Oracle® Coherence

Release Notes for Oracle Coherence

Release 3.4.1

E14002-02

January 2009

ORACLE®

Oracle Coherence Release Notes for Oracle Coherence, Release 3.4.1

E14002-02

Primary Author: Thomas Pfaeffle

Contributing Author: Noah Arliss, Jason Howes, Mark Falco, Alex Gleyzer, Gene Gleyzer, PatrickPeralta

Contents

Preface

This document describes changes and enhancements that have been made to the Oracle Coherence product since the 3.4 release.

Audience

This document is intended for users of Oracle Coherence.

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.

TTY Access to Oracle Support Services

To reach AT&T Customer Assistants, dial 711 or 1.800.855.2880. An AT&T Customer Assistant will relay information between the customer and Oracle Support Services at 1.800.223.1711. Complete instructions for using the AT&T relay services are available at `http://www.consumer.att.com/relay/tty/standard2.html`. After the AT&T Customer Assistant contacts Oracle Support Services, an Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process.

Related Documents

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

- *Getting Started with Oracle Coherence*
- *User's Guide for Oracle Coherence*
- *Developer's Guide for Oracle Coherence*

Conventions

The following text conventions are used in this document:

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

1

Technical Changes and Enhancements

This chapter describes the changes and enhancements made to the Oracle Coherence product since the 3.4 release. This document is accurate at the time of publication. Oracle updates the release notes periodically after the software release.

Oracle Coherence for Java 3.4.1

The following is a list of new features, improvements, and bug fixes in Oracle Coherence for Java 3.4.1

Coherence*Web Enhancements and Fixes

- Added support for Servlet 2.5 containers such as JBoss 4.2.2, Tomcat 6.x, Jetty 6.1.x and Resin 3.1.
- Fixed Servlet 2.4 specification (SRV 6.2.2 and SRV 8.2) compliancy violations.
- Added a high-performance lock-free HTTP session access mode.
- Prevented inconsistent sticky optimization strategy caused by failed Invocation service startup.
- Optimized the memory footprint of idle HTTP sessions.
- Added logging messages that identify HTTP sessions locked by hung or unresponsive threads.
- Added ability to restrict session identifiers to alphanumeric characters.

Coherence*Extend Enhancements and Fixes

- Added a work around to avoid high latencies for small requests on Windows Server 2008.
- Fixed a regression causing a sporadic `ClassCastException` during filter-based `invokeAll` calls.
- Improved the connection termination algorithm for unresponsive or overloaded connections.
- Fixed bugs causing potential `IllegalStateExceptions` and lost events upon proxied cache service restart.

Management Framework Fixes

- Fixed a bug in the `Reporter` causing a failure to start management service on nodes without JMX libraries.

- Fixed a regression in the "refresh-behind" policy causing intermittent failures to retrieve remote attributes.
- Fixed a regression that made the `SeniorMemberId` attribute invisible.

Other Enhancements

- Fixed a regression in Coherence 3.4 causing a potential "Storage not configured" exception during failover.
- The `ReplicatedCache` made fully POF compatible.
- Added log messages helping to identify the cause of a cluster node departure.
- Added the ability to configure logging configuration programmatically.
- Certified the P13N cache provider with WebLogic Portal 10.x.
- Fixed a regression in `CacheFactory` to avoid potential deadlock during initialization when running on Azul platform.
- Implemented work-around for JVM NIO bug on Windows.
- Fixed the `GroupAggregator` so that it returns correct results when used with the `DistinctValues` aggregator.
- Fixed WKA list consistency verification across cluster nodes.

Oracle Coherence for .NET 3.4.1

The following is a list of new features, improvements, and bug fixes in Oracle Coherence for .NET 3.4.1:

- Fixed `NearCache` front `put` so it will not ignore the specified expiration.
- Enhanced POF so it can now be configured from a daemon thread within a `ASP.NET` application.
- Enhanced the Logging framework to work with `Common.Logging` version 1.2.0.
- Fixed the `GroupAggregator` so that it returns correct results when used with the `DistinctValues` aggregator.
- Enhanced the `<include>` element in the `pof-config.xsd` to support web resource URIs.

Oracle Coherence for C++ 3.4.1

The following is a list of new features, improvements, and bug fixes in Oracle Coherence for C++ 3.4.1:

Platform and Environment Support

- Improved support for pre-4.1 GCC versions
- Added Windows Vista support

Other Enhancements and Fixes:

- Near cache front `put` may ignore expiry for caches when invalidation strategy is `NONE`
- Resolved potential for deadlock in `Object::acquireMemberWriteLock`

- Provided for client Subject propagation to remote cache
- Added support for instantiating thread-safe handles outside of managed classes.

2

Documentation Errata

This chapter describes changes, enhancements, and corrections made to the existing Oracle Coherence library (Getting Started, Developer's Guide, and User's Guide). The current Oracle Coherence documentation library can be found at the following URL:

`http://download.oracle.com/docs/cd/E13924_01/index.htm`

This chapter contains the following information:

Installing Coherence Web Session Management Module (3.4.1)

Coherence*Web Session Management Module: Supported Web Containers

Table 2–1 summarizes the Web containers supported by the Coherence*Web Session Management Module and the installation information specific to each supported web container.

Table 2–1 Web Containers Supported by Coherence*Web

Application Server	Server Type Alias*	See this Installation Section
Apache Tomcat 4.1.x	Generic	"General Instructions for Installing Coherence*Web Session Management Module"
Apache Tomcat 5.0.x	Generic	"General Instructions for Installing Coherence*Web Session Management Module"
Apache Tomcat 5.5.x	Generic	"General Instructions for Installing Coherence*Web Session Management Module"
Apache Tomcat 6.0.x	Generic	"General Instructions for Installing Coherence*Web Session Management Module"
Caucho Resin 3.0.x	Resin/3.0.x	"Installing Coherence*Web Session Management Module on Caucho Resin 3.0.x"
Caucho Resin 3.1.x	Resin/3.1.x	"Installing Coherence*Web Session Management Module on Caucho Resin 3.1.x"
IronFlare Orion 2.0.x	Orion/2.0.x	"General Instructions for Installing Coherence*Web Session Management Module"
IBM WebSphere 5.x	WebSphere/5.x	"General Instructions for Installing Coherence*Web Session Management Module"
IBM WebSphere 6.x	WebSphere/6.x	"General Instructions for Installing Coherence*Web Session Management Module"
JBoss Application Server	Jetty/4.2.x, Jetty/5.1.x, *or* Generic	"General Instructions for Installing Coherence*Web Session Management Module"
Jetty 4.2.x	Jetty/4.2.x	"General Instructions for Installing Coherence*Web Session Management Module"
Jetty 5.1.x	Jetty/5.1.x	"General Instructions for Installing Coherence*Web Session Management Module"
Jetty 6.1.x	Generic	"General Instructions for Installing Coherence*Web Session Management Module"
New Atlanta ServletExec 5.0	ServletExec/5.x	"General Instructions for Installing Coherence*Web Session Management Module"
Oracle OC4J 10.1.2.x	Oracle/10.1.2.x	"General Instructions for Installing Coherence*Web Session Management Module"
Oracle OC4J 10.1.3.x	Oracle/10.1.3.x	"General Instructions for Installing Coherence*Web Session Management Module"
Oracle WebLogic 8.x	WebLogic/8.x	"General Instructions for Installing Coherence*Web Session Management Module"

