelement. Requires Coherence Enterprise Edition or higher</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ all—Manage both local and remotely manageable cluster nodes. See <code>&lt;allowed-remote-management&gt;</code> subelement. Requires Coherence Enterprise Edition or higher</td>
</tr>
<tr>
<td><code>&lt;allow-remote-management&gt;</code></td>
<td>Required</td>
<td>Specifies whether this cluster node exposes its managed objects to remote MBeanServer(s). Legal values are: true or false. Default value is false. Preconfigured value is tangosol.coherence.management.remote. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td>refresh-policy</td>
<td>Optional</td>
<td>Specifies the method which is used to refresh remote management information. Legal values are: refresh-ahead, refresh-behind or refresh-expired. Default value is refresh-expired. Preconfigured override is tangosol.coherence.management.refresh.policy</td>
</tr>
<tr>
<td>refresh-expiry</td>
<td>Optional</td>
<td>Specifies the time interval (in milliseconds) after which a remote MBean information will be invalidated on the management node. Legal values are strings representing time intervals. Default value is 1s. Preconfigured override is tangosol.coherence.management.refresh.expiry</td>
</tr>
</tbody>
</table>
The content override attribute `xml-override` can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information.
**member-identity**

Used in: `cluster-config`.

The `member-identity` element contains detailed identity information that is useful for defining the location and role of the cluster member.

**Elements**

Table H–19 describes the elements you can define within the `member-identity` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;cluster-name&gt;</code></td>
<td>Optional</td>
<td>The <code>cluster-name</code> element contains the name of the cluster. To join the cluster all members must specify the same cluster name. It is strongly suggested that cluster-name be specified for production systems, thus preventing accidental cluster discovery among applications. Preconfigured value is <code>tangosol.coherence.cluster</code>. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td><code>&lt;site-name&gt;</code></td>
<td>Optional</td>
<td>The <code>site-name</code> element contains the name of the geographic site that the member is hosted at. For WAN clustering, this value identifies the datacenter within which the member is located, and can be used as the basis for intelligent routing, load balancing and disaster recovery planning. The name is also useful for displaying management information (for example, JMX) and interpreting log entries. It is optional to provide a value for this element. Deployments that spread across more than one geographic site should specify a <code>site-name</code> value. Preconfigured value is <code>tangosol.coherence.site</code>. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td><code>&lt;rack-name&gt;</code></td>
<td>Optional</td>
<td>The <code>rack-name</code> element contains the name of the location within a geographic site that the member is hosted at. This is often a cage, rack or bladeframe identifier, and can be used as the basis for intelligent routing, load balancing and disaster recovery planning (that is, the explicit backing up of data on separate bladeframes). The name is also useful for displaying management information (for example, JMX) and interpreting log entries. It is optional to provide a value for this element. Large scale deployments should always specify a <code>rack-name</code> value. Preconfigured value is <code>tangosol.coherence.rack</code>. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td><code>&lt;machine-name&gt;</code></td>
<td>Optional</td>
<td>The <code>machine-name</code> element contains the name of the physical server that the member is hosted on. This is often the same name as the server identifies itself as (for example, its <code>HOSTNAME</code>, or its name as it appears in a DNS entry). If provided, the <code>machine-name</code> is used as the basis for creating a <code>machine-id</code>, which in turn is used to guarantee that data are backed up on different physical machines to prevent single points of failure (SPOFs). The name is also useful for displaying management information (for example, JMX) and interpreting log entries. It is optional to provide a value for this element. However, it is strongly encouraged that a name always be provided. Preconfigured value is <code>tangosol.coherence.machine</code>. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
</tbody>
</table>
The process-name element contains the name of the process (JVM) that the member is hosted on. This name makes it possible to easily differentiate among multiple JVMs running on the same machine. The name is also useful for displaying management information (for example, JMX) and interpreting log entries. It is optional to provide a value for this element. Often, a single member will exist per JVM, and in that situation this name would be redundant. Preconfigured value is tangosol.coherence.process. See Appendix L, “Command Line Overrides” for more information.

The member-name element contains the name of the member itself. This name makes it possible to easily differentiate among members, such as when multiple members run on the same machine (or even within the same JVM). The name is also useful for displaying management information (for example, JMX) and interpreting log entries. It is optional to provide a value for this element. However, it is strongly encouraged that a name always be provided. Preconfigured value is tangosol.coherence.member. See Appendix L, “Command Line Overrides” for more information.

The role-name element contains the name of the member role. This name allows an application to organize members into specialized roles, such as cache servers and cache clients. The name is also useful for displaying management information (for example, JMX) and interpreting log entries. It is optional to provide a value for this element. However, it is strongly encouraged that a name always be provided. Preconfigured value is tangosol.coherence.role. See Appendix L, “Command Line Overrides” for more information.

The priority element specifies a priority of the corresponding member. The priority is used as the basis for determining tie-breakers between members. If a condition occurs in which one of two members will be ejected from the cluster, and in the rare case that it is not possible to objectively determine which of the two is at fault and should be ejected, then the member with the lower priority will be ejected. Valid values are from 1 to 10. Preconfigured value is tangosol.coherence.priority. See Appendix L, “Command Line Overrides” for more information.
multicast-listener

Used in: cluster-config.

Description

Specifies the configuration information for the Multicast listener. This element is used to specify the address (see `<address>` subelement) and port (see `<port>` subelement) that a cluster will use for cluster wide and point-to-multipoint communications. All nodes in a cluster must use the same multicast address and port, whereas distinct clusters on the same network should use different multicast addresses.

Multicast-Free Clustering

By default, Coherence uses a multicast protocol to discover other nodes when forming a cluster. If multicast networking is undesirable, or unavailable in your environment, the `well-known-addresses` feature may be used to eliminate the need for multicast traffic. If you are having difficulties in establishing a cluster by using multicast, see Chapter 15, "Performing a Multicast Connectivity Test."

Elements

Table H–20 describes the elements you can define within the `multicast-listener` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;address&gt;</code></td>
<td>Required</td>
<td>Specifies the multicast IP address that a Socket will listen or publish on. Legal values are from 224.0.0.0 to 239.255.255.255. Default value depends on the release and build level and typically follows the convention of <code>{build}.{major version}.{minor version}.{patch}</code>. For example, for Coherence Release 2.2 build 255 it is 225.2.2.0. Preconfigured is <code>tangosol.coherence.clusteraddress</code>. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td><code>&lt;port&gt;</code></td>
<td>Required</td>
<td>Specifies the port that the Socket will listen or publish on. Legal values are from 1 to 65535. Default value depends on the release and build level and typically follows the convention of <code>{version}+{{build}</code>. For example, for Coherence Release 2.2 build 255 it is 22255. Preconfigured value is <code>tangosol.coherence.clusterport</code>. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td><code>&lt;time-to-live&gt;</code></td>
<td>Required</td>
<td>Specifies the time-to-live setting for the multicast. This determines the maximum number of &quot;hops&quot; a packet may traverse, where a hop is measured as a traversal from one network segment to another by using a router. Legal values are from 0 to 255. Default value is 4. Preconfigured value is <code>tangosol.coherence.ttl</code>. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td><code>&lt;packet-buffer&gt;</code></td>
<td>Required</td>
<td>Specifies how many incoming packets the operating system will be requested to buffer.</td>
</tr>
</tbody>
</table>
The content override attribute `xml-override` can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information on this attribute.
notification-queueing

Used in: packet-publisher.

Description

The notification-queueing element is used to specify the timing of notifications packets sent to other cluster nodes. Notification packets are used to acknowledge the receipt of packets which require confirmation.

Batched Acknowledgments

Rather then sending an individual ACK for each received packet which requires confirmation, Coherence will batch a series of acknowledgments for a given sender into a single ACK. The <ack-delay-milliseconds> specifies the maximum amount of time that an acknowledgment will be delayed before an ACK notification is sent. By batching the acknowledgments Coherence avoids wasting network bandwidth with many small ACK packets.

Negative Acknowledgments

When enabled cluster nodes will use packet ordering to perform early packet loss detection (see the <use-nack-packets> subelement of <incoming-message-handler>). This allows Coherence to identify a packet as likely being lost and retransmit it well before the packets scheduled (see the <resend-milliseconds> subelement of <packet-delivery>).

Elements

The following table describes the elements you can define within the notification-queueing element.

<table>
<thead>
<tr>
<th>Table H–21 notification-queueing Subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>&lt;ack-delay-milliseconds&gt;</td>
</tr>
<tr>
<td>&lt;nack-delay-milliseconds&gt;</td>
</tr>
</tbody>
</table>
outgoing-message-handler

Used in: acceptor-config, initiator-config.

Description

The outgoing-message-handler specifies the configuration info used to detect dropped client-to-cluster connections. For connection initiators and acceptors that use connectionless protocols (for example, JMS), this information is necessary to proactively detect and release resources allocated to dropped connections. Connection-oriented initiators and acceptors can also use this information as an additional mechanism to detect dropped connections.

Elements

Table H–22 describes the elements you can define within the outgoing-message-handler element.
### Table H–22  outgoing-message-handler Subelement

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<heartbeat-interval>` | Optional | Specifies the interval between ping requests. A ping request is used to ensure the integrity of a connection. The value of this element must be in the following format: 

\[
[d]+[.][d]+? [m]s | MS | M | H | D | d |
\]

where the first non-digits (from left to right) indicate the unit of time duration:

- MS or ms (milliseconds)
- S or s (seconds)
- M or m (minutes)
- H or h (hours)
- D or d (days)

If the value does not contain a unit, a unit of milliseconds is assumed. A value of zero disables ping requests. The default value is zero. |
| `<heartbeat-timeout>` | Optional | Specifies the maximum amount of time to wait for a response to a ping request before declaring the underlying connection unusable. The value of this element must be in the following format: 

\[
[d]+[.][d]+? [m]s | MS | M | H | D | d |
\]

where the first non-digits (from left to right) indicate the unit of time duration:

- MS or ms (milliseconds)
- S or s (seconds)
- M or m (minutes)
- H or h (hours)
- D or d (days)

If the value does not contain a unit, a unit of milliseconds is assumed. The default value is the value of the request-timeout element. |
| `<request-timeout>` | Optional | Specifies the maximum amount of time to wait for a response message before declaring the underlying connection unusable. The value of this element must be in the following format: 

\[
[d]+[.][d]+? [m]s | MS | M | H | D | d |
\]

where the first non-digits (from left to right) indicate the unit of time duration:

- MS or ms (milliseconds)
- S or s (seconds)
- M or m (minutes)
- H or h (hours)
- D or d (days)

If the value does not contain a unit, a unit of milliseconds is assumed. The default value is an infinite timeout. |
outstanding-packets

Used in: flow-control.

Description

Defines the number of unconfirmed packets that will be sent to a cluster node before packets addressed to that node will be deferred. This helps to prevent the sender from flooding the recipient's network buffers.

Auto Tuning

The value may be specified as either an explicit number by using the maximum-packets element, or as a range by using both the maximum-packets and minimum-packets elements. When a range is specified, this setting will be dynamically adjusted based on network statistics.

Elements

Table H–23 describes the elements you can define within the outstanding-packets element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;maximum-packets&gt;</td>
<td>Optional</td>
<td>The maximum number of unconfirmed packets that will be sent to a cluster node before packets addressed to that node will be deferred. It is recommended that this value not be set below 256. Default is 4096.</td>
</tr>
<tr>
<td>&lt;minimum-packets&gt;</td>
<td>Optional</td>
<td>The lower bound on the range for the number of unconfirmed packets that will be sent to a cluster node before packets addressed to that node will be deferred. It is recommended that this value not be set below 16. Default is 64.</td>
</tr>
</tbody>
</table>
packet-buffer

Used in: `unicast-listener, multicast-listener, packet-publisher`.

**Description**

Specifies the size of the operating system buffer for datagram sockets.

**Performance Impact**

Large inbound buffers help insulate the Coherence network layer from JVM pauses caused by the Java Garbage Collector. While the JVM is paused, Coherence is unable to dequeue packets from any inbound socket. If the pause is long enough to cause the packet buffer to overflow, the packet reception will be delayed as the originating node will need to detect the packet loss and retransmit the packet(s).

**It's just a hint**

The operating system will only treat the specified value as a hint, and is not required to allocate the specified amount. In the event that less space is allocated then requested Coherence will issue a warning and continue to operate with the constrained buffer, which may degrade performance. See Chapter 19, "Performance Tuning," for details on configuring your operating system to allow larger buffers.

**Elements**

Table H–24 describes the elements you can define within the `packet-buffer` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;maximum-packets&gt;</code></td>
<td>Required</td>
<td>For <code>unicast-listener, multicast-listener</code> and <code>packet-publisher</code>: Specifies the number of packets of <code>packet-size</code> that the datagram socket will be asked to size itself to buffer. See <code>SO_SNDBUF</code> and <code>SO_RCVBUF</code>. Actual buffer sizes may be smaller if the underlying socket implementation cannot support more than a certain size. Defaults are 32 for publishing, 64 for multicast listening, and 1428 for unicast listening.</td>
</tr>
</tbody>
</table>
packet-bundling

Used in: packet-delivery.

Description

The packet-bundling element contains configuration information related to the bundling of multiple small packets into a single larger packet to reduce the load on the network switching infrastructure.

Default Configuration

The default packet-bundling settings are minimally aggressive allowing for bundling to occur without adding a measurable delay. The benefits of more aggressive bundling will be based on the network infrastructure and the application object's typical data sizes and access patterns.

Elements

Table H–25 describes the elements you can define within the packet-bundling element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;maximum-deferral-time&gt;</td>
<td>Optional</td>
<td>The maximum amount of time to defer a packet while waiting for additional packets to bundle. A value of zero will result in the algorithm not waiting, and only bundling the readily accessible packets. A value greater than zero will cause some transmission deferral while waiting for additional packets to become available. This value is typically set below 250 microseconds to avoid a detrimental throughput impact. If the units are not specified, nanoseconds are assumed. Default value is 1us (microsecond).</td>
</tr>
<tr>
<td>&lt;aggression-factor&gt;</td>
<td>Optional</td>
<td>Specifies the aggressiveness of the packet deferral algorithm. Where as the maximum-deferral-time element defines the upper limit on the deferral time, the aggression-factor influences the average deferral time. The higher the aggression value, the longer the Publisher may wait for additional packets. The factor may be expressed as a real number, and often times values between 0.0 and 1.0 will be allow for high packet utilization while keeping latency to a minimum. Default value is zero.</td>
</tr>
</tbody>
</table>
packet-pool

Used in: incoming-message-handler, packet-publisher.

Description

Specifies the number of packets which Coherence will internally maintain for use in transmitting and receiving UDP packets. Unlike the packet-buffer these buffers are managed by Coherence rather than the operating system, and allocated on the JVM's heap.

Performance Impact

The packet pools are used as a reusable buffer between Coherence network services. For packet transmission, this defines the maximum number of packets which can be queued on the packet-speaker before the packet-publisher must block. For packet reception, this defines the number of packets which can be queued on the incoming-message-handler before the unicast-listener, and multicast-listener must block.

Elements

Table H–26 describes the subelements you can define within the packet-pool element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;maximum-packets&gt;</td>
<td>Required</td>
<td>The maximum number of reusable packets to be used by the services responsible for publishing and receiving. The pools are initially small, and will grow on demand up to the specified limits. Defaults are 2048 for transmitting and receiving.</td>
</tr>
</tbody>
</table>
packet-delivery

Used in: packet-publisher.

Description

Specifies timing and transmission rate parameters related to packet delivery.

Death Detection

The \(<\text{timeout-milliseconds}\>\) and \(<\text{heartbeat-milliseconds}\>\) subelements are used in detecting the death of other cluster nodes.

Elements

Table H–27 describes the elements you can define within the packet-delivery element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;\text{resend-milliseconds})&gt;</td>
<td>Required</td>
<td>For packets which require confirmation, specifies the minimum amount of time in milliseconds to wait for a corresponding ACK packet, before resending a packet. Default value is 200.</td>
</tr>
<tr>
<td>(&lt;\text{timeout-milliseconds})&gt;</td>
<td>Required</td>
<td>For packets which require confirmation, specifies the maximum amount of time, in milliseconds, that a packet will be resent. After this timeout expires Coherence will make a determination if the recipient is to be considered &quot;dead&quot;. This determination takes additional data into account, such as if other nodes are still able to communicate with the recipient. Default value is 60000. <strong>Note:</strong> For production use, the recommended value is the greater of 60000 and two times the maximum expected full GC duration.</td>
</tr>
<tr>
<td>(&lt;\text{heartbeat-milliseconds})&gt;</td>
<td>Required</td>
<td>Specifies the interval between heartbeats. Each member issues a unicast heartbeat, and the most senior member issues the cluster heartbeat, which is a broadcast message. The heartbeat is used by the tcp-ring-listener as part of fast death detection. Default value is 1000.</td>
</tr>
<tr>
<td>(&lt;\text{flow-control})&gt;</td>
<td>Optional</td>
<td>Configures per-node packet throttling and remote GC detection.</td>
</tr>
<tr>
<td>(&lt;\text{packet-bundling})&gt;</td>
<td>Optional</td>
<td>Configures how aggressively Coherence will attempt to maximize packet utilization.</td>
</tr>
</tbody>
</table>
packet-publisher

Used in: `cluster-config`.

**Description**

Specifies configuration information for the Packet publisher, which manages network data transmission.

**Reliable packet delivery**

The Packet publisher is responsible for ensuring that transmitted packets reach the destination cluster node. The publisher maintains a set of packets which are waiting to be acknowledged, and if the ACK does not arrive by the `packet-delivery` resend timeout, the packet will be retransmitted (see `<packet-delivery>` subelement). The recipient node will delay the ACK, to batch a series of ACKs into a single response (see `<notification-queuing>` subelement).

**Throttling**

The rate at which the publisher will accept and transmit packet may be controlled by using the `traffic-jam` and `flow-control` settings. Throttling may be necessary when dealing with slow networks, or small `packet-buffer`.

