Oracle® Fusion Middleware

Release Notes for Oracle Coherence

14c (14.1.2.0.0)
F79654-01
December 2024

Oracle Fusion Middleware Release Notes for Oracle Coherence, 14c (14.1.2.0.0)

F79654-01

Primary Author: Oracle Corporation

Contents

Preface

Release Notes for Oracle Coherence summarizes the release information related to new and updated features, known issues and their workarounds, deprecated and removed functionality, and more.

This preface includes the following topics:

- Audience
- Documentation Accessibility
- Diversity and Inclusion
- Related Documents
- Conventions

Audience

This document is intended for users of Oracle Coherence.

Documentation Accessibility

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

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

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

Related Documents

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

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

You can use the Oracle Coherence Release Notes to learn about important production information such as Coherence certifications, support, and licensing.
This chapter contains the following sections:

Latest Release Information

This document is accurate at the time of publication. Oracle will update the release notes periodically after the software release. You can access the latest information and additions to these release notes at Oracle Help Center.

Purpose of this Document

This document contains the release information for Oracle Coherence.

Oracle recommends you review its contents before installing, or working with the product.

System Requirements and Specifications

Oracle Coherence follows the Fusion Middleware system requirements and certifications for production environments. For more information, see http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html.

For system requirements for installations of development environments, visit:

- Coherence for Java — *Installing Oracle Coherence for Java*.
- C++ Client — *Installing the C++ Client Distribution*.
- .Net Client — *Installing the .NET Client Distribution*.

Certification Information

To see versions of platforms and related software for which Oracle Coherence is certified and supported, go to `https://www.oracle.com/middleware/technologies/fusion-certification.html`.

Product Documentation

For complete documentation on Oracle Coherence, go to Oracle Help Center.

Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support .

Licensing Information

Detailed information regarding license compliance for Oracle Fusion Middleware is available at Licensing Information.

Downloading and Applying Required Patches

To download and install the latest software patch:

1. Log in to `My Oracle Support` to download the latest software patches.
2. Click the **Patches & Updates** tab.
3. Under the Patch Search tab, select **Product or Family (Advanced Search)**, and select the **Include all patches in a product family** check box.
4. Enter **Oracle Coherence** as the product, select the platform and release, and click **Search**.

   A list of currently available patches for Oracle Coherence is returned.
5. Select the required patch and click **Download**.

   You can check the `README` file in the patch distribution for up-to-date information on the software fixes provided by the patch.

2 What's New in this Release

Learn about the features, enhancements, and changes made to Oracle Coherence. Oracle updates the release notes periodically after the software release. This document is accurate at the time of publication.

This chapter includes the following sections:

New Features

This section contains new features for Oracle Coherence that are organized by release.

New and Improved for 14c (14.1.2.0.0)

- **Java Modules Support**
  - You can now run Coherence using Java modules. See Using Java Modules to Build a Coherence Application in *Developing Applications with Oracle Coherence*.
  - Coherence no longer requires the following modules to be explicitly opened or exported: `java.base/java.lang.invoke=com.oracle.coherence`, `java.base/java.lang=org.eclipse.persistence.core`, `java.management/sun.management=com.oracle.coherence`, `java.base/java.util=com.oracle.coherence`. See Using Java Modules to Build a Coherence Application in *Developing Applications with Oracle Coherence*.
- **Core Improvements**
  - **NamedMap API** - A distributed implementation of `java.util.Map` interface. See Performing Basic Cache Operations in *Developing Applications with Oracle Coherence*.
  - **Bootstrap API** - The new bootstrap API enables you to configure and start a Coherence application by building a `com.tangosol.net.Coherence` instance and starting this instance. See Using the Bootstrap API in *Developing Applications with Oracle Coherence*.
  - **Repository API** - The Coherence Repository API provides you with a higher-level, DDD-friendly way to access data managed in Coherence. See Using the Repository API in *Developing Applications with Oracle Coherence*.
  - **Caffeine** - Coherence now adds a Caffeine backing map implementation, enabling you to use Caffeine wherever the standard Coherence local cache can be used. See Integrating Caffeine in *Developing Applications with Oracle Coherence*.
  - **Partition Events Logging** - This feature enables logging of partition unavailable duration when the partition events occur. For example, during partition movements between members. See Logging Partition Events in *Developing Applications with Oracle Coherence*.

  - **Non-Blocking Data Sources** - The new `NonBlockingEntryStore` enables cache stores to respond asynchronously when mutations are made to entries. See Non-Blocking Data Sources in *Developing Applications with Oracle Coherence*.
  - **Coherence Lifecycle Listeners** - Added the ability to register lifecycle listeners with Coherence instances, either through the Coherence API, or through discovery using the Java ServiceLoader. See Starting Cache Servers in *Developing Applications with Oracle Coherence*.
  - **Remote Client MEMBER_JOINED and MEMBER_LEFT MemberEvents** - A proxy now sends `MEMBER_JOINED` and `MEMBER_LEFT` `MemberEvents` to all active services on the proxy when a remote client joins and leaves. This event enables management of a service's server side resources being retained per remote client. If a `MemberListener` is registered on a service and the environment has both remote and cluster member access to a service, the `MemberListener` may need to account for remote client `MemberEvent(s)`. For example, see Example 8-3 in Listening to Member Events section in *Developing Applications with Oracle Coherence*.
  - **Scheduled Backups** - Writing asynchronous backups has been enhanced to enable scheduling of these backups at a time interval after the primary has been written. See Scheduling Backups in *Developing Applications with Oracle Coherence*.
  - **Read Locator** - Coherence now allows for certain requests for data to be targeted to non-primary partition owners (backups) to balance request load or reduce latency. See Using the Read Locator in *Developing Applications with Oracle Coherence*.
  - **Cache Configuration Override** - Similar to the Coherence Cluster override, you can now specify a cache configuration override to override elements of existing cache configuration with new elements at runtime. See Using Cache Configuration Override in *Developing Applications with Oracle Coherence*.
  - **Extend the Operational Configuration File** - Adds support for custom namespace handlers in the Coherence operational configuration file. See Introduction to Extending Configuration Files in *Developing Applications with Oracle Coherence*.
  - **BigDecimal-related aggregators** - These aggregators now support the ability to set BigDecimal properties such as `scale`, `rounding mode`, `stripTrailingZeros`, and `MathContext` (where applicable) for the final result. See *Java API Reference for Oracle Coherence*.
  - **JSON objects in CohQL** - The Coherence Query Language (CohQL) now supports the ability to insert JSON objects as keys or values, as well as to query by and select JSON attributes. See Working with JSON Objects in *Developing Applications with Oracle Coherence*.
  - **Durable Events (Experimental)** - Coherence now supports an experimental feature which allows missed `MapEvents` to be replayed when a client disconnects. See Using Durable Events (Experimental) in *Developing Applications with Oracle Coherence*.
- **Topics Improvements**
  - A number of durability and stability improvements have been applied to make topics more stable during failover.
  - Topics now guarantee at least once delivery, whereas in previous releases this was not the case. A subscriber that is part of a group can commit a processed message to indicate that processing is complete and it should not be redelivered on failover.
  - Topic channels are now fairly allocated to the subscribers in a subscriber group; only a single subscriber receives messages from an allocated channel.

  - Subscribers will be timed-out after a configurable period of inactivity (or failure to heartbeat) causing their channels to be reallocated to remaining subscribers in the same group.
  - Added API methods to determine the number of unreceived elements for a `NamedTopic` subscriber or subscriber group.

  See Using Topics in *Developing Applications with Oracle Coherence*.

- **Persistence**
  - **Persistent Backups** - You can now enable and configure persistent backups which stores backup partitions on a disk, as additional copies of persisted primary one. See Using Persistent Backups in *Developing Applications with Oracle Coherence*.
  - **Parallel Recovery** - The parallel recovery feature enables Coherence to recover data in parallel within a member/process as well as in parallel across the cluster. See Parallel Recovery in *Administering Oracle Coherence*.
- **Distributed Concurrency** - The Coherence Concurrent module provides distributed implementations of the concurrency primitives from the `java.util.concurrent` package such as executors, atomics, locks, semaphores, and latches. See Implementing Concurreny in a Distributed Network in *Developing Applications with Oracle Coherence*.
- **Queues** - Coherence now includes an implementation of Queues as a data structure. See Using Blocking Queues in *Developing Applications with Oracle Coherence*.
- **Serialization/ POF**
  - **Portable Types and POF Maven Plug-in** - This release introduces Portable Types, which provide a way to add support for POF serialization to your classes by using annotations and without the requirement to implement serialization code by hand. See Using Portable Object Format in *Developing Applications with Oracle Coherence*.
  - **POF Configuration Discovery** - It is now possible to make POF configuration files discoverable at runtime by the `ConfigurablePofContext` class instead of needing to put them inside `<include>` elements. See Making POF Configuration Files Discoverable at Runtime in *Developing Applications with Oracle Coherence*.
- **Integrations**
  - **Internal**
    - **CDI Support** - Coherence provides support for Contexts and Dependency Injection (CDI) within the Coherence cluster members to inject Coherence-managed resources, such as NamedMap, NamedCache, and Session instances into CDI managed beans. See Using Contexts and Dependency Injection in *Developing Applications with Oracle Coherence*.
    - **MicroProfile Configuration** - Coherence MicroProfile (MP) Configuration provides support for Eclipse MicroProfile Configuration within Coherence cluster members. See Using Coherence MicroProfile Configuration in *Integrating Oracle Coherence*.
    - **MicroProfile Metrics** - Coherence MicroProfile Metrics provides support for Eclipse MicroProfile Metrics within the Coherence cluster members. See Using Coherence MicroProfile Metrics in *Integrating Oracle Coherence*.
  - **External**
    - **Helidon** - Coherence can be integrated with Helidon through Contexts and Dependency Injection (CDI). See Helidon.
    - **GraphQL Support through Helidon** - Using Helidon integration, you can enable access to Coherence data from GraphQL. See GraphQL.

