Oracle® Fusion Middleware

Installing Oracle Coherence

12c (12.2.1.4.0)
E90861-05
July 2024

Oracle Fusion Middleware Installing Oracle Coherence, 12c (12.2.1.4.0)

E90861-05

Contents

6 Running the Coherence Examples

A Understanding the Oracle Coherence Directory Structure

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 customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

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 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*
- *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.

What's New In This Guide

New and significant changes in *Installing Oracle Coherence*.

This preface includes the following sections:

- New and Changed Features
  New and changed features in *Installing Oracle Coherence* that are organized by release.
- Other Significant Changes in this Document
  Other significant changes in *Installing Oracle Coherence* that are organized by release.

New and Changed Features

New and changed features in *Installing Oracle Coherence* that are organized by release.

New and Changed Features for 12c (12.2.1.4)

Oracle Coherence 12c (12.2.1.4) does not contain any new and changed features for this document.

New and Changed Features for 12c (12.2.1.3)

Oracle Coherence 12c (12.2.1.3) does not contain any new and changed features for this document.

New and Changed Features for 12c (12.2.1.2)

Oracle Coherence 12c (12.2.1.2) does not contain any new and changed features for this document.

New and Changed Features for 12c (12.2.1.1)

Oracle Coherence 12c (12.2.1.1) does not contain any new and changed features for this document.

New and Changed Features for 12c (12.2.1)

Oracle Coherence 12c (12.2.1) includes the following new and changed features for this document.

- Java 8 features example, which demonstrates using lambdas, streams, and default `Map` methods in Coherence. See Coherence Java 8 Features Example.
- Asynchronous `NamedCache` example, which demonstrates using the `AsyncNamedCache` interface. See Coherence Asynchronous Features Example.
- Federated caching example, which demonstrates replicating cache data across two cluster that are configured in an active-active topology. See Coherence Federated Caching Example.

- Persistence example, which demonstrates how to persist and recover cached data. See Coherence Persistence Example.
- REST example, which demonstrates how an application can use Coherence REST to interact with a Coherence cache. See Coherence REST Examples.

Other Significant Changes in this Document

Other significant changes in *Installing Oracle Coherence* that are organized by release.

Other Significant Changes in This Document for 12c (12.2.1.4)

For 12c (12.2.1.4), this guide has been updated in several ways. Following are the sections that have been added or changed.

- Revised C++ client requirements. See Installing the C++ Client Distribution.
- Added compatibility statement. See Coherence 12.2.1.4 Backward Compatibility Exception.

Other Significant Changes in This Document for 12c (12.2.1.3)

For 12c (12.2.1.3), no other significant changes have been made to this guide.

Other Significant Changes in This Document for 12c (12.2.1.2)

For 12c (12.2.1.2), this guide has been updated in several ways. Following are the sections that have been added or changed.

- Revised Linux support for C++ clients. See Supported Environments for Coherence C++ Client.
- Revised REST example instructions to include Binary and JSON pass-through. See Coherence REST Examples.

Other Significant Changes in This Document for 12c (12.2.1.1)

For 12c (12.2.1.1), this guide has been updated in several ways. Following are the sections that have been added or changed.

- Revised instruction for upgrading to Coherence 12.2.*x*. See Upgrading from Version 12.1.*x*.

Other Significant Changes in This Document for 12c (12.2.1)

For 12c (12.2.1), this guide has been updated in several ways. Following are the sections that have been added or changed.

- Revised the JDK requirement. See System Requirements.
- Revised C++ client requirements. See Installing the C++ Client Distribution.
- Revised .NET client requirements. See Installing the .NET Client Distribution.
- Revised instruction for upgrading to Coherence 12.2.*x*. See Upgrading from Version 12.1.*x*.

