Oracle® Fusion Middleware

Installing Oracle Coherence

15c (15.1.1.0.0)
G25094-01
October 2025

Oracle Fusion Middleware Installing Oracle Coherence, 15c (15.1.1.0.0)

G25094-01

Primary Author: Oracle Corporation

Contents

3 Installing a Client Distribution

4 Installing Coherence*Web to an Application Server

5 Upgrading Coherence from Previous Releases

Preface

Installing Oracle Coherence provides instructions for installing Coherence for Java, Coherence for C++, Coherence for .NET, and Coherence*Web. The documentation also includes instructions for upgrading from previous releases and instructions for running the Coherence examples.

This preface includes the following sections:

Audience

Installing Oracle Coherence is intended for the following audiences:

- **Primary Audience** – Application developers who want to install Coherence for application development.
- **Secondary Audience** – System architects and operations personnel who want to understand how to install Coherence components.

The audience must be familiar with Java, C++, and .NET to use this guide.

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.

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*
- *Integrating Oracle Coherence*
- *Managing Oracle Coherence*
- *Securing Oracle Coherence*

- *Java API Reference for Oracle Coherence*
- *C++ API Reference for Oracle Coherence*
- *.NET API Reference for Oracle Coherence*
- *REST API Reference*
- *Release Notes for Oracle Coherence*

Conventions

The following text conventions are used in this document:

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

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.

1

Planning Your Oracle Coherence Installation

This guide will help you install Oracle Coherence. Various topics are covered that should be reviewed thoroughly to help ensure that you do not encounter any problems either during or after the Oracle Coherence installation.
To install standalone Oracle Coherence, there is no prerequisite for Oracle Fusion Middleware Infrastructure. If you do have the Infrastructure on your system, then Oracle Coherence can be integrated with it in a number of ways. For the purposes of this guide, only the standalone mode is considered.

> **Note**
>
> Oracle Coherence can also be installed as part of an Oracle WebLogic Server installation. Installing and configuring Oracle Coherence with WebLogic Server is beyond the scope of this documentation. See Planning the Oracle WebLogic Server Installation in *Installing and Configuring Oracle WebLogic Server and Coherence*.

This chapter contains the following sections:

About Oracle Coherence

Oracle Coherence in-memory data grid is a key component of Oracle's Enterprise Cloud Native Java portfolio. Oracle Coherence predictably scales applications to meet mobile and cloud demands on shared services and infrastructure.

- Provides real-time application processing using parallel query, live event processing, map-reduce aggregation, and parallel transaction processing
- Scales applications linearly and dynamically for predictable cost and reliable delivery of real-time customer experiences
- Enables continuous data availability and transactional integrity across the most demanding multi-data center deployments
- Oracle Coherence's GoldenGate HotCache enables businesses to leverage real-time cache updates to provide always-accurate application information
- Provides operational simplicity through advanced integration with Oracle WebLogic Server, across conventional and cloud environments, and Oracle Exalogic Elastic Cloud

Introducing the Oracle Coherence Standard Installation Topologies

Using Oracle Coherence software together with other application software, you can create a variety of production topologies to suit the needs of your applications, your organization, and your application users.

As a result, it is difficult to provide exact instructions for every possible Oracle Coherence installation. This documentation provides detailed instructions for installing Oracle Coherence only in standalone mode.

For more information about standard installation topologies, see Understanding the Standard Installation Topology in *Planning an Installation of Oracle Fusion Middleware*.

Roadmap for Installing and Configuring Standalone Oracle Coherence

Review the steps that are required to install and Oracle Coherence. Table 1-1 shows the steps required to install and configure standalone Oracle Coherence.

Table 1-1 Roadmap for Standalone Oracle Coherence Installation

Task	Description	For More Information
Verify your system environment	Before beginning the installation, verify that the minimum system and network requirements are met.	Roadmap for Verifying Your System Environment and System Requirements
Obtain the appropriate distribution	To install Oracle Coherence, obtain the distribution.	Understanding and Obtaining the Oracle Coherence Distribution
Determine your installation directories	Verify that the directory into which you want to install Oracle Coherence is accessible by the installer, and exists on systems that meet the minimum requirements.	Understanding the Oracle Coherence Directory Structure
Install Oracle Coherence	Run the installation program to install the software. This transfers the software to your system.	Performing a Coherence Installation
Post-configuration administration and configuration tasks	Discover additional tools and resources to configure and administer Oracle Coherence.	Installing a Client Distribution, Installing Coherence*Web to an Application Server
Upgrade tasks	If you are already working with Coherence, upgrade your applications to use the current release.	Upgrading Coherence from Previous Releases
Run Coherence Examples	The Coherence distribution includes a collection of examples that exercise many Coherence features.	Running the Coherence Examples

Roadmap for Verifying Your System Environment

Oracle Fusion Middleware products are certified to run in different system environments. Table 1-2 identifies important tasks and checks that you must perform to make sure that your environment is properly prepared for installing and configuring Oracle Coherence.

Table 1-2 Roadmap for Verifying Your System Environment

Task	Description	For More Information, See
Verify certification and system requirements.	Verify that your operating system is certified and properly configured for Oracle Fusion Middleware Infrastructure installation and configuration.	Verifying Certification and System Requirements in *Planning an Installation of Oracle Fusion Middleware*.

Table 1-2 (Cont.) Roadmap for Verifying Your System Environment

Task	Description	For More Information, See
Prepare your system for installation.	Verify that the necessary environment variables are set, and you have identified a proper installation user.	Preparing for an Oracle Fusion Middleware Installation in *Planning an Installation of Oracle Fusion Middleware*.

Understanding and Obtaining the Oracle Coherence Distribution

The Oracle Coherence distribution is available as a standalone executable installation program. To obtain the distribution, see Obtaining Product Distributions in *Planning an Installation of Oracle Fusion Middleware*.

2

Installing Oracle Coherence for Java

Several installers are available for installing Oracle Coherence for Java (simply referred to as Coherence). The installers are delivered as executable JAR files and facilitate the installation process. After you have installed Coherence, run the quick example to verify that Coherence is successfully installed.

> **Note**
>
> For instructions about installing a Coherence*Extend client distribution, see Installing a Client Distribution. For instructions about installing Coherence*Web to an application server, see Installing Coherence*Web to an Application Server .

This chapter includes the following topics:

System Requirements

Coherence has different requirements for installation and runtime.

Runtime Requirements

The following are the suggested minimum system requirements for running Coherence in a development environment:

- 100 MB disk space for complete installation (includes API documentation)
- 1 GB of RAM (assuming a maximum Java heap size of 512MB) – This amount of RAM can ideally support a maximum cache size of 150MB on a single node that is configured to store a backup of all data (150MB x 2) and leaves more than a 1/3 of the heap available for scratch and JVM tasks. This recommendation is considered a basic starting point and should not be considered a rule. See JVM Tuning in *Administering Oracle Coherence*.
- JVM (JRE or JDK) 17 or 21. A JDK is often used during development and offers tools for monitoring and troubleshooting Java applications, but a JDK is not required to run Coherence.

  > **Note**
  >
  > Customers that want to integrate with applications that are running older JVM versions can use older Coherence clients; however, the client is constrained to the platform and client features that are supported for that Coherence version.

- Windows or UNIX-based system that supports the required Java Version
- Network adapter

Installation Requirements

The following are the minimum requirements for using the Coherence installer:

> **Note**
>
> The requirements for running the installer are not the same as the requirements for running Coherence.

- 300 MHz CPU
- 512 MB swap space
- 256 color monitor (required for GUI-based installation only)
- Java Development Kit (JDK) version 17 or 21.

Performing a Coherence Installation

Coherence is installed using the Oracle Universal Installer. The installer provides both installation and patching services for Oracle products. The following installers are available for Coherence and detailed in this section.

- `fmw_`*`version`*`_coherence.jar` – A full Coherence installation that can be run in either graphical mode or silent mode. See Performing a Coherence Installation In Graphical Mode and Performing a Coherence Installation In Silent Mode.
- `fmw_`*`version`*`_coherence_quick.jar` – A minimum Coherence installation that is always run in silent mode. The quick installer provides a smaller footprint and does not include API documentation. See Running the Coherence Quick Installer.
- `fmw_`*`version`*`_coherence_quick_supplemental.jar` – A supplemental installation that is always run in silent mode. The supplemental installer contains only API documentation. See Running the Coherence Supplemental Installer.
- `fmw_`*`version`*`_wls.jar` – A full WebLogic Server installation that includes Coherence. See Installing Coherence with WebLogic Server.

Coherence is always installed to an *`ORACLE_HOME`*`/coherence` directory. The complete path to the `coherence` directory is referred to as *`COHERENCE_HOME`* throughout the Coherence documentation.

This section includes the following topics:

Performing a Coherence Installation In Graphical Mode

The Coherence installer is distributed as an executable Java ARchive (JAR) file called `fmw_`*`version`*`_coherence.jar`. Use the `java` command to run the installer on the target computer. For detailed help on the installer's options, use the `-help` argument when running the installer.

For information about the directories created by the installer, see Understanding the Oracle Coherence Directory Structure.

This section includes the following topics:

Starting the Installation Program

To perform a Coherence installation in graphical mode:

1. Copy the `fmw_version_coherence.jar` file to the target computer.
2. From a command prompt, change directories to the location of the `coherence_version.jar` file and execute the following command (assuming that `JAVA_HOME/bin` is located on the computer's `PATH`):

   ```
   java -jar fmw_version_coherence.jar
   ```

   Note: If you are installing with JDK 21, and you receive the following error, click **Next** to continue. This is a known issue and can be ignored.

   ```
   Checking Java version used to launch the installer
   Problem: This JDK version was not certified at the time it was made
   generally available.
   It may have been certified following general availability.
   Recommendation: Check the Supported System Configurations Guide (http://
   www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html) for
   further details.
   Press "Next" if you wish to continue. Expected result: 17.0.8
   Actual result: 21.0.2
   ```

The installation program displays.