        * **Kafka** - Coherence can now integrate with Kafka using Kafka Entry Store and Kafka Sink Connector. See Kafka.
        * **Micronaut** - Coherence now provides integration to Micronaut. See Micronaut Coherence.
        * **Hibernate** - Updated support for Coherence integration with Hibernate. See Integrating Hibernate and Coherence in *Integrating Oracle Coherence*.
        * **Spring** - Coherence can be integrated with Spring, which is a platform for building and running Java-based enterprise applications. See Integrating Spring with Coherence in *Integrating Oracle Coherence*.

- **gRPC** - Coherence introduces the ability to use gRPC to access Coherence caches. See Introduction to gRPC.
    - **gRPC Proxy** - A new Coherence gRPC proxy implementation of the services defined within the Coherence gRPC module. See Using the Coherence gRPC Server in *Developing Remote Clients for Oracle Coherence*.
    - **gRPC Java Client** - The Coherence Java gRPC Client is a library that enables a Java application to connect to a Coherence gRPC proxy server. See Using the Coherence Java gRPC Client in *Developing Remote Clients for Oracle Coherence*.
- **Coherence*Web - Apache Tomcat 9 Support** - Coherence*Web is now supported on Tomcat 9.
- **Clients**
    - **JavaScript Client** - The Coherence JavaScript Client allows Node applications to act as cache clients to a Coherence Cluster using gRPC framework as the network transport. See Coherence JavaScript Client.
    - **Go Client** - The Coherence Go Client allows native Go applications to act as cache clients to a Coherence cluster using gRPC for the network transport. See Coherence Go Client.
    - **Python Client** - The Coherence Python Client allows Python applications to act as cache clients to an Oracle Coherence cluster using gRPC as the network transport. See Coherence Python Client.
    - **Client for .NET** - Coherence for .NET 14.1.2 supports .NET 6 and .NET 8. Coherence for .NET 14.1.2 includes session support for ASP.NET 6 and 8. Prior versions of Coherence for .NET used an `app.config` file. Coherence for .NET 14.1.2 now uses `appsettings.json`, which follows standard .NET practices. Coherence 14.1.2 server ASP.NET session support is not compatible with older Coherence for .NET versions. For more information, see Oracle Coherence for .NET.
- **Security**
    - **SSL Improvements** - Various SSL improvements that enable more flexible configuration and allow customizations through extensions. See Using Private Key and Certificate Files and Using Custom Keystore, Private Key, and Certificate Loaders in *Securing Oracle Coherence*.
    - **New TLS/SSL socket provider configuration element** - `client-auth` element controls whether the socket provider should use one-way or two-way TLS/SSL authentication. See Configuring TLS/SSL Authentication in *Securing Oracle Coherence*.
    - **New Password Provider for DefaultController Keystore** - This password provider allows you to obtain the passwords from the DefaultController keystore. See Using Custom Password Providers in *Securing Oracle Coherence*.

- **Management/Administration**
  - **OpenTelemetry API** - The OpenTelemetry API provides developers with visibility into cache operations within the cluster. See Distributed Tracing in *Developing Applications with Oracle Coherence*.
  - **Coherence Metrics** - Additional dependencies are no longer required when using the coherence-metrics module. In addition, a new Coherence Metrics endpoint for WebLogic managed Coherence servers allows the scraping of metrics using metrics gathering systems such as Prometheus. See Using Oracle Coherence Metrics in *Managing Oracle Coherence*.
  - A new system property, `coherence.metrics.http.path`, that specifies the metrics context root path. See Using Metrics System Properties in *Managing Oracle Coherence*.
  - **Coherence Management over REST** - The dependencies required to enable Management over REST have been reduced significantly. The only additional dependency required to enable management over REST is `coherence-json.jar`. See REST API for Managing Oracle Coherence.
  - **Health Check API** - A new health check API to enable application code to determine the health of the local Coherence member. See Using the Health Check API in *Managing Oracle Coherence*.
  - **Micrometer Metrics** - The coherence-micrometer module provides integration between Coherence metrics and Micrometer allowing Coherence metrics to be published through any of the Micrometer registries. See Using Coherence Micrometer Metrics in *Managing Oracle Coherence*.
  - **Access to Remote MBeans** - The new `jmxserviceurl` script works with the JConsole utility to remotely access MBeans. See Accessing MBeans of a Running Coherence Cluster Using the JConsole Utility in *Managing Oracle Coherence*.
  - **New Reports - Executor, View, Storage, and Proxy Connections** - See Understanding the Executor Report, Understanding the View Report, Understanding the Cache Storage Report, and Understanding the Proxy Connections Report in *Managing Oracle Coherence*.
  - **New Topics Reports** - See Understanding the Topic Report, Understanding the Topic Subscribers Report, Understanding the Topic Subscriber Groups Report in *Managing Oracle Coherence*.
  - **Persistence and Persistence Detail Reports** - Additional columns added to these reports. See Understanding the Persistence Detail Report and Understanding the Persistence Report in *Managing Oracle Coherence*.
  - **New ExecutorMBean** - Provides statistics for the executor services that run in a cluster. See ExecutorMBean in *Managing Oracle Coherence*.
  - **New ViewMBean** - Provides statistics for view caches that run in a cluster. See ViewMBean in *Managing Oracle Coherence*.
  - **New HealthMBean** - Provides information about health checks configured in a cluster. See HealthMBean in *Managing Oracle Coherence*.
  - **ServiceMBean** - Additional attributes added to the ServiceMBean track Persistent Backups storage utilization. See ServiceMBean in *Managing Oracle Coherence*.
  - **StorageManagerMBean** - Additional attributes added to StorageManagerMBean to show Indexing Total Millis and Index Total Units. See StorageManagerMBean in *Managing Oracle Coherence*.

  - **New PagedTopic MBean** - Provides statistics for Topic services running in a cluster. See PagedTopic MBean in *Managing Oracle Coherence*.
  - **New PagedTopicSubscriber MBean** - provides statistics for Topic Subscribers running in a cluster. See PagedTopicSubscriber MBean in *Managing Oracle Coherence*.
  - **New PagedTopicSubscriberGroup MBean** - provides statistics for Topic Subscriber Groups running in a cluster. See PagedTopicSubscriberGroup MBean in *Managing Oracle Coherence*.
  - **WKA Improvements** - Added the ability to provide a comma-separated list of addresses when specifying a Well Known Address (WKA). See Using Well Known Addresses.
  - **Coherence Operator** - A number of major enhancements have been made to the Coherence Operator. See Coherence Operator.
  - **Coherence VisualVM Plug-in** - Updates have been made to support new functionalities added in 14.1.2. See Coherence VisualVM Plugin Releases.
  - **Coherence CLI** - Updates have been made to support new functionalities added in 14.1.2. See Coherence CLI Release.
- **Federation** - Additional attributes added to Origin and Destination MBeans to show errors and replication estimation. See OriginMBean and DestinationMBean in *Managing Oracle Coherence*.
- **Examples** - Coherence guides and tutorials are now hosted on the Coherence GitHub Repository and are documented here: Examples - Guides & Tutorials Overview.

Breaking Changes

Learn about updates in Coherence 14c (14.1.2.0.0) that introduce potentially incompatible changes between Coherence releases.

- PortableType Annotation Now Requires type-id Attribute
- "vendor:" Prefix Removed from Prometheus Generated Metrics
- Changed Return Type of Coherence.start() Method
- MapViewBuilder and ViewBuilder
- Client for .NET

PortableType Annotation Now Requires type-id Attribute

In Coherence 14.1.2.0.0, the `@PortableType` annotation was updated to require a `type-id` attribute.

In previous releases, `type-id` was optional and auto-generated, which could lead to serialization and schema evolvability issues as the `type-id` was not guaranteed to be consistent between each build for each class.

To correct this issue, the `type-id` is now mandatory on the `@PortableType` annotation, and in `coherence-pof-config.xsd`. Therefore, annotations using `@PortableType` without an `id` will fail to compile. You must now supply a unique `type-id` when the annotation is used. For example:

```
@PortableType(id = 1000)
public class Customer {
```

"vendor:" Prefix Removed from Prometheus Generated Metrics

In this release, the default value of the system property "`coherence.metrics.legacy.names`" has been changed from `true` to `false`, to remove the "`vendor:`" prefix from generated Prometheus metrics.

This prefix was deprecated several releases ago.

For more information, see the Coherence Operator documentation, Publish Metrics.

Changed Return Type of Coherence.start() Method

Enhanced the `Coherence.start()` method to return a `CompletableFuture<Coherence>` instead of a `CompletableFuture<Void>`. This allows a more fluent API when using static factory methods to create and start a Coherence instance. This is a breaking change in applications that specifically assign the result of calls to `Coherence.start()` to a `CompletableFuture<Void>` variable.

