Oracle® Coherence

Release Notes for Oracle Coherence

Release 3.5

E14978-01

July 2009

Oracle Coherence Release Notes for Oracle Coherence, Release 3.5

E14978-01

Primary Author: Thomas Pfaeffle

Contributing Author: Noah Arliss, Mark Falco, Alex Gleyzer, Gene Gleyzer, Jason Howes, Adam Leftik, Rob Misek, Patrick Peralta

Contents

Preface

This document describes changes and enhancements that have been made to the Oracle Coherence product since the 3.4.2 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.

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 in the Oracle Coherence documentation set:

- *Getting Started with Oracle Coherence*
- *Developer's Guide for Oracle Coherence*
- *Client Guide for Oracle Coherence*
- *Tutorial for Oracle Coherence*
- *User's Guide for Oracle Coherence*Web*
- *Integration 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 for the 3.5 release. This document is accurate at the time of publication. Oracle updates the release notes periodically after the software release.

Oracle Coherence for Java 3.5

The following is a list of new features and improvements in Oracle Coherence 3.5:

New and Improved Coherence Data Grid Functionality

- Prior to Coherence 3.5 off-heap based backing maps had a capacity limit of 2GB per node. Coherence 3.5 allows setting up backing map capacity on partition by partition base, dramatically increasing the total backing map capacity per 64bit JVM node.
- Introduced POF-based value extractors and updaters (see `PofExtractor` and `PofUpdater`); this feature allows efficient using of partitioned cache queries, aggregations, and invocations without serialization and deserialization of corresponding values.
- Added server side logic that uses the `EntryProcessor.processAll()` method to batch process multiple entries that belong to the same partition.
- Added a "service guardian" functionality to identify and recover from server-side deadlocks.
- Dramatically reduced the CPU consumption during the redistribution of backing maps with large number of entries, and performance of the partition-based queries.
- Prior to Coherence 3.5, to ensure even distribution the number of partitions had to exceed a square of the number of storage-enabled nodes. This constraint has been removed.
- Added several distribution-related `PartitionEvent` types that are delivered to preconfigured partition listeners.
- Added ability to specify an `AddressProvider` for dynamic discovery of the Well-Known-Address (WKA) list, thus enabling dynamic reconfiguration of the WKA based on changes in the environment without cluster restart. This feature is particularly useful for operations in dynamic environments such as Amazon's EC2.
- Added the ability to easily iterate query results on a partition-by-partition or member-by-member basis (see the `com.tangosol.net.partition.`

PartitionedIterator class). This feature is particularly valuable for processing large result sets.

- Added the ability to specify an AddressProvider for dynamic discovery of the Well-Known-Address (WKA) list and ProxyService's acceptor, thus enabling dynamic reconfiguration of the WKA and tcp-acceptor based on changes in the environment without cluster restart. This feature is particularly useful for operations in dynamic environments such as Amazon's EC2.
- Added an automatic detection and resolution of multicast group disconnects.

Coherence*Web Enhancements

- Hardened support for the Coherence*Extend-based deployment topology. See *"Configuring Coherence*Web with Coherence*Extend"* in the *"User's Guide for Oracle Coherence*Web"* for complete details.
- Added various optimizations and fixed several issues with the Coherence*Web WebLogic SPI implementation. See the *"User's Guide for Oracle Coherence*Web"* for more details.
- Fixed several issues with HttpSession-related lifecycle event dispatching.
- Added a new application-scoped locking mode to better support complex application deployments. See *"Session Locking Modes"* in the *"User's Guide for Oracle Coherence*Web"* for more details.
- Optimized the performance of HttpSession access for Web applications with read-mostly access patterns.
- Optimized the performance of the split session mode update and invalidation operations.
- Optimized the performance of the session reaper, dramatically reducing network and CPU utilization by moving the expired session detection into the data grid.
- Added session affinity support for Tomcat.
- Coherence*Web 3.5 has been certified on WebLogic Server 11g (10.3.1) and 10.3. See "Required Patch Levels for Coherence*Web 3.5" on page 2-1 for details.

WebLogic Portal Integration

- Certified the PortalCacheProvider with WebLogic Portal 9.2 and 10.2+.

Other Enhancements and Fixes

- Added index support for Replicated, Local and Continues Query caches.
- Reduced memory usage by indexes.
- Add support for unit calculator to "external-scheme" (SerializationMap).
- Added support for the KeyAssociatedFilter for the Coherence*Extend clients.
- Introduced the new com.tangosol.util.BinaryEntry interface that provides access to entry's data in internal (Binary) format to entry processors and aggregators.
- Improved performance of default hybrid eviction strategy for local caches.
- Improved multiple object handling in the HibernateCacheStore.
- Hardened the ownership conflict resolution algorithm for partitioned services.