Navigating the Installation Screens

Table 2-1 lists the screens in the order that the installation program displays.

If you need additional help with any of the installation screens, click the screen name.

Table 2-1 Oracle Coherence Installation Screens

Screen	Description
Inventory Setup	On UNIX operating systems, this screen will appear if this is the first time you are installing any Oracle product on this host. Specify the location where you want to create your central inventory. Make sure that the operating system group name selected on this screen has write permissions to the central inventory location. For more information about the central inventory, see About the Oracle Central Inventory in *Installing Software with the Oracle Universal Installer*. This screen will not appear on Windows operating systems.
Welcome	This screen introduces you to the product installer.
Installation Location	Use the drop-down list to select an existing *ORACLE_HOME* directory to which Coherence will be installed, or enter an absolute path to create a new Coherence *ORACLE_HOME* directory. Click **Browse** to search for a directory if required. The directory cannot contain an existing Coherence installation.
Installation Type	Select which Coherence options to install.
Prerequisite Checks	This screen displays a list of system checks that are performed to ensure that Coherence is certified on the system.

Table 2-1 (Cont.) Oracle Coherence Installation Screens

Screen	Description
Installation Summary	Verify the installation. Click **Save Response File** if you intend to duplicate this installation on additional computers. A response file is created that can be used to perform a silent install with the exact same installation settings. See Performing a Coherence Installation In Silent Mode.
Installation Progress	This screen allows you to see the progress of the installation.
Installation Complete	This screen appears when the installation is complete. Review the information on this screen, then click **Finish** to dismiss the installer.

Performing a Coherence Installation In Silent Mode

Silent mode allows Coherence to be installed without using a graphical interface and is ideal for remote installations or when incorporating the installation as part of a script. Silent mode typically uses a response file (`.rsp`) that contains the installation parameters as `name=value` pairs. Create a response file by running the installer in graphical mode and then saving the installation parameters to a response file at the Installation Summary screen. Use the saved file to replicate the installation on other computers or modify the file to change the installation as required.

To perform a Coherence installation in silent mode:

1. Copy the `fmw_`*`version`*`_coherence.jar` file and a response file to the target computer.
2. From a command prompt, change directories to the location of the `coherence_`*`version`*`.jar` file and execute the following command (assuming that *`JAVA_HOME`*`/bin` is located on the computer's `PATH`):

   ```
   java -jar fmw_version_coherence.jar -silent -responseFile full_path_to_response_file
   -waitForCompletion
   ```

   On UNIX-based platforms, the installer requires the location of the `oraInst.loc` inventory directory pointer file if it is not found in the default location (`/etc`). If this is the first time that an Oracle product has been installed on this computer, you can use the `createCentralInventory.sh` script to set up an inventory directory pointer file in the `/etc` directory. The script requires root permissions.

   If you want to use a custom location for the `oraInst.loc` file, use the `-invPtrLoc` installer option to specify the location. For example:

   ```
   java -jar fmw_version_coherence.jar -silent -responseFile full_path_to_response_file
   -waitForCompletion -invPtrLoc /MyDirectory/oraInst.loc
   ```

   The contents of the `oraInst.loc` file contains the location and the ownership group for the inventory directory. For example:

   ```
   inventory_loc=/MyDirectory/oraInventory
   inst_group=group
   ```

Running the Coherence Quick Installer

The quick install is distributed as an executable JAR file called `fmw_`*`version`*`_coherence_quick.jar`. Use the `java` command to run the installer on the target

computer. For detailed help on the installer's options, use the `-help` argument when running the installer.

The quick install performs a silent install with no options. The distribution includes less lifecycle tools but does register the Coherence components as part of the Oracle inventory, which allows future lifecycle operations to work. In addition, the installation does not include API documentation. The result is a faster installation process and a smaller installation footprint than the regular Coherence installer and is an ideal method for installing Coherence as part of a script without user interaction.

To perform a Coherence quick installation:

1. Copy the `fmw_`*`version`*`_coherence_quick.jar` file to a directory on the target computer.
2. From a command prompt, change directories to the location of the `fmw_`*`version`*`_coherence_quick.jar` file and execute the following command (assuming that *`JAVA_HOME`*`/bin` is located on the computer's `PATH`):

   Note: If you are installing with JDK 21, and you receive the following error, click **Next** to continue. This is a known issue and can be ignored.

   ```
   Checking Java version used to launch the installer
   Problem: This JDK version was not certified at the time it was made
   generally available.
   It may have been certified following general availability.
   Recommendation: Check the Supported System Configurations Guide (http://
   www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html) for
   further details.
   Press "Next" if you wish to continue. Expected result: 17.0.8
   Actual result: 21.0.2
   ```

   ```
   java -jar fmw_version_coherence_quick.jar ORACLE_HOME=/oracle
   ```

   The value of the `ORACLE_HOME` variable specifies the *`ORACLE_HOME`* directory to which Coherence will be installed. The value must be an absolute path. If the directory already exists, it must be empty or it must be an existing valid *`ORACLE_HOME`*. The directory cannot contain an existing Coherence installation. If the directory does not exist, the installer creates the directory. You can also start the installation from an empty current working directory and omit the `ORACLE_HOME` variable; the current working directory becomes the *`ORACLE_HOME`* directory. For example:

   ```
   cd /oracle
   java -jar /tmp/fmw_version_coherence_quick.jar
   ```

   On UNIX-based platforms, the quick installer attempts to find the `oraInst.loc` inventory directory pointer file in the `/etc` directory. If the file is not found, the `/tmp` directory is used as the inventory directory. If this is the first time that an Oracle product has been installed on this computer, you can use the `createCentralInventory.sh` script to set up an inventory directory pointer file in the `/etc` directory. The script requires root permissions.

   If you want to use a custom location for the `oraInst.loc` file, use the `-invPtrLoc` installer option to specify the location. For example:

   ```
   java -jar fmw_version_coherence_quick.jar -invPtrLoc /MyDirectory/oraInst.loc
   ```

   The contents of the `oraInst.loc` file contains the location and the ownership group for the inventory directory. For example:

```
inventory_loc=/MyDirectory/oraInventory
inst_group=group
```

Running the Coherence Supplemental Installer

The supplemental install is distributed as an executable JAR file called `fmw_version_coherence_quick_supplemental.jar`. The distribution is used to install the API documentation to an existing Coherence installation. The supplemental installer performs a silent install with no options. It is typically used together with the quick installer to perform an installation as part of a script without user interaction. If you do not require the API documentation, then you can skip the supplemental installation.

1. Copy the `fmw_version_coherence_quick_supplemental.jar` file to the *ORACLE_HOME* directory where Coherence is installed.
2. From a command prompt, change directories to the location of the `fmw_version_coherence_quick_supplemental.jar` file and execute the following command (assuming that *JAVA_HOME*`/bin` is located on the computer's `PATH`):

   ```
   java -jar fmw_version_coherence_quick_supplemental.jar
   ```

   The installation starts and status messages are emitted.

Installing Coherence with WebLogic Server

The WebLogic Server installer includes the Coherence distribution and installs Coherence in the same *ORACLE_HOME* directory as WebLogic Server. WebLogic Server includes a Coherence integration that standardizes how Coherence is managed and deployed within a WebLogic Server domain. The integration makes Coherence a subsystem of WebLogic Server and allows Coherence environments to be administered using WebLogic Server tools and infrastructure, such as Jakarta EE-styled packaging and deployment, remote server management, server clusters, WebLogic Scripting Tool (WLST) automation, and configuration through the WebLogic Remote Console. For details about installing Coherence with WebLogic Server, see Planning the Oracle WebLogic Server Installation in *Installing and Configuring Oracle WebLogic Server and Coherence*.

Setting Environment Variables

You can set the `JAVA_HOME` and `COHERENCE_HOME` environment variables. However, they are not required to run Coherence.

- `JAVA_HOME` – This variable is used when running the scripts that are included in the `COHERENCE_HOME/bin` directory. The value of this variable is the full path to the Java installation directory. If `JAVA_HOME` is not set, the scripts use the computer's default Java installation. Set this variable to ensure that the scripts use a specific Java version.
- `COHERENCE_HOME` – This variable is typically set as a convenience. The value of this variable is the full path to the `ORACLE_HOME/coherence` directory.

Running Coherence for the First Time

The `COHERENCE_HOME/bin` directory includes scripts that are used during development and testing and are provided as a design-time convenience. The `cache-server` script starts a cache server using a default configuration. The `coherence` script starts a cache factory instance using a default configuration. The cache factory instance includes a command-line tool that is used to (among other things) create and interact with a cache.

In this scenario, a basic cluster is created and then the command-line tool is used to create and interact with a cache that is hosted in the cluster.

This section includes the following topics:

Create a Basic Cluster

In this step, a basic cluster is created that contains three separate Java processes: a cache server and two simple CLI instances. For simplicity, the three processes are collocated on a single computer. The cache server, by default, is configured to store data. The two cache factory instances, by default, are configured not to store data. As each process is started, they automatically join and become cluster members (also referred to as cluster nodes).

For this example, the Coherence out-of-box default configuration is slightly modified to create a unique cluster which ensures that these cluster members do not attempt to join an existing Coherence cluster that may be running on the network.

> **Note**
>
> The Coherence default behavior is to use multicast to find cluster members. Coherence can be configured to use unicast if a network does not allow the use of multicast. See Using Well Known Addresses in *Developing Applications with Oracle Coherence*.