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 Cloud Application Foundation. 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*.
Prepare your system for installation.	Verify that the necessary environment variables are set, and you have identified a proper installation user.	Prepare Your System for 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 sections:

- System Requirements
  Coherence has different requirements for installation and runtime.
- Performing a Coherence Installation
- Setting Environment Variables
  You can set the *JAVA_HOME* and *COHERENCE_HOME* environment variables. However, they are not required to run Coherence.
- Running Coherence for the First Time
- Integration with Maven
- Installing a Coherence Patch
- Uninstalling Coherence

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 and examples)
- 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) 1.8 or later. 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) 1.8.0_211 or later

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 or examples. 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 and examples. 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
- Performing a Coherence Installation In Silent Mode

- Running the Coherence Quick Installer
- Running the Coherence Supplemental Installer
- Installing Coherence with WebLogic Server

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
- Navigating the Installation Screens

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
   ```

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 Understanding 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.

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

Screen	Description
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.
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 or code examples. 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`):

   ```
   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 and code examples 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 or code examples, 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 Java EE-styled packaging and deployment, remote server management, server clusters, WebLogic Scripting Tool (WLST) automation, and configuration through the Administration 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

- Create a Cache

Create a Basic Cluster

In this step, a basic cluster is created that contains three separate Java processes: a cache server and two cache factory instances. For simplicity, the three processes are collocated on a single computer. The cache server, by default, is configured to store backup data. The two cache factory instances, by default, are configured not to store backup 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 a backup of 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*.

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

Table 2-2 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 the 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
- Installing the C++ Client Distribution
- Installing the .NET Client Distribution
- Compatibility Between Coherence*Extend Versions
  Coherence client distributions support both forward and backwards compatibility with cluster proxies.

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
- Microsoft-Specific Requirements
- Extracting the Coherence for C++ Distribution

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: 2012R2+ Client: Windows 7+	Visual Studio 2012, 2013, 2015, 2017, and 2019	x86, x64
Oracle Solaris 10+	SunPro 5.9+[1,2]	SPARC64, x64
Linux	GCC 4.8.5+, GNU libc 2.1.7+	x86, x64
Apple macOS 10.13+[3]	Xcode 9.4+ (GCC)	x64

1 Specifically Sun C++ 5.9 SPARC Patch 124863-14 or later are supported.
2 Specifically Sun C++ 5.9 x64 Patch 124864-14 or later are supported.

[3] When building C++ applications with Apple OS X, 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 libraries must be installed on the deployment computer.

- Visual C++ Redistributable for Visual Studio 2015, 2017, and 2019
- Visual C++ Redistributable for Visual Studio 2013
- Visual C++ Redistributable for Visual Studio 2012 Update 4

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.

> **Note:**
>
> – For Visual Studio 2015, 2017, and 2019 support, use `\lib\vs2015\coherence.dll`.
>
> – For Solaris, STLport, `/lib/stlport/libcoherence.so`.

Installing the .NET Client Distribution

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

Prerequisites

The following are required to use Coherence for .NET:

- Microsoft .NET 4.0 or higher runtime and SDK
- Supported Microsoft Windows operating system (see the system requirements for the appropriate .NET runtime above)
- MSHelp 2.x runtime, which is included in Visual Studio
- Microsoft Visual Studio 2010 or higher is required to build and run the examples in the `example.zip` file that is provided as part of the Coherence for Java distribution

Running the Installer

Coherence for .NET is distributed as a ZIP file which contains an installer. Use a ZIP utility or the `unzip` command-line utility to extract the installer to a location on the target computer. The following example extracts the installer using the `unzip` utility to the `C:\` directory:

```
unzip C:\path_to_zip\coherence-net-version_number.zip -d C:\
```

To run the installer:

1. From the directory where the ZIP was extracted, double-click the `coherence-net-version.msi` file.
2. Follow the instructions in the installer to complete the installation.

> **Note:**
>
> If the installer indicates that it is rolling back the installation, then run the installer in elevated execution mode. For example, executing the MSI file from a command prompt that was started as an Administrator should enable the installation process to complete. For Windows 7, right-click the command prompt and select **run as Administrator**.