MapViewBuilder and ViewBuilder

Fixed an issue with the generics of `MapViewBuilder` and `ViewBuilder` that would prevent the proper use of the `map()` function. `MapViewBuilder` and `ViewBuilder` have had their class-level generics simplified to `<K, V>` from `<K, V_BACK, V_FRONT>`. The `map()` function has been changed to: `public <U> ViewBuilder<K, U> map(ValueExtractor<? super V, ? extends U> mapper)`, where `U` represents the type of the extracted value. This change also necessitated similar changes to the generics of `NamedMap.view()` and `NamedCache.view()`. These methods have also been simplified to `<K, V> from <K, V_BACK, V_FRONT>`.

Note:

This is a backward-incompatible change, but will have an impact only during compilation.

Client for .NET

- Coherence for .NET configuration, which used to be in `app.config`, is now in `appsettings.json`.
- Coherence 14.1.2 ASP.NET session support is incompatible with older Coherence for .NET versions.
- The following classes that relied on the deprecated .NET `BinaryFormatter` class have been removed from Coherence for .NET: `BinarySerializer`, `OptimizedBinarySerializer`, `BinaryPofSerializer`, and `SafeConfigurablePofContext`.

Deprecated Features

Learn about the deprecated and desupported features of Oracle Coherence.

This section includes the following topics:

- Deprecated Features for 14.1.2.0.0

Deprecated Features for 14.1.2.0.0

A brief description of the deprecated features for 14.1.2.0.0.

This section includes the following topics:

- Oracle Solaris Support
- SafeSortedMap
- ImmutableArrayList.getSortedSet
- Coherence .NET Client
- Memcached Adapter

Oracle Solaris Support

Coherence no longer supports the Oracle Solaris platform for the Coherence for C++ client.

SafeSortedMap

The `SafeSortedMap` class is now depreciated.

ImmutableArrayList.getSortedSet

The deprecated `ImmutableArrayList.getSortedSet` will be removed. This method and the `SortedSet` interface implementation will be removed from `ImmutableArrayList` in a future release.

Coherence .NET Client

Coherence .NET client 14.1.2.0.0 has deprecated `HashDictionary(SerializationInfo info, StreamingContext context)` and `PortableException(SerializationInfo info, StreamingContext context)` constructors and `GetObjectData` methods in `PortableException`, `RequestIncompleteException`, and `HashDictionary` classes, which overrode legacy serialization methods.

Memcached Adapter

The `memcached` adapter is now deprecated.

3

Known Issues and Workarounds

Learn about the known issues at the time of release.
This chapter includes the following section:

Changing the Partition Count When Using Active Persistence

Issue

The partition count cannot be changed when using active persistence. If you change a services partition count, then on restart of the services all active data is moved to the persistence trash and must be recovered after the original partition count is restored. Data that is persisted can only be recovered to services running with the same partition count.

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

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

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

Workaround

To recover the data:

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

Disabling Inlining in Java Versions Greater than 8

When using Java versions greater than Java 8 (for example, Java 11), one of the directives below should be provided as a JVM Option (on the command line). This step is to avoid a segmentation fault (SIGSEGV) that has been observed due to a compiler bug that is being worked on by the Java team.

- `-XX:CompileCommand=exclude,com/tangosol/coherence/component/util/daemon/queueProcessor/service/Grid.onInterval`
- `-XX:-Inline`

The first option will exclude the problematic method from being inlined by the compiler while the latter will disable inlining altogether.

Binary Incompatibility with Older Versions

There is a binary incompatibility between Oracle Coherence release 14.1.1.0 and Oracle Coherence release 14.1.2. This incompatibility requires upgrading users to recompile your applications against 14.1.2. It is a binary incompatibility only; not an API or functional incompatibility.

When you run Coherence-based applications prior to release 14.1.2 with the `coherence.jar` file from that release, you may encounter the following exception at runtime:

```
java.lang.ClassNotFoundException: com.tangosol.net.NamedCache$Option
```

For example, from `Session.getCache("<cache name>")` calls. This exception occurs because the nested class `Option` moved, in a refactoring, from `NamedCache` to its supertype `NamedMap` between 14.1.1.0 and 14.1.2. This refactoring preserves the API compatibility for the previous code using `NamedCache.Option`, but requires recompilation of that code against the 14.1.2 `coherence.jar`.

Workaround

To run a Coherence-based application prior to Oracle Coherence 14.1.2 with that release, you should first recompile the application against 14.1.2. Recompiling will avoid encountering a binary incompatibility exception at runtime.

pof-maven-plugin Version is Non-Compliant

Issue

The `pof-maven-plugin` packaged in the installer has a version qualifier, for example, `14.1.2-0-0-112309` in the plug-in `pom.xml` file and in the Maven plug-in metadata, under `META-INF/maven/plugin.xml`, which is generated and packaged inside the `pof-maven-plugin` JAR file.

Workaround

When the `oracle-maven-sync` plug-in is used to put the `pof-maven-plugin` artifacts into the local Maven repository, the plug-in output will show a message as follows:

```
[INFO] Installing <OracleHome>/coherence/plugins/maven/com/oracle/coherence/pof-maven-
plugin/14.1.2/pof-maven-plugin.14.1.2.jar to $HOME/.m2/repository/com/oracle/coherence/
pof-maven-plugin/14.1.2-0-0-112309/pof-maven-plugin-14.1.2-0-0-112309.jar
[INFO] Installing <OracleHome>/coherence/plugins/maven/com/oracle/coherence/pof-maven-
plugin/14.1.2/pof-maven-plugin.14.1.2.pom to $HOME/.m2/repository/com/oracle/coherence/
pof-maven-plugin/14.1.2-0-0-112309/pof-maven-plugin-14.1.2-0-0-112309.pom
```

When the `pof-maven-plugin` is used and referenced inside the Maven `pom.xml` file, the `pof-maven-plugin` version must be specified as the same exact version with the qualifier, as shown in the output message, for example `14.1.2-0-0-112309`:

```
<groupId>com.oracle.coherence</groupId>
<artifactId>pof-maven-plugin</artifactId>
<version>14.1.2-0-0-112309</version>
```

4

Bugs Fixed and Enhancements in This Release

Learn about the bugs fixed and enhancements in this release.
This chapter includes the following section:

- Oracle Coherence for Java

Oracle Coherence for Java

New features, improvements, and bug fixes added to Oracle Coherence for Java components.

Enhancements and Fixes for 14.1.2.0.0

- **Persistence**
  - Fixed an issue where a service with on-demand persistence mode could be blocked by the recovery quorum from a system property.
  - Improved recovering from persistence by deferring contentious maintenance tasks.
  - Fixed an issue where partition recovery could take a long time for a large cluster, especially with persistence using shared disk storage.
  - Fixed an issue where partition distribution may fail to reach a balanced state within five minutes after recovering from a persistence snapshot.
  - Fixed an issue where a service using active persistence may be terminated with a `PersistenceException` due to reaching the guardian timeout on a blocked operation.
  - Made the opening of persistent stores gradual. Underlying store files, one per partition, will only be created when they start containing data.
  - Fixed an issue with persistent backups where a deadlock situation may occur during partition re-distribution.
  - Added an element to the service list indicating the persistence mode in use.
  - Improved the cleanup process of persistence files to handle case when data is deleted followed by a period with no cache activity.
  - Added log messages to report when individual partitions cannot be accessed due to indexing, partition migration or persistence, and added the `IndexingTotalMillis` attribute to the `StorageManager` MBean.
  - Improved persistence to recover data in parallel within a member/process, in addition to in parallel across the cluster. This allows the cluster, and more importantly the associated data, to be made available as quickly as possible.
  - Fixed a rare case where persistence snapshots could produce a `NullPointerException` if a rolling upgrade to remove cache mappings was not correctly done.
  - Fixed an issue where specifying a wrong value for persistence mode would be silently ignored and default to "`on-demand`".

  - Fixed an issue where rolling restart with persistence done concurrently by two or more nodes may result in the cache service going into the "orphaned" state and require a cluster restart.
  - Fixed an issue where data loss might occur after multiple rolling restarts when backup persistence is enabled.
  - Fixed an issue in persistence to ensure errors are caught in a rare and unexpected part of the recovery protocol.
  - Fixed an issue where a persistence snapshot could contain stale cache data leading to a recovery error.
  - Fixed an issue where a persistence snapshot recovery operation could hang indefinitely.
  - Fixed an issue where using persistent backups can result in an "IllegalArgumentException: unknown extent identifier" error under load and while performing rolling restarts.
  - Fixed an issue where a distributed service with active persistence may be terminated by an async write to an old persistent store.
  - Fixed an issue where a service could repeatedly fail to recover a partition if persistence is using local disk storage.
  - Fixed an issue where a service could repeatedly fail to recover a partition.
  - Fixed an issue where recovering snapshots with indices would result in corrupted index contents.
- **Executor Service**
  - Added support to allow the definition of custom executors for the remote executor service via XML configuration.
  - Added support for advanced task orchestration across multiple JVMs via the `RemoteExecutor` API.
  - Fixed an issue where executor tasks may not execute under high load.
  - Added support for JDK 21 VirtualThread-per-task executors to the Coherence Executor Service.
  - Updated the executor services CronTask to allow the user to configure whether or not the wrapped task should be cloned upon each successful execution or not where as previously, it always performed the clone which prevents the task from maintaining internal state.
  - Fixed an issue that prevented the use of additional filters when using `NamedOrchestration` to orchestrate tasks to the executor service.
  - Fixed an issue which could prevent tasks from being executed by `RemoteExecutors`.
  - Fixed an issue where multiple long running tasks may prevent other tasks from being executed by concurrent Executors.
  - Fixed an issue where the executor service would recreate a failed-over task upon re-execution after a yield.
  - Fixed an issue where the executor service would incorrectly increment the tasks-in-progress count when re-executing a yielded task.
  - Fixed an issue where an `IllegalArgumentException` could be thrown when looking up a named executor when using POF.