To create a basic cluster:

1. Using a text editor, open the *COHERENCE_HOME*`/bin/cache-server` script.
2. Modify the `java_opts` variable to include the `coherence.cluster` system properties as follows:

   ```
   set java_opts="-Xms%memory% -Xmx%memory% -Dcoherence.cluster=cluster_name"
   ```

   Replace *`cluster_name`* with a value that is unique for this cluster. For example, use your name for the cluster name.
3. Save and close the `cache-server` script.
4. Repeat steps 1 to 3 for the *COHERENCE_HOME*`/bin/coherence` script and specify the same value for *`cluster_name`*.
5. Run the `cache-server` script. The cache server starts and output is emitted that provides information about this cluster member.
6. Run 2 instances of the `coherence` script. As each instance is started, output is emitted that provides information about the respective cluster members. Each instance returns a command prompt for the command-line tool.

Create a Cache

In this step, a cache is created and hosted on the basic cluster. A simple string is entered into the cache using the command-line tool of the first cache factory instance. The string is then retrieved from the cache using the command-line tool of the second cache factory instance. The example is simplistic and not very practical, but it does quickly demonstrate the distributed nature of Coherence caches. Moreover, these steps are typically performed directly using the Coherence API.

To create a cache:

1. At the command prompt for either cache factory instance, create a cache named `Test` using the `cache` command:

   ```
   cache Test
   ```

2. At the command prompt, use the `put` command to place a simple string in the new cache by entering a key/value pair (separated by a space):

   ```
   put key1 Hello
   ```

   The command returns and displays `null`. The `put` command always returns the previous value for a given key. The `null` value is returned because this is the first value entered for this key.

3. Switch to the other cache factory instance and from the command prompt create the `Test` cache using the `cache` command:

   ```
   cache Test
   ```

4. From this command prompt, retrieve the string in the cache using the `get` command and entering the key name:

   ```
   get key1
   ```

   The command returns and displays `hello`. Either cache factory process can add or remove cache entries because the processes are part of the same cluster and because the `Test` cache is known to all cluster members. In addition, since the cache server is storing the cache data, either cache factory process (or both) can be shutdown and the cache data persists.

Integration with Maven

Software projects that use Maven can incorporate Coherence into their build process. Maven is a build and dependency system that allows the configuration of project dependencies, 3rd party dependencies and definition of a build lifecycle. Software projects often use Maven to simplify and standardize their build process. If you are new to Maven, see the Maven project page.
Oracle Middleware provides a plug-in that synchronizes an Oracle home directory with a Maven repository and standardizes Maven usage and naming conventions. The plug-in allows Coherence artifacts to be uploaded to a Maven repository, which simplifies how the artifacts are consumed in development projects. See Installing and Configuring Maven for Build Automation and Dependency Management in *Developing Applications Using Continuous Integration*.

In addition, the Maven integration includes an archetype and packaging plug-in for a Coherence Grid Archive (GAR). A Coherence GAR is a module type that is typically used to deploy Coherence applications within a WLS domain. The Maven archetype plug-in generates a GAR structure and provides example configuration files. The packaging plug-in generates a GAR based on a project's contents and dependencies and ensures that the dependencies, source, and configuration files are copied into the GAR.

The Maven plug-in and configuration files for Coherence are located in the *`COHERENCE_HOME`*`/plugins` directory. The Maven GAR plug-in and archetype are installed in the enterprise repository as part of the synchronization plug-in. See Building Oracle Coherence Projects with Maven in *Developing Applications Using Continuous Integration*.

> **Note**
>
> In this guide, the example Maven dependencies are displayed in the following format:
>
> ```
> <dependency>
>     <groupId>${coherence.groupId}</groupId>
>     <artifactId>${coherence.artifactId}</artifactId>
>     <version>${coherence.version}</version>
> </dependency>
> ```
>
> For these examples, you should define the following properties in your `pom.xml` file, to customize any dependencies for the version of Coherence you are running. For example:
>
> **Table 2-2 Properties to Define in the `pom.xml` File**
>
Property	Grid/Enterprise Edition
`coherence.groupId`	`com.oracle.coherence`
`coherence.artifactId`	`coherence` or other
`coherence.version`	`14.1.2-0-0` or later
>
> For Grid and Enterprise editions, you should follow the instructions in this section to install Coherence into the Maven repository.

Installing a Coherence Patch

Coherence periodically releases patches to the Oracle Support Website. See My Oracle Support. Patches are installed using the standard Oracle patching mechanism. See Patching Your Environment Using OPatch in *Patching with OPatch*.

Uninstalling Coherence

Coherence is uninstalled by using the Oracle Fusion Middleware deinstaller. The deinstaller allows you to select which components in a Coherence `ORACLE_HOME` directory to uninstall and can also be used to completely remove a Coherence `ORACLE_HOME` directory.
To uninstall Coherence using the Deinstallation wizard, start the deinstaller. Use either the Coherence `ORACLE_HOME/oui/bin/deinstall.sh` script on UNIX-based platforms or the Coherence `ORACLE_HOME\oui\bin\deinstall.cmd` script on Windows. A shortcut to the script is available on Windows and is located in the **Oracle** program group on the start menu.

> **Note**
>
> If Coherence is installed as part of a WebLogic Server installation, it is not possible to uninstall Coherence separately from WebLogic Server.

Table 2-3 lists the screens in the order that the Deinstallation program displays.

Table 2-3 Oracle Coherence Deinstallation Screens

Screen	Description
Welcome	This screen introduces you to the product deinstaller.
Deinstallation Summary	This screen lists the features that will be deinstalled. Click **Deinstall** to proceed.
Deinstallation Progress	This screen displays and shows all tasks that have succeeded and failed.
Deinstallation Complete	This screen displays and shows a summary of the Deinstallation process. Click **Finish** to close the Deinstallation program.

> **Note**
>
> Additional files in the `ORACLE_HOME` directory must be manually deleted. On Windows, you must also manually delete the **Oracle** program group on the Start menu.

3

Installing a Client Distribution

Coherence provides C++ and .NET client distributions that can be installed as required. There is no separate Java client distribution. Java extend clients are created using Coherence for Java. In addition, the Coherence cluster is implemented in Java. Therefore, Coherence for Java must be installed to use any client distribution.
This chapter includes the following sections:

Installing Coherence for Java

The Coherence for Java distribution is used to build and use Java-based extend clients. To install Coherence for Java, see Installing Oracle Coherence for Java.

Installing the C++ Client Distribution

The Oracle Coherence for C++ distribution is used to develop and run C++ extend clients. The latest version of the distribution can be downloaded at Oracle Coherence Software Downloads. This section contains the following topics:

Supported Environments for Coherence C++ Client

Table 3-1 lists the supported platforms and operating systems for Coherence for C++:

Table 3-1 Platform and Operating System Support for Coherence for C++

Operating System	Compiler	Architecture
Microsoft Windows Server: 2022+ Client: Windows 11+	Visual Studio 2015, 2017, and 2019	x86, x64
Linux	GCC 4.8.5+, GNU libc 2.1.7+	x86, x64, aarch64
Apple macOS 14+[1]	Xcode 15+ (GCC)	aarch64

[1] When building C++ applications with Apple macOS, you must compile with the command "g++" (as opposed to "CC").

Microsoft-Specific Requirements

When deploying on Microsoft Windows, just as with any Visual Studio based application, the corresponding Visual Studio runtime library package must be installed on the deployment computer.

Extracting the Coherence for C++ Distribution

Coherence for C++ is distributed as a ZIP file. Use a ZIP utility or the `unzip` command-line utility to extract the ZIP file to a location on the target computer. The extracted files are organized within a single directory called `coherence-cpp`.

The following example uses the `unzip` utility to extract the distribution to the `/opt` directory which is the suggested installation directory on UNIX-based operating systems. Use the ZIP utility provided with the target operating system if the `unzip` utility is not available.

```
unzip /path_to_zip/coherence-cpp-version_number-platform-architecture-compiler.zip -
d /opt
```

The following example extracts the distribution using the `unzip` utility to the `C:\` directory on the Windows operating system.

```
unzip C:\path_to_zip\coherence-cpp-version_number-platform-architecture-compiler.zip -d
C:\
```

The following list describes the directories that are included in installation directory:

- `bin` – This directory includes `sanka.exe`, which is an application launcher that is used to invoke executable classes embedded within a shared library.
- `doc` – This directory contains Coherence for C++ documentation including the API documentation
- `include` – This directory contains header files that use the Coherence API and must be compiled with an application.
- `lib` – This directory includes the Coherence for C++ library. The `coherence.dll` file is the main development and run-time library and is discussed in detail throughout this documentation.

Installing the .NET Client Distribution

The Oracle Coherence for .NET distribution is used to develop and use .NET extend clients. It is published as a nuget package. The package name is `Coherence` and it can be downloaded from nuget.org.

> **Note**
>
> As of 14.1.2.0.0, the Coherence .NET Client supports .NET 6.0 and later.
>
> For earlier .NET versions, use Coherence 14.1.1.x. In 14.1.1.x, the package name for the .NET Core compatible implementation is `Coherence.Core`, while the package name for the .NET Framework 4.x compatible implementation is `Coherence`. For more information on the 14.1.1.x .NET Client, see Installing the .NET Client Distribution in *Installing Oracle Coherence (14.1.1.0)*.

This section contains the following topics:

Prerequisites

The following are required to use Coherence for .NET:

- Microsoft .NET 6.0 or higher runtime and SDK
- Supported Microsoft Windows operating system

Installing the Coherence .NET Client

Coherence for .NET is distributed as a nuget package and is published to nuget.org. You can install it from either Microsoft Visual Studio or the .NET command-line interface (.NET CLI).

- To install Coherence for .NET using Visual Studio, follow these steps:
  1. In Visual Studio, select **Project**, then **Manage NuGet Packages**.
  2. On the **NuGet Package Manager** page, choose nuget.org as the Package source.
  3. From the **Browse** tab, find Coherence in the list, and then select **Install**.
- To install Coherence for .NET using the .NET CLI tool, follow these steps:
  1. Open a command line interface and change to the directory that contains your project file.
  2. Enter the following command to install the Coherence NuGet package: `dotnet add package Coherence`

     The `dotnet add package` command adds a package reference for Coherence to the project file, and then runs `dotnet restore` to install the package. After the command finishes, you can open the `.csproj` project file to see that the package reference was added.