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

- `bin` – This directory includes the Coherence for .NET library. The `Coherence.dll` file is the main development and run-time library and is discussed in detail throughout this documentation.
- `config` – This directory contains XML schemas for Coherence client configuration files and also includes a POF configuration file for Coherence-defined user types.
- `doc` – This directory contains Coherence for .NET API documentation. The API documentation is available as: HTML Help (`Coherence.chm`), MSHelp 2.0, and MS Help Viewer.

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
12.2.1.0	12.2.1.0.0
12.2.1.1	12.2.1.0.1
12.2.1.1000	12.2.1.1.0
12.2.1.1001	12.2.1.1.1
12.2.1.2010	12.2.1.2.10
12.2.1.10010	12.2.1.10.10

> **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.dll` found in the `bin\` folder to your Microsoft.NET application.

Compatibility Between Coherence*Extend Versions

Coherence client distributions support both forward and backwards compatibility with cluster proxies.

Compatibility for the extend protocol and POF is maintained between the second digit of major releases (for example, 12.1, 12.2, and so on) but may not be maintained between the first digit of major releases (for example, 12.x, 13.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.

Prior to version 12.1.2.0.1, extend clients only support forward compatibility with cluster proxies. That is, extend clients can connect to cluster proxies that have either the same or higher second digit of a major release.

Starting with version 12.1.2.0.1, extend clients support both forward and backward compatibility with cluster proxies. That is, extend clients can connect to cluster proxies that have lower or higher version numbers. For example, a 12.1.2.0.2 extend client can connect to a 12.1.2.0.1 proxy. Extend client backward compatibility is not supported on proxy versions prior to 12.1.2.0.1, including 12.1.2.0.0 and proxy versions 3.7.1 or earlier.

Coherence 12.1.2.0.0 extend clients require 12.1.2.0.0 or later cluster proxies. Coherence 12.1.2 extend clients other than 12.1.2.0.0 (for example 12.1.2.0.1 and 12.1.2.0.2) require 12.1.2.0.1 or later cluster proxies.

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.

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.
- Upgrading from Version 12.1.x
  You can migrate Coherence 12.1.*x* applications to 12.2.1.*x*.
- Upgrading from Version 3.7.1.x
- 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.

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).

Upgrading from Version 12.1.*x*

You can migrate Coherence 12.1.*x* applications to 12.2.1.*x*.

This sections includes the following topics:

- Update JVM
- Update Maven Build Scripts
- Update Cache Configuration File
- Update Address and Port Assignments
- Update Multiple Clusters that Run on the Same Network

- Plan for TCP Usage
- Update Extractor Implementations
- Updated Packaging for Coherence REST on WebLogic Server
- Running coherence.jar for the Coherence Console
- Update CohQL Scripts
- Update the Coherence*Web Configuration
- Migrate to a Supported Web Container
- Remove ActiveCache Integrations
- Remove Encryption Filters
- Remove TopLink Grid Implementations
- Update Classpaths for HotCache
- Update Custom Health Monitors

Update JVM

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

Update 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 12.2.1-0-0. 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*.

Update 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.

Update 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*.

Update 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.

Plan 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*.

Update 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 your extractors to use actual extractors (`ReflectionExtractor`). For example:

```
QueryHelper.createExtractor("key().myKey");
```

should be changed to:

```
new ReflectionExtractor("getMyKey", null,ReflectionExtractor.KEY);
```