  - Fixed an issue where an executor service task submitted with the Debugging option wouldn't log any task execution details unless the `coherence.executor.trace.logging` system property was set to true and `coherence.log.level` is at least seven.
  - Fixed an issue where the executor service in-progress count could be incremented twice per task.
  - Fixed a rare issue with the executor service where a dynamically registered executor could result in an inflight task not completing.
  - Fixed an issue where the concurrent executor service calls `ensureCache` on the service thread during shutdown resulting in a potential deadlock warning message.
  - Fixed a rare issue where a Coherence cache server may be inadvertently restarted when attempting to do an asynchronous shutdown of a registered executor.
  - Fixed an issue with the executor service where task properties could be set after a task had completed.
  - Fixed a rare issue in the executor service where a task executing across multiple members may not complete properly.
  - Updated the system properties for persistence mode for the config service to be consistent with those of the concurrent executor service.
  - Added the ability to configure the number of worker threads the concurrent cache service using two system properties: `coherence.concurrent.distributed.threads.{min|max}`. The concurrent cache service will honor the global system properties `coherence.distributed.threads.{min|max}` if provided and the concurrent versions are not set.
- **Federation**
  - Fixed an issue where federation may get stuck repeatedly requesting a retry of the same cache changes.
  - Fixed an issue where partition distribution for a large cluster could take a long time to reach the desired state.
  - Fixed an issue where in rare cases a partition may be stuck with repeated `MISSING_RECORDS_IN_RANGE` trace messages resulting in changes not being sent to a destination participant after stopping then starting federation.
  - Fixed an issue where the `DestinationController` state in the `DestinationMBean` might not be correct.
  - Enabled TCP KeepAlive for federation connections.
  - Added federation `ReplicateAllPercentComplete` to Coherence metrics.
  - Added currentConnectionCount and mapMembers attributes to the federation `Origin` MBean to provide the current connection count and a list of members from which the connections are established.
  - Added the ability to enable/disable federation trace logging using the federation `Destination` MBean.
  - Added support for specifying a separate `AddressProvider` per (Participant, Service) pair for environments which have multiple `FederatedCache` services and are not using the `NameService` to look up federation connections.
  - Fixed an issue where changes for a partition may stop being federated under rare circumstances.

- Fixed an issue where unacknowledged federation journal records may be garbage collected.
- Fixed an issue where the wrong `TransferEvent` type was reported in some federation trace level log messages
- Added additional metrics and MBean attributes to improve federation `replicateAll` operation monitoring and error reporting.
- Added `StateCode` attribute/metric to Federation `Destination` MBean so users can map a State name to a numeric code.
- Enhanced federation connection related information level logging to provide more information about the connection and more details about the life cycle of the connection.
- Fixed an issue where an `IllegalStateException` may be thrown when federating large entries.
- Fixed an issue where federation may attempt to apply federated changes during service shutdown after the local storage for caches has already been released.
- Fixed an issue where a federation service does not work if a `KeyAssociator` or `KeyPartitioningStrategy` is defined. Cache entries with `PartitionAwareKey` type keys will now bypass `KeyAssociators` and `KeyPartitioningStrategies` for federation services.
- Fixed an issue where incoming federation messages may cause the cluster service to be unable to complete its shutdown process.
- Fixed an issue where a `NullPointerException` may be thrown in some rare cases when federation is preparing a `ReplicateMessage` during partition migration.
- Fixed an issue where a federation transition to the `ERROR` state may not be done cluster wide.
- Fixed an issue where certain types and sizes of federation journal entries which are too large to fit in Elastic Data may not be identified as requiring splitting.
- Added a `TotalRetryResponses` attribute to both `Destination` and `Origin` federation MBeans to track the total number of retry responses to `ReplicateMessages`.
- Added the service name to the federation log message that indicates that an endpoint address could not be created for a remote participant.
- Changed the default calculator to `BINARY` for federation's internal metadata caches.
- Fixed an issue where federation may fail to apply changes to a cache which has been released or destroyed.
- Fixed an issue where there may be unnecessary redundant updates to federation's internal metadata caches in some rare scenarios.
- Fixed an issue where some cache updates may not be federated under very rare circumstances.
- Fixed an issue where the federation destination MBean state attribute may be null in some rare cases.
- Enabled heartbeat messages on federation connections to prevent the dropping of idle connections by load balancers and firewalls.
- Increased the MessageBus ack timeout for federation to tolerate potentially higher latency inter-cluster connections.

  - Fixed an issue in federation where a `NullPointerException` may be thrown by the `EnvelopeAggregator` in some rare cases.
  - Fixed an issue where MessageBus connection migration attempts may occur for federation connections.
  - Fixed an issue where a deadlock may occur in federation between `UnsolicitedEntryAddProcessor` and `FederatedCacheInterceptor`.
- **Clustering/Services**
  - Enhanced the service startup messages for Partitioned, Invocation and Proxy services to display the serializer used.
  - Fixed an issue where the cluster service may be stopped in very rare circumstances due to an unhandled `UnsupportedOperationException`.
  - Fixed an issue where there may be leaked threads as a result of starting and stopping Coherence multiple times within the same JVM instance.
  - Fixed an issue where it was not possible to specifically set the serializer for the Coherence system config separately from the default `coherence.serializer` property.
  - Corrected the displayed version, for certain Coherence versions, of members within the master member set.
  - Fixed an issue where a `NullPointerException` may be thrown during service shutdown.
  - Improved a warning log to describe the potential communication problem due to packet delivery failures to be clearer and more correct.
  - Fixed "Started Cluster" log message to clarify that a JOINING member's version is a transport protocol compatibility version, not the member's actual Coherence version.
  - Improved a warning log to describe the potential communication problem due to packet delivery failures more clear and correct.
  - Enhanced `DefaultServiceFailurePolicy.POLICY_EXIT_PROCESS` from halting the process to graceful exit, allowing registered shutdown listeners to run. The process is halted if graceful exit does not complete within `coherence.shutdown.timeout` duration.
  - Improved the cluster member join algorithm to avoid members blocking each other when there are large number of members joining the cluster simultaneously and system resources are under heavy load.
  - Fixed an issue where a `ClassCastException` may be thrown when transferring partitions containing entries with an expiry.
  - Implemented periodic flushing of pending messages to reduce native memory usage when sending multiple messages at once.
  - Fixed an issue where a PartitionedCache may be terminated due to an unhandled `NullPointerException` in `onBackupListenerRequest()`.
  - Fixed an issue where a Coherence worker thread daemon pool may deadlock if the pool is stopped while a resize task is executing.
  - Fixed an issue where a PartitionedCache service may terminate unexpectedly due to an unhandled `ArrayIndexOutOfBoundsException` being thrown while processing an `UpdateIndexRequest`.
  - Fix an issue in PartitionedCache where a `NullPointerException` may be thrown in `onBackupListenerAllRequest` and `onBackupListenerRequest` when a member sends one of these requests and then suddenly leaves cluster.

  - Fixed an issue where the partitioned cache service thread could hit the guardian timeout while processing deferred events during partition recovery or failover.
  - Fixed an issue where `DefaultCacheServer.shutdown()` does not shut down the Coherence server gracefully.
  - Fixed an issue where partition redistribution could be stuck after a snapshot recovery.
  - Reduced the overhead of key set manipulation when applying indexes
  - Reduced memory allocation when creating a `PartitionSet` instance for a single partition, to improve memory profile of parallel queries against caches with high partition count.
  - Fixed an issue where PartitionedCache's PartitionControl may hang indefinitely due to a discarded backup message.
  - Fixed an issue where a `NullPointerException` could be thrown during partition redistribution.
  - Fixed an issue where a service could be terminated while finalizing a cache request.
- **Caching /Cache-Stores**
  - Fixed an issue where `CacheStore.eraseAll()` had no path to be called on `NamedCache` bulk operations such as `invokeAll()`. With this fix, `CacheStore.eraseAll()` is called when `NamedCache.invokeAll()` is invoked with a "remove" processor or when `NamedCache.keySet().removeAll()` or `NamedCache.entrySet().removeAll()` are called.
  - Added default implementations to CacheLoader and CacheStore interfaces for `loadAll()`, `storeAll()`, and `eraseAll()`.
  - Fixed an issue where the `getOrDefault()` call on a cache that uses RWBM with cache store does not propagate the call through to the cache store when the entry does not exist.
  - Added service-name as an allowable child element of near-scheme
  - Fixed an issue where `InFilter` queries could take longer to return.
  - Fixed an issue where `AsyncNamedCache.putAll()` can result in `CacheLoader/CacheStore load()/loadAll()` calls.
  - Fixed an issue where entry processor invocations may never be re-sent when re-distribution takes place at the same time.
  - Fixed an issue where `ContinuousQueryCache` did not handle `NamedCacheDeactivationListener` registrations correctly. This can cause issues if using View Scheme on Extend or gRPC Proxy servers.
  - Fixed an issue where a write-behind remove might get stuck when there are outstanding pending write-behind operations on the entry.
  - Fixed an issue where `CacheLifecycleEvents` are not emitted for existing caches or on storage-disabled nodes.
  - Fixed an issue where incremental cache eviction could temporarily hold references to evicted cache keys.
  - Fixed an issue where the bootstrap API can deadlock if the `stop()` method is called on a Coherence instance before start-up has completed.
  - Improved `NearCache getOrDefault` and `computeIfAbsent` to utilize the front map.