     ```
     <ItemGroup>
        <PackageReference Include="Coherence" Version="14.1.2.0" />
     </ItemGroup>
     ```

     By default, `dotnet add package` installs the latest version of the package. If you want to install a different version, add the `-v` or `--version` switches. For example, `dotnet add package Coherence.Core --version 14.1.1.13`

> **Note**
>
> In 14.1.1.x, `Coherence.Core` is the package name for the .NET Core compatible implementation while `Coherence` is the package name for the .NET Framework 4.x compatible implementation.

Coherence .NET Version Number Mapping

A Coherence assembly uses a custom version number mapping. Oracle version numbers use 5 digits (*N.N.N.N.N*), but .NET version numbers can only have up to 4 digits (*N.N.N.N*). To support the .NET version convention, the 4th and 5th Oracle digits are combined for the 4th .NET version digit.

The following calculation is used to create the 4th .NET version digit:

`4th .NET digit = 4th Oracle digit * 1000 + 5th Oracle digit`

The following calculations are used to convert the 4th .NET version digit to the 4th and 5th Oracle version digits:

`4th Oracle digit = int(4th .NET digit / 1000)`

`5th Oracle digit = 4th .NET digit - (4th Oracle digit * 1000)`

For example:

.NET Version Number	Oracle Version Number
14.1.2.0	14.1.2.0.0
14.1.2.1	14.1.2.0.1
14.1.2.1000	14.1.2.1.0
14.1.2.1001	14.1.2.1.1
14.1.2.2010	14.1.2.2.10
14.1.2.10010	14.1.2.10.10

> **Note**
>
> For logging, the .NET 4th digit is converted to the Oracle 4th and 5th digits so that logging messages appear the same as Java and C++ log messages.

Deploying Coherence for .NET

Coherence for .NET requires no specialized deployment configuration. Simply add a reference to the Coherence nuget package as described in Installing the Coherence .NET Client.

Compatibility Between Coherence*Extend Versions

The extend protocol supports both forward and backwards version compatibility between extend clients and cluster proxies.

In other words, cluster proxies support both older and newer version extend clients, and extend clients support both older and newer version cluster proxies. Compatibility for the extend protocol and POF is maintained between the second digit of major releases (for example, 14.1, 14.2, and so on) but may not be maintained between the first digit of major releases (for example, 14.x, 15.x, and so on).

> **Note**
>
> Compatibility requires that the serializers in the different Coherence*Extend versions be compatible. For non-Java clients, compatibility requires the use of POF. For Java clients that use `java.io.Serializable` for serialization, the major version of Java Standard Edition used by the client must be the same as, or within one major version of, that used by the cluster.

Backward compatibility to cluster proxies is intended as an upgrade convenience and not as a long term solution. It allows extend clients to upgrade to a new version before the proxy server and cluster. However, a cluster should always be upgraded to the latest version as a best practice. When an extend client and the server it connects to are on different versions, the extend client is limited to the functionality of the older of the two releases or patch set versions.

Coherence Backward Compatibility Exception for 14.1.2.0.0

In general, compatibility for the extend protocol and POF is maintained between the second digit of major Coherence releases (for example, version 14.1.x would be compatible with

version 14.2.x, and so on) but may not be maintained between the first digit of major releases (for example, version 14.x may not be compatible with version 15.x, and so on).

By exception to the extend compatibility policy, Coherence 14.1.2.0.0 extend protocol and POF support spans a broader range of versions. In addition to the standard extend version compatibility support, Coherence 14.1.2.0.0 proxy servers are also compatible with supported 12.x versions (for example, 12.2.1.4.x). And Coherence 14.1.2.0.0 extend clients are also compatible with Coherence 12.2.1.4.x proxy servers.

This exception to the extend compatibility policy may not be included in future Coherence releases. That is, future releases of Coherence proxy servers and extend clients that are a higher version than 14.1.2.0.x may not include this additional support for Coherence 12.x versions.

4

Installing Coherence*Web to an Application Server

Coherence*Web is an HTTP session management module dedicated to managing session state in clustered environments. Built on top of Oracle Coherence, Coherence*Web brings Coherence data grid's data scalability, availability, reliability, and performance to in-memory session management and storage.
Coherence*Web can be deployed to many mainstream application servers such as Oracle WebLogic Server, IBM WebSphere, and Tomcat. For a complete list of supported application servers, see Supported Web Containers in *Administering HTTP Session Management with Oracle Coherence*Web*.

This chapter includes the following sections:

Installing Coherence*Web with WebLogic Server

All of the files which support Coherence*Web are included with the Coherence distribution. If you are using WebLogic Server, then you can install WebLogic Server and Coherence simultaneously. See Planning the Oracle WebLogic Server Installation in *Installing and Configuring Oracle WebLogic Server and Coherence*.
Once you have installed WebLogic Server and Coherence, you can integrate your applications with Coherence*Web without any further configuration. See Using Coherence*Web with WebLogic Server in *Administering HTTP Session Management with Oracle Coherence*Web*.

Installing Coherence*Web with Other Application Servers

Coherence*Web is supported on different application servers, such as IBM WebSphere or Tomcat. The Coherence*Web files are installed as part of the Coherence distribution. However, you must then complete post-installation steps to integrate Coherence*Web with your applications. See Using Coherence*Web on Other Application Servers in *Administering HTTP Session Management with Oracle Coherence*Web*.

5

Upgrading Coherence from Previous Releases

Coherence applications can be upgraded to new Coherence versions to take advantage of new and improved features. The most common upgrading steps are provided and should be followed as required for your application.
This chapter includes the following sections:

General Upgrade Guidelines

Understanding and following some basic guidelines before you upgrade to a new Coherence release can ensure a successful upgrade.

General Instructions:

- Read the Release Notes carefully for any changes to features you may be using.
- Pay particular attention to changes in default behavior.
- Plan a period of QA and Performance testing as subtle changes may impact customer SLA.
- Plan for upgrades to the JVM, if required by the Coherence upgrade.
- Check compatibilities with any external systems.
- Do not combine changes in environment, network, external systems with the planned upgrade (or treat it as a new product release).

Jakarta EE 9.1 Compatibility

Coherence 15.1.1.0.0 has migrated to Jakarta EE 9.1 from Jakarta EE 8, importing types in `jakarta` packages instead of `javax` packages.

The following table describes the mapping of `javax` packages to `jakarta` packages and Maven artifacts used by Coherence 15.1.1.0.0.

Table 5-1 Mapping `javax` packages to `jakarta` packages and Maven artifacts

`javax` Package	`jakarta` Package	Maven Group ID	Maven Artifact ID
`javax.activation`	`jakarta.activation`	`jakarta.activation`	`jakarta.activation-api`
`javax.annotation`	`jakarta.annotation`	`jakarta.annotation`	`jakarta.annotation-api`
`javax.enterprise`	`jakarta.enterprise`	`jakarta.enterprise`	`jakarta.enterprise.cdi-api`
`javax.inject`	`jakarta.inject`	`jakarta.inject`	`jakarta.inject-api`
`javax.interceptor`	`jakarta.interceptor`	`jakarta.interceptor`	`jakarta.interceptor-api`
`javax.json`	`jakarta.json`	`jakarta.json`	`jakarta.json-api`

Table 5-1 (Cont.) Mapping `javax` packages to `jakarta` packages and Maven artifacts

`javax` Package	`jakarta` Package	Maven Group ID	Maven Artifact ID
`javax.json.bind`	`jakarta.json.bind`	`jakarta.json.bind`	`jakarta.json.bind-api`
`javax.persistence`	`jakarta.persistence`	`jakarta.persistence`	`jakarta.persistence-api`
`javax.resource`	`jakarta.resource`	`jakarta.resource`	`jakarta.resource-api`
`javax.servlet`	`jakarta.servlet`	`jakarta.servlet`	`jakarta.servlet-api`
`javax.servlet.jsp`	`jakarta.servlet.jsp`	`jakarta.servlet.jsp`	`jakarta.servlet.jsp-api`
`javax.ws.rs`	`jakarta.ws.rs`	`jakarta.ws.rs`	`jakarta.ws.rs-api`
`javax.xml.bind`	`jakarta.xml.bind`	`jakarta.xml.bind`	`jakarta.xml.bind-api`

> **Note**
>
> Typically, the migration from Jakarta EE 8 to Jakarta EE 9.1 is simply changing EE type references from `javax` to `jakarta`. However, prior to upgrading, we recommend that you review the specifications that you use in your projects.

> **Note**
>
> Coherence*Web supports Jakarta EE 9.1, 10, and 11.

Coherence examples have been updated to use `jakarta` packages where relevant. These examples still work for older versions of Coherence; in such cases, you will need to change from `jakarta` to `javax`.

In addition to the standard APIs being migrated, some of the major dependent libraries have been updated for this migration as well. Most notably:

- Helidon 4.1.x
- Jersey 3.1.x
- Jackson 2.14.x
- Jackson DataBind 2.14.x
- Weld 5.1.1.x
- JAXB Core 4.0x
- JAXB Implementation 4.0.x
- Eclipse MP Config 3.1.x
- Eclipse MP Metrics 5.1.x

> **Note**
>
> If you are using the older `jackson-rs-base` and `jackson-jaxrs-json-provider` libraries, you must migrate to the `jakarta` versions. The Maven groupId for the `jakarta` versions is `com.fasterxml.jackson.jakarta.rs` with the artifactIds being `jackson-jakarta-rs-base` and `jackson-jakarta-rs-json-provider`, respectively.

> **Note**
>
> If you are using the older `jackson-module-jaxb-annotations` library, you must migrate to the `jakarta` versions. The Maven groupId for the `jakarta` version remains the same (`com.fasterxml.jackson.module`), however the artifactId should now be `jackson-module-jakarta-xmlbind-annotations`.