Updated 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
```

Update 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*.

Update 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*.

Migrate 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*.

Remove 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*.

Remove 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*.

Remove 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.

Update Classpaths for HotCache

Applications that use Oracle Coherence GoldenGate HotCache require an additional JAR file to certain JVM classpaths when upgrading from Coherence version 12.1.*x* to 12.2.1.*x*, and you need to refer to 12.2.1.*x* distributions of other JAR files in those same JVM classpaths.

Specifically, all cache server JVMs (storage-enabled cluster members) need to include *ORACLE_HOME*`/coherence/lib/coherence-hotcache.jar` on their classpaths. Likewise, all HotCache JVMs need to include that same JAR file on their classpaths. The classpaths of HotCache JVMs are configured in a properties file. See Configuring HotCache in *Integrating Oracle Coherence*. Classpaths of cache server and HotCache JVMs also need to be modified to refer to 12.2.1.*x* versions of other JAR files used with HotCache. Those classpaths should refer to the following JAR files from the 12.2.1.*x* installation and not older versions of the same JAR files from a 12.1.*x* installation:

- *ORACLE_HOME*`/coherence/lib/coherence.jar`
- *ORACLE_HOME*`/oracle_common/modules/javax.persistence.jar`
- *ORACLE_HOME*`/oracle_common/modules/oracle.toplink/eclipselink.jar`
- *ORACLE_HOME*`/oracle_common/modules/oracle.toplink/toplink-grid.jar`

Update 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* applications to 12.*x*.

> **Note:**
>
> Perform the tasks as required for your Coherence deployment. However, these tasks should be performed only after considering the upgrade issues for 12.1.*x* 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 12*c* (12.2.1.1). 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 12*c* (12.2.1.1):
   - If you used the default `session-cache-config.xml` file in your Coherence release 3.7.1.*x* application, note that the name has been changed to `default-session-cache-config.xml` in 12*c* (12.2.1.1).

     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
- ActiveCache (active-cache.jar) Replaced with Managed Coherence Servers
- New Session Cache Configuration File

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 12*c* (12.2.1.1), 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, As of the 12*c* (12.2.1.1), Coherence cache configurations and services used by Coherence*Web are 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
- Replacement for the Deprecated LH File Manager
- Replacement for the Deprecated NamedCache Lock APIs
- Replacement for the Deprecated XmlConfigurable Interface

Replacement for Deprecated packet-pool and message-pool Elements

The `packet-pool` and `message-pool` elements are deprecated. In Coherence 12*c* (12.2.1.1), 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 the Coherence 12*c* (12.2.1.1) release, 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 12*c* (12.2.1.1).

- New DistributedCache Default for Exalogic Environments
- Connecting from Remote RMI Clients
- Key Associations on the Coherence*Extend Client
- Changes to Invalidation Strategy for Near Caches
- New Cache Configuration Element: resource-config
- Changes to Invocable API Behavior

New DistributedCache Default for Exalogic Environments

All `DistributedCache` instances now default to the Infiniband Message Bus (IMB) transport in Exalogic environments. The transport is configured within the `<reliable-transport>` service parameter. See DistributedCache Service Parameters in *Developing Applications with Oracle Coherence*.

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* (12.2.1.1) 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 12.*x* 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 12.*x*, 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 5-1 Selecting Coherence and GoldenGate versions for Upgrade

HotCache Coherence Cluster Member Version	Minimum JDK Version	Minimum OGG/OGGBD Version[1]
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 5-2 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 5-2 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 5-2 (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 **12.2.1.4.0**, you must also add: • `javax.persistence.jar`

6

Running the Coherence Examples

The Coherence distribution provides a collection of example code. The examples show how to use multiple Coherence features and are implemented the same across Java, C++, and .NET. This chapter includes the following sections:

- Overview of Coherence Examples
  The Coherence examples are grouped into feature sets. Each feature set contains multiple examples that demonstrate key functionality.
- Obtaining the Examples