- Made deserialization routines in `DistributedCache` service more robust, preventing custom processor serialization failure to cause the service failure.

  As of Coherence 3.5, the `PortableObject` interface no longer extends `Serializable` interface.
- Added a local-only configuration setting that allows custom MBeans to be registered without the `nodeId` attribute.
- Added a work around the WebSphere 6.1 MBean name mangling, which prevented MBean re-registering during application re-deployment.
- Turned the remote JMX management feature on by default.
- Fixed a concurrency issue in the `ConfigurablePofContext`.
- Fixed a potential deadlock caused by concurrent `NamedCache.destroy()` calls.
- Allowed the network filters to be configured for the `Cluster` service.
- Fixed a serialization-related regression in the ReplicatedCache service causing it to return `Binary` object.
- Fixed a regression for the key-based map `removeMapListener` API during redistribution.
- Allowed non-serializable objects to be used in ReplicatedCache service with custom `Serializers`.

Backward Compatibility

The `AbstractExtractor` implementation has an additional ability to operate against the `java.util.Map.Entry` instead of just the value. This allows any sub-classed to extract a desired value using all available information on the corresponding `Map.Entry` object and is intended to be used in advanced custom scenarios, when application code must look at both key and value at the same time or can make some very specific assumptions regarding to the implementation details of the underlying `Entry` object. The only impact of that change is that custom subclasses of the `EntryExtractor` have to change the parameter signature for the `extractFromBinary()` method.

Oracle Coherence for .NET 3.5

Other Enhancements and Fixes:

- Introduced POF-based value extractors and updaters (in `Tangosol.Util.Extractor` namespace). This feature allows efficient using of partitioned cache queries, aggregations and invocations without serialization and deserialization of corresponding values.
- Added index support for the Local and Continuous Query caches.
- Added support for the `KeyAssociatedFilter`.
- Improved performance of default hybrid eviction strategy for local caches.
- Fixed a concurrency issue in the `ConfigurablePofContext`.

Oracle Coherence for C++ 3.5

Additional Platform Support

- Solaris on SPARC, x86, x64
- Apple OS X on x64

Performance Improvements

- Optimized memory management for short lived heap objects.
- Added automatic thread based escape analysis.
- Significantly reduced the cost of reference counting.
- Significantly optimized RTTI-based cast operations.
- Dramatically reduced contention for the Socket read/write operations.
- Added `coherence::lang::FinalHandle`, `FinalView`, and `FinalHolder` for reduced cost thread-safe data member reads.

Other Enhancements and Fixes

- Introduced POF-based value extractors and updaters (`PofExtractor` and `PofUpdater` in `coherence/util/extractor` namespace); this feature allows the efficient use of partitioned cache queries, aggregations, and invocations without serialization and deserialization of corresponding values.
- Added index support for the Local and Continues Query caches.
- Added `coherence::util::extractor::TypedExtractor` for local and remote `ReflectionExtractor` functionality
- Added support for the `coherence::util::filter::KeyAssociatedFilter`.
- Added the `coherence::util::ArrayList` class.
- Added stack trace support on Windows Vista.
- Added OOME and memory corruption detection for Arrays.
- Removed coherence-debug shared library; release library is now suitable for linking with debug applications.

2

Documentation Errata

This chapter describes changes, enhancements, and corrections made to the Oracle Coherence documentation library for 3.5. The library can be found at the following URL:

`http://download.oracle.com/docs/cd/E14526_01/index.htm`

Required Patch Levels for Coherence*Web 3.5

Table 2–1 describes the WebLogic Server patch levels required to use Coherence*Web 3.5 with WebLogic Server.

*Table 2–1 Required Patch Levels to use Coherence*Web 3.5 with WebLogic Server*

	WebLogic Server 10.3	WebLogic Server 11g (10.3.1)
WebLogic Smart Update	Patch ID: 6W2W	No Patch Required

See *Overview Of Configuration and Deployment* in the *User's Guide for Oracle Coherence*Web* for more information on accessing WebLogic Smart Update.

Coherence*Web 3.5 is Not Certified for Use on WebLogic Server 9.2 MP1 using Coherence*Web SPI

The *User's Guide for Oracle Coherence*Web* makes references to WebLogic Server 9.2 MP1 supporting Coherence*Web 3.5 using Coherence*Web SPI. This is an error. Coherence*Web 3.5 is not certified for use on WebLogic Server 9.2 MP1.