- Fixed an issue where a `ClassCastException` or `NullPointerException` may be thrown by `InvocableMap` during service config processing in some very rare scenarios.
- Fixed an issue where `ExtensibleConfigurableCacheFactory.DependenciesHelper newInstance()` methods present in Coherence CE are not present in commercial Coherence versions.
- Fixed an issue where, in certain cases, remote invocation using an ArrayFilter would use excessive CPU.
- Fixed an issue where `CacheMappingRegistry.register()` or `SchemeMappingRegistry.register()` will throw a `NullPointerException` if the registry is not initialized.
- Fixed an issue where pending events could remain on backup members for a longer time than expected if not acknowledged by clients.
- Fixed a performance regression in InFilter.
- Fixed an issue where CQC synchronization with the reference cache may miss updates when its initialization runs concurrent with changes.
- Fixed an issue where services could restart during graceful shutdown when using the Bootstrap API.
- Fixed an issue where put, get, and remove operations from a gRPC client or when using `AsyncNamedMap` or `AsyncNamedCache` did not trigger the cache store.
- Fixed an issue where query processing enhancements introduced a performance degradation when using indices in filter-based calls (`entrySet`, `invokeAll` with filter or aggregations).
- Fixed an issue where near cache key lock(s) were not being properly released when the back map is truncated and the near cache is using the PRESENT invalidation strategy. The observable failure is thread(s) hung waiting for near cache key locks that are never released.
- Fixed an issue where a `NullPointerException` may be thrown traversing `BinaryRadixTree` nodes during a `CompactSerializationCache` operation.
- Added new `ValueExtractor` factory methods to the `com.tangosol.util.Extractors` class.
- Narrowed down the return type of factory methods in `Extractors`, `Processors`, and `Aggregators` classes to eliminate the need for casting of created instances.
- Added support for Java records to `UniversalExtractor`.
- Added the ability to pass a custom Executor to `AsyncNamedCache` to use to complete the invoked futures instead of using the Coherence common pool. This is useful if it is a requirement to strictly enforce order of completion of async futures.
- Enhanced `AsynchronousAgent` to complete async API responses using daemon pool instead of service thread.
- Improved performance of filter-based aggregators by leveraging partitioned index.
- Improved the filter reordering logic for composite filters.
- Improved parallelism of queries, aggregations and bulk entry processor requests by splitting them by partition, instead of by member.
- Added `CacheEvent.isExpired()`. On `ENTRY_DELETED`, `isExpired()` will return true if the entry was evicted due to expiry.

  - Added write-behind support for cache store `erase()` and `eraseAll()` operations.
  - Fixed an issue where using a conditional index on a key extractor resulted in the corresponding index not being updated when entry values were modified, and queries would return incorrect results.
  - Fixed an issue where partitioned queries took longer to execute than before, this fix now provides as fast or faster execution due to the ability to run queries in parallel across partitions.
  - Fixed an issue where query results could include an entry that does not match specified filter under heavy concurrent updates.
  - Fixed an issue where calling `AsyncNamedCache.put()` ignored any expiry value configured for the cache, causing entries to never be expired.
  - Fixed a performance issue which can occur when a large number of cache entries expire at approximately the same time.
  - Fixed an issue where calling `AsyncNamedMap` values or `entrySet` methods with a Filter could fail to return all of the values or entries.
  - Fixed an issue in `DistinctValues.accumulate()` that results in `onAggregateFilterRequest()` throwing a `NotSerializableException`.
  - Enhanced index support, as part of partitioning indices, to avoid index contents being stored more than necessary.
  - Fixed an issue where a `NearCache` may not detect and release a lock on a cache key that is held by a terminated thread, resulting in a "Detected state corruption on KEY..." log message.
  - Fixed an issue where Caffeine could not be configured or used as a near cache front map.
  - Fixed an issue where an `EntryProcessorEvent.EXECUTED` event raised by an invokeAll may incorrectly contain an empty entry set.
  - Fixed an issue where expired entries are not evicted for `ReadWriteBackingMap`.
  - Fixed an issue where `NullPointerException` could occur in the index rebuild thread during failover, and lead to worker threads hanging waiting for index ready.
  - An internal ForkJoinPool is now used to run queries in parallel across all owned partitions.
  - Enhanced Coherence gRPC proxy and client to be configurable using the Coherence operational and cache configuration files. Added support for configuring gRPC secure sockets using the same socket provider approach used in the rest of Coherence.
- **Security**
  - Added support for using a password provider for the keystore password in the `AccessController`.
  - Added the ability to configure Coherence socket providers to use TLS/SSL private key and certificate files instead of keystores and to be able to load keystores, private keys and certificates from custom sources instead of the file system.
  - Fixed an issue where a service restart could throw a `SecurityException` with the message "No security token available" when the security framework is enabled.
  - Added an option to configure a global socket provider that will be applied to all network sockets created by Coherence. This allows a single place to configure TLS settings

    that apply to Coherence cluster communication, extend proxy and client communication, gRPC channels, and such.

  - Fixed an issue where hostname verification could fail due to missing peer certificates when using Coherence with TLS enabled on Java 17 and higher.
  - Fixed an issue where traffic over TLS 1.3 connections may hang.
  - Added an enhancement to allow the client auth mode to be configured for an SSL socket provider. Previously this behavior was fixed to "required" if a trust store was configured. The enhancement allows the mode to be "none", "wanted" or "required".
  - Enhanced the default SSL `HostnameVerifier` to be able to verify wild-card Subject Alternate Names and to disable localhost matching by default. Matching of localhost can be enabled via a system property.

- **CohQL**
  - The JLine console input Java library is no longer included as part of a Coherence installation. For JLine functionality when using CohQL or Coherence console command line tools, download and include the JLine library in the tool classpath.
  - Fixed a regression in query.sh by adding new command line argument "-v" to indicate that the CohQL statement should be echo'ed to output. Also fixed unhandled JLine exceptions for EOF and user interruption.
  - Fixed an issue where correct CohQL comparison expressions could raise an exception stating "The use of identifier on both sides of an expression is not supported".
  - Fixed an issue where the history file was not getting saved to disk for CohQL query console and Coherence console when using `jline.jar`.
  - Fixed an issue where CohQL could return incorrect results when compound conditions are used with parenthesis on the first part of the query statement.
  - Enhanced CohQL to print out the Exception call stack when trace is turned on.
  - Fixed an issue where a `StackOverflowError` could be thrown when comparing two identifiers.
  - Fixed an issue where backup and restore commands would use the default Java serializer rather than the pass-through binary serializer.
  - Fixed a thread safety issue in `QueryHelper` and `FilterBuilder` that may result in corruption of the underlying parser token table.
  - Fixed a thread safety issue in CohQL `QueryHelper` and `FilterBuilder` that may result in corruption of the underlying parser token table.

- **Elastic Data**
  - Fixed an issue where federation `JournalRecord` cache garbage collection may be very slow when there are a large number of entries to be removed.
  - Fixed an issue where queries using an index with Elastic Data were running slower than they should.
  - Replaced the usage of synchronized blocks with locks to prevent hangs and allow for better thread utilization in `JournalCache`.

- **Coherence*Web**
  - Added support for and certified running Coherence*Web on the following web containers: Wildfly/Undertow web container, Jetty 9.4 and Tomcat 9.
  - Certified Coherence*Web support for servlet spec 3.1.

- Fixed an issue where a Coherence*Web thread may hang waiting to acquire a lock when releasing ownership of a session.
- Added additional session information to the session log: "Session Reaping Cycle Seconds" "Session Logger Level", "Session Log Invalidation Exceptions", and "Configuration Consistency Required".
- Fixed an issue in Coherence*Web where overflow attributes in a session may not be saved.
- Removed the "web-sessions" report. The same information is available in the "cache-effectiveness" report.
- Fixed an issue in Coherence*Web to disallow invalid characters in MBean names.
- Fixed an issue where session invalidation might lead to "no usable session model error" and stuck threads.
- Added a warning level log message for when HTTP session reaping takes longer than coherence-reaperdaemon-cycle-seconds.
- Fixed an issue in Coherence*Web where incorrect session data may be returned if there are members in the cluster with a lower patch level than 12.2.1.4.16, which does not have the "splitting large values" feature.
- Fixed an issue in Coherence*Web where a lock on an invalidated session might not be released, causing other threads waiting on the lock to be stuck indefinitely, when a session is invalidated and replaced by a new one.
- Fixed an issue where when `coherence.session.log.invalidation.exceptions` is set to false, reaper still logged invalidation exceptions.
- Fixed an issue in Coherence*Web where many threads are stuck due to the session reaper not releasing the lock on the session being reaped.
- Fixed an issue where `ServletRequestEvents` are not dispatched when Coherence*Web is used.
- Fixed an issue with the Coherence*Web installer to skip instrumenting and retain integrity of signed nested jar files.
- Fixed an issue in Coherence*Web where a Servlet may fail with com.tangosol.net.RequestTimeoutException: Failed to obtain cluster ownership leading to struck thread.
- Fixed an issue in Coherence*Web where an application thread could end up blocked waiting for a lock on a `SegmentedConcurrentMap` due to the lock not being released by the reaper thread.
- Fixed an issue in Coherence*Web where a thread could block indefinitely in `com.tangosol.util.SegmentedConcurrentMap$LockableEntry.waitForNotify`.
- Fixed an issue where if a war-scoped deployment is used, a "java.lang.IllegalStateException: SafeNamedCache was explicitly destroyed" exception may be thrown due to Coherence*Web not cleaning up its cache references when handling a contextDestroyed event.
- Changed Coherence*Web to no longer use replicated caches.
- Added a Coherence*Web Context Parameter "`coherence-use-elastic-data`" to enable/disable the use of elastic data for storing session data. The default setting is true.
- Enhanced Coherence*Web to support sessions containing data that exceeds the Elastic Data maximum entry size limit.