  The Coherence examples are included in the `coherence_`*version*`.jar` or `wls_`*version*`.jar` installer file.
- How to Build the Examples
- How to Run the Examples
  The Coherence examples are run using the scripts in the `examples` directory. Scripts are provided for the Java, C++, and .NET (C#) programming languages.
- Coherence Basic Features Example
  The Coherence Basic Features Examples are a collection of examples that demonstrate basic functionality using a simplified contact information tracker application:
- Coherence Security Examples
- Coherence Live Events Example
- Coherence Java 8 Features Example
- Coherence Asynchronous Features Example
- Coherence Federated Caching Example
  The federated caching example starts two clusters: ClusterA and ClusterB. The clusters are configured in a federation topology and cached data is actively synchronized between the two clusters.
- Coherence Persistence Example
- Coherence REST Examples
  The Coherence REST examples shows how to create a basic web-based application that uses the Coherence REST API. The example uses the Grizzly HTTP server to receive client HTTP requests.

Overview of Coherence Examples

The Coherence examples are grouped into feature sets. Each feature set contains multiple examples that demonstrate key functionality.

Basic Features

The Coherence Basic Features Examples include the following:

Table 6-1 Coherence Basic Features Examples

Example Name	Description
Basic Data Access	Getting, putting and removing data from the Coherence Data Grid. See Basic Data Access Example.
Data Loading	Loading example data into the Coherence Data Grid. See Loader Example.
Parallel Querying	Querying the Coherence Data Grid including the use of indexes. See Query Example.
Observable	Listening for changes to data in the Coherence Data Grid. See Observer Example.
Processing	Co-locating data processing with the data itself in the Coherence Data Grid. See Processor Example.
Query Language	How to use the Coherence Query Language. See Query Example.

Security Features

The Coherence Security Examples include the following:

Table 6-2 Coherence Security Examples

Example Name	Description
Password Example	Requiring a password to access Coherence. See Password Example.
Access Control Example	Simplified role based access control. See Access Control Example.
Password Identity Transformer	Creates a custom security token that contains the required password and then adds a list of Principal names. See Password Identity Transformer.
Password Identity Asserter	Asserts that the security token contains the required password and then constructs a Subject based on a list of Principal names. See Password Identity Asserter.
Entitled Cache Service	Wraps a cache service for access control. See Entitled Cache Service.
Entitled Invocation Service	Wraps an invocation service for access control. See Entitled Invocation Service.
Entitled Named Cache	Wraps a named cache for access control. See Entitled Named Cache.

Live Events

The Coherence Live Events Examples are available for the Java platform only. They include the following:

Table 6-3 Coherence Live Events Examples

Example Name	Description
EventsExamples	Illustrates various features within Live Events, such as providing mean elapsed times split by event type, the different semantics in throwing exceptions in pre-events compared to post-events, and logging of partition movement when enabled. See EventsExamples.
TimedTraceInterceptor	Provides timings between pre- and post-commit events for different types of events. See TimedTraceInterceptor.

Table 6-3 (Cont.) Coherence Live Events Examples

Example Name	Description
CantankerousInterceptor	Responds with runtime exceptions at either pre- or post-commit time, based on the type of key being inserted. See CantankerousInterceptor.
RedistributionInterceptor	Logs partition events when enabled. See RedistributionInterceptor.
RedistributionInvocable	Defines three actionable states that will be executed on various members of the cluster. The states are enable logging performed by the RedistributionInterceptor, disable logging, or terminate the JVM that the invocable (RedistributionInvocable) is executed on. See RedistributionInvocable.
LazyProcessor	Creates a superficial delay between the processing of events. See LazyProcessor.

Java 8 Features

The Coherence Java 8 Examples demonstrate how to use Coherence with various features introduced in Java 8. The examples are available for the Java platform only. They include the following:

Table 6-4 Coherence Java 8 Examples

Example Name	Description
StreamsExample	Queries a cache using the `Stream` API. See Streams.
LambdaExample	Performs cache operations using lambda expressions. See Lambda.
MapDefaultMethodExample	Performs cache operations using overridden default methods from the `Map` API. See Map Default Method.

Asynchronous Features

The Coherence Asynchronous Examples demonstrate how to use asynchronous processing using the `AsyncNamedCache` interface. The examples are available for the Java platform only. They include the following:

Table 6-5 Coherence Asynchronous Examples

Example Name	Description
DataAccessExample	Performs cache operations asynchronously. See Asynchronous Data Access.
ProcessorExample	Process cache entries asynchronously. See Asynchronous Entry Processor.
AggregatorExample	Aggregates cache entries asynchronously. See Asynchronous Aggregator.

Federated Caching

The Coherence Federated Caching Examples demonstrates how to configure and use Federated Caching. Two clusters are started and are configured to use Active-Active replication, which means data can be replicated from either cluster to the other. Each cluster includes a GUI to insert data or clear the caches in either cluster. The Federated Caching

Examples also includes steps to configure SSL between cluster participants. The examples are available for the Java platform only.

Persistence

The Coherence Persistence Examples demonstrate how to save and restore the contents of a cache to disk. The examples are available for the Java platform only. They include the following:

Table 6-6 Coherence Persistence Examples

Example Name	Description
Basic Snapshot Operations	Persists the contacts cache to disk. Destroys the cache and then reloads the cache from disk. See Basic Snapshot Operations.
Persistence Notifications	Subscribes to persistence notifications to determine the duration of persistence operations. See Persistence Notifications.
Persistence Operations in Parallel	Runs persistence operations in parallel across multiple services. See Persistence Operations in Parallel.
SFTP Archiver	Creates a custom snapshot archiver which uses SFTP to store and retrieve snapshots. See Archiving Snapshots with a Custom Archiver.

REST

The Coherence REST Examples demonstrates how applications can interact with a Coherence Cache over the HTTP protocol. The example is a single web-based application that exercise serval Coherence features. The examples are available for the Java platform only and the client-side application is written using JavaScript. They include the following:

Table 6-7 Coherence REST Examples

Example Name	Description
Products Example	Insert, edit and remove entries from a cache and update entries using an entry processor. See Products Page.
Department Example	Insert, edit, and remove entries from a cache. See Departments Page.
Contacts Example	Insert, edit, and remove entries from a cache. Query entries in a cache. See Contacts Page.
Server-Sent Events	Add listeners and monitor events for the products, department, and contacts caches. See Server-Sent Events.

Obtaining the Examples

The Coherence examples are included in the `coherence_`*`version`*`.jar` or `wls_`*`version`*`.jar` installer file.

The Coherence examples appear as an installation option in the Oracle Universal Installer and are installed to *`COHERENCE_HOME`*`/examples`.

If you installed Coherence using `coherence_quick_`*`version`*`.jar`, you can obtain the examples by running the `coherence_quick_supp_`*`version`*`.jar` supplemental installer file. The supplemental installer contains only API documentation and examples. Note that the `coherence_quick_`*`version`*`.jar` quick installer file does not install the examples or API documentation.

How to Build the Examples

The Coherence examples' source files must be built using the scripts in the `examples` directory. Scripts are provided for the Java, C++, and .NET (C#) programming languages.

> **Note:**
>
> You must build and run the Java example even for C++ and .NET. This is because the cache server runs in Java.

This section contains the following topics:

- How to Build the Java Examples
- How to Build the .NET Examples
- How to Build the C++ Examples

How to Build the Java Examples

This section contains the following topics:

- Prerequisites for Java
- Directory Structure for Java
- Build Instructions for Java

Prerequisites for Java

To build the example, you must have Coherence and a Java Development Kit (JDK) 1.8 or later. Ensure that the following environment variables are set.

Environment Variable	Description
`$COHERENCE_HOME`	Make sure that the `COHERENCE_HOME` environment variable points to the location of the Coherence installation directory.
`$JAVA_HOME`	Make sure that the `JAVA_HOME` environment variable points to the location of a 1.8 or greater JDK before building the example. A Java runtime 1.8 or greater is needed to run the example

Directory Structure for Java

The directory structure described below is relative to the `examples` directory.

Table 6-8 Directory Structure for Java

Directory Name	Description
`java/bin`	Scripts for building and executing the example. There are two sets of scripts. Scripts with no file extension are bash scripts. Scripts with a `.cmd` file extension are Windows command scripts. The following description refers to the script names without specifying the file extension. • `build`—builds an example
`java/src`	All example source. The examples are in the `com.tangosol.examples.<example name>` package. The classes for objects stored in the cache are in the `com.tangosol.examples.pof` package.