If you are using WebLogic Server 9.2 MP1, you can run Coherence*Web 3.4.2, patch 6.

Use an Override File Instead of Editing tangosol-coherence.xml

Steps 5, 6, and 7 of the *Restricting Coherence to Your Own Host* section in the *Tutorial for Oracle Coherence*, describe how to extract `tangosol-coherence.xml` from the `coherence.jar` file and edit it to change the value of the multicast listener port. Editing the `tangosol-coherence.xml` file and replacing it in the JAR is not a recommended practice. The preferred way to change the behavior defined in the `tangosol-coherence.xml` file is by using an override file: in this case, `tangosol-coherence-override-dev.xml`.

The `tangosol-coherence-override-dev.xml` file can be found in the `coherence.jar` file. After editing this file (in this case to define a unique value for the port system-property value for the multicast listener), save the file and add it to the server's classpath. When the server is executed, the values in the `tangosol-coherence-override-dev.xml` file will override any corresponding settings in the `tangosol-coherence.xml` file.

Updated Sample File to Define a Spring-Aware Cache Factory

The sample file to define a Spring-aware cache factory in the *Integrating an Oracle Coherence CacheFactory with Spring* chapter of the *Integration Guide for Oracle Coherence* has been updated for bug fixes and to remove private accessors. See the updated sample in Example 2–1.