  - Fixed a memory leak in Coherence*Web `SplitHttpSessionCollection`.
- **Serialization/POF**
  - Fixed an issue in POF that would result in `java.io.StreamCorruptedException:` unknown user type:5.
  - Fixed an issue where all "`\uXXXX`" character sequences (for example: "`\usr\bin`") were assumed to be a Unicode escape sequence when writing a JSON value.
  - Fixed an issue that resulted in unnecessary deserialization of entry values during write-behind.
  - Fixed an issue where `ScriptAggregator`, `ScriptFilter` and `ScriptProcessor` could not be serialized using JSON.
  - Fixed an issue where deserialization of 4-byte UTF-8 sequences would fail.
  - Fixed an issue where automatic discovery of the Coherence JSON serializer at start-up could fail with Java 17 and later.
  - Fixed an issue where `PortableTypeGenerator` may generate an incorrect implementation for "`public Evolvable getEvolvable(int nTypeId)`".
  - Fixed an issue where it was not possible to specifically set the serializer for the Coherence system config separately from the default `coherence.serializer` property.
  - Improved `PortableTypeGenerator` to report an error when a POF annotated field is declared as final.
  - Improved JSON serialization of `Big{Decimal,Integer}` so these types may be better handled by gRPC clients.
  - Added Helidon JEP-290 serialization configuration files to allow Coherence to work with Helidon.
  - Enabled configuring lambdas serialization mode in the operational configuration by setting the `<cluster-config/lambdas-serialization>` element to "`static`" or "`dynamic`".
  - Added a new Gradle plug-in for POF serialization that instruments classes at build time (similar to the Coherence POF Maven plug-in).
  - Improved the efficiency of several of the methods on `AsyncNamedMap` and `AsyncNamedCache` by eliminating unnecessary serialization or deserialization wherever possible.
  - Fixed an issue where deserialization of `Optional<Object>` fails with "`java.lang.ClassNotFoundException`" due to the incorrect ClassLoader being used.
  - Fixed an issue that prevented the serialization/deserialization of lambdas with JDK 21.
  - Removed identity token deserialization in `NameService TcpAcceptor` processing.
  - Added support for high-performance (raw) serialization of primitive arrays.
  - The `coherence-json` module's POF configuration file is now auto discoverable when using POF serialization.
  - Added support for use of Java records as portable types.
  - Added support for the use of final fields within portable types.
  - Added maven and gradle plug-ins for instrumenting Coherence `PortableTypes` to the Installer.

  - Enhanced POF deserialization error messages to show the ID of the field being deserialized.
  - Improved deserialization performance of very large byte arrays (> 100MB).
  - Fixed `QueryMap.values()` methods that accept Filter argument to eliminate eager deserialization of returned values when called against distributed cache, to bring them in-line with the existing `keySet` and `entrySet` implementations, which deserialize returned results lazily.
  - Added `PofWriter.writeByteArray` overload that takes array offset and length as arguments.
  - Added support for serializing Protobuf messages using POF.
  - Added enhancement enabling specifying an `ExternalizableLiteSerializer` for an `ExternalizableLite` class using class annotation `@ExternalizableType(serializer=ImplOfExternalizableLiteSerializer.class)`.
  - Added functionality to allow the list of included POF configuration files to be discoverable at runtime using the Java `ServiceLoader`
  - Fixed an issue that prevents `ExtractorComparator` to be used with `PofExtractor`.
  - Fixed a regression in `PortableException` when Java serialisation introduced in 12.2.1.4.5 and 14.1.1.0.1 that made it incompatible with earlier Coherence versions. Applications using POF are not affected by this change or this bug.
- **Configuration**
  - Added enhancements to configuration system property macros enabling nested definition and shell substring expansion.
  - Added cache configuration element cache-values which allows a view-scheme to be configured to cache both keys and values (the default), or only keys.
  - Allow custom namespace handlers to be used in the operational configuration file.
  - Added system property `coherence.discovery.address` for providing the discovery address.
  - Fixed an issue where a custom namespace handler cannot be used within a backing-map-scheme element.
  - Fixed an issue where XSD schema validation may fail in coherence-operational-config.xsd when using JDK22 or greater and setting `-Djdk.xml.jdkcatalog.resolve=strict`.
  - Fixed an issue where the cluster service did not honor the `coherence.service.startup.timeout` and `coherence.service.clusterservice.startup.timeout` system properties.
  - Added support for macro parameter expansion to `<cdi:bean/>` content expression.
  - Added a debug system property "`coherence.debug.operational.config`" that when set to true, causes Coherence to dump the stack of the thread that loads the Coherence operational config to standard out.
  - Added the ability to configure distributed service partition count using two system properties: `coherence.service.partitions` and `coherence.service.<distributed-service>.partitions`.
  - Fixed an issue where the `coherence.distributed.partitioncount` system property was not honored when using the default cache configuration.

- Fixed an issue that incorrectly raises an `AssertionException` when setting the `coherence.distributed.threads.min` system property when the property is applied to a service using the deprecated thread-count configuration that is configured with a smaller value.
- Fixed an issue where Coherence fails to load a cache configuration file if the `<reconnect-interval>` for a `<view-scheme>` was in form of a time unit, for example "30s".
- Fixed an issue where a `NullPointerException` could be thrown if the interface for `multicast-listener` is incorrectly specified.
- Fixed an issue where `cluster-quorum-policy` attributes could not be overridden with a system-property.
- Improved the configuration of the hostname-verifier with the introduction of an "`action`" element with permitted values of "`allow`" or "`default`"; see the XSD for a thorough description.
- Improved the description for the various managed-nodes options in a schema for coherence operational config so that it is easier for the developer to gain a quick understanding of what each mode means.
- Added system property `coherence.reflect.filter` to enable configuring allow list and/or deny list of classes accessible via reflection.
- Added security enhancement that enables configuring between secure static lambdas and more convenient to use dynamic lambdas.
- Configured XML processors to disable access to external entities to prevent XML eXternal Entity injection (XXE).
- Added support for system property `coherence.guardian.log.threaddump.interval` that can be set to a time duration to reduce the frequency that guardian service thread dumps appear in server log.
- Added service-name as an allowable child element of near-scheme.
- Fixed an issue where `coherence.mode` would be effectively ignored at the cluster level.
- Added a message to stdout which prints the Coherence logging configuration when the Coherence logger is set to level 6 or higher.
- Reduced the frequency of the messages displayed (to once every 60 seconds) when the `BinaryMemoryCalculator` cannot calculate the index size.
- Added system properties "`coherence.publisher.resend.interval`" and "`coherence.publisher.delivery.timeout`" for setting the packet delivery "`<resend-milliseconds>`" and "`<timeout-milliseconds>`" element values.
- Added system property `coherence.join.timeout` which can be used to control the cluster join timeout.
- Added system property `coherence.daemonpool.debug` (defaults to false) to control the display of log messages pertaining to dynamic thread pool sizing. Set this property to true to see these messages.

• **Topics**
  - Improved the removal of topic group subscribers created by departed cluster members, by making it more aggressive and hence speeding up reallocation of channels to the remaining subscribers.

- Fixed an issue where the last polled position for a subscriber group in a channel was not properly rolled back on subscriber fail over, causing some messages to never be received.
- Fixed an issue where increasing the channel count for a topic may cause an `UnsupportedOperationException`.
- Fixed an issue where a topic subscriber could stop receiving messages from a newly allocated channel after the previous owner of the channel departed.
- Fixed an issue where new channels are not allocated to topic subscribers if a publisher increases the channel count. This particularly applies when performing a cluster restart using active persistence.
- Fixed an issue where topic subscribers channel allocations were sometimes not cleaned up when the subscribers owning member departed from the cluster.
- Fixed a race condition which could cause a `PagedTopic` to miss messages when cancelling futures returned by a Subscriber.
- Enhanced topics to allow the channel count to be changed for an existing topic.
- Fixed an issue with where cancelling a `CompletableFuture` returned by a `NamedTopic` Subscriber stopped the subscriber from receiving more messages. Cancelled or completed futures are now handled correctly.
- Fixed `NamedTopic` methods `isDestroyed` and `isReleased` to return true when appropriate.
- Fixed an issue where topic subscribers failed to be cleaned up when their owning member departs the cluster. This is especially relevant when running Coherence on a single core.
- Fixed a race condition in the concurrent executor service `RecordingSubscriber` which can result in the `RecordingSubscriber` reporting an incorrect state.
- Fixed a potential `NullPointerException` when registering a topic MBean
- Fixed an issue where a topic subscriber could redeliver previously committed messages.
- Fixed a possible race condition when calling seek operations on a topic subscriber that has in-flight receive operations.
- Fixed an issue where topic subscribers could become disconnected and hang attempting to reconnect. The fix for this issue means that when using topics is it not possible to perform a rolling upgrade from versions prior to 14.1.1-2206-4. If a rolling upgrade is required it must be done in two stages, first to a version 14.1.1-2206-4 or higher then to the 14.1.1-2206-9. This rolling upgrade restriction only affects applications that are using topics.