Upgrading from Version 14.1.2.x

You can migrate Coherence 14c (14.1.2.0.0) to 15c (15.1.1.0.0).

Coherence 15.1.1.0.0 has migrated to Jakarta EE 9.1 from Jakarta EE 8, importing types in `jakarta` packages instead of `javax` packages. For detailed information on the mapping of `javax` packages to `jakarta` packages and Maven artifacts, see Jakarta EE 9.1 Compatibility.

Upgrading from Version 14.1.1.x

You can migrate Coherence 14.1.1.x to 15c (15.1.1.0.0).

Coherence 15.1.1.0.0 has migrated to Jakarta EE 9.1 from Jakarta EE 8, importing types in `jakarta` packages instead of `javax` packages. For detailed information on the mapping of `javax` packages to `jakarta` packages and Maven artifacts, see Jakarta EE 9.1 Compatibility.

Updating JVM

The minimum supported JVM version for Coherence has changed. See Runtime Requirements.

Update Maven Build Scripts

The version is now 15.1.1-0-0. If you are using Maven to create, build, and deploy Oracle Coherence applications, then you must change your scripts accordingly. See Building Oracle Coherence Projects with Maven in *Developing Applications Using Continuous Integration*.

Performing a Rolling Restart (Optional)

Install Oracle Coherence 15c (15.1.1.0.0) in a new Oracle Home. Start the new 15c (15.1.1.0.0) Coherence servers and bring down the servers of the previous Coherence version in a rolling fashion.

When upgrading a cluster, a rolling restart can only be used to upgrade patch set releases or patch set update releases, but not major or minor releases. Therefore, it is not supported to perform a rolling upgrade to Coherence 15c (15.1.1.0.0) from previous releases. See Release Number Format in *Administering Oracle Fusion Middleware*.

If you are applying a patch, see Performing a Rolling Restart in *Developing Applications with Oracle Coherence* for details on how to perform a rolling restart of a Coherence cluster without losing cache data.

Recompiling Your Application

You must recompile your application code against the newly patched Coherence version and a supported Java JDK/JRE. See Runtime Requirements.

Upgrading from Version 12.2.1.x

You can migrate Coherence 12.2.1.x to 15c (15.1.1.0.0).

Coherence 15.1.1.0.0 has migrated to Jakarta EE 9.1 from Jakarta EE 8, importing types in `jakarta` packages instead of `javax` packages. For detailed information on the mapping of `javax` packages to `jakarta` packages and Maven artifacts, see Jakarta EE 9.1 Compatibility.

This section includes the following topics:

Updating JVM

The minimum supported JVM version for Coherence has changed. See Runtime Requirements.

Updating Maven Build Scripts

The version is now 15.1.1-0-0. If you are using Maven to create, build, and deploy Oracle Coherence applications, then you must change your scripts accordingly. See Building Oracle Coherence Projects with Maven in *Developing Applications Using Continuous Integration*.

Performing a Rolling Restart (Optional)

Install Oracle Coherence 15c (15.1.1.0.0) in a new Oracle Home. Start the new 15c (15.1.1.0.0) Coherence servers and bring down the servers of the previous Coherence version in a rolling fashion.

When upgrading a cluster, a rolling restart can only be used to upgrade patch set releases or patch set update releases, but not major or minor releases. Therefore, it is not supported to perform a rolling upgrade to Coherence 15c (15.1.1.0.0) from previous releases. See Release Number Format in *Administering Oracle Fusion Middleware*.

If you are applying a patch, see Performing a Rolling Restart in *Developing Applications with Oracle Coherence* for details on how to perform a rolling restart of a Coherence cluster without losing cache data.

Upgrading from Version 12.1.*x*

You can migrate Coherence 12.1.x to 15c (15.1.1.0.0).

Coherence 15.1.1.0.0 has migrated to Jakarta EE 9.1 from Jakarta EE 8, importing types in `jakarta` packages instead of `javax` packages. For detailed information on the mapping of `javax` packages to `jakarta` packages and Maven artifacts, see Jakarta EE 9.1 Compatibility.

This section includes the following topics:

Updating JVM

The minimum supported JVM version for Coherence has changed. See Runtime Requirements.

Updating Maven Build Scripts

The `maven-gar-plugin` plug-in and `maven-gar-archetype` archetype have been refactored to `gar-maven-plugin` and `gar-maven-archetype`, respectively. Also, the version is now 15.1.1-0-0. If you are using Maven to create, build, and deploy Oracle Coherence applications, then you must change your scripts accordingly. See Building Oracle Coherence Projects with Maven in *Developing Applications Using Continuous Integration*.

Updating Cache Configuration File

A new default cache configuration file is included in the `coherence.jar` library. The new default configuration is not backwards compatible with the previous configuration. If your solution relies on the previous default cache configuration file, then the proper work around is to author a new cache configuration file that defines the required cache mappings and override the default cache configuration file. If your solution does not rely on the default cache configuration file, then no update is required.

Updating Address and Port Assignments

Significant enhancements have been made to simplify the way Coherence addresses and ports are configured and may require updates to your solution. The enhancements include:

- Coherence now uses port 7574 as the default cluster port for multicast communication and `239.192.0.0` as the default address. Addresses and ports that are explicitly configured are still used. However, solutions that rely on the previous defaults need to be updated to use the new defaults. See Specifying a Cluster's Multicast Address and Port in *Developing Applications with Oracle Coherence*.
- Unicast Ports are now automatically selected. Unicast ports that are explicitly configured are still used. However, solutions that relied on the previous default ports need to be updated accordingly. For most use cases, unicast ports do not need to be explicitly configured. See Specifying a Cluster Member Unicast Address and Port in *Developing Applications with Oracle Coherence*.
- WKA addresses now use the cluster port. WKA addresses which contain an explicit port are still respected but it is recommended that the new form which does not include a port be used instead as it provides increased availability. However, solutions that relied on the previous default port need to be updated accordingly. See Specifying WKA Addresses in *Developing Applications with Oracle Coherence*.
- The Name service now automatically uses the cluster port. Proxy addresses that are explicitly configured are still used. However, extend clients that rely on the Name service to find a proxy and rely on the previous default Name service port must be updated to use the new default. Extend clients that run on the same network as the proxy and use the Name service are no longer required to configure an address or a port, so long as they have an operational configuration which is compatible with the cluster. See Defining a Single Proxy Service Instance in *Developing Remote Clients for Oracle Coherence*.

Updating Multiple Clusters that Run on the Same Network

Multiple clusters can now share a cluster port and Multicast or WKA address. For most use cases, there is no reason to change the cluster port, or multicast address. Note that clusters configured to use SSL do not support sharing. In addition, clusters that are configured to only support IPv4 (`-DpreferIPv4Stack=true`) can only share with other clusters that are configured to only support IPv4. The use of `-DpreferIPv4Stack=true` is generally not necessary. If your solution includes multiple clusters on the same network, consider using the Coherence defaults addresses and port and not explicitly configuring addresses and ports. Note that when using shared addresses and ports the selection of a unique cluster name is required.

Planning for TCP Usage

The default protocol that is used between clustered data services has changed from UDP to TCP message bus (TMB). UDP is still used for cluster maintenance while TCP is used for workloads which may be more performance sensitive. Most networks are already optimally configured for TCP and do not require Coherence-specific configuration. In addition, there should be very little network load difference between UDP and TCP. A message bus test utility is provided that can be used test TMB performance between network nodes. See Running the Message Bus Test Utility and TCP Considerations in *Administering Oracle Coherence*.

Solutions that require the use of a firewall between cluster members should ensure that the cluster port (7574) is open for both UDP and TCP for both multicast and unicast configurations as well as port 7 for Coherence TcpRing/IpMonitor death detection. Lastly, ensure that the unicast port range is open for both UDP and TCP traffic and that the unicast listen port range is explicitly set rather then relying upon a system assigned ephemeral port. See Changing the Default Unicast Port in *Developing Applications with Oracle Coherence*.

Updating Extractor Implementations

The `QueryHelper.createExtractor()` API does not produce value extractors that are equivalent with previous versions of Coherence. Do not use `QueryHelper.createExtractor()` for indexes and extend client filters if you have extend clients running previous versions of Coherence. Instead, you should change the extractors to use the actual extractors (`ReflectionExtractor`).

For example, change `QueryHelper.createExtractor("key().myKey");` to `new ReflectionExtractor("getMyKey", null,ReflectionExtractor.KEY);`

Updating Packaging for Coherence REST on WebLogic Server

WebLogic Server now includes the `coherence-rest.jar` library in the server classpath. Existing Coherence REST applications that are deployed on WebLogic server should be repackaged and the `coherence-rest.jar` library should be removed from the application. See Deploying to WebLogic Server in *Developing Applications with Oracle Coherence*.

Running coherence.jar for the Coherence Console

Executing `java -jar coherence.jar` starts a `DefaultCacheServer` instance rather than the legacy Coherence console. If your solution depends on the console, you can start the console using the `bin/coherence` script or directly using:

```
java -cp coherence.jar com.tangosol.net.CacheFactory
```

Updating CohQL Scripts

The `BACKUP CACHE` and `RESTORE CACHE` statements available in CohQL are deprecated. Applications or scripts that relied on these commands must be updated to use Coherence persistence and the new persistence statements. See Persisting Cache Data to Disk in *Developing Applications with Oracle Coherence*.

Updating the Coherence*Web Configuration

The default Coherence*Web session configuration file no longer includes a near cache definition. Applications that were dependent on the near cache configuration must override the default configuration file and define a near cache definition. See Defining Near Cache Schemes in *Developing Applications with Oracle Coherence*.