*Table 2–1 (Cont.) Web Containers Supported by Coherence*Web*

Application Server	Server Type Alias*	See this Installation Section
Oracle WebLogic 9.x	WebLogic/9.x	"General Instructions for Installing Coherence*Web Session Management Module"
Oracle WebLogic 10.x	WebLogic/10.x	"Installing Coherence*Web Session Management Module on Oracle WebLogic 10.x"
Sun ONE 6.x	Generic	"General Instructions for Installing Coherence*Web Session Management Module"
Sun ONE 7.x	Generic	"General Instructions for Installing Coherence*Web Session Management Module"
Sun ONE 8.x	Generic	"General Instructions for Installing Coherence*Web Session Management Module"

Notes:

* The server type alias passed to the Coherence*Web installer though the `-server` command line option.

General Instructions for Installing Coherence*Web Session Management Module

To enable Coherence*Web in your Java EE application, you must run a ready-to-deploy application (recommended) through the automated installer before deploying it. The automated installer prepares the application for deployment.

To install Coherence*Web for the Java EE application you are deploying:

- Make sure that the application directory and the `.ear` file or `.war` file are not being used or accessed by another process.
- 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.
- Complete the application inspection step by running the following command and specifying the full path to your application and the name of your server found in the chart above (replacing the `<app-path>` and `<server-type>` with them in the command line below):

  ```
  java -jar webInstaller.jar <app-path> -inspect -server:<server-type>
  ```

  The system will create (or update, if it already exists), the coherence-web.xml configuration descriptor file for your Java EE application in the directory where the application is located. This configuration descriptor contains the default Coherence*Web settings for your application recommended by the installer. You may proceed to the install step or review and modify the settings, based on your requirements, before running the install step. For example, you can enable certain features by setting the `context-param` options in the `coherence-web.xml` configuration descriptor:

  – The setting in Table 2–2 will cluster all `ServletContext` ("global") attributes so that servers in a cluster will share the same values for those attributes, and

will also receive the events specified by the Servlet Specification when those attributes change:

Table 2–2 Settings to Cluster ServletContext Attributes

Parameter	Value
param-name	coherence-servletcontext-clustered
param-value	true

– The setting in Table 2–3 allows an application to enumerate all of the sessions that exist within the application, or to obtain any one of those sessions to examine or manipulate:

Table 2–3 Settings to Enumerate All Sessions in the Application

Parameter	Value
param-name	coherence-enable-sessioncontext
param-value	true

– The setting in Table 2–4 enables you to increase the length of the `HttpSession` ID, which is generated using a `SecureRandom` algorithm; the length can be any value, although in practice it should be small enough to fit into a cookie or a URL (depending on how session IDs are maintained.) Increasing the length can decrease the chance of a session being purposefully hijacked:

Table 2–4 Settings to Increase Length of HTTPSession ID

Parameter	Value
param-name	coherence-session-id-length
param-value	32

– By default, the `HttpSession` ID is managed in a cookie. If the application supports URL encoding, set the option described in Table 2–5 to enable it:

Table 2–5 Settings to Support URI Encoding

Parameter	Value
param-name	coherence-session-urlencode-enabled
param-value	true

After double-checking that these changes have been made, save the file and exit the editor; remember to return back to the Coherence library directory if you are working from a shell or command line.

- Go through the Coherence*Web application installation step by running the following command and specifying the full path to your application (replacing the `<app-path>` with it in the command line below):

```
java -jar webInstaller.jar <app-path> -install
```

The installer requires a valid `coherence-web.xml` configuration descriptor for its use in the same directory in which the application is located.

- Deploy the updated application and verify that everything functions as expected, using the load balancer if necessary. Please remember that the load balancer is only intended for testing and should not be used in a production environment.

Installing Coherence*Web Session Management Module on Caucho Resin 3.0.x

Follow these additional steps when installing the Coherence*Web Session Management Module into a Caucho Resin 3.0.x server:

1. From within the Coherence library directory, extract the `coherence-web.jar` from the `webInstaller.jar`:

   ```
   jar -xvf webInstaller.jar web-install/coherence-web.jar
   ```

   This will extract the `coherence-web.jar` file into a subdirectory named `web-install`. Use the following commands to move the `coherence-web.jar` file up one level into the `library` directory:

   On Windows:

   ```
   move web-install\coherence-web.jar .
   rmdir web-install
   ```

   On UNIX:

   ```
   mv web-install/coherence-web.jar .
   rmdir web-install
   ```

2. For each Resin install that will be running in the server cluster, update the libraries using the following command (note that it is broken up into multiple lines only for formatting purposes; this is a single command entered on one line):

   ```
   java -cp coherence.jar;coherence-web.jar
   com.tangosol.coherence.servlet.WebPluginInstaller <resin-home-path> -install
   ```

   For example, on Windows:

   ```
   java -cp coherence.jar;coherence-web.jar
   com.tangosol.coherence.servlet.WebPluginInstaller C:\opt\resin30 -install
   ```

3. Follow the general installation steps for the Coherence*Web Session Management Module, specifying Resin/3.0.x for the server type.

Installing Coherence*Web Session Management Module on Caucho Resin 3.1.x

Follow these additional steps when installing the Coherence*Web Session Management Module into a Caucho Resin 3.1.x server:

1. From within the Coherence library directory, extract the `coherence-web.jar` from the `webInstaller.jar`:

   ```
   jar -xvf webInstaller.jar web-install/coherence-web.jar
   ```

   This will extract the `coherence-web.jar` file into a subdirectory named `web-install`. Use the following commands to move the `coherence-web.jar` file up one level into the `library` directory:

   On Windows:

   ```
   move web-install\coherence-web.jar .
   rmdir web-install
   ```

On UNIX:

```
mv web-install/coherence-web.jar .
rmdir web-install
```

2. For each Resin install that will be running in the server cluster, update the libraries using the following command (note that it is broken up into multiple lines only for formatting purposes; this is a single command entered on one line):

```
java -cp coherence.jar;coherence-web.jar
com.tangosol.coherence.servlet.WebPluginInstaller <resin-home-path> -install
```

For example, on Windows:

```
java -cp coherence.jar;coherence-web.jar
com.tangosol.coherence.servlet.WebPluginInstaller C:\opt\resin31 -install
```

3. Follow the general installation steps for the Coherence*Web Session Management Module, specifying Resin/3.1.x for the server type.