**Elements**

`Table H–28` describes the elements you can define within the `packet-publisher` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;enabled&gt;</code></td>
<td>Required</td>
<td>Specifies if TCMP clustering is enabled. For Coherence editions which support both Coherence Extend and Coherence TCMP based clustering, this feature allows TCMP to be disabled to ensure that a node only connects by using the Extend protocol. Default value is <code>true</code>. Preconfigured value is <code>tangosol.coherence.tcmp.enabled</code>. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td><code>&lt;packet-size&gt;</code></td>
<td>Required</td>
<td>Specifies the UDP packet sizes to use.</td>
</tr>
<tr>
<td><code>&lt;packet-delivery&gt;</code></td>
<td>Required</td>
<td>Specifies timing parameters related to reliable packet delivery.</td>
</tr>
<tr>
<td><code>&lt;notification-queueing&gt;</code></td>
<td>Required</td>
<td>Contains the notification queue related configuration info.</td>
</tr>
<tr>
<td><code>&lt;burst-mode&gt;</code></td>
<td>Required</td>
<td>Specifies the maximum number of packets the publisher may transmit without pausing.</td>
</tr>
<tr>
<td><code>&lt;traffic-jam&gt;</code></td>
<td>Required</td>
<td>Specifies the maximum number of packets which can be enqueued on the publisher before client threads block.</td>
</tr>
<tr>
<td><code>&lt;packet-buffer&gt;</code></td>
<td>Required</td>
<td>Specifies how many outgoing packets the operating system will be requested to buffer.</td>
</tr>
<tr>
<td><code>&lt;packet-pool&gt;</code></td>
<td>Required</td>
<td>Specifies how many outgoing packets Coherence will buffer before blocking.</td>
</tr>
<tr>
<td><code>&lt;priority&gt;</code></td>
<td>Required</td>
<td>Specifies a priority of the packet publisher execution thread. Legal values are from 1 to 10. Default value is 6.</td>
</tr>
</tbody>
</table>
The content override attribute xml-override can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information.
packet-size

Used in: packet-publisher.

Description

The packet-size element specifies the maximum and preferred UDP packet sizes (see the <maximum-length> and <preferred-length> subelements). All cluster nodes must use identical maximum packet sizes. For optimal network utilization this value should be 32 bytes less then the network MTU.

Note: When specifying a UDP packet size larger then 1024 bytes on Microsoft Windows a registry setting must be adjusted to allow for optimal transmission rates. See "Datagram size (Microsoft Windows)" on page 19-3 for details.

Elements

Table H–29 describes the subelements you can define within the packet-size element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| <maximum-length>     | Required          | Specifies the maximum size, in bytes, of the UDP packets that will be sent and received on the unicast and multicast sockets. This value should be at least 512; recommended value is 1468 for 100Mb, and 1Gb Ethernet. This value must be identical on all cluster nodes.  
  Note: Some network equipment cannot handle packets larger than 1472 bytes (IPv4) or 1468 bytes (IPv6), particularly under heavy load. If you encounter this situation on your network, this value should be set to 1472 or 1468 respectively. The recommended values is 32 bytes less then the network MTU setting. Default value is 1468. |
| <preferred-length>   | Required          | Specifies the preferred size, in bytes, of UDP packets that will be sent and received on the unicast and multicast sockets. This value should be at least 512 and cannot be greater than the maximum-length value; it is recommended to set the value to the same as the maximum-length value. Default value is 1468. |
packet-speaker

Used in: cluster-config.

Description

Specifies configuration information for the Packet speaker, used for network data transmission.

Offloaded Transmission

The Packet speaker is responsible for sending packets on the network. The speaker is used when the packet-publisher detects that a network send operation is likely to block. This allows the Packet publisher to avoid blocking on IO and continue to prepare outgoing packets. The Publisher will dynamically choose whether to use the speaker as the packet load changes.

Elements

Table H–30 describes the subelements you can define within the packet-speaker element.

Table H–30  packet-speaker Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;volume-threshold&gt;</td>
<td>Optional</td>
<td>Specifies the packet load which must be present for the speaker to be activated.</td>
</tr>
<tr>
<td>&lt;priority&gt;</td>
<td>Required</td>
<td>Specifies a priority of the packet speaker execution thread. Legal values are from 1 to 10. Default value is 8.</td>
</tr>
</tbody>
</table>

The content override attribute xml-override can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information on this attribute.
pause-detection

Used in: flow-control.

Description
Remote Pause detection allows Coherence to detect and react to a cluster node becoming unresponsive (likely due to a long GC). When a node is marked as paused, packets addressed to it will be sent at a lower rate until the node resumes responding. This remote GC detection is used to avoid flooding a node while it is incapable of responding.

Elements
Table H–31 describes the subelements you can define within the pause-detection element.

Table H–31  pause-detection Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;maximum-packets&gt;</td>
<td>Optional</td>
<td>The maximum number of packets that will be resent to an unresponsive cluster node before assuming that the node is paused. Specifying a value of 0 will disable pause detection. Default is 16.</td>
</tr>
</tbody>
</table>
security-config

Used in: coherence.

Elements

Table H–32 describes the subelements you can define within the security-config element.

Table H–32  security-config Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;enabled&gt;</td>
<td>Required</td>
<td>Specifies whether the security features are enabled. All other configuration elements in the security-config group will be verified for validity and used if and only if the value of this element is true. Legal values are true or false. Default value is false. Preconfigured value is tangosol.coherence.security. See Appendix L, “Command Line Overrides” for more information.</td>
</tr>
<tr>
<td>&lt;login-module-name&gt;</td>
<td>Required</td>
<td>Specifies the name of the JAAS LoginModule that should be used to authenticate the caller. This name should match a module in a configuration file will be used by the JAAS (for example specified by using the -Djava.security.auth.login.config Java command line attribute). For details please refer to the Sun Login Module Developer's Guide.</td>
</tr>
<tr>
<td>&lt;access-controller&gt;</td>
<td>Required</td>
<td>Contains the configuration information for the class that implements com.tangosol.net.security.AccessController interface, which will be used by the security framework to check access rights for clustered resources and encrypt/decrypt node-to-node communications regarding those rights. See Chapter 7, “Security Framework” for more information.</td>
</tr>
<tr>
<td>&lt;callback-handler&gt;</td>
<td>Optional</td>
<td>Contains the configuration information for the class that implements javax.security.auth.callback.CallbackHandler interlace which will be called if an attempt is made to access a protected clustered resource when there is no identity associated with the caller.</td>
</tr>
</tbody>
</table>

The content override attribute xml-override can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information.
service-guardian

Used in: cluster-config

Description

Specifies the configuration of the service-guardians, which detect and attempt to resolve service deadlocks.

Elements

Table H–33 describes the subelements you can define within the service-guardian element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;timeout-milliseconds&gt;</td>
<td>Required</td>
<td>The timeout value used to guard against deadlocked or unresponsive services. It is recommended that service-guardian/timeout-milliseconds be set equal to or greater than the packet-delivery/timeout-milliseconds value. A timeout of 0 will disable service guardians. Default value is 65000. Preconfigured override is tangosol.coherence.guard.timeout</td>
</tr>
</tbody>
</table>
| <service-failure-policy> | Required          | Specifies the action to take when an abnormally behaving service cannot be terminated gracefully by the service-guardian. Legal values are:  
  - <class-name> - configuration element providing the name of a Java class that provides an implementation for the com.tangosol.net.ServiceFailurePolicy interface  
  - exit-cluster - cause the local node to stop the cluster services  
  - exit-process - cause the JVM to terminate abruptly  
  Default value is exit-cluster. |

The content override attribute xml-override can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information.
services

Used in: cluster-config.

Description

Specifies the configuration for Coherence services.

Service Components

The types of services which can be configured includes:

- ReplicatedCache—A cache service which maintains copies of all cache entries on all cluster nodes which run the service.
- ReplicatedCache.Optimistic—A version of the ReplicatedCache which uses optimistic locking.
- DistributedCache—A cache service which evenly partitions cache entries across the cluster nodes which run the service.
- SimpleCache—A version of the ReplicatedCache which lacks concurrency control.
- LocalCache—A cache service for caches where all cache entries reside in a single cluster node.
- InvocationService—A service used for performing custom operations on remote cluster nodes.

Elements

Table H–34 describes the subelements you can define for each services element.
The content override attributes `xml-override` and `id` can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document.

### Table H–34 services Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;service-type&gt;</code></td>
<td>Required</td>
<td>Specifies the canonical name for a service, allowing the service to be referenced from the <code>service-name</code> element in cache configuration caching schemes. See &quot;caching-schemes&quot; on page D-20 for more information.</td>
</tr>
<tr>
<td><code>&lt;service-component&gt;</code></td>
<td>Required</td>
<td>Specifies either the fully qualified class name of the service or the relocatable component name relative to the base Service component. Legal values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ ReplicatedCache</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ ReplicatedCache.Optimistic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ DistributedCache</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ SimpleCache</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ LocalCache</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ InvocationService</td>
</tr>
<tr>
<td><code>&lt;use-filters&gt;</code></td>
<td>Optional</td>
<td>Contains the list of <code>filters</code> names to be used by this service. For example, specify <code>use-filter</code> as follows</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;use-filters&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;filter-name&gt;gzip&lt;/filter-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;/use-filters&gt;</code></td>
</tr>
<tr>
<td><code>&lt;init-params&gt;</code></td>
<td>Optional</td>
<td>Specifies the initialization parameters that are specific to each service-component. For more service specific parameter information see:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;DistributedCache Service Parameters&quot; on page I-3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;ReplicatedCache Service Parameters&quot; on page I-7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ &quot;InvocationService Parameters&quot; on page I-8</td>
</tr>
</tbody>
</table>
shutdown-listener

Used in: cluster-config.

Description

Specifies the action a cluster node should take upon receiving an external shutdown request. External shutdown includes the "kill" command on UNIX and Ctrl-C on Windows and UNIX.

Elements

Table H–35 describes the elements you can define within the shutdown-listener element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| <enabled>  | Required          | Specifies the type of action to take upon an external JVM shutdown. Legal values:  
  ■ none—perform no explicit shutdown actions  
  ■ force—perform "hard-stop" the node by calling Cluster.stop()  
  ■ graceful—perform a "normal" shutdown by calling Cluster.shutdown()  
  ■ true—same as force  
  ■ false—same as none  
  
  Note: For production use, the suggested value is none unless testing has verified that the behavior on external shutdown is exactly what is desired. Default value is force. Preconfigured value is tangosol.coherence.shutdownhook. See Appendix L, "Command Line Overrides" for more information. |

The content override attribute xml-override can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information.
socket-address

Used in: well-known-addresses, tcp-initiator.

Elements

Table H–36 describes the subelements you can define within the socket-address element.

<table>
<thead>
<tr>
<th>Table H–36 socket-address Subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>&lt;address&gt;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>&lt;port&gt;</td>
</tr>
</tbody>
</table>
tcp-ring-listener

Used in: cluster-config.

Description

The TCP-ring provides a means for fast death detection of another node within the cluster. When enabled the cluster nodes form a single "ring" of TCP connections spanning the entire cluster. A cluster node is able to use the TCP connection to detect the death of another node within a heartbeat interval (default is one second; see the <heartbeat-milliseconds> subelement of packet-delivery). If disabled, the cluster node must rely on detecting that another node has stopped responding to UDP packets for a considerately longer interval (see the <timeout-milliseconds> subelement of packet-delivery). When the death has been detected it is communicated to all other cluster nodes.

Elements

Table H–37 describes the subelements you can define within the tcp-ring-listener element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;enabled&gt;</td>
<td>Required</td>
<td>Specifies whether the tcp ring listener should be enabled to detect node failures faster. Legal values are true and false. Default value is true. Preconfigured value is tangosol.coherence.tcpring. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td>&lt;maximum-socket-closed-exceptions&gt;</td>
<td>Required</td>
<td>Specifies the maximum number of tcp ring listener exceptions that will be tolerated before a particular member is considered really gone and is removed from the cluster. This value is used only if the value of tcp-ring-listener/enabled is true. Legal values are integers greater than zero. Default value is 2.</td>
</tr>
<tr>
<td>&lt;priority&gt;</td>
<td>Required</td>
<td>Specifies a priority of the tcp ring listener execution thread. Legal values are from 1 to 10. Default value is 6.</td>
</tr>
</tbody>
</table>

The content override attribute xml-override can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information on this attribute.
traffic-jam

Used in: packet-publisher.

Description

The traffic-jam element is used to control the rate at which client threads enqueue packets for the Packet publisher to transmit on the network. When the limit is exceeded any client thread will be forced to pause until the number of outstanding packets drops below the specified limit. To limit the rate at which the Publisher transmits packets see the flow-control, and burst-mode elements.

Tuning

Specifying a limit which is to low, or a pause which is to long may result in the publisher transmitting all pending packets, and being left without packets to send. An ideal value will ensure that the publisher is never left without work to do, but at the same time prevent the queue from growing uncontrollably. It is therefore recommended that the pause remain quite short (10ms or under), and that the limit on the number of packets be kept high (that is, greater than 5000). As of Coherence 3.2 a warning will be periodically logged if this condition is detected.

Traffic Jam and Flow Control

When flow-control is enabled the traffic-jam operates in a point-to-point mode, only blocking a send if the recipient has too many packets outstanding. It is recommended that the traffic-jam/maximum-packets value be greater than the value (see the <maximum-packets> subelement of outstanding-packets). When flow-control is disabled, the traffic-jam will take all outstanding packets into account.

Elements

Table H–38 describes the subelements you can define within the traffic-jam element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;maximum-packets&gt;</td>
<td>Required</td>
<td>Specifies the maximum number of pending packets that the Publisher will tolerate before determining that it is clogged and must slow down client requests (requests from local non-system threads). Zero means no limit. This property prevents most unexpected out-of-memory conditions by limiting the size of the resend queue. Default value is 8192.</td>
</tr>
<tr>
<td>&lt;pause-milliseconds&gt;</td>
<td>Required</td>
<td>Number of milliseconds that the Publisher will pause a client thread that is trying to send a message when the Publisher is clogged. The Publisher will not allow the message to go through until the clog is gone, and will repeatedly sleep the thread for the duration specified by this property. Default value is 10.</td>
</tr>
</tbody>
</table>
**unicast-listener**

Used in: `cluster-config`.

**Description**

Specifies the configuration information for the Unicast listener. This element is used to specify the address and port that a cluster node will bind to, to listen for point-to-point cluster communications.

**Automatic Address Settings**

By default Coherence will attempt to obtain the IP to bind to using the `java.net.InetAddress.getLocalHost()` call. On machines with multiple IPs or NICs you may need to explicitly specify the address (see the `<address>` subelement). Additionally if the specified port is already in use, Coherence will by default auto increment the port number until the binding succeeds (see the `<port>` and `<auto>` subelements).

**Multicast-Free Clustering**

By default Coherence uses a multicast protocol to discover other nodes when forming a cluster. If multicast networking is undesirable, or unavailable in your environment, the `well-known-addresses` feature may be used to eliminate the need for multicast traffic. If you are having difficulties in establishing a cluster by using multicast, see Chapter 15, "Performing a Multicast Connectivity Test."

**Elements**

Table H–39 describes the subelements you can define within the `unicast-listener` element.

<table>
<thead>
<tr>
<th>Table H–39</th>
<th>unicast-listener Subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Required/Optional</td>
</tr>
<tr>
<td><code>&lt;well-known-addresses&gt;</code></td>
<td>Optional</td>
</tr>
<tr>
<td><code>&lt;machine-id&gt;</code></td>
<td>Required</td>
</tr>
<tr>
<td><code>&lt;address&gt;</code></td>
<td>Required</td>
</tr>
<tr>
<td><code>&lt;port&gt;</code></td>
<td>Required</td>
</tr>
</tbody>
</table>
Table H–39  (Cont.) unicast-listener Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;port-auto-adjust&gt;</td>
<td>Required</td>
<td>Specifies whether the unicast port will be automatically incremented if the specified port cannot be bound to because it is already in use. Legal values are true or false. It is recommended that this value be configured to false for production environments. Default value is true. Preconfigured value is tangosol.coherence.localport.adjust. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td>&lt;packet-buffer&gt;</td>
<td>Required</td>
<td>Specifies how many incoming packets the operating system will be requested to buffer.</td>
</tr>
<tr>
<td>&lt;priority&gt;</td>
<td>Required</td>
<td>Specifies a priority of the unicast listener execution thread. Legal values are from 1 to 10. Default value is 8.</td>
</tr>
<tr>
<td>&lt;ignore-socket-closed&gt;</td>
<td>Optional</td>
<td>Specifies whether the unicast listener will ignore socket exceptions that indicate that a Member is unreachable. Deprecated as of Coherence 3.2.</td>
</tr>
<tr>
<td>&lt;maximum-socket-closed -exceptions&gt;</td>
<td>Optional</td>
<td>Specifies the maximum number of unicast listener exceptions that will be tolerated before a particular member is considered really gone and is removed from the cluster. Deprecated as of Coherence 3.2.</td>
</tr>
</tbody>
</table>

The content override attribute xml-override can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information on this attribute.
volume-threshold

Used in: packet-speaker

Description

Specifies the minimum outgoing packet volume which must exist for the speaker daemon to be activated.

Performance Impact

When the packet load is relatively low it may be more efficient for the speaker's operations to be performed on the publisher's thread. When the packet load is high using the speaker allows the publisher to continue preparing packets while the speaker transmits them on the network.

Elements

Table H–40 describes the elements you can define within the packet-speaker element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;minimum-packets&gt;</td>
<td>Required</td>
<td>Specifies the minimum number of packets which must be ready to be sent for the speaker daemon to be activated. A value of 0 will force the speaker to always be used, while a very high value will cause it to never be used. If unspecified, it will be set to match the packet-buffer, this is the default.</td>
</tr>
</tbody>
</table>
well-known-addresses

Used in: unicast-listener.

---

**Note:** This is not a security-related feature, and does not limit the addresses which are allowed to join the cluster. See the authorized-hosts element for details on limiting cluster membership.

Use of the Well Known Addresses (WKA) feature is not supported by Caching Edition. If you are having difficulties in establishing a cluster by using multicast, see Chapter 15, "Performing a Multicast Connectivity Test".

---

**Description**

By default, Coherence uses a multicast protocol to discover other nodes when forming a cluster. If multicast networking is undesirable, or unavailable in your environment, the Well Known Addresses feature may be used to eliminate the need for multicast traffic. When in use the cluster is configured with a relatively small list of nodes which are allowed to start the cluster, and which are likely to remain available over the cluster lifetime. There is no requirement for all WKA nodes to be simultaneously active at any point in time. This list is used by all other nodes to find their way into the cluster without the use of multicast, thus at least one well known node must be running for other nodes to be able to join.

---

**Example**

**Example H–2** illustrates a configuration for two well-known-addresses with the default port.

**Example H–2  Configuration for Two Well-Known-Addresses**

```xml
<well-known-addresses>
  <socket-address id="1">
    <address>192.168.0.100</address>
    <port>8088</port>
  </socket-address>
  <socket-address id="2">
    <address>192.168.0.101</address>
    <port>8088</port>
  </socket-address>
</well-known-addresses>
```

---

**Elements**

Table H–41 describes the subelements you can define within the well-known-addresses element.
The content override attribute `xml-override` can be optionally used to fully or partially override the contents of this element with XML document that is external to the base document. See "Element Attributes" on page H-57 for more information about this attribute.
Element Attributes

The optional `id` and `xml-override` attributes can be used to override the contents of an element. These attributes can appear, either individually or together, within the following elements:

Table H–42 lists the elements that can use `id` or `xml-override`, or both.