Migrating to a Supported Web Container

Coherence*Web no longer supports the following web containers: Apache Tomcat 5.5.*n*, Apache Tomcat 6.0.*n*, Caucho Resin 3.1.*n*, IBM WebSphere 5.*n*, IBM WebSphere 6.*n*, IBM WebSphere 7.*n*, Sun GlassFish 2.*n*, Sun Application Server 8.*n*, Oracle OC4J 10.1.3.*n*, Oracle OC4J 10.1.2.*n*, Oracle GlassFish 3.*n*, Oracle GlassFish 4.*n*, Jetty 6.1.*n*, Jetty 5.1.*n*, JBoss Application Server. Applications that require Coherence HTTP session management must be migrated to use a supported web container version. See Supported Web Containers in *Administering HTTP Session Management with Oracle Coherence*Web*.

Removing ActiveCache Integrations

The `active-cache.jar` library that was previously used to integrate Coherence with WebLogic Server has been removed from the WLS distribution. Solutions that rely on the Coherence and WLS integration must be re-factored to use the Managed Coherence Server integration instead. See Deploying Coherence Applications to WebLogic Server in *Administering Oracle Coherence*.

Removing Encryption Filters

Encryption filters are no longer available and can no longer be used. Solutions that rely on encryption filters must now be configured to use SSL. See Using SSL to Secure Communication in *Securing Oracle Coherence*.

Removing TopLink Grid Implementations

TopLink Grid has been deprecated in the TopLink product. Applications must be re-architected to use the Coherence API in their data access layers instead of using the JPA API.

Updating Custom Health Monitors

The hexadecimal receive string that is required to ping Coherence from a BIG-IP LTM custom health monitor has changed. If your solution makes use of a BIG-IP LTM custom health monitor to ping Coherence, then you must update the monitor to use the new hexadecimal string. See Using Advanced Health Monitoring in *Developing Remote Clients for Oracle Coherence*.

Upgrading from Version 3.7.1.*x*

You can migrate Coherence 3.7.1.x to 15c (15.1.1.0.0).
Coherence 15.1.1.0.0 has migrated to Jakarta EE 9.1 from Jakarta EE 8, importing types in `jakarta` packages instead of `javax` packages. For detailed information on the mapping of `javax` packages to `jakarta` packages and Maven artifacts, see Jakarta EE 9.1 Compatibility.

> **Note**
>
> Perform the tasks as required for your Coherence deployment. However, these tasks should be performed only after considering the upgrade issues for 15c (15.1.1.0.0) which may supersede these instructions. See Upgrading from Version 12.1.*x*.

This section includes the following topics:

Upgrading Applications Using Coherence and Coherence*Web on WebLogic Server

Follow these instructions for upgrading applications running on WebLogic Server that use Coherence and Coherence*Web.

1. In an existing WebLogic Server domain:
   - Stop and undeploy the applications that use Coherence*Web.
   - Undeploy the `coherence.jar` and `coherence-web-spi.war` files if they are deployed.
2. Follow the steps to upgrade WebLogic Server and its domains to WebLogic Server 15c (15.1.1.0.0). See Roadmap for Upgrading Your Application Environment in *Upgrading Oracle WebLogic Server*.
3. Modify your applications to remove all references to the `coherence.jar` file:
   - In the `weblogic.xml` file, remove the `<library-ref>` element that refers to the `coherence-web-spi` file.
   - In the `META-INF/MANIFEST.MF` file, remove the following lines that identify Coherence as an extension:

     ```
     Extension-List: coherence
     coherence-Extension-Name: coherence
     ```

   - Remove any explicit references to the `coherence.jar` file in the classpath.
4. Modify your applications to use the required settings for Coherence 15c (15.1.1.0.0):
   - If you have used the default `session-cache-config.xml` file in your Coherence release 3.7.1.*x* application, note that the name is `default-session-cache-config.xml` in 15c (15.1.1.0.0).

     For example, if you used this context parameter value in Coherence release 3.7.1.*x* application:

     ```
     coherence.cacheconfig=session-cache-config.xml
     ```

     change it to `default-session-cache-config.xml`:

     ```
     coherence.cacheconfig=default-session-cache-config.xml
     ```

You should not have to change the session cache file name. If you created a custom `session-cache-config.xml`, you should be able to leave the file name as it is.

- If your application is in an EAR file, then the packaging for the custom `session-cache-config` file has changed. See Using a Custom Session Cache Configuration File in *Administering HTTP Session Management with Oracle Coherence*Web*.

5. Redeploy your applications on WebLogic Server.

Upgrading Coherence*Extend

For all Extend client customers (Java, C++, and .NET), you must upgrade the cluster side before upgrading the Coherence*Extend clients. This is in compliance with the Coherence client and proxy upgrade policy. See Compatibility Between Coherence*Extend Versions in *Installing Oracle Coherence*.

Upgrading Coherence*Web

The following sections describe upgrade considerations for Coherence*Web.

Coherence*Web SPI Reserved for Older Versions of WebLogic

The `coherence-web-spi.war` file, which was included in previous releases of Coherence*Web, is deprecated. If you are using WebLogic Server 15c (15.1.1.0.0), you should not have to work with or reference this file. If you attempt to deploy the `coherence-web-spi.war` file to WebLogic Server 12*c* (12.2.1.1), it will be ignored.

ActiveCache (active-cache.jar) Replaced with Managed Coherence Servers

ActiveCache (`active-cache.jar`), the collection of WebLogic Server features which allow deployed applications to easily use Coherence data caches and seamlessly incorporate Coherence*Web for session management, has been deprecated since the 12.1.2. release.

Users must migrate to Managed Coherence Servers when developing new WebLogic Server/ Coherence applications for the current release. See Deploying Coherence Applications to WebLogic Server in *Administering Oracle Coherence*.

New Session Cache Configuration File

In previous releases, Coherence cache configurations and services used by Coherence*Web SPI were defined in the `session-cache-config.xml` file. Coherence cache configurations and services used by Coherence*Web are now defined in the `default-session-cache-config.xml` file, which can be found in the `coherence-web.jar` file. The default cache and services configuration defined in the `default-session-cache-config.xml` file should satisfy most Web applications.

You can create your own custom session cache configuration by packaging a file named `session-cache-config.xml` in your Web application. See Using a Custom Session Cache Configuration File in *Administering HTTP Session Management with Oracle Coherence*Web*.

Upgrading ActiveCache Applications on WebLogic Server

The 11g Release 1 (10.3.6) version of ActiveCache is documented in About ActiveCache in *Oracle Fusion Middleware Using ActiveCache*. This version of ActiveCache will work with

WebLogic Server and Coherence 12.1.2 but some of the documented steps are no longer required.

> **Note**
>
> ActiveCache has been deprecated since the 12.1.2 release. Users must migrate to Managed Coherence Servers. See Deploying Coherence Applications to WebLogic Server in *Administering Oracle Coherence*.

- Choose the ActiveCache Deployment Topology in *Oracle Fusion Middleware Using ActiveCache* describes the several different combinations of application and data tiers, or *cluster topologies*, in which ActiveCache can be deployed. In upgrading applications using ActiveCache, you should not use the Out-of-Process topology except for backward compatibility. In the current release, WebLogic Out-of-Process topology is the preferred approach. Using managed Coherence servers makes the WebLogic Out-of-Process topology easier to configure.
- Locate the Cache Configuration File in *Oracle Fusion Middleware Using ActiveCache* describes the location where you place the cache configuration file. The location where you store the cache configuration file determines the cache scope; that is, the visibility of the caches to deployed applications. The approaches described in this section will work, but putting the cache configuration in the system classpath is a bad practice unless there is only one and will only ever be one application using Coherence in the server.

  Oracle recommends that you use a GAR file when you package your application. The cache configuration file is packaged in the GAR file. For more information on the GAR file and its packaging structure, see Packaging Coherence Applications in *Developing Oracle Coherence Applications for Oracle WebLogic Server*.
- Configuring Application-Server Scoped Coherence Clusters in *Oracle Fusion Middleware Using ActiveCache* describes a configuration such that all deployed applications on WebLogic Server instances that are directly accessing Coherence caches become part of one Coherence cluster. In the procedure, **do not** perform Step 1: do not put the `coherence.jar` and `active-cache.jar` files in the system classpath. The `active-cache.jar` file uses the classpath in the `MANIFEST` file to add the Coherence integration module to the classpath. In release 12.1.2, the Coherence integration module will always be in the server classpath, in addition to the `coherence.jar` file.
- Configuring EAR-Scoped Coherence Clusters in *Oracle Fusion Middleware Using ActiveCache* describes a configuration such that all deployed applications within each EAR become part of one Coherence cluster. Caches will be visible to all modules in the EAR. The procedure described in this section will not work as described. Because `coherence.jar` is already in the system classpath, you must follow the steps documented in the for using a filtering Classloader.

  The only reason to use the EAR-scoped approach is to isolate your application from other Coherence applications. That use case is better handled by the application isolation provided by a GAR file, or by using the `scope` element in the cache configuration file. Another use case is to use a different version of `coherence.jar` than is in the system classpath but using a different version should be discouraged.
- Configuring WAR-Scoped Clusters in *Oracle Fusion Middleware Using ActiveCache* describes a configuration such that each deployed Web application becomes its own Coherence cluster. Caches will be visible to the individual modules only. In the procedure, **do not** perform Steps 1 and 2. The `coherence.jar` and `active-cache.jar` should not be deployed as shared libraries nor should they appear in the `MANIFEST` file. You can perform

Step 3 to reference the Coherence cluster system resource, but making the managed server a member of the Coherence cluster is the preferred approach.

- Example 3-10 tangosol-coherence-override.xml in *Oracle Fusion Middleware Using ActiveCache* displays a custom cache configuration file that contains a logging configuration. The logging configuration is not required.
- Start a Cache Server in *Oracle Fusion Middleware Using ActiveCache* describes several different ways of starting the cache server. The Out-of-Process topology should be replaced with managed Coherence servers. The procedure for starting a cache server using node manager should be performed by using managed Coherence servers, instead of using the external cache server managed by WebLogic Server.