Installing Coherence*Web Session Management Module on Oracle OC4J 10.1.2.x

Follow these additional steps when installing the Coherence*Web Session Management Module into a Oracle OC4J 10.1.2.x server:

1. From within the Coherence `library` directory, extract the `coherence-web.jar` from the `webInstaller.jar`:

```
jar -xvf webInstaller.jar web-install/coherence-web.jar
```

This will extract the `coherence-web.jar` file into a subdirectory named `web-install`. Use the following commands to move the `coherence-web.jar` file up one level into the `library` directory:

On Windows:

```
move web-install\coherence-web.jar .
rmdir web-install
```

On UNIX:

```
mv web-install/coherence-web.jar .
rmdir web-install
```

2. For each OC4J install that will be running in the server cluster, update the libraries using the following command (note that it is broken up into multiple lines only for formatting purposes; this is a single command entered on one line):

```
java -cp coherence.jar;coherence-web.jar
com.tangosol.coherence.servlet.WebPluginInstaller <oc4j-home-path> -install
```

For example, on Windows:

```
java -cp coherence.jar;coherence-web.jar
com.tangosol.coherence.servlet.WebPluginInstaller C:\opt\oracle1012\j2ee\home
```

3. Follow the general installation steps for the Coherence*Web Session Management Module, specifying Oracle/10.1.2.x for the server type.

Installing Coherence*Web Session Management Module on Oracle WebLogic 10.x

Follow these additional steps when installing the Coherence*Web Session Management Module into an Oracle WebLogic 10.x server:

1. From within the Coherence library directory, extract the `coherence-web.jar` from the `webInstaller.jar`:

   ```
   jar -xvf webInstaller.jar web-install/coherence-web.jar
   ```

   This will extract the `coherence-web.jar` file into a subdirectory named `web-install`. Use the following commands to move the `coherence-web.jar` file up one level into the `library` directory:

   On Windows:

   ```
   move web-install\coherence-web.jar .
   rmdir web-install
   ```

   On UNIX:

   ```
   mv web-install/coherence-web.jar .
   rmdir web-install
   ```

2. For each WebLogic 10.x install that will be running in the server cluster, update the libraries using the following command (note that it is broken up into multiple lines only for formatting purposes; this is a single command entered on one line):

   ```
   java -cp coherence.jar;coherence-web.jar
   com.tangosol.coherence.servlet.WebPluginInstaller <wls-home-path> -install
   ```

   For example, on Windows:

   ```
   java -cp coherence.jar;coherence-web.jar
   com.tangosol.coherence.servlet.WebPluginInstaller C:\bea\weblogic\wlserver_10
   -install
   ```

3. Follow the general installation steps for the Coherence*Web Session Management Module, specifying WebLogic/10.x for the server type.

How the Coherence*Web Installer instruments a Java EE application

During the inspect step, the Coherence*Web Installer performs the following tasks:

1. Generates a template `coherence-web.xml` configuration file that contains basic information about the application and target web container along with a set of default Coherence*Web configuration context parameters appropriate for the target web container. If an existing `coherence-web.xml` configuration file exists (for example, from a previous run of the Coherence*Web Installer), the context parameters in the existing file are merged with those in the generated template.
2. Enumerates the JSPs from each web application in the target Java EE application and add information about each JSP to the `coherence-web.xml` configuration file.
3. Enumerates the TLDs from each web application in the target Java EE application and add information about each TLD to the `coherence-web.xml` configuration file.

During the install step, the Coherence*Web Installer performs the following tasks:

1. Creates a backup of the original Java EE application so that it can be restored during the uninstall step.
2. Add the Coherence*Web configuration context parameters generated in Step (1) of the inspect step to the `web.xml` descriptor of each web application contained in the target Java EE application.
3. Unregisters any application-specific `ServletContextListener`, `ServletContextAttributeListener`, `ServletRequestListener`, `ServletRequestAttributeListener`, `HttpSessionListener`, and `HttpSessionAttributeListener` classes (including those registered by TLDs) from each Web application.
4. Registers a Coherence*Web `ServletContextListener` in each `web.xml` descriptor. At runtime, the Coherence*Web `ServletContextListener` will propagate each `ServletContextEvent` to each application-specific `ServletContextListener`.
5. Registers a Coherence*Web `ServletContextAttributeListener` in each `web.xml` descriptor. At runtime, the Coherence*Web `ServletContextAttributeListener` will propagate each `ServletContextAttributeEvent` to each application-specific `ServletContextAttributeListener`.
6. Wraps each application-specific Servlet declared in each `web.xml` descriptor with a Coherence*Web `SessionServlet`. At runtime, each Coherence*Web `SessionServlet` will delegate to the wrapped Servlet.
7. Adds the following directive to each JSP enumerated in Step (2) of the inspect step:

   ```
   <%@ page extends="com.tangosol.coherence.servlet.api22.JspServlet" %>
   ```

During the uninstall step, the Coherence*Web Installer replaces the instrumented Java EE application with the backup of the original version created in Step (1) of the install process.

Testing HTTP session management (without a dedicated loadbalancer)

Coherence comes with a light-weight software load balancer; it is only intended for testing purposes. The load balancer is very useful when testing functionality such as Session Management and is very easy to use.

1. Start multiple application server processes, on one or more server machines, each running your application on a unique IP address and port combination.
2. Open a command (or shell) window.
3. Change the current directory to the Coherence library directory (*%COHERENCE_HOME%*`\lib` on Windows and *$COHERENCE_HOME*`/lib` on UNIX).
4. Make sure that the paths are configured so that the Java command will run.
5. Start the software load balancer with the following command lines (each of these command lines makes the application available on the default HTTP port, which is port 80).

   To test load-balancing locally on one machine with two application server instances on ports `7001` and `7002`:

   ```
   java -jar coherence-loadbalancer.jar localhost:80 localhost:7001 localhost:7002
   ```

   To run the load-balancer locally on a machine named server1 that load balances to port `7001` on `server1`, `server2` and `server3`:

```
java -jar coherence-loadbalancer.jar server1:80 server1:7001 server2:7001
server3:7001
```

Assuming the above command line, an application that previously was accessed with the URL `http://server1:7001/my.jsp` would now be accessed with the URL `http://server1:80/my.jsp` or just `http://server1/my.jsp`.