<table>
<thead>
<tr>
<th>Table H–42</th>
<th>Elements that can use id or xml-override, or Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>authorized-hosts</td>
<td>cluster-config</td>
</tr>
<tr>
<td>filter-name</td>
<td>filters</td>
</tr>
<tr>
<td>init-param</td>
<td>logging-config</td>
</tr>
<tr>
<td>services</td>
<td>shutdown-listener</td>
</tr>
</tbody>
</table>

Table H–43 describes the functionality of the `id` and `xml-override` attributes.

<table>
<thead>
<tr>
<th>Table H–43</th>
<th>id and xml-override Attribute Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Required/Optional</td>
</tr>
<tr>
<td><code>xml-override</code></td>
<td>Optional</td>
</tr>
<tr>
<td><code>id</code></td>
<td>Optional</td>
</tr>
</tbody>
</table>
Initialization Parameter Settings

The `<init-param>` element in the Coherence operational configuration deployment descriptor defines initialization parameters for a service or filter. The parameters that appear under `init-param` will be different, depending on the service or filter you are working with.

The following sections describe the parameters that can be configured for these services and filters:

- DistributedCache Service Parameters
- ReplicatedCache Service Parameters
- InvocationService Parameters
- ProxyService Parameters
- Compression Filter Parameters

The tables in each section describe the specific `<param-name>` — `<param-value>` pairs that can be configured for various elements. The Parameter Name column refers to the value of the `param-name` element and Value Description column refers to the possible values for the corresponding `param-value` element.

For example, the sample entry in Table I–1 means that the `init-params` element may look like the configuration in Example I–1 or Example I–2.

<table>
<thead>
<tr>
<th>Parameter Value</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>local-storage</td>
<td>Specifies whether this member of the DistributedCache service enables the local storage. Legal values are true or false. Default value is true. Preconfigured value is <code>tangosol.coherence.distributed.localstorage</code>. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
</tbody>
</table>

Example I–1 Sample `init-param` Configuration

```xml
...<init-params>
  <init-param>
    <param-name>local-storage</param-name>
    <param-value>false</param-value>
  </init-param>
</init-params>
...```

or as follows:
Example I–2  Another Sample init-param Configuration

...  
  <init-params>
  <init-param>
    <param-name>local-storage</param-name>
    <param-value>true</param-value>
  </init-param>
  </init-params>
  ...

DistributedCache Service Parameters

DistributedCache <services> elements support the parameters described in Table I–2. These settings may also be specified as part of the <distributed-scheme> element in the Cache Configuration Elements descriptor coherence-cache-config.xml.

Table I–2  DistributedCache Service Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value, Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup-count</td>
<td>Specifies the number of members of the DistributedCache service that hold the</td>
</tr>
<tr>
<td></td>
<td>backup data for each unit of storage in the cache. Value of 0 means that in the case</td>
</tr>
<tr>
<td></td>
<td>of abnormal termination, some portion of the data in the cache will be lost. Value</td>
</tr>
<tr>
<td></td>
<td>of N means that if up to N cluster nodes terminate immediately, the cache data will</td>
</tr>
<tr>
<td></td>
<td>be preserved. To maintain the distributed cache of size M, the total memory usage in</td>
</tr>
<tr>
<td></td>
<td>the cluster does not depend on the number of cluster nodes and will be in the order</td>
</tr>
<tr>
<td></td>
<td>of M(N+1). Recommended values are 0, 1 or 2. Default value is 1.</td>
</tr>
<tr>
<td>backup-storage/class-name</td>
<td>Only applicable with the custom type. Specifies a class name for the custom storage</td>
</tr>
<tr>
<td></td>
<td>implementation. If the class implements com.tangosol.run.xml.XmlConfigurable</td>
</tr>
<tr>
<td></td>
<td>interface then upon construction the setConfig method is called passing the entire</td>
</tr>
<tr>
<td></td>
<td>backup-storage element.</td>
</tr>
<tr>
<td>backup-storage/directory</td>
<td>Only applicable with the file-mapped type. Specifies the path name for the directory</td>
</tr>
<tr>
<td></td>
<td>that the disk persistence manager (com.tangosol.util.nio.MapBufferManager) will use</td>
</tr>
<tr>
<td></td>
<td>as “root” to store files in. If not specified or specifies a non-existent directory,</td>
</tr>
<tr>
<td></td>
<td>a temporary file in the default location is used. Default value is the default temporary</td>
</tr>
<tr>
<td></td>
<td>directory designated by the Java runtime.</td>
</tr>
<tr>
<td>backup-storage/initial-size</td>
<td>Only applicable with the off-heap and file-mapped types. Specifies the initial buffer</td>
</tr>
<tr>
<td></td>
<td>size in bytes. The value of this element must be in the following format: [\d]+[.][\d]</td>
</tr>
</tbody>
</table>
### DistributedCache Service Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value, Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup-storage/scheme-name</td>
<td>Only applicable with the scheme type. Specifies a scheme name for the ConfigurableCacheFactory.</td>
</tr>
<tr>
<td>backup-storage/type</td>
<td>Specifies the type of the storage used to hold the backup data. Legal values are:</td>
</tr>
<tr>
<td></td>
<td>■ on-heap—The corresponding implementations class is java.util.HashMap.</td>
</tr>
<tr>
<td></td>
<td>■ off-heap—The corresponding implementations class is com.tangosol.util.nio.BinaryMap using com.tangosol.util.nio.DirectBufferManager. Only available with JDK 1.4 and later.</td>
</tr>
<tr>
<td></td>
<td>■ file-mapped—The corresponding implementations class is com.tangosol.util.nio.BinaryMap using com.tangosol.util.nio.MappedBufferManager. Only available with JDK 1.4 and later.</td>
</tr>
<tr>
<td></td>
<td>■ custom—The corresponding implementations class is the class specified by the backup-storage/class element.</td>
</tr>
<tr>
<td></td>
<td>■ scheme—The corresponding implementations class is the map returned by the ConfigurableCacheFactory for the scheme referred to by the backup-storage/scheme-name element.</td>
</tr>
<tr>
<td></td>
<td>Default value is on-heap.</td>
</tr>
<tr>
<td></td>
<td>Preconfigured value is tangosol.coherence.distributed.backup. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
<tr>
<td>key-associator/class-name</td>
<td>Specifies the name of a class that implements the com.tangosol.net.partition.KeyAssociator interface. This implementation must have a zero-parameter public constructor.</td>
</tr>
<tr>
<td>key-partitioning/class-name</td>
<td>Specifies the name of a class that implements the com.tangosol.net.partition.KeyPartitioningStrategy interface. This implementation must have a zero-parameter public constructor.</td>
</tr>
<tr>
<td>lease-granularity</td>
<td>Specifies the lease ownership granularity. Available since release 2.3. Legal values are:</td>
</tr>
<tr>
<td></td>
<td>■ thread</td>
</tr>
<tr>
<td></td>
<td>■ member</td>
</tr>
<tr>
<td></td>
<td>A value of thread means that locks are held by a thread that obtained them and can only be released by that thread. A value of member means that locks are held by a cluster node and any thread running on the cluster node that obtained the lock can release it.</td>
</tr>
<tr>
<td>local-storage</td>
<td>Specifies whether this member of the DistributedCache service enables local storage.</td>
</tr>
<tr>
<td></td>
<td>Normally this value should be left unspecified within the configuration file, and instead set on a per-process basis using the tangosol.coherence.distributed.localstorage system property. This allows cache clients and servers to use the same configuration descriptor.</td>
</tr>
<tr>
<td></td>
<td>Legal values are true or false. Default value is true.</td>
</tr>
<tr>
<td></td>
<td>Preconfigured value is tangosol.coherence.distributed.localstorage. See Appendix L, &quot;Command Line Overrides&quot; for more information.</td>
</tr>
</tbody>
</table>
The `partition-count` element specifies the number of partitions that a partitioned (distributed) cache will be "chopped up" into. Each member running the partitioned cache service that has the local-storage option set to true will manage a "fair" (balanced) number of partitions.

The number of partitions should be a prime number and sufficiently large such that a given partition is expected to be no larger than 50MB in size.

Good defaults for example service storage sizes are provided below:

<table>
<thead>
<tr>
<th>service storage</th>
<th>partition-count</th>
</tr>
</thead>
<tbody>
<tr>
<td>100M</td>
<td>257</td>
</tr>
<tr>
<td>1G</td>
<td>509</td>
</tr>
<tr>
<td>10G</td>
<td>2039</td>
</tr>
<tr>
<td>50G</td>
<td>4093</td>
</tr>
<tr>
<td>100G</td>
<td>8191</td>
</tr>
</tbody>
</table>

A list of first 1,000 primes can be found at:

http://primes.utm.edu/lists/small/1000.txt

Valid values are positive integers.

Default value is the value specified in the `tangosol-coherence.xml` descriptor.

The `partition-listener/class-name` specifies the name of a class that implements the `com.tangosol.net.partition.PartitionListener` interface. This implementation must have a zero-parameter public constructor.

The `request-timeout` specifies the maximum amount of time a client will wait for a response before abandoning the original request. The request time is measured on the client side as the time elapsed from the moment a request is sent for execution to the corresponding server node(s) and includes the following:

- the time it takes to deliver the request to an executing node (server)
- the interval between the time the task is received and placed into a service queue until the execution starts
- the task execution time
- the time it takes to deliver a result back to the client

Legal values are positive integers or zero (indicating no default timeout).

The `standard-lease-milliseconds` specifies the duration of the standard lease in milliseconds. When a lease has aged past this number of milliseconds, the lock will automatically be released. Set this value to zero to specify a lease that never expires. The purpose of this setting is to avoid deadlocks or blocks caused by stuck threads; the value should be set higher than the longest expected lock duration (for example, higher than a transaction timeout). It's also recommended to set this value higher than the `packet-delivery-timeout-milliseconds` value.

Legal values are from positive long numbers or zero. Default value is 0.

The `task-hung-threshold` specifies the amount of time in milliseconds that a task can execute before it is considered "hung".

**Note:** a posted task that has not yet started is never considered as hung. This attribute is applied only if the Thread pool is used (the `thread-count` value is positive).

Legal values are positive integers or zero (indicating no default timeout).
DistributedCache Service Parameters

(task-timeout)

(task-timeout) Specifies the default timeout value in milliseconds for tasks that can be timed-out (for example, implement the com.tangosol.net.PriorityTask interface), but don’t explicitly specify the task execution timeout value. The task execution time is measured on the server side and does not include the time spent waiting in a service backlog queue before being started. This attribute is applied only if the thread pool is used (the thread-count value is positive).

Legal values are positive integers or zero (indicating no default timeout).

(thread-count)

Specify the number of daemon threads used by the distributed cache service. If zero, all relevant tasks are performed on the service thread.

Legal values are from positive integers or zero.

Default value is 0. Preconfigured value is tangosol.coherence.distributed.threads. See Appendix L, "Command Line Overrides" for more information.

(transfer-threshold)

Specifies the primary buckets distribution threshold in kilobytes. When a new node joins the distributed cache service or when a member of the service leaves, the remaining nodes perform a task of bucket ownership re-distribution. During this process, the existing data gets rebalanced along with the ownership information. This parameter indicates a preferred message size for data transfer communications. Setting this value lower will make the distribution process take longer, but will reduce network bandwidth utilization during this activity.

Legal values are integers greater than zero.

Default value is 512 (0.5MB). Preconfigured value is tangosol.coherence.distributed.transfer. See Appendix L, "Command Line Overrides" for more information.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value, Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>task-timeout</td>
<td>Specifies the default timeout value in milliseconds for tasks that can be timed-out (for example, implement the com.tangosol.net.PriorityTask interface), but don’t explicitly specifi...</td>
</tr>
<tr>
<td>thread-count</td>
<td>Specifies the number of daemon threads used by the distributed cache service. If zero, all relevant tasks are performed on the service thread.</td>
</tr>
<tr>
<td>transfer-threshold</td>
<td>Specifies the threshold for the primary buckets distribution. When a new node joins the distributed cache service or when a member of the service leaves, the remaining nodes perform a task of bucket ownership re-distribution. During this process, the existing data gets rebalanced along with the ownership information. This parameter indicates a preferred message size for data transfer communications. Setting this value lower will make the distribution process take longer, but will reduce network bandwidth utilization during this activity.</td>
</tr>
</tbody>
</table>

Default value is 512 (0.5MB). Preconfigured value is tangosol.coherence.distributed.transfer. See Appendix L, "Command Line Overrides" for more information.
ReplicatedCache Service Parameters

ReplicatedCache services elements support the parameters described in Table I–3. These settings may also be specified as part of the replicated-scheme element in the Cache Configuration Elements descriptor coherence-cache-config.xml.

Table I–3  ReplicatedCache Service Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value Description</th>
</tr>
</thead>
</table>
| lease-granularity    | Specifies the lease ownership granularity. Available since release 2.3.Legal values are:  
  - thread  
  - member  
  A value of thread means that locks are held by a thread that obtained them and can only be released by that thread. A value of member means that locks are held by a cluster node and any thread running on the cluster node that obtained the lock can release it.  
  Default value is thread. |
| mobile-issues        | Specifies whether lease issues should be transferred to the most recent lock holders.  
  Legal values are true or false.  
  Default value is false. |
| standard-lease-      | Specifies the duration of the standard lease in milliseconds. When a lease has aged past this number of milliseconds, the lock will automatically be released. Set this value to zero to specify a lease that never expires. The purpose of this setting is to avoid deadlocks or blocks caused by stuck threads; the value should be set higher than the longest expected lock duration (for example, higher than a transaction timeout). It’s also recommended to set this value higher then packet-delivery/timeout-milliseconds value.  
  Legal values are from positive long numbers or zero.  
  Default value is 0. |
| milliseconds         |                                                                                                                                                                                                                   |
InvocationService elements support the following parameters listed in Table I–4. These settings may also be specified as part of the invocation-scheme element in the Cache Configuration Elements descriptor coherence-cache-config.xml.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value, Description</th>
</tr>
</thead>
</table>
| request-timeout    | Specifies the default timeout value in milliseconds for requests that can time-out (for example, implement the com.tangosol.net.PriorityTask interface), but don’t explicitly specify the request timeout value. The request time is measured on the client side as the time elapsed from the moment a request is sent for execution to the corresponding server node(s) and includes the following:  
  ■ the time it takes to deliver the request to an executing node (server)  
  ■ the interval between the time the task is received and placed into a service queue until the execution starts  
  ■ the task execution time  
  ■ the time it takes to deliver a result back to the client  
  Legal values are positive integers or zero (indicating no default timeout). |
| task-hung-threshold| Specifies the amount of time in milliseconds that a task can execute before it is considered “hung”. Note: a posted task that has not yet started is never considered as hung. This attribute is applied only if the Thread pool is used (the thread-count value is positive). |
| task-timeout       | Specifies the default timeout value in milliseconds for tasks that can be timed-out (for example, implement the com.tangosol.net.PriorityTask interface), but don’t explicitly specify the task execution timeout value. The task execution time is measured on the server side and does not include the time spent waiting in a service backlog queue before being started. This attribute is applied only if the thread pool is used (the thread-count value is positive).  
  Legal values are positive integers or zero (indicating no default timeout). |
| thread-count       | Specifies the number of daemon threads to be used by the invocation service. If zero, all relevant tasks are performed on the service thread.  
  Legal values are from positive integers or zero. Preconfigured value is tangosol.coherence.invocation.threads. See Appendix L, “Command Line Overrides” for more information.  
  Default value is 0. |
ProxyService parameters support the parameters described in Table I-5. These settings may also be specified as part of the `proxy-scheme` element in the `Cache Configuration Elements` descriptor `coherence-cache-config.xml`.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>thread-count</td>
<td>Specifies the number of daemon threads to be used by the proxy service. If zero, all relevant tasks are performed on the service thread. Legal values are from positive integers or zero. Default value is 0.</td>
</tr>
<tr>
<td>task-hung-threshold</td>
<td>Specifies the amount of time in milliseconds that a task can execute before it is considered &quot;hung&quot;. Note: a posted task that has not yet started is never considered as hung. This attribute is applied only if the Thread pool is used (the <code>thread-count</code> value is positive). Legal values are positive integers or zero (indicating no default timeout).</td>
</tr>
<tr>
<td>task-timeout</td>
<td>Specifies the default timeout value in milliseconds for tasks that can be timed-out (for example, implement the <code>com.tangosol.net.PriorityTask</code> interface), but don't explicitly specify the task execution timeout value. The task execution time is measured on the server side and does not include the time spent waiting in a service backlog queue before being started. This attribute is applied only if the thread pool is used (the <code>thread-count</code> value is positive). Legal values are positive integers or zero (indicating no default timeout).</td>
</tr>
</tbody>
</table>
Compression Filter Parameters

The compression filters (com.tangosol.net.CompressionFilter), supports the parameters described in Table I–6 (see java.util.zip.Deflater for details).

**Table I–6  Compression Filter Parameters**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer-length</td>
<td>Specifies compression buffer length in bytes. Legal values are positive integers or zero. Default value is 0.</td>
</tr>
<tr>
<td>level</td>
<td>Specifies the compression level. Legal values are: default, compression, speed, none. Default value is default.</td>
</tr>
<tr>
<td>strategy</td>
<td>Specifies the compressions strategy. Legal values are: gzip, huffman-only, filtered, default. Default value is gzip.</td>
</tr>
</tbody>
</table>
This appendix provides a listing of the elements that can be used to specify POF user types. POF user type configuration elements are defined in the `pof-config.dtd` file that can be found in the `coherence.jar` file.

You can find additional information about the POF user type configuration file in the Javadoc for the `ConfigurablePofContext` class.

**POF User Type Deployment Descriptor**

Use the POF user type deployment descriptor to specify the various user types which are being passed into the cluster.

**Document Location**

The name and location of the descriptor defaults to `pof-config.xml`. The default POF user type descriptor (packaged in `coherence.jar`) will be used unless a custom file is found within the application's classpath. The name and location of the descriptor can also be configured using system property `tangosol.pof.config`. It is recommended that all nodes within a cluster use identical POF user type descriptors.

**Document Root**

The root element of the POF user type descriptor is `pof-config`. This is where you may begin specifying your user types.

**Document Format**

The POF user type descriptor should begin with the following `DOCTYPE` declaration:

```xml
<!DOCTYPE pof-config SYSTEM "pof-config.dtd">
```

The format of the document and the nesting of elements is illustrated in Example J–1.

**Example J–1  Format of a POF User Type Configuration File (pof-config.xml)**