Replacements for Deprecated Features

The following sections describe replacements for features that have been deprecated since Coherence 12.1.2.

Replacement for Deprecated packet-pool and message-pool Elements

The `packet-pool` and `message-pool` elements are deprecated. In Coherence 14c, the API will now take care of sizing. To upgrade, remove the elements from any configuration files.

Replacement for the Deprecated LH File Manager

The LH store manager is deprecated as of Coherence 12.1.2 release. Use Berkeley DB for similar functionality.

Replacement for the Deprecated NamedCache Lock APIs

The `NamedCache lock` APIs are deprecated. Use the locking support that is provided by the entry processor API instead (`EntryProcessor` for Java and C++, `IEntryProcessor` for .NET).

Replacement for the Deprecated XmlConfigurable Interface

The `com.tangosol.run.xml.XmlConfigurable` interface has been deprecated since the Coherence 12.1.2 release. Coherence used this interface to inject XML parameters into instances of custom classes.

In Coherence 14*c*, you can initialize parameters by writing XML which nests `<instance>` and `<class-scheme>` (or any other custom namespace) inside of `<param-value>` elements.

For example, given the following Java code:

```
public class MyClass
  {
  public MyClass(String s, OtherClass o, int i) { ... }
  }

public class OtherClass
  {
  public OtherClass(String s) { ... }
  }
```

You can initialize the `MyClass` and `OtherClass` classes by writing the following XML. In the XML, the `MyClass` class is initialized with the string `Hello World` and the integer `42`. The

instance of the `OtherClass` class which appears in the `MyClass` class, is initialized with the string `Goodbye World`.

```
<instance>
  <class-name>MyClass</class-name>
    <init-params>
      <init-param>
        <param-value>Hello World</param-value>
      </init-param>
      <init-param>
        <param-value>
          <instance>
            <class-name>OtherClass</class-name>
              <init-params>
                <init-param>
                  <param-value>Goodbye World</param-value>
                </init-param>
              </init-params>
          </instance>
        </param-value>
      </init-param>
      <init-param>
        <param-value>42</param-value>
      </init-param>
    </init-params>
  </instance>
```

Other Upgrade Issues

The following sections describe issues that you might need to consider when upgrading to Coherence 15c (15.1.1.0.0).*x*.

Connecting from Remote RMI Clients

When connecting from a remote RMI client (different physical computer), add the `java.rmi.server.hostname` RMI system property to the script with the value set to the cluster member's IP address. The address ensures that the RMI stubs that are sent to the client contain the correct server address. See Allowing Remote Access to Oracle Coherence MBeans in *Managing Oracle Coherence*.

Key Associations on the Coherence*Extend Client

Key association is now processed on the extend client by default. Existing client implementations (including Java clients) that rely on key association on the cluster must set the `defer-key-association-check` parameter in order to force the processing of key classes on the cluster.

To force key association processing to be done on the cluster side instead of by the extend client, set the `<defer-key-association-check>` element, within a `<remote-cache-scheme>` element, in the client-side cache configuration to true. For example:

```
<remote-cache-scheme>
   ...
   <defer-key-association-check>true</defer-key-association-check>
</remote-cache-scheme>
```

See Deferring the Key Association Check in *Developing Remote Clients for Oracle Coherence*.

Changes to Invalidation Strategy for Near Caches

The default near cache invalidation strategy `auto` has changed to ensure that reduced network traffic is prioritized over performance. Set the invalidation strategy to `all` for pre-12*c* default behavior. See Near Cache Invalidation Strategies in *Developing Applications with Oracle Coherence*.

New Cache Configuration Element: resource-config

The `resource-config` element contains the configuration information for a class that extends the `com.sun.jersey.api.core.ResourceConfig` class. The instance is used by the HTTP acceptor to load resource and provider classes for the Coherence REST application that is mapped to the specified context path. Multiple resource configuration classes can be configured and mapped to different context paths. See Deploying with the Embedded HTTP Server in *Developing Remote Clients for Oracle Coherence*.

Changes to Invocable API Behavior

Applications that use the Invocable API may receive an error when upgrading from Coherence 3.7.1 to Coherence 14c due to a change in serialization requirements. In Coherence 3.7.1, if an Invocable is sent to a number of nodes including itself, then there is a chance that it will begin local execution before having been serialized for transmission to the remote members. If the Invocable updates non-transient state, this state will be leaked to the other nodes as part of the delayed serialization.

In Coherence 14c, applications that use the Invocable API on local members must make sure that their classes (such as entry processors and aggregators) are serializable.

Upgrading from Coherence HotCache 12.2.1.x to Later Versions

When you upgrade from various Oracle Coherence GoldenGate HotCache versions, you must take the following considerations into account.

For information on how Oracle Coherence and HotCache work together, see Integrating with Oracle Coherence GoldenGate HotCache in *Integrating Oracle Coherence*.

Prerequisites for Oracle GoldenGate and GoldenGate Big Data

For more information on installing or upgrading Oracle GoldenGate (OGG) and Oracle GoldenGate Big Data (OGGBD), see:

- *Installing Oracle GoldenGate Classic Architecture 21c*
- Upgrading Oracle GoldenGate Classic in *Upgrading Oracle GoldenGate 21c*
- Installing Oracle GoldenGate Classic for Big Data in *Installing and Upgrading Oracle GoldenGate for Big Data 21c*
- Upgrading Oracle GoldenGate Classic for Big Data in *Installing and Upgrading Oracle GoldenGate for Big Data 21c*

Table 5-2 Selecting Coherence and GoldenGate versions for Upgrade

HotCache Coherence Cluster Member Version	Minimum JDK Version	Minimum OGG/OGGBD Version[1]
14.1.2.0.x	17	21.11.0.0.0.004
14.1.1.2206.x	11	21.8.0.0.0.3
14.1.1.x, 12.2.1.4.x	8	19.1.0.0.4.002

1 Oracle recommends using the latest available patch.

While it is recommended to use a HotCache Coherence cluster member, if it is not possible to run with the minimum OGG/OGGBD version supported by targeted upgrade Coherence version, then running with a HotCache Coherence extend client allows running with OGG/OGGBD version lower than minimum versions listed in table above. For information about running as a HotCache extend client, see Provide Coherence*Extend Connection Information in *Integrating Oracle Coherence*.

Updates to srccapt.prm

The `RecoveryOptions OverwriteMode` command is now obsolete and should be removed from `srccapt.prm`.

Updates to the HotCache Properties File

You must update the `.properties` file that contains the configuration for HotCache.

Table 5-3 contains descriptions of the changes you must make when moving between HotCache releases. For descriptions of all required properties, see Create a Properties File with GoldenGate for Java Properties in *Integrating Oracle Coherence*. Make sure you refer to the correct version of the documentation for your target upgrade.

Table 5-3 Summary of Changes to the HotCache Properties File

Property	Required Changes
`gg.handler.hotcache.type`	For Oracle GoldenGate Application Adapters 12.2.0 or later, set `gg.handler.hotcache.type=oracle.toplink.goldengate.CoherenceAdapter1220`

Table 5-3 (Cont.) Summary of Changes to the HotCache Properties File

Property	Required Changes
`gg.classpath`	**Note**: You can find any non-Coherence jars mentioned below from the Coherence installation under `ORACLE_HOME/oracle_common/modules`. For **all** releases, add the following items to `gg.classpath`: • `coherence.jar` • `coherence-hotcache.jar` • `oracle.toplink/eclipselink.jar` • `oracle.toplink/toplink-grid.jar` • Application domain classes • Various XML configuration files For **14.1.2.0.0**, you must also add: • `jakarta.persistence.jakarta.persistence-api.jar` • `jakarta.xml.bind-api.jar` • `jaxb-impl.jar, jaxb-xjc.jar` • `oracle.jdbc/ojdbc11.jar` (for an Oracle database) For non-Oracle databases, add an equivalent JDK 11 certified JDBC driver for your chosen database. For **14.1.1.2206**, you must also add: • `javax.persistence.jar` • `oracle.jdbc/ojdbc11.jar` (for an Oracle database) For non-Oracle databases, add an equivalent JDK 11 certified JDBC driver for your chosen database. For **14.1.1.0.0**, you must also add: • `javax.persistence.jar` For **12.2.1.4.0**, you must also add: • `javax.persistence.jar`

6

Running the Coherence Examples

Coherence guides and tutorials are now hosted on the Coherence GitHub Repository and are documented here: Examples - Guides & Tutorials Overview. These examples can be run against Coherence 15c (15.1.1.0.0) and later.

> **Note**
>
> In the rest of the document, examples refer to both guides and tutorials.

This chapter includes the following sections:

About Coherence Examples

There are two subsets of examples: Guides and Tutorials.

Guides - These simple guides are designed to be a quick hands-on introduction to a specific feature of Coherence. In most cases, they require nothing more than a Coherence JAR and an IDE (or a text editor). Guides are typically built as a combination of Maven and Gradle project, including the corresponding wrappers for those tools, making them simple to build as stand-alone projects without needing to build the whole Coherence source tree.

Tutorials - These tutorials provide a deeper understanding of larger Coherence features and concepts that cannot usually be explained with a few simple code snippets. They might, for example, require a running Coherence cluster to properly show a feature. Tutorials are typically built as a combination of Maven and Gradle project, including the corresponding wrappers for those tools, making them simple to build as stand-alone projects without needing to build the whole Coherence source tree.

Obtaining the Examples

The examples are hosted on the Coherence GitHub repository.
In a directory of your choice, clone the examples using the following:

```
git clone https://github.com/oracle/coherence.git
```

This command clones the examples into a directory called `coherence`, off the current directory. You can find the examples in the `prj/examples` sub-directory.

The documentation for the examples is available online. See Examples - Guides & Tutorials Overview.

Prerequisites

Each example has documentation outlining the prerequisites for building and running the specific example.