Table 2–6 describes the supported command line options:

Table 2–6 Command Line Options to Test HTTP Session Management

Option	Description
-backlog	Sets the TCP/ IP accept backlog option to the specified value, for example: `-backlog=64`
-random	Specifies the use of a random load-balancing algorithm (default).
-roundrobin	Specifies the use of a round-robin load-balancing algorithm
-threads	Uses the specified number of request/ response thread pairs (so the total number of additional daemon threads will be two times the specified value), for example: `-threads=64`

Make sure that your application uses only relative re-directs or the address of the load-balancer.

Using Coherence and WebLogic Portal (3.4.1)

Coherence integrates closely with Oracle WebLogic Portal to provide WAN-capable clustered caching for portal applications. Specifically, Coherence includes the following integration points:

- P13N CacheProvider SPI Implementation
- Sharing Data Between WSRP-Federated Portals Using Coherence—This section provides a blueprint for efficiently sharing data between WSRP-federated portals that leverages Coherence and the WebLogic Portal Custom Data Transfer mechanism

P13N CacheProvider SPI Implementation

Internally, WebLogic Portal uses its own caching service to cache portal, personalization, and commerce data as described here:

`http://download.oracle.com/docs/cd/E13155_01/wlp/docs103/javadoc/com/bea/p13n/cache/package-summary.html`

WebLogic Portal 8.1.6 and later includes an SPI for the P13N caching service that can be implemented by third party cache vendors. Coherence includes a P13N `CacheProvider` SPI implementation that—when installed into a WebLogic Portal application—will transparently manage cached P13N data without requiring code changes. Additionally, combining Coherence and WebLogic Portal gives you extreme flexibility in your choice of cache topologies.

For example, if you find that your Portal servers are bumping into the 4GB heap limit (on 32-bit JVMs) or are experiencing slow GC times, you can reduce your Portal JVM heap size and GC times by leveraging a cache client/server topology to move serializable P13N state out of your Portal JVMs and into one or more dedicated Coherence cache servers. You can also leverage the Coherence Management Framework to closely monitor statistics to better tune your P13N cache settings.

Finally, the Coherence `CacheProvider` also allows your portlets to leverage Coherence caching service by using the standard P13N Cache API.

To install the Coherence P13N `CacheProvider`, copy the `coherence-wlp.jar` and `coherence.jar` libraries included in the `lib` directory of the Coherence installation to the `APP-INF/lib` directory of your WebLogic Portal application. Additionally, you must configure the Coherence P13N `CacheProvider` as the default provider in the `p13n-cache-config.xml` file found in each of your WLP application's `META-INF` directory. Specifically, add the following line immediately before the first `<cache>` element:

```
<default-provider-id>com.tangosol.coherence.weblogic</default-provider-id>
```

See the Javadoc for the `PortalCacheProvider` class for details on configuring the Coherence `CacheProvider` and Coherence caches used by the provider. See also the following document for a list of some of the caches used by WebLogic Portal:

http://download.oracle.com/docs/cd/E13155_01/wlp/docs103/caches/caches.html

Sharing Data Between WSRP-Federated Portals Using Coherence

The Web Services for Remote Portlets (WSRP) protocol was designed to support the federation of portals hosted by arbitrary portal servers and server clusters. Developers use WSRP to aggregate content and the user interface (UI) from various portlets hosted by other remote portals. By itself, WSRP does not address the challenge of implementing scalable, reliable, and high-performance federated portals that create, access, and manage the lifecycle of data shared by distributed portlets. Fortunately, WebLogic Portal provides an extension to the WSRP specification that—when coupled with Oracle Coherence—allows WSRP Consumers and Producers to create, view, modify, and control concurrent access to shared, scoped data in a scalable, reliable, and highly performant manner.

See the following document for complete details:

http://www.oracle.com/technology/pub/articles/dev2arch/2005/11/federated-portal-cache.html

Using JDeveloper for Coherence Development

This section provides basic instructions on how to setup JDeveloper when developing Coherence solutions.

- Running Coherence in JDeveloper
- Viewing Thread Dumps in JDeveloper
- Creating a Cache Configuration in JDeveloper

Running Coherence in JDeveloper

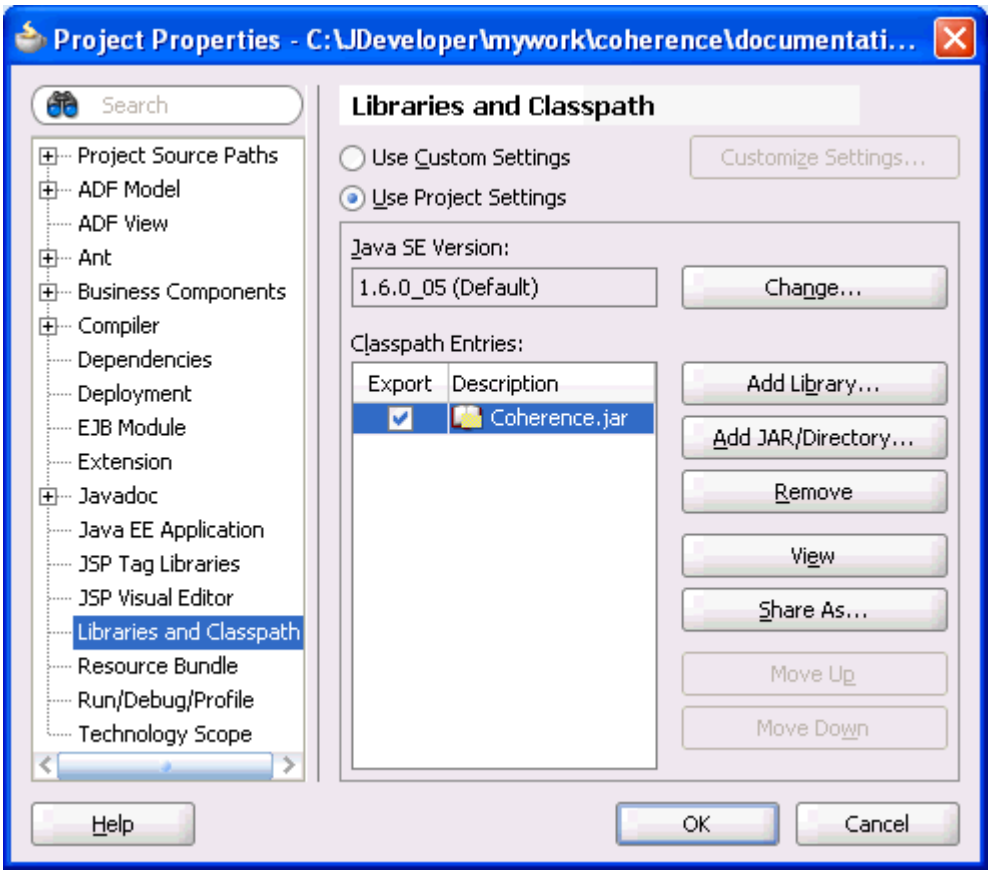
JDeveloper can be used to run cache server (`DefaultCacheServer`) and cache (`CacheFactory`) instances. Each instance is started as a separate Java process and emits standard output to the process' log. Input (such as cache commands) can be entered directly in the process as if it were started from the command line. This configuration facilitates development and testing Coherence solutions.