`java/classes`	The class files output from a build. This directory will not exist until the build script is executed.
`java/resource/config`	The common Coherence configuration files required by the examples.
`java/resource/<example name>`	If an example has configuration that is required instead of the common configuration, it will have its own directory. The security example uses configuration files from `java/resource/security`.
`$COHERENCE_HOME/lib`	Coherence libraries used for compiling and running the example.

Build Instructions for Java

Execute the build script with the name of the example collection:

- `bin/build contacts`
- `bin/build security`
- `bin/build events`
- `bin/build java8`
- `bin/build async`
- `bin/build federation`
- `bin/build persistence`

The script builds the POF package files and then the files for the particular example. The `contacts` example is required for the other examples and should always be built first.

On Windows, change directories to the `/bin` directory then run the scripts.

How to Build the .NET Examples

This section contains the following topics:

- Prerequisites for .NET
- Directory Structure for .NET
- Build Instructions for .NET

Prerequisites for .NET

To build the example, you must have Coherence for .NET and Visual Studio 2008 or later or Visual Studio 2008 Express or later. See Prerequisites.

To run the example, you will need the Java version of Coherence and a Java Development Kit (JDK) 1.8 or greater. The Java version is required because the Coherence*Extend proxy and cache servers require Java. Also, the examples depend on Java example classes that must be built before running the proxy and cache server. See the Java example `readme.txt` file for instructions on how to build and run.

Directory Structure for .NET

The directory structure described below is relative to the `examples` directory.

Table 6-9 Directory Structure for .NET

Directory Name	Description
dotnet\src	All example source. The examples are in the `Tangosol.Examples.<example name>` namespace. The classes for objects stored in the cache are in the `Tangosol.Examples.Pof` namespace. The examples are in the Visual Studio 2008 examples solution. Each example has its own Visual Studio 2008 project in the `src` directory. For example, `src` contains projects for the `contacts` and `security` examples. The Coherence configuration files required by the example.
`src\pof\config`	The common Coherence configuration files required by the examples.
`src\<example name>\config`	If an example has configuration that is required instead of the common configuration, it will have its own directory. The security example uses configuration files from `security\config`.

Build Instructions for .NET

Open the examples project from the `examples\dotnet\src\contacts.csproj` directory with Visual Studio

When installing Coherence for the .NET Framework, the installer registers the `coherence.dll` library with the assembly registry. The included Visual Studio projects have a reference to `coherence.dll` in the default location. If another version of the library is desired, or it was not installed in the default location, the Coherence reference can be overridden when configuring the reference, be sure to set the `local copy` attribute to `true`. This setting will copy and register the correct `coherence.dll` in the `bin\debug` directory.

After Coherence for .NET is configured, in Visual Studio select **Build** then **Build Solution** from the menu, **Build Solution (F6)**, etc., to build the solution.

The build for the `contacts` example will copy `resource\contacts.csv` to the build output directory (`examples\dotnet\src\bin\Debug`).

How to Build the C++ Examples

This section contains the following topics:

Prerequisites for C++

To run the examples, you will need the Java version of Coherence and a Java Development Kit (JDK) 1.8 or greater. The Java version is required because the Coherence*Extend proxy and cache servers require Java. Also, the examples depend on Java example classes that must be built before running the proxy and cache server. See the Java examples `readme.txt` for instructions on how to build and run.

Ensure that the following environment variables are set:

Environment Variable	Description
`%COHERENCE_HOME%`	Make sure that the `COHERENCE_HOME` environment variable points to the location of the unpacked Coherence directory.
`%JAVA_HOME%`	Make sure that the `JAVA_HOME` environment variable points to the location of a 1.8 or greater JDK before building the examples. A Java runtime 1.8 or greater is needed to run the examples.
`%COHERENCE_CPP_HOME%`	Make sure that the `COHERENCE_CPP_HOME` environment variable points to the location of the unpacked C++ development environment. Compiler environments supported.

Directory Structure for C++

The directory structure described below is relative to the `examples` directory.

Table 6-10 Directory Structure for C++

Directory Name	Description