- **Management/Metrics**
  - Fixed the reset statistics URL for federation in management over REST.
  - Added the ability to specify a domain name suffix for Coherence MBeans by setting the `coherence.management-config.domain-name-suffix` in the operational override configuration.
  - Added additional logging when metrics registration fails.
  - Added an error level log message for when federation is stopped due to the connect retry timeout being reached
  - Updated the Management over REST Swagger documentation to call out certain features that are only available in Grid Edition.

- Enhanced Coherence REST server logging to assist development time debugging by logging handled exceptions for a REST HTTP Response of BAD_REQUEST (status 400) at log level 6 or higher.
- Improved `ClusterNodeMBean.setLoggingLevel()` to apply the change to all supported destinations except SLF4J because it does not support the feature.
- Added a REST endpoint to the `ClusterMemberResource` to return the response of the `reportEnvironment` MBean operation of the `ClusterNodeMBean`, providing details about the Java environment and system properties.
- Fixed an issue where some cache metrics would fail to register when using Micrometer.
- Added a new report (`report-transaction.xml`) to show `TransactionManager` MBean details.
- Added `ServiceMBean` attribute `StatusHACode` for metrics support.
- Added example showing how to monitor `StatusHA` for rolling redeploys
- Added service name and nodeId to the result of the Cache MBean query. In addition, you can now get the MBean type (type), cluster name (cluster), member name (member), and cache tier (tier) attributes from the Cache MBean query.
- Added `reportEnvironment` operation to the `ClusterNode` MBean to provide details about the Java environment and system properties.
- Added an MBean operation and Management over REST endpoint to retrieve the Coherence Cluster configuration.
- Fixed an issue where a deadlock could occur when a node becomes the dynamic management senior while there are other management nodes in the cluster.
- Removed deprecated gRPC session classes from the coherence-java-client module.
- Fixed an issue where not all MBean operations honored read-only management mode.
- Fixed an issue with Management over REST and JMX where queries over all members of a large cluster may fail with an `InstanceNotFoundException` if a cluster member is shut down in middle of the query computation.
- Fixed an issue where the metric metadata value retrieved from the Coherence metrics endpoint when using Prometheus text format includes an additional space in the metric type name.
- Fixed an issue where performing a rolling upgrade would cause a `NullPointerException` and make the cache service restart.
- Fixed an issue where a rolling upgrade was not possible in some cases when using view caches due to a version compatibility issue.
- Coherence cumulative OPatch patches now supersede earlier applied Coherence cumulative patches, if any. A rollback of an earlier patch may no longer be required.
- Fixed an issue where the `forceRecovery` operation was not present in the `management-swagger.json` for Management over REST.
- Fixed an issue where the Coherence health API could report ready before all services had started.
- Fixed an issue where the NodeId column was missing from the view report.
- Added a Management over REST endpoint to retrieve the view caches in a cluster.

- Fixed an issue where the actual listen port was not being displayed for HTTP listeners which are configured to bind to port 0 (ephemeral).
- Fixed an issue where when the management senior leaves the cluster registered health checks could disappear on the new management senior.
- Fixed an issue where some reporter group files do not honor the `coherence.reporter.frequency` system property.
- Fixed an issue where a `RequestTimeoutException` may be thrown when setting an MBean attribute due to the operation using a small timeout value instead of the service request timeout.
- Fixed an issue where the `ServiceMBean.TaskMaxBacklog` might not be updated to have the maximum task backlog.
- Fixed an issue where the `ConnectionManager` MBean `resetStatistics` operation was missing from REST API and Swagger documentation.
- Corrected the descriptions of some Health Check MBean attributes.
- Fixed an issue where the `MetricsHttpProxy` service fails to restart on shutdown or unexpected restart of a Coherence member.
- Fixed an issue where the cluster service thread may be blocked on a member that is assuming the JMX cluster member role.
- Fixed an issue where the Coherence Reporter proxy reports do not account for members joining and leaving the cluster.
- Removed the shaded MVEL2 library from coherence-rest.jar and switched to using built in Coherence classes for Coherence REST query processing. MVEL can still be used for query processing if desired by adding the library (mvel2.jar) to the classpath.
- Fixed an issue where Management over REST queries could fail if non-Coherence MBeans exist with the same type field in the ObjectName, for example "`type=Service`".
- Fixed an issue where `Connector$Register` may throw a `NullPointerException` during a rolling upgrade.
- Added `ListenerKeyCount`, `ListenerFilterCount` , and `ListenerRegistrationCount` to `report-cache-storage.xml`.
- Improved the process of loading management-http-config.xml so that the file can be overridden by placing another `management-http-config.xml` file in the classpath before coherence.jar.
- Improved logging to report the location of the management-config.xml file loaded by the cluster member
- Added the `reportPartitionStats` operation to the `StorageManager` MBean to report cache partition sizes for a cache.
- Added the `ClearCount` attribute to the `StorageManager` MBean which shows how many times `clear()` has been called on a cache.
- Added `size()` operation on `StorageManager` MBean to get distributed cache total size.
- Added UNIX and Windows script `jmxserviceurl.[sh|cmd]` to print the Coherence JMX server URL to use to connect using Jconsole.
- Additional JMX attributes are now exposed as metrics on the `SimpleStrategyMBean`.

- Enhanced the Coherence `Node` and `Service` MBeans to always have reliable transport information in the `TransportStatus` attribute.
- Added `getClusterDescription`, `getServiceDescription`, and `getNodeDescription` operations to the `Cluster`, `Service`, and `ClusterNode` MBeans to retrieve details about a cluster, service, and member.
- Added `clearCache` and `truncateCache` operations to `StorageManager` MBean
- Fixed a performance regression during `StorageManager` MBean population caused by expensive collection of unique keys across index partitions, by removal of a Content attribute from a default (non-verbose) `IndexInfo` string representation.
- Fixed an issue in management over REST where an empty response could be returned instead of an expected empty collection.
- Added `CharacterLimit` attribute to the `CoherenceLoggingParamsBean` to allow user to set character limit for Coherence logging message in managed Coherence server.
- Removed the following deprecated HTTP servers supported by Coherence: `coherence-http-grizzly`, `coherence-http-jetty`, and `coherence-http-simple`.
- Added additional columns to the memory status report to show memory information in megabytes. Additionally ensured that in all reports the display of report values never uses exponential notation.
- Added integration with the Microprofile Health API so server Coherence health checks via MP Health endpoints.
- Added new attributes starting with Client to `ConnectionMBean` to identify a Connection to its client when a load balancer is between proxy and client. These attributes are mapped to metrics tags on `Coherence.Connnection.*` metric values.
- Fixed an issue where "`java.lang.IllegalArgumentException: Operation vmUnlockCommercialFeatures()` cannot be invoked" is thrown when invoking JFR related MBean operations when running with the Java Enterprise Performance Pack.
- Fixed an issue where the Cache `Units` attribute or metric could be negative for large caches when the unit factor is greater than 1.

• **Integrations**
  - Added CDI support for response caching.
  - Fixed an issue where an `IllegalStateException` could be thrown when terminating a Coherence/Spring Boot application.
  - Fixed an issue with the Bootstrap API that could prevent a Coherence Session from being found when using Spring Boot.
  - Fix a performance regression by removing unnecessary JEP-290 filter checking of the array length of a String or Binary.
  - Removed the `coherence-helidon-grpc`, `coherence-helidon-client`, and `coherence-helidon-proxy` modules. While a Coherence gRPC server and client still work in a Helidon application, it is no longer possible to automatically serve Coherence gRPC proxy endpoints on the Helidon MP gRPC server. Coherence configures and creates its own independent gRPC clients and server.
  - Integrated support for OpenTelemetry. See the documentation for further details.
  - Updated server-side JavaScript integration to work with GraalVM for JDK 21 and GraalVM Truffle 23.1.4 libraries.

  - Enabled the `com.tangosol.net.Coherence` class main method to be able to run a GAR server to be consistent with `DefaultCacheServer`.
  - Fixed an issue where a GAR would not shut down properly when using Interceptors defined on abstract classes.
  - Added defensive guards against OpenTracing tracer implementations that don't conform to the specification.
- **WebLogic Managed Coherence Servers**
  - Fixed an issue where requesting the default session using the bootstrap API inside WebLogic Managed Coherence fails to return a valid session.
  - Fixed an issue where undeploying a GAR application may hang due to the underlying cache service being unable to gracefully shut down.
  - Fixed an issue where deserialization of reflection-based extractors may be rejected when running in WebLogic.
  - Fixed an issue where auto-discovered session configurations were not started and stopped correctly when running in a GAR application using WebLogic Managed Coherence.
  - Fixed an issue where an `IllegalStateException` is thrown when a Coherence client runs inside an OSB Java callout utility.
  - Fixed an issue where the Coherence gRPC proxy is not usable in clusters running in WebLogic Managed Coherence.
- **Hot Cache**
  - Enhanced Coherence HotCache to support Oracle GoldenGate 12.3.2.1.x and 19.1.0.x.
  - Fixed HotCache process to report `CoherenceAdapter` MBean statistics when GoldenGate ggsci command stats is called.
  - Changed HotCacheAdapter to log only one warning message per missing table column on an insert event. This warning only occurs when GoldenGate Extract does not extract all table columns.
  - Upgraded Coherence HotCache to depend on Oracle GoldenGate Big Data 21.8 that supports JRE 11.
- **Miscellaneous**
  - Fixed rolling upgrade for distributed lambdas by removing production mode defaulting to static lambdas.
  - Improved Coherence to work correctly when `coherence.jar` is shaded into another jar.
  - Fixed an issue where JPMS `--add-exports java.management/sun.management=com.oracle.coherence` was required when using Berkeley Database JE database for storage.
  - Added the ability to show the Coherence version without starting a cluster via `java -jar coherence.jar --version`.
  - Improved error reporting during the cluster service halting process to help identify what caused the cluster service to halt and any issues that may have been encountered while halting.
  - Added the ability to register lifecycle listeners with Coherence instances, either via the Coherence API, or via discovery using the Java `ServiceLoader`.