To run Coherence in JDeveloper:

1. In JDeveloper, create a new Generic Application, which includes a single project. If you are new to JDeveloper, consult the Online Help for detailed instructions.
2. In the Application Navigator, double-click the new project. The Project Properties dialog box displays.
3. Select the **Libraries and Classpath** node. The Libraries and Classpath page displays
4. On the Libraries and Classpath page, click **Add JAR/Directory**. The Add Archive or Directory dialog box displays.
5. From the directory tree, select *COHERENCE_HOME*`\lib\coherence.jar` and click **Select**. The `coherence.jar` library displays in the Classpath Entries list as shown in Figure 2–1:

Note: The `tangosol.jar` library is also required in the classpath when using a Coherence 3.3.x release.

Figure 2–1 Setting the Coherence Library on the Classpath



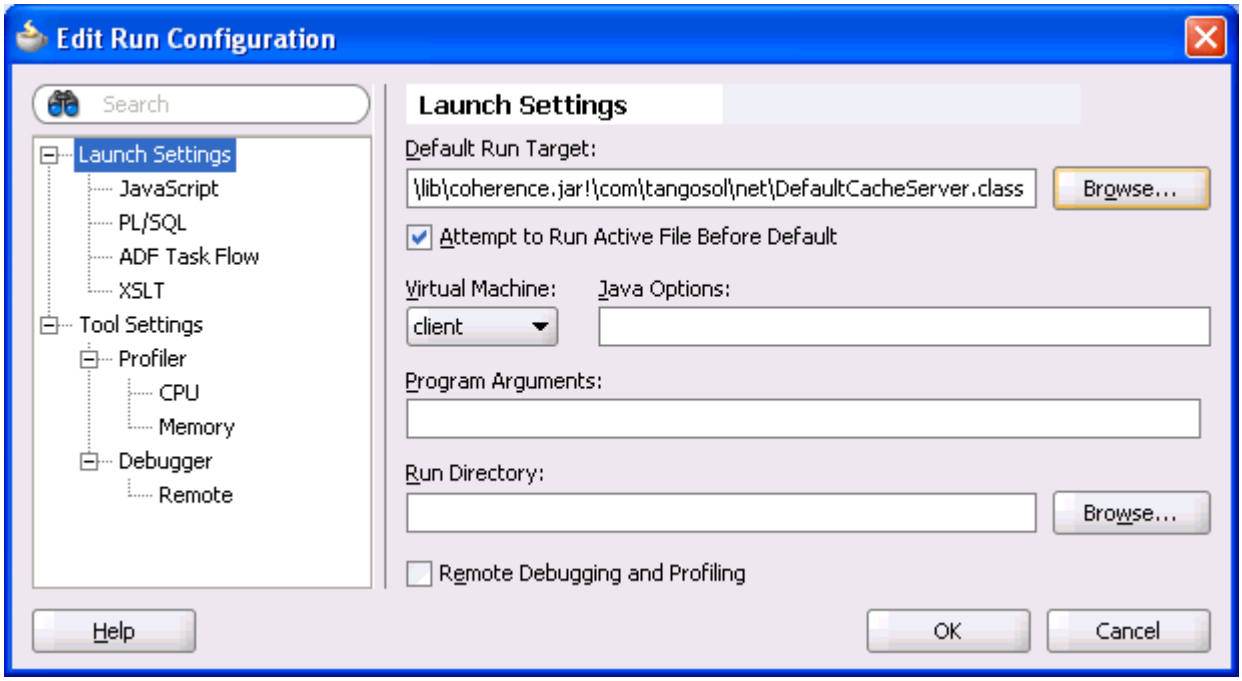
Setting the Coherence Library on the Classpath

****

6. From the Project Properties dialog box, select the **Run/Debug/Profile** node. The Run/Debug/Profile page displays.
7. From the Run/Debug/Profile page, click **New**. The Create Run Configuration dialog box displays.

8. In the Name text box, enter a name for the new run configuration. In the Copy Settings From drop-down box, choose **default**. Click **OK**. The new run configuration displays in the Run Configuration list.
9. From the Run Configuration list, select the new Run Configuration and click **Edit**. The Edit Run Configuration dialog box displays and the Launch Settings node is selected.
10. From the Launch Settings page, click **Browse** to select a Default Run Target. The Choose Default Run Target dialog box displays.
11. From the directory tree, select *COHERENCE_HOME*`\lib\coherence.jar\com\tangosol\net\DefaultCacheServer.class` and click **Open**. The `DefaultCacheServer` class is entered as the default run target as shown in Figure 2–2:

Figure 2–2 Setting DefaultCacheServer as a Default Run Target



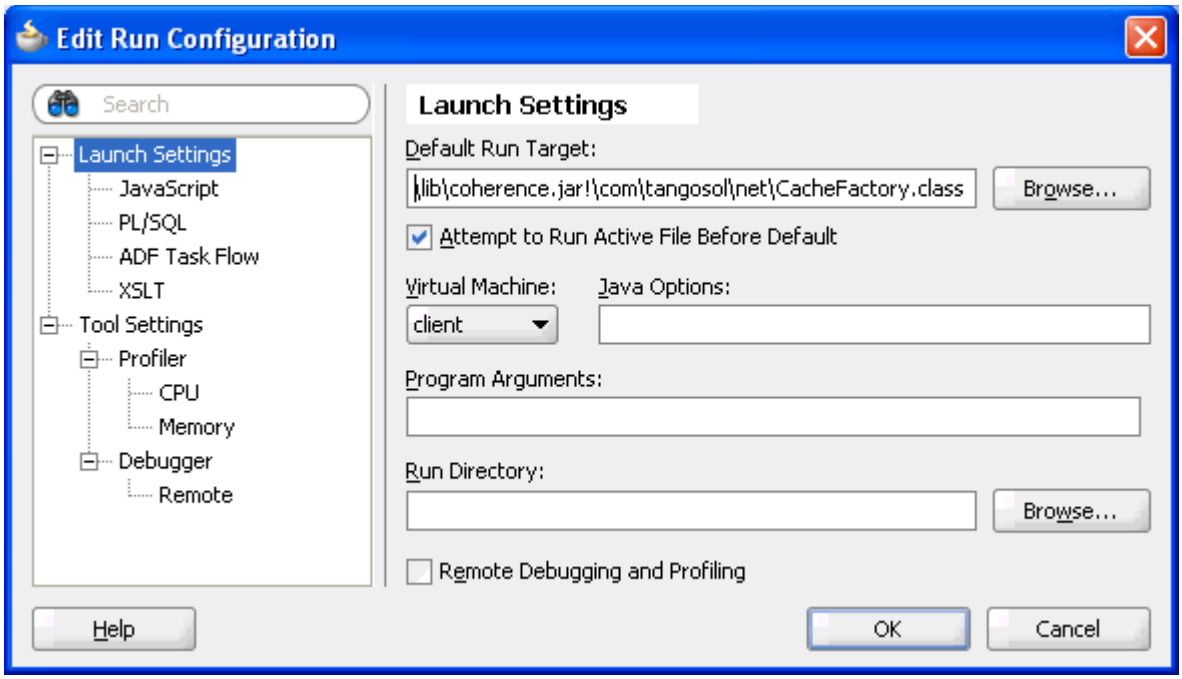
Setting DefaultCacheServer as the Default Run Target in the Edit Run Page

****

> **Tip:** Use the Java Options text box to set Coherence system properties.

12. Select the Tool Settings Node. The Tool Settings page displays.
13. From the Additional Runner Options section, click the **Allow Program Input** check box. A check mark in the box indicates that the option is selected.
14. Click **OK**.
15. Repeat Steps 6 through 14 and select *COHERENCE_HOME*`\lib\coherence.jar\com\tangosol\net\CacheFactory.class` as the default run target as shown in Figure 2–3:

Figure 2–3 Setting CacheFactory as a Default Run Target

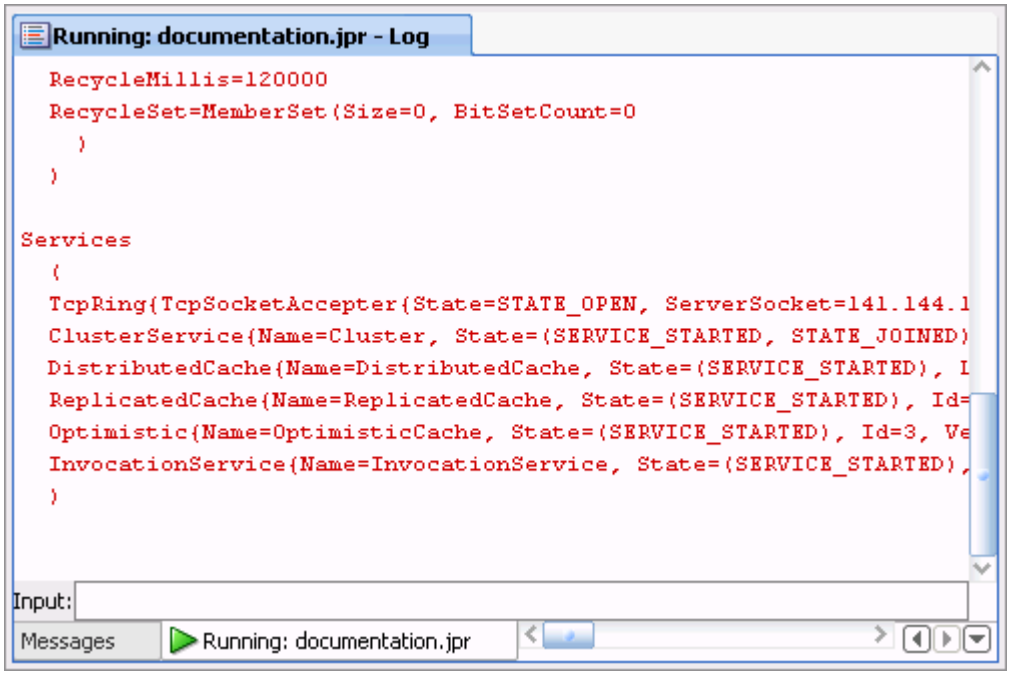


Setting CacheFactory as a Default Run Target in the Edit Run page

****

16. Click **OK** to close the Project Properties dialog box.
17. Use the Run button drop-down list to select and start the run configuration for the cache server. A cache server instance is started and output is shown in the process's log tab as shown in Figure 2–4:

Figure 2–4 Log Output for a Cache Server Instance

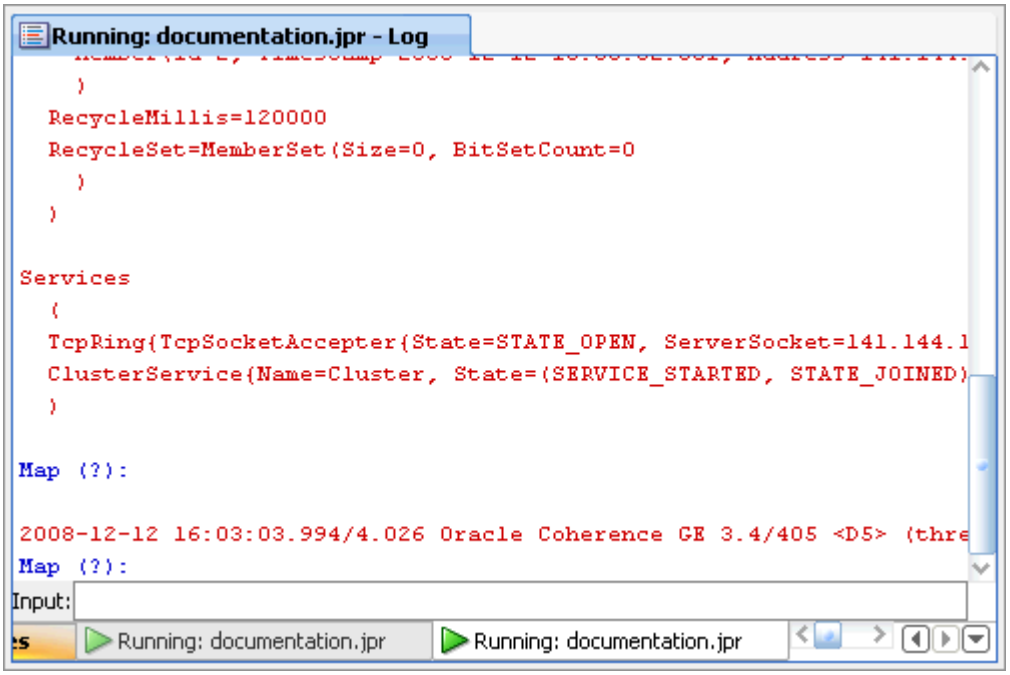


Log Output for a Cache Server Instance

****

18. Use the Run button drop-down list to select and start the run configuration for the cache. A cache instance is started and output is shown in the process's log tab as shown in Figure 2–5.