```xml
<pof-config>
  <user-type-list>
  ..
  </user-type-list>
  <user-type>
    <type-id>53</type-id>
    <class-name>com.mycompany.data.Trade</class-name>
    <serializer>
      <class-name>com.tangosol.io.pof.PortableObjectSerializer</class-name>
      <init-params>
      ..
      </init-params>
  </user-type>
</pof-config>
```
<init-param>
  <param-type>int</param-type>
  <param-value>(type-id)</param-value>
</init-param>
</init-params>
</serializer>
</user-type>

<user-type>
  <type-id>54</type-id>
  <class-name>com.mycompany.data.Position</class-name>
</user-type>

..
<include>file:/my-pof-config.xml</include>

..
</user-type-list>

<allow-interfaces>false</allow-interfaces>
<allow-subclasses>false</allow-subclasses>
</pof-config>

Command Line Override
Oracle Coherence provides a powerful Command Line Setting Override Feature, which allows any element defined in this descriptor to be overridden from the Java command line if it has a system-property attribute defined in the descriptor.
Element Index

Table J–1 lists all elements which may be used from within a POF user type configuration.

<table>
<thead>
<tr>
<th>Element</th>
<th>Used In:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;allow-interfaces&gt;</td>
<td>&lt;pof-config&gt;</td>
</tr>
<tr>
<td>&lt;allow-subclasses&gt;</td>
<td>&lt;pof-config&gt;</td>
</tr>
<tr>
<td>&lt;class-name&gt;</td>
<td>&lt;user-type&gt;,&lt;serializer&gt;</td>
</tr>
<tr>
<td>&lt;include&gt;</td>
<td>&lt;user-type-list&gt;</td>
</tr>
<tr>
<td>&lt;init-param&gt;</td>
<td>&lt;init-params&gt;</td>
</tr>
<tr>
<td>&lt;init-params&gt;</td>
<td>&lt;serializer&gt;</td>
</tr>
<tr>
<td>&lt;param-type&gt;</td>
<td>&lt;init-param&gt;</td>
</tr>
<tr>
<td>&lt;param-value&gt;</td>
<td>&lt;init-param&gt;</td>
</tr>
<tr>
<td>&lt;pof-config&gt;</td>
<td>root element</td>
</tr>
<tr>
<td>&lt;serializer&gt;</td>
<td>&lt;user-type&gt;</td>
</tr>
<tr>
<td>&lt;type-id&gt;</td>
<td>&lt;user-type&gt;</td>
</tr>
<tr>
<td>&lt;user-type&gt;</td>
<td>&lt;user-type-list&gt;</td>
</tr>
<tr>
<td>&lt;user-type-list&gt;</td>
<td>&lt;pof-config&gt;</td>
</tr>
</tbody>
</table>
allow-interfaces

Used in: `<pof-config>`

Description

The `allow-interfaces` element indicates whether the `user-type class-name` can specify Java interface types in addition to Java class types.

Valid values are `true` or `false`. Default value is `false`.

Elements

Terminal element.
allow-subclasses

Used in: `<pof-config>`

Description

The `allow-subclasses` element indicates whether the `user-type class-name` can specify a Java class type that is abstract, and whether sub-classes of any specified `user-type class-name` will be permitted at runtime and automatically mapped to the specified super-class for purposes of obtaining a serializer.

Valid values are `true` or `false`. Default value is `false`.

Elements

Terminal element.
class-name

Used in: `<user-type>`, `<serializer>`

Description

The `class-name` element specifies the name of a Java class or interface.

Within the `user-type` element, the `class-name` element is required, and specifies the fully qualified name of the Java class or interface that all values of the user type are type-assignable to.

Within the `serializer` element, the `class-name` element is required.

Elements

Terminal element.
include

Used in: `<user-type-list>`

Description

The include element specifies the location of a `pof-config` file to load `user-type` elements from. The value is a locator string (either a valid path or URL) that identifies the location of the target `pof-config` file.

Elements

Terminal element.
init-param

Used in: <init-params>

Description

The init-param element provides a type for a configuration parameter and a corresponding value to pass as an argument.

Elements

Table J–2 describes the subelements you can define within the init-param element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;param-type&gt;</td>
<td>Required</td>
<td>The param-type element specifies the Java type of initialization parameter. Supported types are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ string—indicates that the value is a java.lang.String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ boolean—indicates that the value is a java.lang.Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ int—indicates that the value is a java.lang.Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ long—indicates that the value is a java.lang.Long</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ double—indicates that the value is a java.lang.Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ decimal—indicates that the value is a java.math.BigDecimal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ file—indicates that the value is a java.io.File</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ date—indicates that the value is a java.sql.Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ time—indicates that the value is a java.sql.Timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ datetime—indicates that the value is a java.sql.Timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ xml—indicates that the value is the entire init-param XmlElement.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The value is converted to the specified type, and the target constructor or method must have a parameter of that type for the instantiation to succeed.</td>
</tr>
<tr>
<td>&lt;param-value&gt;</td>
<td>Required</td>
<td>The param-value element specifies a value of the initialization parameter. The value is in a format specific to the type of the parameter. There are four reserved values that can be specified. Each of these values is replaced at runtime with a specific runtime value before the constructor is invoked:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ {type-id}—replaced with the Type ID of the User Type;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ {class-name}—replaced with the name of the class for the User Type;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ {class}—replaced with the Class for the User Type;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ {class-loader}—replaced with the ConfigurablePofContext's ContextClassLoader.</td>
</tr>
</tbody>
</table>
init-params

Used in: <serializer>, <well-known-addresses>

Description
The init-params element contains zero or more arguments (each as an init-param) that correspond to the parameters of a constructor of the class that is being configured.

Elements
Table J–3 describes the elements you can define within the init-params element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;init-param&gt;</td>
<td>Required</td>
<td>The init-param element provides a type for a configuration parameter and a corresponding value to pass as an argument.</td>
</tr>
</tbody>
</table>
param-type

Used in: <init-param>

Description

The `param-type` element specifies the Java type of initialization parameter. Supported types are:

- `string`—indicates that the value is a `java.lang.String`
- `boolean`—indicates that the value is a `java.lang.Boolean`
- `int`—indicates that the value is a `java.lang.Integer`
- `long`—indicates that the value is a `java.lang.Long`
- `double`—indicates that the value is a `java.lang.Double`
- `decimal`—indicates that the value is a `java.math.BigDecimal`
- `file`—indicates that the value is a `java.io.File`
- `date`—indicates that the value is a `java.sql.Date`
- `time`—indicates that the value is a `java.sql.TimeDatetimetime`
- `datetime`—indicates that the value is a `java.sql.Timestamp`
- `xml`—indicates that the value is the entire `init-param` `XmlElement`.

The value is converted to the specified type, and the target constructor or method must have a parameter of that type in order for the instantiation to succeed.

Elements

Terminal element.
param-value

Used in: `<init-param>`

Description
The `param-value` element specifies a value of the initialization parameter. The value is in a format specific to the type of the parameter.

There are four reserved values that can be specified. Each of these values is replaced at runtime with a specific runtime value before the constructor is invoked:

- `{type-id}`—replaced with the Type ID of the User Type;
- `{class-name}`—replaced with the name of the class for the User Type;
- `{class}`—replaced with the Class for the User Type;
- `{class-loader}`—replaced with the ConfigurablePofContext's ContextClassLoader.

Elements
Terminal element.
**pof-config**

*root element*

**Description**

The `pof-config` element is the root element of the POF user type configuration descriptor.

**Elements**

Table J–4 describes the elements you can define within the `pof-config` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;allow-interfaces&gt;</td>
<td>Optional</td>
<td>The <code>allow-interfaces</code> element indicates whether the user-type class-name can specify Java interface types in addition to Java class types. Valid values are <code>true</code> or <code>false</code>. Default value is <code>false</code>.</td>
</tr>
<tr>
<td>&lt;allow-subclasses&gt;</td>
<td>Optional</td>
<td>The <code>allow-subclasses</code> element indicates whether the user-type class-name can specify a Java class type that is abstract, and whether sub-classes of any specified user-type class-name will be permitted at runtime and automatically mapped to the specified super-class for purposes of obtaining a serializer. Valid values are <code>true</code> or <code>false</code>. Default value is <code>false</code>.</td>
</tr>
<tr>
<td>&lt;user-type-list&gt;</td>
<td>Required</td>
<td>The <code>user-type-list</code> element contains zero or more user-type elements. Each POF user type that will be used must be listed in the <code>user-type-list</code>. The <code>user-type-list</code> element may also contain zero or more include elements. Each include element is used to add user-type elements defined in another <code>pof-config</code> file.</td>
</tr>
</tbody>
</table>
**serializer**

Used in: `<acceptor-config>`, `<distributed-scheme>`, `<initiator-config>`, `<invocation-scheme>`, `<optimistic-scheme>`, `<replicated-scheme>`, `<user-type>`

**Description**

This element may be used either as part of a service scheme element such as `proxy-scheme/acceptor-config`, and `distributed-scheme`, or as part of a `user-type` element within a POF configuration file for specifying a POFSerializer.

**Usage Within Service Schemes**

Specifies the class configuration info for a `com.tangosol.io.Serializer` implementation used by the service to serialize and deserialize user types.

For example, the following configures a `ConfigurablePofContext` that uses the default `coherence-pof-config.xml` configuration file to write objects to and read from a stream:

```xml
<serializer>
  <class-name>com.tangosol.io.pof.ConfigurablePofContext</class-name>
</serializer>
```

**Usage Within user-type**

The serializer element specifies what `PofSerializer` to use to serialize and deserialize a specific user type.

A `PofSerializer` is used to serialize and deserialize user type values to and from a POF stream. Within the serializer element, the `class-name` element is required, and zero or more constructor parameters can be defined within an `init-params` element.

If the serializer element is omitted, then the user type is assumed to implement the `PortableObject` interface, and the `PortableObjectSerializer` implementation is used as the `PofSerializer`.

If the `init-params` element is omitted from the serializer element, then the following four constructors are attempted on the specific `PofSerializer` implementation, in this order:

- `(int nTypeId, Class clz, ClassLoader loader)`
- `(int nTypeId, Class clz)`
- `(int nTypeId)`
- `()`

**Elements**

`Table J–5` describes the elements you can define within the `serializer` element.
Table J–5  serializer Subelements

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;class-name&gt;</code></td>
<td>Required</td>
<td>Specifies the name of the serializer.</td>
</tr>
<tr>
<td><code>&lt;init-params&gt;</code></td>
<td>Optional</td>
<td>The <code>init-params</code> element contains zero or more arguments (each as an <code>init-param</code>) that correspond to the parameters of a constructor of the class that is being configured.</td>
</tr>
</tbody>
</table>
**type-id**

**Used in**: `<user-type>`

**Description**

The `type-id` element specifies an integer value \( n \geq 0 \) that uniquely identifies the user type.

If none of the `user-type` elements contains a `type-id` element, then the type IDs for the user types will be based on the order in which they appear in the `user-type-list`, with the first user type being assigned the type ID 0, the second user type being assigned the type ID 1, and so on.

However, it is strongly recommended that user types IDs always be specified, to support schema versioning and evolution.

---

**Note:** Reserved IDs: The first 1000 IDs are reserved for Coherence internal use.

---

**Elements**

Terminal element.
The `user-type` element contains the declaration of a POF user type. A POF user type is a uniquely identifiable, portable, versionable object class that can be communicated among systems regardless of language, operating system, hardware and location.

Within the `user-type` element, the `type-id` element is optional, but its use is strongly suggested to support schema versioning and evolution.

Within the `user-type` element, the `class-name` element is required, and specifies the fully qualified name of the Java class or interface that all values of the user type are type-assignable to.

If the `serializer` element is omitted, then the user type is assumed to implement the `PortableObject` interface, and the `PortableObjectSerializer` implementation is used as the `PofSerializer`.

### Elements

Table J–6 describes the elements you can define within the `user-type` element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;class-name&gt;</code></td>
<td>Required</td>
<td>The <code>class-name</code> element specifies the name of a Java class or interface. Within the <code>user-type</code> element, the <code>class-name</code> element is required, and specifies the fully qualified name of the Java class or interface that all values of the user type are type-assignable to. Within the <code>serializer</code> element, the <code>class-name</code> element is required.</td>
</tr>
<tr>
<td><code>&lt;serializer&gt;</code></td>
<td>Optional</td>
<td>The <code>serializer</code> element specifies what <code>PofSerializer</code> to use to serialize and deserialize a specific user type. A <code>PofSerializer</code> is used to serialize and deserialize user type values to and from a POF stream. Within the <code>serializer</code> element, the <code>class-name</code> element is required, and zero or more constructor parameters can be defined within an <code>init-params</code> element. If the <code>serializer</code> element is omitted, then the user type is assumed to implement the <code>PortableObject</code> interface, and the <code>PortableObjectSerializer</code> implementation is used as the <code>PofSerializer</code>. If the <code>init-params</code> element is omitted from the <code>serializer</code> element, then the following four constructors are attempted on the specific <code>PofSerializer</code> implementation, and in this order:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ (int nTypeId, Class clz, ClassLoader loader)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ (int nTypeId, Class clz)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ (int nTypeId)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ ()</td>
</tr>
<tr>
<td><code>&lt;type-id&gt;</code></td>
<td>Optional</td>
<td>The <code>type-id</code> element specifies an integer value ( n \geq 0 ) that uniquely identifies the user type. If none of the <code>user-type</code> elements contains a <code>type-id</code> element, then the type IDs for the user types will be based on the order in which they appear in the <code>user-type-list</code>, with the first user type being assigned the type ID 0, the second user type being assigned the type ID 1, and so on. However, it is strongly recommended that user types IDs always be specified, to support schema versioning and evolution.</td>
</tr>
</tbody>
</table>
user-type-list

Used in: <pof-config>

Description

The user-type-list element contains zero or more user-type elements. Each POF user type that will be used must be listed in the user-type-list.

The user-type-list element may also contain zero or more include elements. Each include element is used to add user-type elements defined in another pof-config file.

Elements

The following table describes the elements you can define within the user-type-list element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;include&gt;</td>
<td>Required</td>
<td>The include element specifies the location of a pof-config file to load user-type elements from. The value is a locator string (either a valid path or URL) that identifies the location of the target pof-config file.</td>
</tr>
<tr>
<td>&lt;user-type&gt;</td>
<td>Required</td>
<td>The user-type element contains the declaration of a POF user type. A POF user type is a uniquely identifiable, portable, versionable object class that can be communicated among systems regardless of language, operating system, hardware and location. Within the user-type element, the type-id element is optional, but its use is strongly suggested to support schema versioning and evolution. Within the user-type element, the class-name element is required, and specifies the fully qualified name of the Java class or interface that all values of the user type are type-assignable to. If the serializer element is omitted, then the user type is assumed to implement the PortableObject interface, and the PortableObjectSerializer implementation is used as the PofSerializer.</td>
</tr>
</tbody>
</table>
This appendix provides a description of the elements that can be used to configure MBeans.

**MBEANS IN THE COHERENCE DEPLOYMENT DESCRIPTOR**

The MBean configuration elements are defined in the `coherence.dtd` XML file, which is packaged in `coherence.jar`.

**DOCUMENT ROOT**

The root element of the POF user type descriptor is `mbeans`. This is where you may begin configuring your MBean.

**DOCUMENT FORMAT**

The format and nesting of the MBean configuration elements is illustrated in Example K–1.

*Example K–1  Format and Nesting of MBean Configuration Elements*

```xml
<mbeans>
  <mbean>
    <mbean-class>
    <mbean-factory>
    <mbean-query>
    <mbean-accessor>
    <mbean-name>
      <enabled>
      <extend-lifecycle>
    </mbean>
  </mbeans>
```

```xml
```
### MBean Configuration Element Index

#### Table K–1  MBean Configuration Element Index

<table>
<thead>
<tr>
<th>Element</th>
<th>Used In:</th>
</tr>
</thead>
<tbody>
<tr>
<td>extend-lifecycle</td>
<td>mbean</td>
</tr>
<tr>
<td>enabled</td>
<td>mbean</td>
</tr>
<tr>
<td>mbean</td>
<td>mbeans</td>
</tr>
<tr>
<td>mbean-accessor</td>
<td>mbean</td>
</tr>
<tr>
<td>mbean-class</td>
<td>mbean</td>
</tr>
<tr>
<td>mbean-factory</td>
<td>mbean</td>
</tr>
<tr>
<td>mbean-name</td>
<td>mbean</td>
</tr>
<tr>
<td>mbean-query</td>
<td>mbean</td>
</tr>
<tr>
<td>mbeans</td>
<td>root element</td>
</tr>
</tbody>
</table>
extend-lifecycle

Used in: mbean

Description
Specifies if the MBean should extend beyond the node connection life cycle. If false, the MBean will be destroyed and re-created when a node is disconnected from the grid. If true, the MBean will maintain the statistics and values across connections.

Example
<extend-lifecycle>true</extend-lifecycle>
enabled

Used in: mbean

Description

The enabled element specifies either true if the MBean should be instantiated or false if the MBean should be ignored.

Example

<enabled>true</enabled>
mbean

Used in: mbeans.

Description

The mbean element contains a list of elements to be instantiated and registered with the Coherence Management infrastructure.

Elements

Table K–2 describes the subelements you can define within the mbeans element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;mbean-class&gt;</td>
<td>Optional</td>
<td>Specifies the class of the standard MBean to instantiate.</td>
</tr>
<tr>
<td>&lt;mbean-factory&gt;</td>
<td>Optional</td>
<td>Specifies the class of the factory used to instantiate the MBean.</td>
</tr>
<tr>
<td>&lt;mbean-query&gt;</td>
<td>Optional</td>
<td>Specifies the JMX ObjectName query pattern used to retrieve the MBeans.</td>
</tr>
<tr>
<td>&lt;mbean-accessor&gt;</td>
<td>Optional</td>
<td>Specifies the accessor method on the &lt;mbean-factory&gt; used to instantiate the MBean.</td>
</tr>
<tr>
<td>&lt;mbean-name&gt;</td>
<td>Required</td>
<td>Specifies the ObjectName prefix for the MBean.</td>
</tr>
<tr>
<td>&lt;local-only&gt;</td>
<td>Optional</td>
<td>Controls the MBean visibility across the cluster. If set to true, the MBean is registered only with a local MBeanServer and is not accessible by other cluster nodes; otherwise the nodeId... key attribute is added to its name and the MBean will be visible from any of the “managing” nodes (the ones that have the managed-nodes element set to values of all or remote-only). Default value is false.</td>
</tr>
<tr>
<td>&lt;enabled&gt;</td>
<td>Optional</td>
<td>Specifies if the MBean should be registered on this instance. Default value is false.</td>
</tr>
<tr>
<td>&lt;extend-lifecycle&gt;</td>
<td>Optional</td>
<td>Specifies if the MBean should extend beyond the node connection life cycle. If false, the MBean will be destroyed and re-created when a node is disconnected from the grid. If true, the MBean will maintain the statistics and values across connections.</td>
</tr>
</tbody>
</table>
mbean-accessor

Used in: mbean

Description

The mbean-accessor element contains the accessor method name on the MBean Factory that instantiates the MXBean. The MBean factory class must be in the class path to correctly instantiate. The `<mbean-accessor>` element requires an mbean-factory element.

Example