As a minimum, you will require the following:

- JDK 17
- Maven 3.8+ or Gradle 4+. See Maven and Gradle.
- You can also import the code straight into your IDE such as `IntelliJ` IDEA.

Running Examples Using Coherence 15c (15.1.1.0.0)

Perform the following procedures to run examples using Coherence 15c (15.1.1.0.0).

This section includes the following topics:

Installing the Prerequisites

After you have installed Oracle Coherence, you must install the required Coherence Maven POMs and JARs to run the examples. This step is required only once.

Set the `COHERENCE_HOME` environment variable to the **coherence** directory under your installation directory. For example, if Coherence is installed in the `/u01/coherenceHome` location, then `COHERENCE_HOME=/u01/coherenceHome/coherence`.

Run the following commands to install the various artifacts which are the minimum requirements for most of the examples.

> **Note**
>
> You can also use the `maven-sync` plug-in to install all artifacts. See Introduction to the Maven Synchronization Plug-In in *Developing Applications Using Continuous Integration*.

For Linux/Mac:

```
mvn install:install-file  -Dpackaging=pom -Dfile=$COHERENCE_HOME/plugins/
maven/com/oracle/coherence/coherence-bom/15.1.1/coherence-bom.15.1.1.pom \
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence-
bom/15.1.1/coherence-bom.15.1.1.pom

mvn install:install-file -Dfile=$COHERENCE_HOME/lib/coherence.jar \
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence/
15.1.1/coherence.15.1.1.pom

mvn install:install-file -Dfile=$COHERENCE_HOME/lib/coherence-bedrock.jar \
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence-
bedrock/15.1.1/coherence-bedrock.15.1.1.pom

mvn install:install-file -Dfile=$COHERENCE_HOME/lib/coherence-bedrock-testing-
support.jar \
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence-
bedrock-testing-support/15.1.1/coherence-bedrock-testing-support.15.1.1.pom

mvn install:install-file -Dfile=$COHERENCE_HOME/lib/coherence-grpc.jar \
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence-
grpc/15.1.1/coherence-grpc.15.1.1.pom
```

```
mvn install:install-file -Dfile=$COHERENCE_HOME/lib/coherence-grpc-proxy.jar \
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence-
grpc-proxy/15.1.1/coherence-grpc-proxy.15.1.1.pom

mvn install:install-file -Dfile=$COHERENCE_HOME/lib/coherence-java-client.jar
\
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence-
java-client/15.1.1/coherence-java-client.15.1.1.pom

mvn install:install-file -Dfile=$COHERENCE_HOME/lib/coherence-json.jar \
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence-
json/15.1.1/coherence-json.15.1.1.pom
```

Required for GraphQL:

```
mvn install:install-file -Dfile=$COHERENCE_HOME/lib/coherence-cdi-server.jar \
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence-
cdi-server/15.1.1/coherence-cdi-server.15.1.1.pom

mvn install:install-file -Dfile=$COHERENCE_HOME/lib/coherence-cdi.jar \
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence-
cdi/15.1.1/coherence-cdi.15.1.1.pom
```

For Windows:

```
mvn install:install-file  -Dpackaging=pom -Dfile=%COHERENCE_HOME%
\plugins\maven\com\oracle\coherence\coherence-bom\15.1.1\coherence-
bom.15.1.1.pom ^
    -DpomFile=%COHERENCE_HOME%\plugins\maven\com\oracle\coherence\coherence-
bom\15.1.1\coherence-bom.15.1.1.pom

mvn install:install-file -Dfile=%COHERENCE_HOME%\lib\coherence.jar ^
    -DpomFile=%COHERENCE_HOME%
\plugins\maven\com\oracle\coherence\coherence\15.1.1\coherence.15.1.1.pom

mvn install:install-file -Dfile=%COHERENCE_HOME%\lib\coherence-bedrock.jar ^
    -DpomFile=%COHERENCE_HOME%\plugins\maven\com\oracle\coherence\coherence-
bedrock\15.1.1\coherence-bedrock.15.1.1.pom

mvn install:install-file -Dfile=%COHERENCE_HOME%\lib\coherence-bedrock-
testing-support.jar ^
    -DpomFile=%COHERENCE_HOME%\plugins\maven\com\oracle\coherence\coherence-
bedrock-testing-support\15.1.1\coherence-bedrock-testing-support.15.1.1.pom

mvn install:install-file -Dfile=%COHERENCE_HOME%\lib\coherence-grpc.jar ^
    -DpomFile=%COHERENCE_HOME%\plugins\maven\com\oracle\coherence\coherence-
grpc\15.1.1\coherence-grpc.15.1.1.pom

mvn install:install-file -Dfile=%COHERENCE_HOME%\lib\coherence-grpc-proxy.jar
^
    -DpomFile=%COHERENCE_HOME%\plugins\maven\com\oracle\coherence\coherence-
grpc-proxy\15.1.1\coherence-grpc-proxy.15.1.1.pom

mvn install:install-file -Dfile=%COHERENCE_HOME%\lib\coherence-java-
```

```
client.jar ^
    -DpomFile=%COHERENCE_HOME%\plugins\maven\com\oracle\coherence\coherence-
java-client\15.1.1\coherence-java-client.15.1.1.pom

mvn install:install-file -Dfile=%COHERENCE_HOME%\lib\coherence-json.jar ^
    -DpomFile=%COHERENCE_HOME%\plugins\maven\com\oracle\coherence\coherence-
json\15.1.1\coherence-json.15.1.1.pom
```

Required for GraphQL:

```
mvn install:install-file -Dfile=%COHERENCE_HOME%\lib\coherence-cdi-server.jar
^
    -DpomFile=%COHERENCE_HOME%\plugins\maven\com\oracle\coherence\coherence-
cdi-server\15.1.1\coherence-cdi-server.15.1.1.pom

mvn install:install-file -Dfile=%COHERENCE_HOME%\lib\coherence-cdi.jar ^
    -DpomFile=%COHERENCE_HOME%\plugins\maven\com\oracle\coherence\coherence-
cdi\15.1.1\coherence-cdi.15.1.1.pom
```

Required for POF annotations processing:

For Linux/Mac:

```
mvn install:install-file -Dfile=$COHERENCE_HOME/plugins/maven/com/oracle/
coherence/pof-maven-plugin/15.1.1/pof-maven-plugin.15.1.1.jar \
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/pof-maven-
plugin/15.1.1/pof-maven-plugin.15.1.1.pom
mvn install:install-file -Dfile=$COHERENCE_HOME/plugins/maven/com/oracle/
coherence/com.oracle.coherence.gradle.plugin/15.1.1/
com.oracle.coherence.gradle.plugin.15.1.1.jar \
    -DpomFile=$COHERENCE_HOME/plugins/maven/com/oracle/coherence/
com.oracle.coherence.gradle.plugin/15.1.1/
com.oracle.coherence.gradle.plugin.15.1.1.pom
```

For Windows:

```
mvn install:install-file -Dfile=%COHERENCE_HOME%
\plugins\maven\com\oracle\coherence\pof-maven-plugin\15.1.1\pof-maven-
plugin.15.1.1.jar ^
    -DpomFile=%COHERENCE_HOME%\plugins\maven\com\oracle\coherence\pof-maven-
plugin\15.1.1\pof-maven-plugin.15.1.1.pom
mvn install:install-file -Dfile=%COHERENCE_HOME%
\plugins\maven\com\oracle\coherence\com.oracle.coherence.gradle.plugin\15.1.1\
com.oracle.coherence.gradle.plugin.15.1.1.jar ^
    -DpomFile=%COHERENCE_HOME%
\plugins\maven\com\oracle\coherence\com.oracle.coherence.gradle.plugin\15.1.1\
com.oracle.coherence.gradle.plugin.15.1.1.pom
```

Building and Running Any Example

To run any of the examples, you must change the group id and revision using the `-D` option in `mvnw` or the `-P` option in `gradlew` commands to set the Coherence version and group ID you are using.

Examples are shown below:

Maven:

```
./mvnw -Dcoherence.group.id=com.oracle.coherence -Drevision=15.1.1-0-0 clean
verify
```

Gradle:

```
./gradlew -PcoherenceGroupId=com.oracle.coherence -
PcoherenceVersion=15.1.1-0-0 clean build
```

A

Understanding the Oracle Coherence Directory Structure

The standalone Oracle Coherence installation creates multiple directories on your system. Take some time to learn about the directory structure and the files it contains. Table A-1 describes the directories that are installed in *COHERENCE_HOME*.

Table A-1 Directory Description for Oracle Coherence

Directory or File	Description
`bin`	This directory includes a set of common scripts for performing different tasks, such as: starting a cache server, starting development tools, and performing network tests. The scripts are provided in both Windows (`.cmd`) and UNIX-based (`.sh`) formats.
`doc`	This directory contains the Coherence Java API Reference and a link to the Coherence documentation on the Oracle Technology Network (OTN). The Coherence Java API Reference is distributed as a JAR file and must be extracted. The JAR can also be imported into an IDE for easy access during development. To extract the Coherence Java API Reference, execute the following command from the `/api` directory (assuming that *JAVA_HOME*`/bin` is located on the computer's `PATH`): `jar -xvf CoherenceJavaDoc.jar`
`lib`	`lib` – This directory includes all delivered libraries. The `coherence.jar` library is the main development and run-time library and is discussed in detail throughout the Coherence documentation.
`plugins`	This directory contains plug-ins for common integrations. Coherence provides a plug-in for Maven and VisualVM. The Maven plug-ins are used to integrate Coherence as part of a Maven build process. See Integration with Maven. The Coherence VisualVM plug-in provides Coherence monitoring. See Using the Coherence VisualVM Plug-In in *Managing Oracle Coherence*. **Note**: Although there is a VisualVM plug-in in this directory, Oracle recommends you to use the open-source plug-in. See https://github.com/oracle/coherence-visualvm.