Figure 2–5 Log Output for a Cache Client Instance



Log Output for a Cache Client Instance

****

19. From the Cache Factory's Running Log tab, use the Input text box located at the bottom of the tab to interact with the cache instance. For example, type `help` and press **Enter** to see a list of valid commands.

Viewing Thread Dumps in JDeveloper

Java allows you to dump a list of threads and all their held locks to standard output. This is achieved in Linux environments using the `kill` command and in Windows environments using `ctrl+break`. Thread dumps are very useful for troubleshooting during development (for example, finding deadlocks).

When developing Coherence solutions in JDeveloper, you can view thread dumps directly in a process's log tab. This is achieved by sending the above signals to the Java process running in JDeveloper.

To view thread dumps in JDeveloper:

1. From a shell or command prompt, use `JDK_HOME/bin/jps` to get the Process ID (PID) of the Java process for which you want to view a thread dump. If you are using JDK 1.4 or below, use `ps -ef` on Linux or `tlist` on Windows to get the PID.

2. On Linux, use `kill -3 PID` to send a `QUIT` signal to the process. On Windows, you must use a third-party tool (such as SendSignal) in order to send a `ctrl+break` signal to a remote Java process.

   The thread dump is viewable in the process's log in JDeveloper.

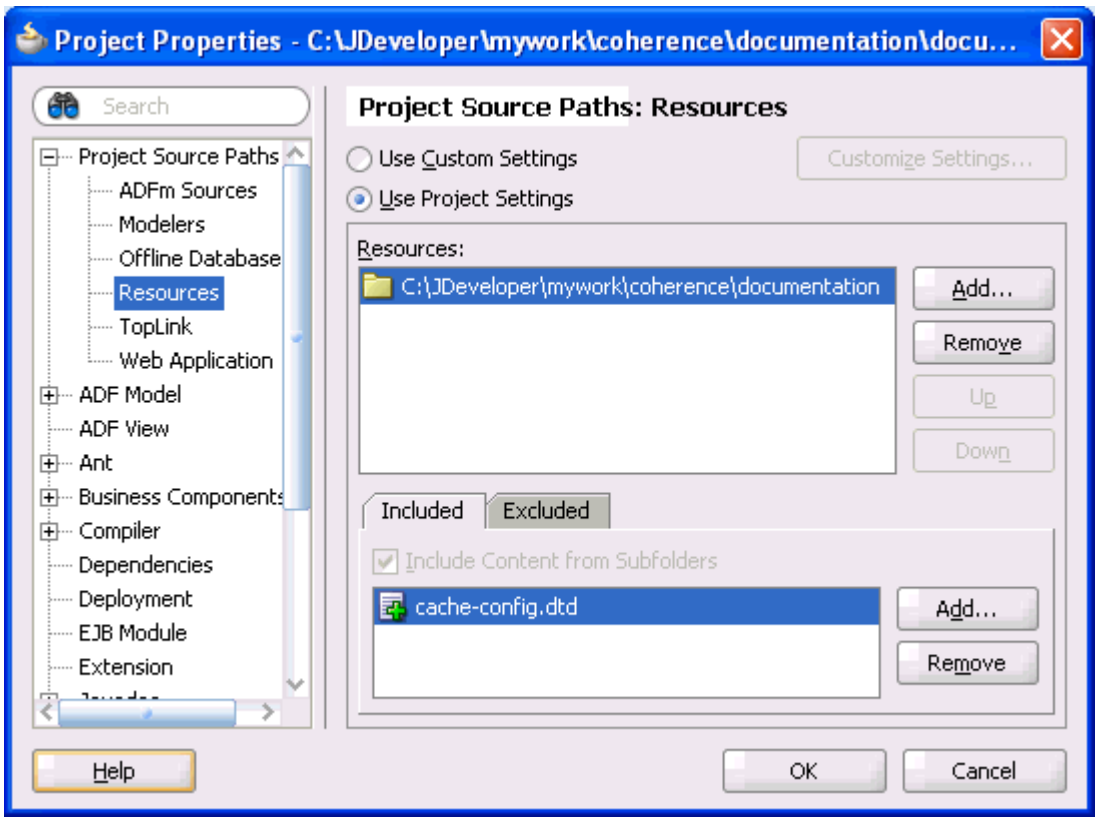
Creating a Cache Configuration in JDeveloper

JDeveloper can be used to create a Coherence cache configuration file. JDeveloper loads the `cache-config.dtd` and lists all the DTD's elements in the Component Palette. In addition, JDeveloper validates the cache configuration file against the DTD and provides XML code completion.

To create a cache configuration in JDeveloper:

1. Extract `cache-config.dtd` from the *COHERENCE_HOME*`\lib\coherence.jar` library to a directory on your computer.
2. In the JDeveloper Application Navigator, double-click your coherence project. The Project Properties dialog box displays.
3. Expand the **Project Source Paths** node and click **Resources**. The Resources page displays.
4. In the Resources section, click **Add** to find and select the directory where you extracted the DTD file.
5. In the Included tab, click **Add** and select the `cache-config.dtd` file.
6. Click **OK**. The DTD is listed in the Included tab as shown in Figure 2–6.

Figure 2–6 Project DTD Listed Under Resources in the Project Properties Dialog Box



Project DTD Listed Under Resources in the Project Properties Dialog Box

****

7. Click **OK** to close the Project Properties dialog box. The DTD is listed in the Application Navigator under the Resources folder for your project.
8. From the File menu, click **New**. The New Gallery dialog box displays.
9. From the Categories section, expand the **General** node and click **XML**.
10. Select **XML Document** and click **OK**. The Create XML File dialog box displays.
11. Enter `coherence-cache-config.xml` as the file name and save it to the same directory where the DTD is located.
12. Click **OK**. The cache configuration file is created, opened for editing, and listed in the Application Navigator under the resources folder for your project.
13. Add the following `DOCTYPE` at the beginning of the file:

    ```
    <?xml version="1.0" encoding="windows-1252" ?>
    <!DOCTYPE cache-config SYSTEM "cache-config.dtd">
    ```

    The Component Palette refreshes and lists all the elements available from the `cache-config.dtd` file.
14. Save the `coherence-cache-config.xml` file.