```
<mbean-factory>java.lang.management.ManagementFactory</mbean-factory>
<mbean-accessor>getMemoryMXBean</mbean-accessor>
```
mbean-class

Used in: mbean

Description
The mbean-class element contains the class name of a standard MBean. The MBean class must be in the class path to correctly instantiate.

Example
<mbean-class>com.oracle.custom.mbeans.query</mbean-class>
mbean-factory

Used in: mbean

Description

The mbean-factory element contains the class name of a MBean factory that instantiates MXBeans. The MBean factory class must be in the class path to correctly instantiate. The <mbean-factory> element requires an mbean-accessor element.

Example

<mbean-factory>java.lang.management.ManagementFactory</mbean-factory>
<mbean-accessor>getMemoryMXBean</mbean-accessor>
mbean-name

Used in: mbean

Description

The mbean-name element contains the ObjectName prefix for the resulting MBeans. This ObjectName prefix should be a comma-separated Key=Value pair. The Coherence MBean naming convention stipulates that the name should begin with a "type"/"value" pair (that is, type=Platform)

Example

To prefix the custom mbeans with type=Platform:

<mbean-name type=Platform</mbean-name>
mbean-query

Used in: mbean

Description

The mbean-query element contains a JMX ObjectName query pattern. The query pattern is executed against a local MBean Server and the resulting objects are registered with the Coherence Management infrastructure. This allows the for a single point of consolidation of MBeans for the grid.

Example

<mbean-query>java.lang:*</mbean-query>

Will include all the MBeans under the java.lang domain in the Coherence management infrastructure.

Notes

- A local MBean Server must be enabled.
mbeans

Used in: management-config

Description

The mbeans element is the root element of the custom mbean configuration file. It contains a list of mbean elements to be instantiated and registered with the coherence management infrastructure.

Elements

Table K–3 describes the elements you can define within the mbeans element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;mbean&gt;</td>
<td>Required</td>
<td>Specifies the MBean type, implementation, and ObjectName that will be instantiated and registered with the Coherence Management service.</td>
</tr>
</tbody>
</table>
Command Line Overrides

Both the Coherence Operational Configuration deployment descriptor tangosol-coherence.xml and the Coherence Cache Configuration deployment descriptor coherence-cache-config.xml can assign a Java command line option name to any element defined in the descriptor. Some elements already have these Command Line Setting Overrides defined. You can create your own or change the predefined ones.

This feature is useful when you need to change the settings for a single JVM, or to be able to start different applications with different settings without making them use different descriptors. The most common application is passing a different multicast address and/or port to allow different applications to create separate clusters.

To create a Command Line Setting Override, add a system-property attribute, specifying the string you would like to assign as the name for the java command line option to the element you want to create an override to. Then, specify it in the Java command line, prepended with "-D".

Override Example

For example, to create an override for the IP address of the multi-home server to avoid using the default localhost, and instead specify a specific the IP address of the interface we want Coherence to use (for instance, 192.168.0.301). We would like to call this override tangosol.coherence.localhost.

First, add a system-property to the cluster-config, unicast-listener, or address element:

```
<address system-property="tangosol.coherence.localhost">localhost</address>
```

which will look as follows with the property we added:

```
<address system-property="tangosol.coherence.localhost">localhost</address>
```

Then use it by modifying the Java command line:

```
java -jar coherence.jar
```

Specify the IP address, 192.168.0.301 (instead of the default localhost specified in the configuration) as follows:

```
java -Dtangosol.coherence.localhost=192.168.0.301 -jar coherence.jar
```
Preconfigured Override Values

Table L–1 lists all of the preconfigured override values:

<table>
<thead>
<tr>
<th>Override Option</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>tangosol.coherence.cluster</td>
<td>Cluster name. See &quot;member-identity&quot; on page H-27.</td>
</tr>
<tr>
<td>tangosol.coherence.clusteraddress</td>
<td>Cluster (multicast) IP address. See &lt;multicast-listener-address&gt; subelement of &quot;multicast-listener&quot; on page H-29.</td>
</tr>
<tr>
<td>tangosol.coherence.clusterport</td>
<td>Cluster (multicast) IP port. See &lt;multicast-listener-port&gt; subelement of &quot;multicast-listener&quot; on page H-29.</td>
</tr>
<tr>
<td>tangosol.coherence.distributed.backup</td>
<td>Data backup storage location. See backup-storage/type subelement in &quot;DistributedCache Service Parameters&quot; on page I-3.</td>
</tr>
<tr>
<td>tangosol.coherence.distributed.backupcount</td>
<td>Number of data backups. See backup-count subelement in &quot;DistributedCache Service Parameters&quot; on page I-3.</td>
</tr>
<tr>
<td>tangosol.coherence.distributed.localstorage</td>
<td>Local partition management enabled. See local-storage subelement in &quot;DistributedCache Service Parameters&quot; on page I-3.</td>
</tr>
<tr>
<td>tangosol.coherence.distributed.threads</td>
<td>Thread pool size. See thread-count subelement in &quot;DistributedCache Service Parameters&quot; on page I-3.</td>
</tr>
<tr>
<td>tangosol.coherence.distributed.transfer</td>
<td>Partition transfer threshold. See transfer-threshold subelement in &quot;DistributedCache Service Parameters&quot; on page I-3.</td>
</tr>
<tr>
<td>tangosol.coherence.invocation.threads</td>
<td>Invocation service thread pool size. See thread-count subelement in &quot;InvocationService Parameters&quot; on page I-8.</td>
</tr>
<tr>
<td>tangosol.coherence.localhost</td>
<td>Unicast IP address. See &lt;unicast-listener-address&gt; subelement in &quot;unicast-listener&quot; on page H-52.</td>
</tr>
<tr>
<td>tangosol.coherence.localport</td>
<td>Unicast IP port. See &lt;unicast-listener-port&gt; subelement in &quot;unicast-listener&quot; on page H-52.</td>
</tr>
<tr>
<td>tangosol.coherence.localport.adjust</td>
<td>Unicast IP port auto assignment. See &lt;unicast-listener-auto&gt; subelement in &quot;unicast-listener&quot; on page H-52.</td>
</tr>
<tr>
<td>tangosol.coherence.log</td>
<td>Logging destination. See &lt;logging-config-destination&gt; subelement in &quot;logging-config&quot; on page H-21.</td>
</tr>
<tr>
<td>tangosol.coherence.log.level</td>
<td>Logging level. See &lt;logging-config-level&gt; subelement in &quot;logging-config&quot; on page H-21.</td>
</tr>
<tr>
<td>tangosol.coherence.machine</td>
<td>Machine name. See &quot;member-identity&quot; on page H-27.</td>
</tr>
<tr>
<td>Override Option</td>
<td>Setting</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
</tr>
<tr>
<td>tangosol.coherence.management</td>
<td>JMX management mode. See &quot;management-config&quot; on page H-25.</td>
</tr>
<tr>
<td>tangosol.coherence.management.readonly</td>
<td>JMX management read-only flag. &quot;management-config&quot; on page H-25.</td>
</tr>
<tr>
<td>tangosol.coherence.management.remote</td>
<td>Remote JMX management enabled flag. See &quot;management-config&quot; on page H-25.</td>
</tr>
<tr>
<td>tangosol.coherence.member</td>
<td>Member name. See &quot;member-identity&quot; on page H-27.</td>
</tr>
<tr>
<td>tangosol.coherence.mode</td>
<td>Operational mode. See &quot;license-config&quot; on page H-20.</td>
</tr>
<tr>
<td>tangosol.coherence.override</td>
<td>Deployment configuration override filename.</td>
</tr>
<tr>
<td>tangosol.coherence.priority</td>
<td>Priority. See &quot;member-identity&quot; on page H-27.</td>
</tr>
<tr>
<td>tangosol.coherence.process</td>
<td>Process name &quot;member-identity&quot; on page H-27.</td>
</tr>
<tr>
<td>tangosol.coherence.rack</td>
<td>Rack name. See &quot;member-identity&quot; on page H-27.</td>
</tr>
<tr>
<td>tangosol.coherence.role</td>
<td>Role name. See &quot;member-identity&quot; on page H-27.</td>
</tr>
<tr>
<td>tangosol.coherence.security</td>
<td>Cache access security enabled flag. See &quot;security-config&quot; on page H-44.</td>
</tr>
<tr>
<td>tangosol.coherence.security.keystore</td>
<td>Security access controller keystore file name. See &quot;security-config&quot; on page H-44.</td>
</tr>
<tr>
<td>tangosol.coherence.security.password</td>
<td>Keystore or cluster encryption password. &quot;Encryption Filters&quot; on page 8-1.</td>
</tr>
<tr>
<td>tangosol.coherence.security.permissions</td>
<td>Security access controller permissions file name. See &quot;security-config&quot; on page H-44.</td>
</tr>
<tr>
<td>tangosol.coherence.site</td>
<td>Site name. See &quot;member-identity&quot; on page H-27.</td>
</tr>
<tr>
<td>tangosol.coherence.tcmp.enabled</td>
<td>TCMP enabled flag. See &lt;packet-publisher-enabled&gt; subelement in &quot;packet-publisher&quot; on page H-39.</td>
</tr>
<tr>
<td>tangosol.coherence.tcpring</td>
<td>!TCP ring enabled flag. See &quot;tcp-ring-listener&quot; on page H-50.</td>
</tr>
<tr>
<td>tangosol.coherence.wka</td>
<td>Well known IP address. See &quot;well-known-addresses&quot; on page H-55.</td>
</tr>
<tr>
<td>tangosol.coherence.wka.port</td>
<td>Well known IP port. See &quot;well-known-addresses&quot; on page H-55.</td>
</tr>
</tbody>
</table>
The following sections in this appendix provide information on deploying Coherence to various platforms.

- Deploying to AIX
- Deploying to Oracle JRockit JVMs
- Deploying to Cisco Switches
- Deploying to Foundry Switches
- Deploying to IBM BladeCenters
- Deploying to IBM JVMs
- Deploying to Linux
- Deploying to OS X
- Deploying to Solaris
- Deploying to Sun JVMs
- Deploying to Virtual Machines
- Deploying to Windows
- Deploying to z OS

**Deploying to AIX**

When deploying Coherence on AIX please be aware of the following:

**Socket Buffers sizes and JVMs**

There is an issue with IBM's 1.4.2, and 1.5 JVMs for AIX which may prevent them from allocating socket buffers larger than 64K (Oracle 2MB). This issue has been addressed in IBM's 1.4.2 SR7 SDK and 1.5 SR3 SDK. See “Operating System Tuning” on page 19-1.

**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 you will need to 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 Oracle JRockit JVMs

When deploying Coherence on JRockit JVMs please be aware of the following:

JRockit and the Native Posix Thread Library (NPTL)

When running JRockit on Linux, Oracle recommends using 2.6 kernels, and ensuring that the NPTL is enabled. Please see Oracle’s documentation regarding this issue.

AtomicLong

When available Coherence will make use of the highly concurrent AtomicLong class, which allows concurrent atomic updates to long values without requiring synchronization. JRockit 1.4 JVMs do not fully support AtomicLong and thus if Coherence detects that it is being run on a JRockit 1.4 JVM it will default to a safe but slower synchronized implementation, and will output the following log message.

sun.misc.AtomicLong is not supported on this JVM; using a synchronized counter.

Upgrading to JRockit 1.5 will allow the use of the highly concurrent implementation.

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 are the parameters to configure this setting on JRockit JVMs:

-XXexitOnOutOfMemory

Deploying to Cisco Switches

When deploying Coherence with Cisco switches please be aware of the following:

Buffer Space and Packet Pauses

Under heavy UDP packet load some Cisco switches may run out of buffer space and exhibit frequent multi-second communication pauses. 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 configuration 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 will cause a partial communication disconnect for the associated Coherence node(s) and they will be 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=xxx.xxx.x.xxx, Port=8088, MachineId=48991)) to this member (Member(Id=1, Timestamp=Sat Sept 13 11:51:11 EST 2008, Address=xxx.xxx.x.xxx, Port=8088, MachineId=49002)) as recently as 5 seconds ago.

To confirm the issue you may run the using the same multicast address and port as the running cluster. If the issue affects a multicast test node its logs will show that at some point it will suddenly stop receiving multicast test messages. See Chapter 15, "Performing a Multicast Connectivity Test".

The following test logs show the issue:

Example M–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 16:44:24 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: 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: 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: 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: 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,
Deploying to IBM BladeCenters

Platform-Specific Deployment Considerations

When deploying Coherence on IBM BladeCenters please be aware of the following:

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 will generally have an even MAC address and the second will have 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 will all be 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, assuming that the IP addresses do not follow the same even/odd pattern.
- This setting would need to be applied across all switches carrying cluster traffic.
- Randomize the MAC address assignments by swapping them between the first and second NIC on alternating machines.

Note that at 16:44:27 the first test node stops receiving multicast packets from other machines. The operating system continues to properly forward multicast traffic from other processes on the same machine, 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 Coherence well-known-addresses feature. See “well-known-addresses” on page H-55.

Deploying to Foundry Switches

When deploying Coherence with Foundry switches please be aware of the following:

Multicast Connectivity

Foundry switches have shown to exhibit difficulty in handing multicast traffic. When deploying on with Foundry switches it is recommend that you use the to ensure that all machines which will be part of the Coherence cluster can communicate over multicast. See Chapter 15, "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 "well-known-addresses" on page H-55.
Deploying to IBM JVMs

When deploying Coherence on IBM JVMs please be aware of the following:

UDP Socket Buffer Sizes

There is an issue with IBM’s 1.4.2, and 1.5 JVMs which may prevent them from allocating socket buffers larger than 64K (Note that buffers of 2MB are recommended for Coherence). This issue has been addressed in IBM’s 1.4.2 SR7 SDK and 1.5 SR3 SDK. For performance reasons it is suggested that the patch be applied.

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 are the parameters to configure this setting on IBM JVMs (1.5 and above):

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:


Deploying to Linux

When deploying Coherence on Linux please be aware of the following:

Native POSIX Thread Library (NPTL)

Early versions of the NPTL are prone to deadlock, especially when combined with 2.4 Linux Kernels. The kernel version and NPTL version can be obtained by executing the following commands:

uname -a
getconf GNU_LIBPTHREAD_VERSION

If running on a 2.4 kernel, it is recommended that you avoid using any version of the NPTL, and revert to using LinuxThreads. This can be done by setting the LD_ASSUME_KERNEL environment variable before launching Java.
export LD_ASSUME_KERNEL=2.4.19
getconf GNU_LIBPTHREAD_VERSION

If running on a 2.6 kernel, it is recommended that you use a 1.0 or higher version of NPTL. If upgrading the NPTL version is not possible then it is then recommended that you switch to LinuxThreads.

NPTL related issues are known to occur with Red Hat Linux 9 and Red Hat Enterprise Linux 3, and are also likely to effect any 2.4 based Linux distribution with a backported version of the NPTL. See http://java.sun.com/developer/technicalArticles/JavaTechandLinux/RedHat for more details on this issue.

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 boot 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. 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.

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 machines. 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 OS X

When deploying Coherence on OS X please be aware of the following:

Multicast and IPv6

OS X defaults to running multicast over IPv6 rather then IPv4. If you run in a mixed IPv6/IPv4 environment you will need to 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.

Unique Multicast Addresses and Ports

On OS X it is suggested that each Coherence cluster use a unique multicast address and port, as some versions of OS X will not take both into account when delivering packets. See the multicast-listener for details on configuring the address.
Socket Buffer Sizing

Generally Coherence prefers 2MB or higher buffers, but in the case of 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 sweet spot. See “packet-buffer” on page H-35 for details on specifying the amount of buffer space Coherence will request.

Deploying to Solaris

When deploying Coherence on Solaris please be aware of the following:

Solaris 10 (x86 and SPARC)

Note: If running on Solaris 10, please review Sun issues 102712 and 102741 which relate to packet corruption and multicast disconnections. These will 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.

Sun Alert 102712:
Possible Data Integrity Issues on Solaris 10 Systems Using the e1000g Driver for the Intel Gigabit Network Interface Card (NIC)

Sun Alert 102741:
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

Solaris 10 Networking

If running on Solaris 10, please review Sun issues 102712 and 102741 which relate to packet corruption and multicast disconnections. These will 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.

Deploying to Sun JVMs

When deploying Coherence on Sun JVMs please be aware of the following:

Heap Sizes

With 1.4 JVMs Coherence recommends keeping heap sizes below 1GB in size per JVM. Multiple cache servers can be used allow a single machine to achieve higher capacities. With Sun's 1.5 JVMs, heap sizes beyond 1GB are reasonable, though GC tuning is still advisable to minimize long GC pauses. See Sun's GC Tuning Guide for tuning details. It is also advisable to run with fixed sized heaps as this generally lowers GC times.

AtomicLong

When available Coherence will make use of the highly concurrent AtomicLong class, which allows concurrent atomic updates to long values without requiring
synchronization. Sun 1.4 client JVMs include an implementation which is not stable on some multiprocessor systems. If Coherence detects that it is being run on a Sun 1.4 client JVM it will default to a safe but slower synchronized implementation, and will output the following log message.

sun.misc.AtomicLong is not supported on this JVM; using a synchronized counter.

It is suggested that you run your 1.4 JVMs 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 are the parameters to configure this setting on Sun JVMs:

Unix:

-XX:OnOutOfMemoryError="kill -9 %p"

Windows:

-XX:OnOutOfMemoryError="taskkill /F /PID %p"

---

**Note:** Note: as of December 2008, this flag is available on newer versions of 1.4.2 and 1.6, but not on 1.5.

---

**Deploying to Virtual Machines**

**Supported Deployment**

Coherence is supported within virtual machine environments, and there should see no functional differences between running it there or in a non-virtualized operating system.

**Multicast Connectivity**

Using virtualization adds another layer to your network topology, and like all other layers it must be properly configured to support multicast networking. See "multicast-listener" on page H-29.

**Performance**

It is less likely that a process running in a virtualized OS will be able to fully use gigabit Ethernet. This is not specific to Coherence, and will be visible most network intensive virtualized applications.

See the following VMWare article covering their network performance as compared to non-virtualized operating systems.
Fault Tolerance

From a Coherence fault tolerance perspective there is more configuration which needs to occur to ensure that cache entry backups reside on physically separate hardware. Manual machine identity must be configured so that Coherence can ensure that backups are not inadvertently stored on the same physical machine as the primary. This can be configured by using the machine-id element within the operational configuration file. See the configuration for “unicast-listener” on page H-52 for details.

Deploying to Windows

When deploying Coherence on Windows please be aware of the following:

Performance Tuning

Out of the box Windows is not optimized for background processes and heavy network loads. This may be addressed by running the optimize.reg script included in the Coherence installation’s bin directory. See “Operating System Tuning” on page 19-1 for details on the optimizations which will be performed.

Personal Firewalls

If running a firewall on a machine you may have difficulties in forming a cluster consisting of multiple computers. This can be resolved by either:

- Disable the firewall, though this is generally not recommended.
- Grant full network access to the Java executable which will run Coherence.
- Open up individual address and ports for Coherence.
- By default Coherence will use TCP and UDP ports starting at 8088, subsequent nodes on the same machine will use increasing port numbers. Coherence may also communicate over multicast, the default address and port will differ with based on the release. See “unicast-listener” on page H-52 and “multicast-listener” on page H-29 for details on address and port configuration.

Deploying to z/OS

When deploying Coherence on z/OS please be aware of the following:

EBCDIC

When deploying Coherence into environments where the default character set is EBCDIC rather than ASCII, please make sure that Coherence the 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.MulticastSocket to use the necessary port, but that information appears to be ignored on z/OS when there already 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 "well-known-addresses" on page H-55 for more information.
Best Practices for Coherence Extend

This chapter describes best practices for configuring and running Coherence® Extend. The following sections are included in this chapter:

- Run Proxy Servers with Local Storage Disabled
- Do Not Run a Near Cache on a Proxy Server
- Configure Heap NIO Space to be Equal to the Max Heap Size
- Set Worker Thread Pool Sizes According to the Needs of the Application
- Be Careful When Making InvocationService Calls
- Be Careful When Placing Collection Classes in the Cache
- Run Multiple Proxies Instead of Increasing Thread Pool Size
- Configure POF Serializers for Cache Servers
- Use Node Locking Instead of Thread Locking

Run Proxy Servers with Local Storage Disabled

Each server in a partitioned cache, including the proxy server, can store a portion of the data. The proxy server has the responsibility of accepting POF formatted data from the client (either Java, C++, or .NET), deserializing POF data to get the Java objects, serializing the Java objects, then placing the resulting data in the cluster. These tasks can be expensive in terms of CPU and memory. You can preserve resources on the proxy server by disabling its local storage.

There are several ways in which you can disable storage:

Local storage for a proxy server can be enabled or disabled with the tangosol.coherence.distributed.localstorage Java property. For example:

```
-Dtangosol.coherence.distributed.localstorage=false
```

You can also disable storage in the cache configuration file. See the description of the `<local-storage>` element in "distributed-scheme" on page D-27.

Storage can also be disabled for the proxy server by modifying the `<local-storage>` setting in its `tangosol-coherence.xml` (or `tangosol-coherence-override.xml`) file. Example N–1 illustrates setting `<local-storage>` to `false` in the `tangosol-coherence-override.xml` file.

**Example N–1  Disabling Storage in tangosol-coherence-override.xml**

```xml
<!-- Example using tangosol-coherence-override.xml
```
Do Not Run a Near Cache on a Proxy Server

By definition, a near cache provides local cache access to recently- and/or often-used data. If a proxy server is configured with a near cache, it will locally cache data accessed by its remote clients. It is unlikely that these clients will be consistently accessing the same subset of data, thus resulting in a low hit ratio on the near cache. This will result in higher heap usage and more network traffic on the proxy nodes with little to no benefit. For these reasons, it is recommended that a near cache not be used on a proxy server. To ensure that the proxy server is not running a near cache, remove all near schemes from the cache configuration being used for the proxy. See "Near Cache" for more information.

Configure Heap NIO Space to be Equal to the Max Heap Size

NIO memory is used for the TCP connection into the proxy and for POF serialization and deserialization. Older Java installations tended to run out of heap memory because it was configured too low. Newer Java JDKs will configure off heap NIO space equal to the max heap space. On Sun JVMs, this can also be set manually with this value:

-XX:MaxDirectMemorySize=512M

Set Worker Thread Pool Sizes According to the Needs of the Application

Client applications can be classified into two general categories: active and passive. In active applications, the Coherence*Extend client sends many requests, such as put, get, and so on, to the proxy. These requests are serviced by the proxy service. The proxy will deserialize POF data put into the cache, and serialize data it returns to the client. For these tasks, configure a larger number of daemon (worker) threads for the proxy service.

In passive applications, the client waits on events (such as map listeners) based on some specified criteria. Events are serviced by the DistributedCache service. This service uses worker threads to push events to the client. For these tasks, the thread
pool configuration for the DistributedCache service should include a sufficient number of worker threads.

Note that near caches on extend clients will use map listeners under the covers for the invalidation strategies of ALL, PRESENT, and AUTO. Applications that are write-heavy that use near caches will generate many map events.

**Be Careful When Making InvocationService Calls**

InvocationService allows a member of a service to invoke arbitrary code on any node in the cluster. On Coherence*Extend however, InvocationService calls are serviced by the proxy that the client is connected to by default. When sending the call through a proxy, you cannot choose the particular node on which the code will run.

**Be Careful When Placing Collection Classes in the Cache**

If a Coherence*Extend client puts a collection object, (such as an ArrayList, HashSet, HashMap, and so on) directly into the cache, it is deserialized as an immutable array. If you then extract it and cast it to its original type, then a ClassCastException will be returned. As an alternative, use a Java interface object (such as a List, Set, Map, and so on) or encapsulate the collection object in another object. Both of these techniques are illustrated in the following example:

**Example N–2  Casting an ArrayList Object**

class ExtendExample
{
    @SuppressWarnings({ "unchecked" })
    public static void main(String asArgs[])
    {
        System.setProperty("tangosol.coherence.cacheconfig", "client-config.xml");
        NamedCache cache = CacheFactory.getCache("test");

        // Create a sample collection
        List list = new ArrayList();
        for (int i = 0; i < 5; i++)
        {
            list.add(String.valueOf(i));
        }
        cache.put("list", list);

        List listFromCache = (List) cache.get("list");

        System.out.println("Type of list put in cache: " + list.getClass());
        System.out.println("Type of list in cache: " + listFromCache.getClass());

        Map map = new TreeMap();
        for (Iterator i = list.iterator(); i.hasNext();)
        {
            Object o = i.next();
            map.put(o, o);
        }
        cache.put("map", map);

        Map mapFromCache = (Map) cache.get("map");

        System.out.println("Type of map put in cache: " + map.getClass());
        System.out.println("Type of map in cache: " + mapFromCache.getClass());
    }
}
Run Multiple Proxies Instead of Increasing Thread Pool Size

The proxy performs POF/EL conversions in the service thread. A single proxy instance can easily bottleneck on a single core due to POF/EL conversions. Running multiple proxy instances on the same box (instead of increasing the thread pool size) helps spread the load across more cores.

Configure POF Serializers for Cache Servers

One of the tasks the proxy server performs is to deserialize POF data into Java objects. If you run C++ or .NET applications and store data to the cache, then the conversion to Java objects could be viewed as an unnecessary step. In the current release of Coherence, you have the option of configuring a POF serializer for cache servers. This will have the effect of storing POF format data directly in the cache.

This can have the following impact on your applications:

- .NET or C++ clients that only perform puts or gets will not require a Java version of the object. Java versions will still be required if deserializing on the server side (for entry processors, cache stores, and so on).
- POF serializers remove the requirement to serialize/deserialize on the proxy, thus reducing their memory and CPU requirements.

Example N–3 illustrates a fragment from example-pof-server.xml, which configures a POF serializer for the distributed cache. A full POF configuration file example, is attached to this topic.

Example N–3 Configuring a POF Serializer for a Distributed Cache

```
...<distributed-scheme>
   <scheme-name>dist-default</scheme-name>
   <serializer>
      <class-name>com.tangosol.io.pof.ConfigurablePofContext</class-name>
      <init-params>
         <init-param>
            <param-type>string</param-type>
            <param-value>custom-types-pof-config.xml</param-value>
         </init-param>
      </init-params>
   </serializer>
   <backing-map-scheme>
      <local-scheme/>
   </backing-map-scheme>
   <autostart>true</autostart>
</distributed-scheme>
...
Use Node Locking Instead of Thread Locking

Coherence*Extend clients can send lock, put, and unlock requests to the cluster. The proxy holds the locks for the client. The requests for locking and unlocking can be issued at the thread level or the node level. In thread level locking, a particular thread instance belonging to the proxy (Thread 1, for example) issues the lock request. If any other threads (Thread 3, for example) issue an unlock request, they will be ignored. A successful unlock request can be issued only by the thread that issued the initial lock request. This can cause application errors since unlock requests will not succeed unless the original thread that issues the lock is also the one that receives the request to release the lock.

In node level locking, if a particular thread instance belonging to the proxy (Thread 1, for example) issues the lock request, then any other thread (Thread 3, for example) can successfully issue an unlock request.

As an alternative to using locks, Coherence recommends that you use the EntryProcessor API instead. EntryProcessors are described in “Chapter 2, Implement Transactions, Locks, and Concurrency.”
Coherence provides a data grid by dynamically, transparently, and automatically partitioning the data set across all storage enabled nodes in a cluster. We have been doing some scale out testing on our new Dual 2.3GHz PowerPC G5 Xserve cluster and here is one of the tests that we have performed using the data grid aggregation feature.

The new InvocableMap tightly binds the concepts of a data grid (that is, partitioned cache) and the processing of the data stored in the grid. When you take the InvocableMap and combine it with the linear scalability of Coherence itself you get an extremely powerful solution. The following tests show that you can take an application that Coherence provides you (the developer, the engineer, the architect, and so on) the ability to build an application when that can scale out to handle any SLA requirement, any increase in throughput requirements. For example, the following test demonstrate Coherence’s ability to linearly increase the number of trades aggregated per second as you increase hardware. That is, if one machine can aggregate $X$ trades per second, if you add a second machine to the data grid you will be able to aggregate $2X$ trades per second, if you add a third machine to the data grid you will be able to aggregate $3X$ trades per second and so on.

All of the Data Grid capabilities described below are features of Coherence Enterprise Edition and higher.

The Data

First, we need some data to aggregate. Example O–1 illustrates a Trade object with a three properties Id, Price, and Symbol.

**Example O–1  Trade Object Defining Three Properties**

```java
package com.tangosol.examples.coherence.data;

import com.tangosol.util.Base;
import com.tangosol.util.ExternalizableHelper;
import com.tangosol.io.ExternalizableLite;
import java.io.IOException;
import java.io.NotActiveException;
import java.io.DataInput;
import java.io.DataOutput;

/**

```
* Example Trade class
*
* @author erm 2005.12.27
*/
public class Trade
extends Base
implements ExternalizableLite
{
    /**<
     * Default Constructor
     */
    public Trade()
    {
    }

    public Trade(int iId, double dPrice, String sSymbol)
    {
        setId(iId);
        setPrice(dPrice);
        setSymbol(sSymbol);
    }

    public int getId()
    {
        return m_iId;
    }

    public void setId(int iId)
    {
        m_iId = iId;
    }

    public double getPrice()
    {
        return m_dPrice;
    }

    public void setPrice(double dPrice)
    {
        m_dPrice = dPrice;
    }

    public String getSymbol()
    {
        return m_sSymbol;
    }

    public void setSymbol(String sSymbol)
    {
        m_sSymbol = sSymbol;
    }

    /**<
     * Restore the contents of this object by loading the object's state from the
     * passed DataInput object.
     *
     * @param in the DataInput stream to read data from to restore the
     *           state of this object
     *
     * @throws IOException        if an I/O exception occurs
     */
Configure a Partitioned Cache

The cache configuration is easy through the XML Cache Configuration Elements. Example O–2 defines one wildcard cache-mapping that maps to one caching-scheme which has unlimited capacity:

Example O–2 Mapping a cache-mapping to a caching-scheme with Unlimited Capacity

```xml
<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">
<cache-config>
  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>*</cache-name>
      <scheme-name>example-distributed</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>
  <caching-schemes>
    <!--
    Distributed caching scheme.
    -->
    <distributed-scheme>
      <scheme-name>example-distributed</scheme-name>
      <service-name>DistributedCache</service-name>
    </distributed-scheme>
  </caching-schemes>
</cache-config>
```

```java
/**
 * @throws NotActiveException if the object is not in its initial state, and
 * therefore cannot be deserialized into
 */
public void readExternal(DataInput in)
    throws IOException
{
    m_iId     = ExternalizableHelper.readInt(in);
    m_dPrice  = in.readDouble();
    m_sSymbol = ExternalizableHelper.readSafeUTF(in);
}

/**
 * Save the contents of this object by storing the object's state into the
 * passed DataOutput object.
 *
 * @param out the DataOutput stream to write the state of this object to
 *
 * @throws IOException if an I/O exception occurs
 */
public void writeExternal(DataOutput out)
    throws IOException
{
    ExternalizableHelper.writeInt(out, m_iId);
    out.writeDouble(m_dPrice);
    ExternalizableHelper.writeSafeUTF(out, m_sSymbol);
}

private int    m_iId;
private double m_dPrice;
private String m_sSymbol;
```
Add an Index to the Price Property

Example O–3 illustrates the code to add an index to the Price property. Adding an index to this property increases performance by allowing Coherence to access the values directly rather than having to deserialze each item to accomplish the calculation.

Example O–3  Adding an Index to the Price Property

```java
ReflectionExtractor extPrice  = new ReflectionExtractor("getPrice");
m_cache.addIndex(extPrice, true, null);
```

In our tests the aggregation speed was increased by more than 2x after an index was applied.

Code to perform a Parallel Aggregation

Example O–4 illustrates the code to perform a parallel aggregation across all JVMs in the data grid. The aggregation is initiated and results received by a single client. That is, a single "low-power" client is able to use the full processing power of the cluster/data grid in aggregate to perform this aggregation in parallel with just one line of code.

Example O–4  Perform a Parallel Aggregation Across all JVMs in the Grid

```java
Double DResult;
DResult = (Double) m_cache.aggregate((Filter) null, new DoubleSum("getPrice"));
```

The Testing Environment and Process

Performing a "Test Run"

A test run does several things:
1. Loads 200,000 trade objects into the data grid.
2. Adds indexes to **Price** property.
3. Performs a **parallel** aggregation of all trade objects stored in the data grid. This aggregation step is done 20 times to obtain an "average run time" to ensure that the test takes into account garbage collection.
4. Loads 400,000 trade objects into the data grid.
5. Repeats steps 2 and 3.
6. Loads 600,000 trade objects into the data grid.
7. Repeats steps 2 and 3.
8. Loads 800,000 trade objects into the data grid.
9. Repeats steps 2 and 3.
10. Loads 1,000,000 trade objects into the data grid.
11. Repeats steps 2 and 3.

**Client Considerations:** The test client itself is run on an Intel Core Duo iMac which is marked as local storage disabled. The command line used to start the client was:

```
java ... -Dtangosol.coherence.distributed.localstorage=false -Xmx128m -Xms128m com.tangosol.examples.coherence.invocable.TradeTest
```

**This "Test Suite" (and Subsequent Results) Includes Data from Four "Test Runs":**

1. Start 4 JVMs on one Xserve - Perform a "test run"
2. Start 4 JVMs on each of two Xserves - Perform a "test run"
3. Start 4 JVMs on each of three Xserves - Perform a "test run"
4. Start 4 JVMs on each of four Xserves - Perform a "test run"

**Server Considerations:** In this case a "JVM" refers to a cache server instance (that is, a data grid node) that is a standalone JVM responsible for managing/storing the data. I used the DefaultCacheServer helper class to accomplish this.

The command line used to start the server was:

```
java ... -Xmx384m -Xms384m -server com.tangosol.net.DefaultCacheServer
```

**JDK Version**

The JDK used on both the client and the servers was Java 2 Runtime Environment, Standard Edition (build 1.5.0_05-84)

**The Results**

As you can see in the following graph the average aggregation time for the aggregations decreases **linearly** as more cache servers/machines are added to the data grid!

---

**Note:** The lowest and highest times were not used in the calculations below resulting in a data set of eighteen results used to create an average.
Similarly, the following graph illustrates how the aggregations per second scales linearly as you add more machines! When moving from 1 machine to 2 machines the trades aggregated per second double, when moving from 2 machines to 4 machines the trades aggregated per second double again.
Combining the Coherence data grid (that is, partitioned cache) with the InvocableMap features enables:

- Applications to scale out data grid calculations linearly;
- Groups to meet increasingly aggressive SLAs by dynamically/transparently adding more resources to the data grid. That is, if you need to achieve 1,837,932 trade aggregations per second all that is required is to start 16 more cache servers across four more machines.

**Note:** FAILOVER!

The above aggregations will complete successfully and correctly even if one of the cache servers or and entire machine fails during the aggregation!
The Portable Object Format (POF) allows object values to be encoded into a binary stream in such a way that the platform/language origin of the object value is both irrelevant and unknown. The Portable Invocation Format (PIF) allows method invocations to be similarly encoded into a binary stream. These two formats (referred together as PIF-POF) are derived from a common binary encoding substrate that is described in this appendix. The binary format is provided here for informative purposes and is not a requirement for using PIF-POF. See "The Portable Object Format" in the Oracle Coherence Getting Started Guide for more information on using PIF-POF.

The following sections are included in this appendix:

- **Stream Format**
- **Binary Formats for Predefined Types**
- **Binary Format for User Types**

### Stream Format

The PIF-POF stream format is octet-based; a PIF-POF stream is a sequence of octet values. For the sake of clarity, this documentation treats all octets as unsigned 8-bit integer values in the range 0x00 to 0xFF (decimal 0 to 255). Byte-ordering is explicitly not a concern since (in PIF-POF) a given octet value that is represented by an unsigned 8-bit integer value is always written and read as the same unsigned 8-bit integer value.

A PIF stream contains exactly one Invocation. An Invocation consists of an initial POF stream that contains an Integer Value for the remaining length of the Invocation, immediately followed by a POF stream that contains an Integer Value that is the conversation identifier, immediately followed by a POF stream that contains a User Type value that is the message object. The remaining length indicates the total number of octets used to encode the conversation identifier and the message object; the remaining length is provided so that a process receiving an Invocation is able to determine when the Invocation has been fully received. The conversation identifier is used to support multiple logical clients and services multiplexed through a single connection, just as TCP/IP provides multiple logical port numbers for a given IP address. The message object is defined by the particular high-level conversational protocol.

A POF stream contains exactly one Value. The Value contains a Type Identifier, and if the Type Identifier does not imply a value, then it is immediately trailed by a data structure whose format is defined by the Type Identifier.
Integer Values

The stream format relies extensively on the ability to encode integer values in a compact form. Coherence refers to this integer binary format as a **packed integer**. This format uses an initial octet and one or more trailing octets as necessary; it is a variable-length format.

Table P–1 describes the three regions in the first octet.