Example 2–1 Updated Sample for a Spring-Aware Cache Factory

```
/*
* SpringAwareCacheFactory.java
*
* Copyright 2001-2007 by Oracle. All rights reserved.
*
* Oracle is a registered trademarks of Oracle Corporation and/or its affiliates.
*
* This software is the confidential and proprietary information of
* Oracle Corporation. You shall not disclose such confidential and
* proprietary information and shall use it only in accordance with the
* terms of the license agreement you entered into with Oracle.
*
* This notice may not be removed or altered.
*/
package com.tangosol.coherence.spring;

import com.tangosol.net.BackingMapManagerContext;
import com.tangosol.net.DefaultConfigurableCacheFactory;

import com.tangosol.run.xml.SimpleElement;
import com.tangosol.run.xml.XmlElement;
import com.tangosol.run.xml.XmlHelper;

import com.tangosol.util.ClassHelper;

import java.util.Iterator;

import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.BeanFactoryAware;

import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.context.support.FileSystemXmlApplicationContext;


/**
* SpringAwareCacheFactory provides a facility to access caches declared
* in a "cache-config.dtd" compliant configuration file, similar to its super
* class {@link DefaultConfigurableCacheFactory}.  In addition, this factory
* provides the ability to reference beans in a Spring application context
* via the use of a class-scheme element.
*
* <p>This factory can be configured to start its own Spring application
* context from which to retrieve these beans.  This can be useful for stand-alone
* JVMs such as cache servers.  It can also be configured at runtime with a
* pre-configured Spring bean factory.  This can be useful for Coherence
* applications running in an environment that is itself responsible for starting
* the Spring bean factory, such as a web container.
*
* @see #instantiateAny(CacheInfo, XmlElement,
        BackingMapManagerContext, ClassLoader)
*
*/
public class SpringAwareCacheFactory
        extends DefaultConfigurableCacheFactory
        implements BeanFactoryAware
    {
    // ----- constructors -------------------------------------------------

    /**
    * Construct a default DefaultConfigurableCacheFactory using the
    * default configuration file name.
    */
    public SpringAwareCacheFactory()
        {
        super();
        }

    /**
    * Construct a SpringAwareCacheFactory using the specified path to
    * a "cache-config.dtd" compliant configuration file or resource.  This
    * will also create a Spring ApplicationContext based on the supplied
    * path to a Spring compliant configuration file or resource.
    *
    * @param sCacheConfig location of a cache configuration
    * @param sAppContext  location of a Spring application context
    */
    public SpringAwareCacheFactory(String sCacheConfig, String sAppContext)
        {
        super(sCacheConfig);

        azzert(sAppContext != null && sAppContext.length() > 0,
                "Application context location required");

        m_beanFactory = sCacheConfig.startsWith("file:") ? (BeanFactory)
            new FileSystemXmlApplicationContext(sAppContext) :
            new ClassPathXmlApplicationContext(sAppContext);

        // register a shutdown hook so the bean factory cleans up
        // upon JVM exit
        ((AbstractApplicationContext) m_beanFactory).registerShutdownHook();
        }

    /**
    * Construct a SpringAwareCacheFactory using the specified path to
    * a "cache-config.dtd" compliant configuration file or resource and
    * the supplied Spring BeanFactory.
    *
    * @param sPath       the configuration resource name or file path
    * @param beanFactory Spring BeanFactory used to load Spring beans
    */
    public SpringAwareCacheFactory(String sPath, BeanFactory beanFactory)
        {
        super(sPath);

        m_beanFactory = beanFactory;
    }


// ----- extended methods -------------------------------------------

/**
* Create an Object using the "class-scheme" element.
* <p/>
* In addition to the functionality provided by the super class,
* this will retreive an object from the configured Spring BeanFactory
* for class names that use the following format:
* <pre>
* &lt;class-name&gt;spring-bean:sampleCacheStore&lt;/class-name&gt;
* </pre>
*
* Parameters may be passed to these beans via setter injection as well:
* <pre>
*   &lt;init-params&gt;
*     &lt;init-param&gt;
*       &lt;param-name&gt;setEntityName&lt;/param-name&gt;
*       &lt;param-value&gt;{cache-name}&lt;/param-value&gt;
*     &lt;/init-param&gt;
*   &lt;/init-params&gt;
* </pre>
*
* Note that Coherence will manage the lifecycle of the instantiated Spring
* bean, therefore any beans that are retreived using this method should be
* scoped as a prototype in the Spring configuration file, for example:
* <pre>
*   &lt;bean id="sampleCacheStore"
*         class="com.company.SampleCacheStore"
*         scope="prototype"/&gt;
* </pre>
*
* @param info      the cache info
* @param xmlClass  "class-scheme" element.
* @param context   BackingMapManagerContext to be used
* @param loader    the ClassLoader to instantiate necessary classes
*
* @return a newly instantiated Object
*
* @see DefaultConfigurableCacheFactory#instantiateAny(
*        CacheInfo, XmlElement, BackingMapManagerContext, ClassLoader)
*/
public Object instantiateAny(CacheInfo info, XmlElement xmlClass,
        BackingMapManagerContext context, ClassLoader loader)
    {
    if (translateSchemeType(xmlClass.getName()) != SCHEME_CLASS)
        {
        throw new IllegalArgumentException(
                "Invalid class definition: " + xmlClass);
        }

    String sClass = xmlClass.getSafeElement("class-name").getString();

    if (sClass.startsWith(SPRING_BEAN_PREFIX))
        {
        String sBeanName = sClass.substring(SPRING_BEAN_PREFIX.length());

        azzert(sBeanName != null && sBeanName.length() > 0,
                    "Bean name required");

            XmlElement xmlParams = xmlClass.getElement("init-params");
            XmlElement xmlConfig = null;
            if (xmlParams != null)
                {
                xmlConfig = new SimpleElement("config");
                XmlHelper.transformInitParams(xmlConfig, xmlParams);
                }

            Object oBean = getBeanFactory().getBean(sBeanName);

            if (xmlConfig != null)
                {
                for (Iterator iter = xmlConfig.getElementList().iterator();
iter.hasNext();)
                    {
                    XmlElement xmlElement = (XmlElement) iter.next();

                    String sMethod = xmlElement.getName();
                    String sParam  = xmlElement.getString();
                    try
                        {
                        ClassHelper.invoke(oBean, sMethod, new Object[]{sParam});
                        }
                    catch (Exception e)
                        {
                        ensureRuntimeException(e,"Could not invoke " + sMethod +
                                "(" + sParam + ") on bean " + oBean);
                        }
                    }
                }
            return oBean;
            }
        else
            {
            return super.instantiateAny(info, xmlClass, context, loader);
            }
        }

    /**
    * Get the Spring BeanFactory used by this CacheFactory
    * @return the Spring {@link BeanFactory} used by this CacheFactory
    */
    public BeanFactory getBeanFactory()
        {
        azzert(m_beanFactory != null, "Spring BeanFactory == null");
        return m_beanFactory;
        }

    /**
    * Set the Spring BeanFactory used by this CacheFactory
    * @param beanFactory the Spring {@link BeanFactory} used by this CacheFactory
    */
    public void setBeanFactory(BeanFactory beanFactory)
        {
        m_beanFactory = beanFactory;
        }
```

```
// ----- data fields ----------------------------------------------------

/**
* Spring BeanFactory used by this CacheFactory
*/
private BeanFactory m_beanFactory;

/**
* Prefix used in cache configuration "class-name" element to indicate
* this bean is in Spring
*/
private static final String SPRING_BEAN_PREFIX = "spring-bean:";
}
```

Index