Corrections to the C++ Object Model

This section describes corrections to the *Understanding the C++ Object Model* topic that can be found in the *Users Guide for Oracle Coherence*.

In the **Threading** section, there is a reference to a `coherence::lang::ThreadLocal` class. This is an error. The correct name of the class is `coherence::lang::ThreadLocalReference`.

In the **Object Immutability** section, there is a sentence where the words `Handle` and `View` were transposed. The incorrect text reads:

- Note that once immutable, an object cannot revert to being mutable. You cannot cast away const-ness to turn a `Handle` into a `View` as this would violate the proved immutability.

The corrected text (in *italic*s) should read:

- Note that once immutable, an object cannot revert to being mutable. You cannot cast away const-ness to turn a *View* into a *Handle* as this would violate the proved immutability.

In the **Thread Safe Handles** section, the paragraph that reads:

- These handle types may be read and written from multiple thread without the need for additional synchronization. They are primarily intended for use as the data-members of other managed classes, and they make use of their parent's internal atomic state to provide thread-safety. When using these handle types it is recommended that they be read into a normal stack based handle if they will be accessed more than once within a code block. This assignment to a normal stack based handle is thread-safe, and once completed allows for essentially free dereferencing of the stack based handle. Note that when initializing thread-safe handles a reference to the parent class must be supplied as the first parameter, this reference can be obtained by calling `self()` on the parent object.

Should be corrected to the following (corrected text in *italic*s):

- These handle types may be read and written from multiple thread without the need for additional synchronization. They are primarily intended for use as the data-members of other managed classes, and they make use of their parent's internal atomic state to provide thread-safety. When using these handle types it is recommended that they be read into a normal stack based handle if they will be accessed more than once within a code block. This assignment to a normal stack based handle is thread-safe, *and after it completes*, allows for essentially free dereferencing of the stack based handle. Note that when initializing thread-safe handles a reference to *a guardian Object* must be supplied as the first parameter, this reference can be obtained by calling `self()` on the *enclosing* object.

The paragraph that reads:

- The same basic technique can be applied to non-managed classes as well. This is made possible, by using the defined synchronization point as the "parent" for the thread-safe handles. With this approach you must ensure that the "parent" `m_hMutex` Object outlives the thread-safe handle `m_vsName`. This is easily accomplished by declaring them as data-members of the same class, and declaring the "parent" before the handle.

Should be replaced by the following (corrected text in italics):

- *The same basic technique can be applied to non-managed classes as well. Since non-managed classes do not extend `coherence::lang::Object` they cannot be used as the guardian of thread-safe handles. It is possible however to use another Object as the guardian. When taking this approach it is crucial to ensure that the guardian Object outlives the guarded thread-safe handle. To facilitate this Coherence starting with Coherence 3.4.1 you can obtain an immortal guardian from `coherence::lang::System` by calling the `System::common()`.*

The code in **Example 2-18: Thread-safe Handle as a Non-Managed Class** should be replaced with the following:

```
class Employee
    {
    public:
        Employee(String::View vsName, int32_t nId)
            : m_vsName(System::common(), vsName), m_nId(nId)
            {
            }

    public:
        String::View getName() const
            {
            return m_vsName;
            }

        void setName(String::View vsName)
            {
            m_vsName = vsName;
            }

        int32_t getId() const
            {
            return m_nId;
            }

    private:
        MemberView<String>   m_vsName;
```

```
    const int32_t        m_nId;
};
```

Corrections to Date Formats in Reporter Output

The date formats described in the *Analyzing Reporter Content* chapter have changed. The previous date formats recorded only the year, month, and day when the report was run (`YYYYMMDD`). The new formats have been updated to also record the hour (`YYYYMMDDHH`). Table 2–7 summarizes the changes from the old format to the new format.

Table 2–7 Old and New Date Formats used by the Reporter

	Old Value	New Value
Date Format	`YYYYMMDD`	`YYYYMMDD`
File Name Example	`20090131-network-health.txt`	`2009013113-network-health.txt`
Date Example	January 31, 2009	January 31, 2009 at 1:00 PM

Corrections to Coherence Features by Edition

The table in the *Coherence Client Editions* section of the *Coherence Features by Edition* appendix is missing a row for the C++ client under API Language. The corrected row should read as follows:

Table 2–8 Coherence Client Editions

		Data Client (see note 9)	Real Time Client (see note 10) *(configured as Extend/TCP Client; see note 11)*	Real Time Client (see note 10) *(configured as Compute Client)*
API Language	Java	Yes	Yes	Yes
	.NET	Yes	Yes	No
	C++	Yes	Yes	No

Moreover, Notes 9, 10, and 11 are missing from the text. They are reproduced below.

9. Data Client may be used with all Coherence Server Editions.

10. Real Time Client may only be used with Grid Edition.

11. Extend/TCP is an abbreviation for Coherence*Extend configured for transport over TCP/IP.

Corrections to How to Manage Coherence using JMX

In *How to Manage Coherence Using JMX*, **Table 22-4: Values for the tangosol.coherence.management.refresh.policy Property** incorrectly identifies `refresh-expired` as being the default value. The correct default value for the property is `refresh-ahead`.

Corrections to distributed-scheme Element Description

The description of the `backup-count` attribute of the `distributed-scheme` element incorrectly lists `0`, `1`, or `2` as the recommended values. The correct recommended values should be `0` or `1`.

Corrections to Production Checklist

This section describes changes and enhancements made to the *Production Checklist* appendix:

In the *JVM* section, the bullet points that read:

- Using identical heap size values for both `-Xms` and `-Xmx` will yield substantially better performance, as well as "fail fast" memory allocation.

Should be corrected to the following (corrected text in *italics*)

- Using identical heap size values for both `-Xms` and `-Xmx` will yield substantially better *performance on Sun and JRockit JVMs*, as well as "fail fast" memory allocation. *See the specific Deployment Considerations for various JVMs below*.

The following bullet point should be added to the list:

- 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.

Corrections to Platform-Specific Deployment Considerations

This section describes the changes and enhancements made to the *Platform-Specific Deployment Considerations* appendix.

Sun JVMs

In the *Deploying to Sun JVMs* section, add the following section:

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.

IBM JVMs

In the *Deploying to IBM JVMs* section, add the following two sections:

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:

`http://www.ibm.com/developerworks/java/jdk/diagnosis/`

Oracle JRocket JVMs

In the *Deploying to Oracle JRocket JVMs* section add the following section:

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
```