<table>
<thead>
<tr>
<th>Region Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80</td>
<td>Continuation indicator</td>
</tr>
<tr>
<td>0x40</td>
<td>Negative indicator</td>
</tr>
<tr>
<td>0x3F</td>
<td>integer value (6 binary LSDs)</td>
</tr>
</tbody>
</table>

Table P–2 describes the two regions in the trailing octets.

<table>
<thead>
<tr>
<th>Region Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80</td>
<td>Continuation indicator</td>
</tr>
<tr>
<td>0x7F</td>
<td>integer value (next 7 binary LSDs)</td>
</tr>
</tbody>
</table>

Example P–1 illustrates writing a 32-bit integer value to an octet stream as supported in Coherence.

**Example P–1  Writing a 32-bit Integer Value to an Octet Stream**

```java
public static void writeInt(DataOutput out, int n)
    throws IOException {
    int b = 0;
    if (n < 0) {
        b = 0x40;
        n = ~n;
    }
    b |= (byte) (n & 0x3F);
    n >>>= 6;
    while (n != 0) {
        b |= 0x80;
        out.writeByte(b);
        b = (n & 0x7F);
        n >>>= 7;
    }
    out.writeByte(b);
}
```

Example P–2 illustrates reading a 32-bit integer value from an octet stream as supported in Coherence.

**Example P–2  Reading a 32-bit Integer Value from an Octet Stream**

```java
public static int readInt(DataInput in)
    throws IOException {
    int b = 0;
    if ((b = in.readByte()) & 0x80) {
        b = in.readByte() | (b & 0x7F);
        b <<= 7;
    }
    return b;
}
```
```java
{  
    int b = in.readUnsignedByte();
    int n = b & 0x3F;
    int cBits = 6;
    boolean fNeg = (b & 0x40) != 0;
    while ((b & 0x80) != 0)  
    {
        b = in.readUnsignedByte();
        n |= ((b & 0x7F) << cBits);
        cBits += 7;
    }
    if (fNeg)
    {
        n = ~n;
    }
    return n;
}
```

Integer values used within this documentation without an explicit Type Identifier are assumed to be 32-bit signed integer values that have a decimal range of \(-2^{31}\) to \(2^{31}-1\).

Table P–3 illustrates some integer value examples.

<table>
<thead>
<tr>
<th>Value</th>
<th>Binary Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x00</td>
</tr>
<tr>
<td>1</td>
<td>0x01</td>
</tr>
<tr>
<td>2</td>
<td>0x02</td>
</tr>
<tr>
<td>99</td>
<td>0xA301</td>
</tr>
<tr>
<td>9999</td>
<td>0x8F9C01</td>
</tr>
<tr>
<td>-1</td>
<td>0x40</td>
</tr>
<tr>
<td>-2</td>
<td>0x41</td>
</tr>
<tr>
<td>-99</td>
<td>0xE201</td>
</tr>
<tr>
<td>-9999</td>
<td>0xCE9C01</td>
</tr>
</tbody>
</table>

**Type Identifiers**

A Type Identifier is encoded in the binary stream as an Integer Value. Type Identifiers greater than or equal to zero are user Type Identifiers. Type Identifiers less than zero are predefined ("intrinsic") type identifiers.

Table P–4 lists the predefined identifiers.

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1 (0x40)</td>
<td>int16</td>
</tr>
<tr>
<td>-2 (0x41)</td>
<td>int32</td>
</tr>
<tr>
<td>-3 (0x42)</td>
<td>int64</td>
</tr>
<tr>
<td>-4 (0x43)</td>
<td>int128*</td>
</tr>
<tr>
<td>-5 (0x44)</td>
<td>float32</td>
</tr>
<tr>
<td>-6 (0x45)</td>
<td>float64</td>
</tr>
</tbody>
</table>
Type Identifiers less than or equal to -33 are a combination of a type and a value. This form is used to reduce space for these commonly used values.

Table P–5 lists the type identifiers that combine type and value.

**Table P–5**  **Type Identifiers that Combine a Type and a Value**

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-33 (0x60)</td>
<td>boolean:false</td>
</tr>
<tr>
<td>-34 (0x61)</td>
<td>boolean:true</td>
</tr>
<tr>
<td>-35 (0x62)</td>
<td>string:zero-length</td>
</tr>
<tr>
<td>-36 (0x63)</td>
<td>collection:empty</td>
</tr>
<tr>
<td>-37 (0x64)</td>
<td>reference:null</td>
</tr>
</tbody>
</table>
The PIF-POF Binary Format

This section describes the binary formats for the predefined (“intrinsic”) type identifiers that are supported with PIF-POF. The types are: int, Decimal, Floating Point, Boolean, Octet, Octet String, Char, Char String, Date, Year-Month Interval, Time, Time Interval, Date-Time, Date-Time Interval, Collections, Arrays, Sparse Arrays, Key-Value Maps (Dictionaries), Identity, and Reference.

### Int

Four signed integer types are supported: int16, int32, int64, and int128. If a type identifier for one of the integer types is encountered in the stream, it is immediately followed by an Integer Value.

#### Table P-5 (Cont.) Type Identifiers that Combine a Type and a Value

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-38 (0x65)</td>
<td>floating-point:+infinity</td>
</tr>
<tr>
<td>-39 (0x66)</td>
<td>floating-point:-infinity</td>
</tr>
<tr>
<td>-40 (0x67)</td>
<td>floating-point:NaN</td>
</tr>
<tr>
<td>-41 (0x68)</td>
<td>int:-1</td>
</tr>
<tr>
<td>-42 (0x69)</td>
<td>int:0</td>
</tr>
<tr>
<td>-43 (0x6A)</td>
<td>int:1</td>
</tr>
<tr>
<td>-44 (0x6B)</td>
<td>int:2</td>
</tr>
<tr>
<td>-45 (0x6C)</td>
<td>int:3</td>
</tr>
<tr>
<td>-46 (0x6D)</td>
<td>int:4</td>
</tr>
<tr>
<td>-47 (0x6E)</td>
<td>int:5</td>
</tr>
<tr>
<td>-48 (0x6F)</td>
<td>int:6</td>
</tr>
<tr>
<td>-49 (0x70)</td>
<td>int:7</td>
</tr>
<tr>
<td>-50 (0x71)</td>
<td>int:8</td>
</tr>
<tr>
<td>-51 (0x72)</td>
<td>int:9</td>
</tr>
<tr>
<td>-52 (0x73)</td>
<td>int:10</td>
</tr>
<tr>
<td>-53 (0x74)</td>
<td>int:11</td>
</tr>
<tr>
<td>-54 (0x75)</td>
<td>int:12</td>
</tr>
<tr>
<td>-55 (0x76)</td>
<td>int:13</td>
</tr>
<tr>
<td>-56 (0x77)</td>
<td>int:14</td>
</tr>
<tr>
<td>-57 (0x78)</td>
<td>int:15</td>
</tr>
<tr>
<td>-58 (0x79)</td>
<td>int:16</td>
</tr>
<tr>
<td>-59 (0x7A)</td>
<td>int:17</td>
</tr>
<tr>
<td>-60 (0x7B)</td>
<td>int:18</td>
</tr>
<tr>
<td>-61 (0x7C)</td>
<td>int:19</td>
</tr>
<tr>
<td>-62 (0x7D)</td>
<td>int:20</td>
</tr>
<tr>
<td>-63 (0x7E)</td>
<td>int:21</td>
</tr>
<tr>
<td>-64 (0x7F)</td>
<td>int:22</td>
</tr>
</tbody>
</table>
The four signed integer types vary only by the length that is required to support the largest value of the type using the common “twos complement” binary format. The Type Identifier, one of int16, int32, int64, or int128 is followed by an Integer Value in the stream. If the Integer Value is outside of the range supported by the type (-2^15 to 2^15-1 for int16, -2^31 to 2^31-1, for int32, -2^63 to 2^63-1 for int64, or -2^127 to 2^127-1 for int128,) then the result is undefined and may be bitwise truncation or an exception.

Additionally, there are a number of Type Identifiers that combine the int designation with a value into a single byte for purpose of compactness. As a result, these Type Identifiers are not followed by an Integer Value in the stream, since the value is included in the Type Identifier.

Table P–6 illustrates these type identifiers.

<table>
<thead>
<tr>
<th>Value</th>
<th>int16</th>
<th>int32</th>
<th>int64</th>
<th>int128</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x69</td>
<td>0x69</td>
<td>0x69</td>
<td>0x69</td>
</tr>
<tr>
<td>1</td>
<td>0x6A</td>
<td>0x6A</td>
<td>0x6A</td>
<td>0x6A</td>
</tr>
<tr>
<td>2</td>
<td>0x6B</td>
<td>0x6B</td>
<td>0x6B</td>
<td>0x6B</td>
</tr>
<tr>
<td>99</td>
<td>0x40A301</td>
<td>0x41A301</td>
<td>0x42A301</td>
<td>0x43A301</td>
</tr>
<tr>
<td>9999</td>
<td>0x408F9C01</td>
<td>0x418F9C01</td>
<td>0x428F9C01</td>
<td>0x438F9C01</td>
</tr>
<tr>
<td>-1</td>
<td>0x68</td>
<td>0x68</td>
<td>0x68</td>
<td>0x68</td>
</tr>
<tr>
<td>-2</td>
<td>0x4041</td>
<td>0x4141</td>
<td>0x4241</td>
<td>0x4341</td>
</tr>
<tr>
<td>-99</td>
<td>0x40E201</td>
<td>0x41E201</td>
<td>0x42E201</td>
<td>0x43E201</td>
</tr>
<tr>
<td>-9999</td>
<td>0x40CE9C01</td>
<td>0x41CE9C01</td>
<td>0x42CE9C01</td>
<td>0x43CE9C01</td>
</tr>
</tbody>
</table>

The Java type equivalents are short (int16), int (int32), long (int64) and BigInteger (int128). Since BigInteger is capable of representing much larger values, it is not possible to encode all BigInteger values in the int128 form; values out of the int128 range are basically unsupported, and would result in an exception or would use a different encoding, such as a string encoding.

Coercion of Integer Types

To enable the efficient representation of numeric data types, an integer type is coerced into any of the following types by a stream recipient:

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1 (0x40)</td>
<td>int16</td>
</tr>
<tr>
<td>-2 (0x41)</td>
<td>int32</td>
</tr>
<tr>
<td>-3 (0x42)</td>
<td>int64</td>
</tr>
<tr>
<td>-4 (0x43)</td>
<td>int128</td>
</tr>
<tr>
<td>-5 (0x44)</td>
<td>float32</td>
</tr>
<tr>
<td>-6 (0x45)</td>
<td>float64</td>
</tr>
<tr>
<td>-7 (0x46)</td>
<td>float128</td>
</tr>
<tr>
<td>-8 (0x47)</td>
<td>decimal32</td>
</tr>
</tbody>
</table>
In other words, if the recipient reads any of the above types from the stream and it encounters an encoded integer value, it automatically converts that value into the expected type. This capability allows a set of common (that is, small-magnitude) octet, character, integer, decimal and floating-point values to be encoded using the single-octet integer form (Type Identifiers in the range -41 to -64).

For purposes of unsigned types, the integer value -1 is translated to 0xFF for the octet type, and to 0xFFFF for the char type. (In the case of the char type, this does unfortunately seem to imply a UTF-16 platform encoding; however, it does not violate any of the explicit requirements of the stream format.)

### Decimal

There are three floating-point decimal types supported: decimal32, decimal64, and decimal128. If a type identifier for one of the decimal types is encountered in the stream, it is immediately followed by two packed integer values. The first integer value is the unscaled value, and the second is the scale. These values are equivalent to the parameters to the constructor of Java's BigDecimal class: 

```java
java.math.BigDecimal(BigInteger unscaledVal, int scale).
```

In addition to the coercion of integer values into decimal values supported as described in "Coercion of Integer Types" on page P-6, the constant type+value identifiers listed in Table P–8 are used to indicate special values supported by IEEE 754r.

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-38 (0x65)</td>
<td>floating-point:+infinity</td>
</tr>
<tr>
<td>-39 (0x66)</td>
<td>floating-point:-infinity</td>
</tr>
<tr>
<td>-40 (0x67)</td>
<td>floating-point:NaN</td>
</tr>
</tbody>
</table>

Java does not provide a standard (that is, portable) decimal type; rather, it has the awkward BigDecimal implementation that was intended originally for internal use in Java's cryptographic infrastructure. In Java, the decimal values for positive and negative infinity, as well as not-a-number (NaN), are not supported.

### Floating Point

Three base-2 floating point types are supported: float32, float64, and float128. If a type identifier for one of the floating point types is encountered in the stream, it is immediately followed by a fixed-length floating point value, whose binary form is defined by IEEE 754/IEEE754r. Floating point numbers are written to the stream using the IEEE 754 format, and using the IEEE 754r format for the float128 type.
In addition to the coercion of integer values into decimal values as described in "Coercion of Integer Types" on page P-6, the constants in Table P–9 are used to indicate special values supported by IEEE-754.

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-38 (0x65)</td>
<td>floating-point:+infinity</td>
</tr>
<tr>
<td>-39 (0x66)</td>
<td>floating-point:-infinity</td>
</tr>
<tr>
<td>-40 (0x67)</td>
<td>floating-point:NaN</td>
</tr>
</tbody>
</table>

Other special values defined by IEEE-754 are encoded using the full 32-bit, 64-bit or 128-bit format, and may not be supported on all platforms. Specifically, by not providing any means to differentiate among them, Java only supports one `NaN` value.

**Boolean**

If the type identifier for Boolean occurs in the stream, it is followed by an integer value, which represents the Boolean value `false` for the integer value of zero, or `true` for all other integer values.

While it is possible to encode Boolean values as described in "Coercion of Integer Types" on page P-6, the only values for the Boolean type are `true` and `false`. As such, the only expected binary formats for Boolean values are the predefined (and compact) forms described in Table P–10.

<table>
<thead>
<tr>
<th>Type ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-33 (0x60)</td>
<td>boolean:false</td>
</tr>
<tr>
<td>-34 (0x61)</td>
<td>boolean:true</td>
</tr>
</tbody>
</table>

**Octet**

If the type identifier for Octet occurs in the stream, it is followed by the octet value itself, which is by definition in the range 0 to 255 (0x00 to 0xFF). As described in "Coercion of Integer Types" on page P-6, the compact form of integer values can be used for Octet values, with the integer value -1 being translated as 0xFF.

Table P–11 lists the integer values that may be used as Octet values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Octet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0x00)</td>
<td>0x69</td>
</tr>
<tr>
<td>1 (0x01)</td>
<td>0x6A</td>
</tr>
<tr>
<td>2 (0x02)</td>
<td>0x6B</td>
</tr>
<tr>
<td>99 (0x63)</td>
<td>0x4B63</td>
</tr>
<tr>
<td>254 (0xFE)</td>
<td>0x4BFE</td>
</tr>
<tr>
<td>255 (0xFF)</td>
<td>0x68</td>
</tr>
</tbody>
</table>
Octet String

If the type identifier for Octet String occurs in the stream, it is followed by an Integer Value for the length \( n \) of the string, and then \( n \) octet values.

An Octet String of zero length is encoded using the "string:zero-length" Type Identifier.

Char

If the type identifier for Char occurs in the stream, it is followed by a UTF-8 encoded character. As described in the section on "Coercion of Integer Types" on page P-6, the compact form of integer values may be used for Char values, with the integer value -1 being translated as 0xFFFF.

Example P–3 illustrates writing a character value to an octet stream.

Example P–3 Writing a Character Value to an Octet Stream

```java
public static void writeChar(DataOutput out, int ch)
throws IOException
{
    if (ch >= 0x0001 && ch <= 0x007F)
    {
        // 1-byte format: 0xxx xxxx
        out.write((byte) ch);
    }
    else if (ch <= 0x07FF)
    {
        // 2-byte format: 110x xxxx, 10xx xxxx
        out.write((byte) (0xC0 | ((ch >>> 6) & 0x1F)));
        out.write((byte) (0x80 | ((ch ) & 0x3F)));
    }
    else
    {
        // 3-byte format: 1110 xxxx, 10xx xxxx, 10xx xxxx
        out.write((byte) (0xE0 | ((ch >>> 12) & 0x0F)));
        out.write((byte) (0x80 | ((ch >>> 6) & 0x3F)));
        out.write((byte) (0x80 | ((ch ) & 0x3F)));
    }
}
```

Example P–4 illustrates reading a character value from an octet stream.