- Fixed an issue with `coherence-json` module removing `jackson-annotations` as a required runtime library.
- Fixed an issue to prevent InjectorProvider from throwing a `ClassNotFound` exception if the `javax.annotation.Priority` annotation isn't available on the classpath.
- Fixed an issue where the service thread would not heartbeat when all daemon pool threads are stuck.
- Fixed an issue where gRPC did not correctly support key association.
- Fixed an issue where the Coherence default `HostnameVerifier` may erroneously reject a valid host name.
- Fixed an issue where an `IllegalAccessException` may be thrown in `DefaultMemberIdentity.makeProcessName()` with Java 17 or later.
- Fixed an issue where a race condition is possible in `SafeHashMap` on ARM processors.
- Removed the Sun Codemodel shaded dependency from coherence.jar due to its dropped support in Java 17.
- Fixed an issue where `UnsolicitedCommitEvents` may not fire for some entries which are part of a bulk update such as a `clear()`.
- Changed `async()` to throw `UnsupportedOperationException` for Extend caches. Changed `async()` for near and view caches to call `async()` on the back cache.
- Fixed an issue where `LifecycleEvent.DISPOSING` was not being emitted for the system ($SYS) `ConfigurableCacheFactory`.
- Fixed an issue where Enums were not automatically discovered when enabling type discovery.
- Fixed an issue that would lead to a leak of a view cache service if the cluster service was restarted.
- Fixed an issue where near and local caches incorrectly share the same service name.
- Fixed an issue where delayed service join may inadvertently start a `DaemonPool` even when `DaemonPool` is disabled.
- Fixed an issue where the memory used by cache backing map entries is higher than it should be.
- Fixed a performance regression introduced by the `Binary.hashCode` change.
- Updated the dependencies listed in the Coherence gRPC, Helidon, JSON, and MicroProfile POM files.
- Fixed an issue where calling `clear()` on a cache from a gRPC client removed entries using a synthetic delete instead of a real delete, and hence appears as an eviction.
- Fixed poor performance in `SafeSortedMap` methods `getEntrySet` and `getEntry` when there are a large number of entries in the map and ensured no `SafeSortedMap` method returns `SafeSortedMap.NULL` for an entry key or value.
- Fixed a potential thread deadlock where an initializing `ContinuousQueryCache` receives and attempts to process a cache truncation event.
- Fixed an issue where MessageBus would heartbeat at double the configured interval.
- Fixed an issue where a `NullPointerException` would be raised when attempting to get a session name from a remote gRPC Session.

- Fixed an issue where gRPC client connections did not fail over correctly during a rolling restart of the gRPC proxy members in the cluster.
- Fixed an issue where a `NullPointerException` may be thrown when a `ReplicatedCache` service is shutting down.
- Changed the `NamedCache.entrySet(Filter)` implementation to execute query by partition instead of by member, in order to improve parallelism and avoid exceeding the 2GB message limit when executing large queries.
- Fixed an issue where a `NearCache` using invalidation strategy present failed to release a key lock within `get/getAll`, resulting in a "Detected state corruption on KEY..." log message.
- Fixed an issue where an `UnsupportedOperationException` is thrown when accessing a read-only cache entry when sliding-expiry is enabled.
- Fixed an issue where using non-observable maps, such as `SafeHashMap`, as backing maps can result in data loss when cluster members leave.
- Fixed an issue where a Coherence `LifecycleListener` discovered using the `ServiceLoader` can be registered twice and hence receive events multiple times.
- Fixed an issue where `TcpRing.close.keys()` may throw an unhandled `ClosedSelectorException` which can cause the Cluster service to terminate unexpectedly.
- Fixed an issue where a MessageBus connection with heartbeats enabled may throw an `OutOfMemoryError` when reestablishing a dropped connection.
- Fixed an issue where the underlying exception is not properly logged when the `SimpleServiceMonitor` fails to restart services.
- Fixed an issue where the high-units setting for a transactional-scheme was being ignored.
- Fixed an issue where a `ConfigurableCacheFactorySession` would leak an event interceptor if constructed with an instance of `ExtensibleConfigurableCacheFactory`.
- Fixed an issue where a distributed service could release an unowned partition leading to `IllegalStateException`.
- Fixed an issue where `SafeSortedMap` concurrent access would result in inconsistencies under high stress situations.
- Fixed an issue where near, view, and continuous query caches may contain stale data after snapshot recovery.
- Improved cache operations to use an interruptible lock so that operations can be interrupted after the specified timeout.
- Fixed an issue where an `AssertionException` could be thrown if the partition backup count is greater than 1.
- Fixed an issue where a deadlock could occur during partition backup transfers.
- Fixed an issue where an `AssertionException` could be thrown by the `PartitionSet.intersects` method.
- Fixed an issue where the `MANIFEST.MF` file inside the Coherence Maven plugin related jars are not updated by Coherence patches.
- Fixed an issue where a heavy key listener can be overridden by a lite listener from the same member.

- Fixed an issue where a `NullPointerException` could be thrown during concurrent query with `BetweenFilter` and an entry remove operation.
- Fixed an issue where `archiveSnapshot` could throw an `ArrayIndexOutOfBoundsException` if the partition count is less than the storage enabled member size.
- Fixed an issue where a service could fail to start and join the cluster.
- Excluded maven build artifacts from distributed library jars.
- Fixed an issue where the `PrimingListener` optimization of `NearCache` in Coherence*Extend was broken.
- Fixed an issue in the Java security example where listeners were not removed when a client disconnected from proxy, causing events to be dispatched to those listeners.
- Fixed a race condition between `TransferEvent.ASSIGNED` and pre-commit `EntryEvents`. Interceptors registered for `TransferEvent.ASSIGNED` should fully complete prior to any `EntryEvents` for the same partition.
- Fixed an issue where a cluster member may run into a `StackOverflowError` and shut down when processing a malformed deeply nested filter.
- Improved `ConverterCollections.getNamedCache` to support generics and correctly implement methods `putIfAbsent`, remove, replace, and merge. The return types were widened for some of the factory methods in `ConverterCollections` to be interface and not implementation based.
- Fixed an issue that could yield a `java.io.StreamCorruptedException: invalid type: 64` during a rolling upgrade.
- Fixed a memory leak which can occur in an *Extend client when using a custom `AddressProvider`.
- Fixed an issue where the service thread could be terminated while blocked waiting for an index rebuild to finish.
- Corrected the `MANIFEST.MF` Bundle-Version in Coherence jar files.
- Fixed an issue where a cache remove operation could get lost during partition transfer.
- Fixed an issue where active recovery may commence while the service is suspended.
- Fixed an issue with binary map detection within a `ContinuousQueryCache`.
- Fixed a race condition in `ConfigurableCacheFactory` where concurrent `releaseCache()` or `destroyCache()` calls could result in an `IllegalArgumentException` being thrown.
- Fixed an issue in which different editions cannot join the same cluster.
- Fixed an issue where TcpRing may incorrectly trigger eviction of a temporarily network unreachable cluster member.
- Fixed an issue where application requests may hang if a cluster connection is handshaking concurrently with the delivery of a backlog excessive event.
- Fixed an issue where assigning partitions for a large partition count can inadvertently trigger the guardian.
- Fixed an issue with calls to `BMC.getReadOnlyEntry` not initiating a read-through when the aggregator is not limited to present entries.
- Corrected the generics type usage in the Filter signature for MapEventFilter.

- Added an option to allow inconsistent query results which can provide faster results when there are frequent concurrent entry mutations. This option can be set by adding `ALLOW_INCONSISTENCIES` to an aggregator's `characteristics()` or globally by setting the `coherence.query.retry` system property to 0 (zero).
- Fixed an issue where concurrent create and release of a cache may cause the `ScopedCacheReferenceStore` to fail to store references to the cache when it is created leading to an exception if that cache is later released.
- Fixed an issue where a query with a `StreamingAggregator` could take very long time if there are frequent concurrent entry mutations.
- Fixed an issue where a registered `MapTrigger` may not be called.
- Fixed an issue where old members may remain in the service member set which can lead to endless rejection of new members trying to join the service.
- Fixed a non-optimal deadlock avoidance rollback when using `BackingMapContext.getBackingMapEntry` to enlist entries in a different cache while concurrently executing an `invokeAll` on the other cache.
- Fixed an issue where cluster join may automatically fallback to multicast if all of the configured WKA addresses are unresolvable.