Example P–4 Reading a Character Value from an Octet Stream

```java
public static char readChar(DataInput in)
throws IOException
{
    char ch;

    int b = in.readUnsignedByte();
    switch ((b & 0xF0) >>> 4)
    {
    case 0x0: case 0x1: case 0x2: case 0x3:  
    case 0x4: case 0x5: case 0x6: case 0x7:  
    // 1-byte format: 0xxx xxxx
        ch = (char) b;
        break;
    case 0xC: case 0xD:
    ```
P-10

Oracle Coherence Developer's Guide

// 2-byte format: 110x xxxx, 10xx xxxx
int b2 = in.readUnsignedByte();
if ((b2 & 0xC0) != 0x80)
    {
        throw new UTFDataFormatException();
    }
ch = (char) (((b & 0x1F) << 6) | b2 & 0x3F);
break;
}
case 0xE:
    {
        // 3-byte format: 1110 xxxx, 10xx xxxx, 10xx xxxx
        int n = in.readUnsignedShort();
        int b2 = n >>> 8;
        int b3 = n & 0xFF;
        if ((b2 & 0xC0) != 0x80 || (b3 & 0xC0) != 0x80)
            {
                throw new UTFDataFormatException();
            }
        ch = (char) (((b & 0x0F) << 12) |
            ((b2 & 0x3F) << 6) |
            b3 & 0x3F);
        break;
    }
default:
    {
        throw new UTFDataFormatException(
            "illegal leading UTF byte: " + b);
    }

return ch;
}

Char String

If the type identifier for Char String occurs in the stream, it is followed by an Integer Value for the length \(n\) of the UTF-8 representation string in octets, and then \(n\) octet values composing the UTF-8 encoding described above. Note that the format length-encodes the octet length, not the character length.

A Char String of zero length is encoded using the string:zero-length Type Identifier. Table P–12 illustrates the Char String formats.

<table>
<thead>
<tr>
<th>Values</th>
<th>Char String Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x62 (or 0x4E00)</td>
<td>&quot;ok&quot; 0x4E026F6B</td>
</tr>
</tbody>
</table>

Date

Date values are passed using ISO8601 semantics. If the type identifier for Date occurs in the stream, it is followed by three Integer Values for the year, month and day, in the ranges as defined by ISO8601.
Year-Month Interval

If the type identifier for Year-Month Interval occurs in the stream, it is followed by two Integer Values for the number of years and the number of months in the interval.

Time

Time values are passed using ISO8601 semantics. If the type identifier for Time occurs in the stream, it is followed by five Integer Values, which may be followed by two more Integer Values. The first four Integer Values are the hour, minute, second and fractional second values. Fractional seconds are encoded in one of three ways:

- 0 indicates no fractional seconds.
- [1..999] indicates the number of milliseconds.
- [-1..-999999999] indicates the negated number of nanoseconds.

The fifth Integer Value is a time zone indicator, encoded in one of three ways:

- 0 indicates no time zone.
- 1 indicates Universal Coordinated Time (UTC).
- 2 indicates a time zone offset, which is followed by two more Integer Values for the hour offset and minute offset, as described by ISO8601.

The variable fractional and time zone encodings do add complexity to the parsing of a Time Value, but provide for much more complete support of the ISO8601 standard and the variability in the precision of clocks, while achieving a high degree of binary compactness. While time values tend to have no fractional encoding or millisecond encoding, the trend over time is toward higher time resolution.

Time Interval

If the type identifier for Time Interval occurs in the stream, it is followed by four Integer Values for the number of hours, minutes, seconds and nanoseconds in the interval.

Date-Time

Date-Time values are passed using ISO8601 semantics. If the type identifier for Date-Time occurs in the stream, it is followed by eight or ten Integer Values, which correspond to the Integer Values that compose the Date and Time values.

Coercion of Date and Time Types

Date Value can be coerced into a Date-Time Value. Time Value can be coerced into a Date-Time Value. Date-Time Value can be coerced into either a Date Value or a Time Value.

Day-Time Interval

If the type identifier for Day-Time Interval occurs in the stream, it is followed by five Integer Values for the number of days, hours, minutes, seconds and nanoseconds in the interval.

Collections

A collection of values, such as a bag, a set, or a list, are encoded in a POF stream using the Collection type. Immediately following the Type Identifier, the stream contains the
Collection Size, an Integer Value indicating the number of values in the Collection, which is greater than or equal to zero. Following the Collection Size, is the first value in the Collection (if any), which is itself encoded as a Value. The values in the Collection are contiguous, and there is exactly \( n \) values in the stream, where \( n \) is equal to the Collection Size.

If all the values in the Collection have the same type, then the Uniform Collection format is used. Immediately following the Type Identifier (uniform-collection), the uniform type of the values in the collection is written to the stream, followed by the Collection Size \( n \) as an Integer Value, followed by \( n \) values \textbf{without their Type Identifiers}. Note that values in a Uniform Collection cannot be assigned an identity, and that (as a side-effect of the explicit type encoding) an empty Uniform Collection has an explicit content type.

Table P–13 illustrates examples of Collection and Uniform Collection formats for several values.

**Table P–13  Collection and Uniform Collection Formats for Various Values**

<table>
<thead>
<tr>
<th>Values</th>
<th>Collection Format</th>
<th>Uniform Collection Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x63 (or 0x5500)</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0x55016A</td>
<td>0x56410101</td>
</tr>
<tr>
<td>1,2,3</td>
<td>0x55036A6B6C</td>
<td>0x564103010203</td>
</tr>
<tr>
<td>1, &quot;ok&quot;</td>
<td>0x55026A4E026F6B</td>
<td>n/a</td>
</tr>
</tbody>
</table>

**Arrays**

An indexed array of values is encoded in a POF stream using the Array type. Immediately following the Type Identifier, the stream contains the Array Size, an Integer Value indicating the number of elements in the Array, which must be greater than or equal to zero. Following the Array Size is the value of the first element of the Array (the zero index), assuming that there is at least one element in the array, which is itself encoded using as a Value. The values of the elements of the Array are contiguous, and there must be exactly \( n \) values in the stream, where \( n \) is equal to the Array Size.

If all the values of the elements of the Array have the same type, then the Uniform Array format is used. Immediately following the Type Identifier (uniform-array), the uniform type of the values of the elements of the Array is written to the stream, followed by the Array Size \( n \) as an Integer Value, followed by \( n \) values \textbf{without their Type Identifiers}. Note that values in a Uniform Array cannot be assigned an identity, and that (as a side-effect of the explicit type encoding) an empty Uniform Array has an explicit array element type.

Table P–14 illustrates examples of Array and Uniform Array formats for several values.

**Table P–14  Array and Uniform Array Formats for Various Values**

<table>
<thead>
<tr>
<th>Values</th>
<th>Array Format</th>
<th>Uniform Array Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x63 (or 0x5700)</td>
<td>0x63 (or 0x584100) – This example assumes an element type of Int32.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0x57016A</td>
<td>0x58410101</td>
</tr>
<tr>
<td>1,2,3</td>
<td>0x57036A6B6C</td>
<td>0x584103010203</td>
</tr>
<tr>
<td>1, &quot;ok&quot;</td>
<td>0x57026A4E026F6B</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Sparse Arrays

For arrays whose element values are sparse, the Sparse Array format allows indexes to be explicitly encoded, implying that any missing indexes have a default value. The default value is false for the Boolean type, zero for all numeric, octet and char types, and null for all reference types. The format for the Sparse Array is the Type Identifier (sparse-array), followed by the Array Size \( n \) as an Integer Value, followed by not more than \( n \) index/value pairs, each of which is composed of an array index encoded as an Integer Value \( i \) \((0 \leq i < n)\) whose value is greater than the previous element's array index, and an element value encoded as a Value; the Sparse Array is finally terminated with an illegal index of -1.

If all the values of the elements of the Sparse Array have the same type, then the Uniform Sparse Array format is used. Immediately following the Type Identifier (uniform-sparse-array), the uniform type of the values of the elements of the Sparse Array is written to the stream, followed by the Array Size \( n \) as an Integer Value, followed by not more the \( n \) index/value pairs, each of which is composed of an array index encoded as an Integer Value \( i \) \((0 <= i < n)\) whose value is greater than the previous element's array index, and a element value encoded as a Value without a Type Identifier; the Uniform Sparse Array is finally terminated with an illegal index of -1. Note that values in a Uniform Sparse Array cannot be assigned an identity, and that (as a side-effect of the explicit type encoding) an empty Uniform Sparse Array has an explicit array element type.

Table P–15 illustrates examples of Sparse Array and Uniform Sparse Array formats for several values.

<table>
<thead>
<tr>
<th>Values</th>
<th>Sparse Array format</th>
<th>Uniform Sparse Array format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x63 (or 0x590040)</td>
<td>0x63 (or 0x5A410040) – This example assumes an element type of Int32.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0x5901006A40</td>
<td>0x5A4101000140</td>
</tr>
<tr>
<td>1,2,3</td>
<td>0x5903006A016B026C40</td>
<td>0x5A410300010102020340</td>
</tr>
<tr>
<td>1,5,9</td>
<td>0x5909006A046E087240</td>
<td>0x5A410900010405080940</td>
</tr>
<tr>
<td>1,&quot;ok&quot;</td>
<td>0x5905006A044E026F6B40</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Key-Value Maps (Dictionaries)

For key/value pairs, a Key-Value Map (also known as Dictionary data structure) format is used. There are three forms of the Key-Value Map binary encoding:

- The generic map encoding is a sequence of keys and values;
- The uniform-keys-map encoding is a sequence of keys of a uniform type and their corresponding values;
- The uniform-map encoding is a sequence of keys of a uniform type and their corresponding values of a uniform type.

The format for the Key-Value Map is the Type Identifier (map), followed by the Key-Value Map Size \( n \) as an Integer Value, followed by \( n \) key/value pairs, each of which is composed of a key encoded as Value, and a corresponding value encoded as a Value.

Table P–16 illustrates several examples of key/value pairs and their corresponding binary format.
If all of the keys of the Key-Value Map are of a uniform type, then the encoding uses a more compact format, starting with the Type Identifier (uniform-keys-map), followed by the Type Identifier for the uniform type of the keys of the Key-Value Map, followed by the Key-Value Map Size $n$ as an Integer Value, followed by $n$ key/value pairs, each of which is composed of a key encoded as a Value \textit{without a Type Identifier}, and a corresponding value encoded as a Value.

Table P–17 illustrates several examples of the binary formats for Key/Value pairs where the Keys are of uniform type.

<table>
<thead>
<tr>
<th>Values</th>
<th>Binary format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1=&quot;ok&quot;</td>
<td>0xB01A4E026F6B</td>
</tr>
<tr>
<td>1=&quot;ok&quot;, 2=&quot;no&quot;</td>
<td>0xB02A4E026F6B6B4E026E6F</td>
</tr>
</tbody>
</table>

If all of the keys of the Key-Value Map are of a uniform type, and all the corresponding values of the map are also of a uniform type, then the encoding uses a more compact format, starting with the Type Identifier (uniform-map), followed by the Type Identifier for the uniform type of the keys of the Key-Value Map, followed by the Type Identifier for the uniform type of the values of the Key-Value Map, followed by the Key-Value Map Size $n$ as an Integer Value, followed by $n$ key/value pairs, each of which is composed of a key encoded as a Value \textit{without a Type Identifier}, and a corresponding value encoded as a Value \textit{without a Type Identifier}.

Table P–18 illustrates several examples of the binary formats for Key/Value pairs where the Keys and Values are of uniform type.

<table>
<thead>
<tr>
<th>Values</th>
<th>Binary format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1=&quot;ok&quot;</td>
<td>0x5C4101014E026F6B</td>
</tr>
<tr>
<td>1=&quot;ok&quot;, 2=&quot;no&quot;</td>
<td>0x5C4102014E026F6B024E026E6F</td>
</tr>
</tbody>
</table>

### Identity

If the type identifier for Identity occurs in the stream, it is followed by an Integer Value, which is the Identity. Following the Identity is the value that is being identified, which is itself encoded as a Value.

Any value within a POF stream that occurs more than once is labeled with an Identity, and subsequent instances of that value within the same POF stream are replaced with a Reference. For platforms that support "by reference" semantics, the identity represents a serialized form of the actual object identity.
An Identity is an Integer Value that is greater than or equal to zero. A value within the POF stream will have at most one Identity. Note that values within a uniform data structure cannot be assigned an identity.

Reference

A Reference is a pointer to an Identity that has already been encountered inside the current POF stream, or a null pointer.

For platforms that support "by reference" semantics, the reference in the POF stream becomes a reference in the realized (deserialized) object, and a null reference in the POF stream becomes a null reference in the realized object. For platforms that do not support "by reference" semantics, and for cases in which a null reference is encountered in the POF stream for a non-reference value (for example, a primitive property in Java), the default value for the type of value is used.

Table P–19 illustrates examples of binary formats for several "by reference" semantics.

<table>
<thead>
<tr>
<th>Value</th>
<th>Binary Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id #1</td>
<td>0x5F01</td>
</tr>
<tr>
<td>Id #350</td>
<td>0x5F9E05</td>
</tr>
<tr>
<td>null</td>
<td>0x60</td>
</tr>
</tbody>
</table>

Support for forward and outer references is not required by POF. In POF, both the identity that is referenced and the value that is being referenced by the identity have already occurred within the POF stream. In the first case, a reference will not be made to an identity that has not yet been encountered, and in the second case, a reference will not be made within a complex value (such as a collection or a user type) to that complex value itself.

Binary Format for User Types

All non-intrinsic types are referred to as User Types. User Types are composed of zero or more indexed values (also known as fields, properties, and attributes), each of which has a Type Identifier. Furthermore, User Types are versioned, supporting both forward and backward compatibility.

User Types have a Type Identifier with a value greater than or equal to zero. The Type Identifier has no explicit or self-describing meaning within the stream itself; in other words, a Value does not contain a type (or "class") definition. Instead, the encoder (the sender) and the decoder (the receiver) share an implicit understanding, called a Context, which includes the necessary meta-data, including the user type definitions.

The binary format for a User Type is very similar to that of a Sparse Array; conceptually, a User Type can be considered a Sparse Array of property values. The format for User Types is the Type Identifier (an Integer Value greater than or equal to zero), followed by the Version Identifier (an Integer Value greater than or equal to zero), followed by index/value pairs, each of which is composed of a Property Index encoded as an Integer Value \( i \) \((0 <= i)\) whose value is greater than the previous Property Index, and a Property Value encoded as a Value; the User Type is finally terminated with an illegal Property Index of -1.
Like the Sparse Array, any property that is not included as part of the User Type encoding is assumed to have a default value. The default value is false for the Boolean type, zero for all numeric, octet and char types, and null for all reference types.

**Versioning of User Types**

Versioning of User Types supports the addition of properties to a User Type, but not the replacement or removal of properties that existed in previous versions of the User Type. By including the versioning capability as part of the general binary contract, it is possible to support both backwards and forwards compatibility.

When a sender sends a User Type value of a version $v_1$ to a receiver that supports version $v_2$ of the same User Type, the receiver uses default values for the additional properties of the User Type that exist in $v_2$ but do not exist in $v_1$.

When a sender sends a User Type value of a version $v_2$ to a receiver that only supports version $v_1$ of the same User Type, the receiver treats the additional properties of the User Type that exist in $v_2$ but do not exist in $v_1$ as opaque. If the receiver must store the value (persistently), or if the possibility exists that the value will ever be sent at a later point, then the receiver stores those additional opaque properties for later encoding. Sufficient type information is included to allow the receiver to store off the opaque property values in either a typed or binary form; when the receiver re-encodes the User Type, it must do so using the Version Indicator $v_2$, since it is including the unaltered $v_2$ properties.
The following sections are included in this appendix:

- TCMP Log Messages
- Configuration Log Messages
- Partitioned Cache Service Log Messages

**TCMP Log Messages**

The following are TCMP-related log messages:

**Experienced a \( n_1 \) ms communication delay (probable remote GC) with Member \( s \)**

**Parameters:** \( n_1 \) - 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 acknowledgement 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 acknowledgement has been received from the specified node, and that it has likely emerged from its GC.

**Action:** Prolonged and frequent GC’s can adversely affect cluster performance and availability. If these warnings are seen frequently, review your JVM heap and GC configuration and tuning. See Chapter 19, "Performance Tuning,” for more details.

**Failed to satisfy the variance: allowed=\( n_1 \) actual=\( n_2 \)**

**Parameters:** \( n_1 \) - the maximum allowed latency in milliseconds; \( n_2 \) - 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 UDP communications 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” on page H-17. Failure to satisfy that latency causes this message to be logged and increases the latency threshold, which will be reflected in a follow up message.

Action: If the latency consistently stays very high (over 100 milliseconds), consult your network administrator and see Chapter 16, “Performing a Datagram Test for Network Performance.”

Created a new cluster "%s1" with Member(%s2)

Parameters: %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, see "multicast-listener" on page H-29), 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, see "member-identity" on page H-27, or calculated based on the multicast listener address and port, or the "well-known-addresses" on page H-55 list). The Member information includes the node id, creation timestamp, unicast address and port, location, process id, role, etc.)

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)

Parameters: %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

Parameters: %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
TCMP Log Messages

Log Message Glossary

Q-3

Member(%s) left Cluster with senior member %n

Parameters:  %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 (e.g. 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)

Parameters:  %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 nodes earlier than others. Most commonly, a node connected via 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.

Service %s joined the cluster with senior service member %n

Parameters:  %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 known at this time, it will be shown as "n/a".

Action:  None.

Member %n1 joined Service %s with senior member %n2

Parameters:  %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

Parameters: %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

Parameters: %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. That same message is sent in the case of 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)

Parameters: %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
TCMP Log Messages

Log Message Glossary

Q-5

Rejecting connection to member %n using TcpSocket(%s)

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.

Parameters: %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)

Parameters: %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 UDP 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 or not 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.

**Parameters:** \%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; b) extremely long GC pause; c) swapped out process. ) In that case, other cluster nodes may choose to revoke the cluster membership from 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).

---

**Configuration Log Messages**

The following are configuration-related log messages:

**java.io.IOException: Configuration file is missing: "tangosol-coherence.xml"**

**Parameters:** n/a  
**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"**

**Parameters:** \%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 via system properties or programatically, verify that the reported URI matches the expected location.
Loaded operational overrides from "%s"

Parameters:  %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 via system properties, descriptor override or programmatically, verify that the reported URI matches the expected location.

Optional configuration override "%s" is not specified

Parameters:  %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

Parameters:  %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

Parameters:  %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"

Parameters:  %s - the URI (file or resource) of the cache configuration descriptor
Severity:  3-Informational
Partitioned Cache Service Log Messages

The following are partitioned cache-related log messages:

**Asking member %n1 for %n2 primary partitions**

**Parameters:**  
%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 at the end of the re-distribution process. This message demarcates a beginning of the transfer request to a specified node for a number of partitions to move toward the “fair” ownership distribution.

**Action:** None.

**Transferring %n1 out of %n2 primary partitions to member %n3 requesting %n4**

**Parameters:**  
%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)**

**Parameters:**  
%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
Partitioned Cache Service Log Messages

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 machines that are different from the primary owners' machines. 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

Parameters: \%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

Parameters: \%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

Parameters: \%n1

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 will postpone subsequent scheduled distribution checks.

Action: None.
Partitioned Cache Service Log Messages

Limiting primary transfer to \%n1 KB (\%n2 partitions)

**Parameters:** \%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 requestor. The size of the transfer is limited by the `<transfer-threshold>` element located within a `<distributed-scheme>` 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

**Parameters:** \%n1 - the time in milliseconds before the next distribution check will be 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 will wait 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

**Parameters:** \%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)

**Parameters:** \%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
Partitioned Cache Service Log Messages

Log Message Glossary

Q-11

%n1> Ownership conflict for partition %n2 with member %n3 (%n4!=%n5)

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.

Parameters: %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.

Assigned %n1 orphaned primary partitions

Parameters: %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.

java.lang.RuntimeException: Storage is not configured

Parameters: None

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.
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.

Parameters:  %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 will 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 (e.g., 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 occured during filter evaluation: %s; removing the filter...

Parameters:  %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 registered on this cache. As a result, some MapEvents may not have been issued. Additionally, to prevent further failures, the filter (and associated MapListener) will be 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...

Parameters:  %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 registered on this cache. As a result, some MapEvents may not have been issued. Additionally, to prevent further failures, the filter (and associated MapListener) will be 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 index rebuild: %s

Parameters: %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

Parameters: %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

Parameters: %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 implementation of the Filter 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

Parameters: %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 will properly instantiate a backing map for the specified cache.

BackingMapManager %s1: failed to instantiate a cache: %s2

Parameters: %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 will properly instantiate a backing map for the specified cache.

BackingMapManager %s1: failed to release a cache: %s2

Parameters: %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 will properly release a backing map for the specified cache.

Unexpected event during backing map operation: key=%s1; expected=%s2; actual=%s3

Parameters: %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
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.

Parameters:  
%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

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 (e.g. 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.

Repeating %s1 for %n1 out of %n2 items due to re-distribution of %s2

Parameters:  
%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 (e.g. due to re-distribution), the request is rejected by the (former) partition owner and is automatically resent to the new partition owner.

Action: None.