`cpp\bin`	Scripts for building and executing the examples. Scripts with no file extension are bash scripts. Scripts with a `.cmd` file extension are Windows command scripts. The following description refers to the script names without specifying any file extension.
`cpp`	All example source organized under the *<example name>* (such as `contacts` and `security`) and `pof` directories.
`cpp\contacts`	The `contacts` example source. The examples are in the `coherence::examples` namespace. The next level of the name after `examples` represents a related set of example classes. "`Driver`" in `coherence::examples::LoaderExample` is the Loader for the `contacts` example. In other words, the name of the example is the name after `coherence::examples`.
`cpp\security`	The `security` example source. The examples are in the `coherence::examples` namespace.

Table 6-10 (Cont.) Directory Structure for C++

Directory Name	Description
cpp\pof	The data model is represented in this directory plus any classes that are serialized. The rationale is to show how to utilize an already existing data model and expose it in Coherence. The model classes do not contain any Coherence-specific code to prove this point. However, there is a serializer that is associated with each model type. For example the Contact has a ContactSerializer class whose purpose is to register the model type with Coherence and serialization operations. The generated output will be in the form of a dynamic library.
cpp\config	The common Coherence configuration files required by the examples.
cpp\config\<*example name*>	If an example has configuration that is required instead of the common configuration, it will have its own directory. The security example uses configuration files from config/security.
cpp\<*example name*>\out	The object files output from a build. This directory will not exist until the build script is executed.
%COHERENCE_CPP_HOME%\include	Contains the Coherence header files.
%COHERENCE_CPP_HOME%\lib	Contains the Coherence library.

Build Instructions for C++

This section contains the following information:

Build Instructions for C++ on Windows

Open a development environment command prompt. This should have been installed with Visual Studio or the platform SDK. Go to the C++ examples directory and type bin\build.cmd <*example name*>. This will build both the pof (model) and the example executable. For example, bin\build.cmd contacts or bin\build.cmd security

The model will put the pof.lib and pof.dll file under cpp\pof\out. These are needed for building and running the contacts and security examples.

The executable contacts.exe will be generated in cpp\contacts\out directory. The executable security.exe will be generated in cpp\security\out directory.

To run the contacts example, type bin\run.cmd contacts after starting a proxy server and cache servers: bin/run-cache-server. The cache server also runs a proxy service which allows connections from Coherence*Extend clients.

As an alternative, in any command window you can cd to the C++ bin directory and run vcvars32.bat before trying to build the examples. With a default install of Visual Studio, the bin directory is C:\Program Files\Mircorsoft Visual Studio 9.0\vc\bin. Follow the previous instructions for running the build script.

Build Instructions for C++ on Linux/Mac and Solaris

Open a command shell. Go to the C++ examples directory and type bin/build <*example name*>. This will build both the pof (model) and the contacts examples executable.

The model dynamic library and `lib` file will be put in `cpp/pof/out`. These are needed for building and running the `contacts` and `security` examples.

The executable contacts, will be generated in `cpp/contacts/out` or `cpp/security/out`.

How to Run the Examples

The Coherence examples are run using the scripts in the `examples` directory. Scripts are provided for the Java, C++, and .NET (C#) programming languages.

> **Note:**
>
> The Coherence examples are distributed as source, so they must first be built. See How to Build the Examples.

This section contains the following topics:

- How to Run the Java Examples
- How to Run the .NET Examples
- How to Run the C++ Examples

How to Run the Java Examples

This section contains the following topics:

- Prerequisites for Java
- Directory Structure for Java
- Instructions for Java

Prerequisites for Java

To run the examples, you must have Coherence installed and use the currently supported JDK. See System Requirements.

Environment Variable	Description
`$COHERENCE_HOME`	Make sure that the `COHERENCE_HOME` environment variable points to the location of the unpacked Coherence directory.
`$JAVA_HOME`	Make sure that the `JAVA_HOME` environment variable points to the location of a supported JDK before building the examples.

Directory Structure for Java

The directory structure described below is relative to the `examples` directory, the directory into which the examples were unzipped.

Table 6-11 Directory Structure for Java

Directory Name	Description
`java/bin`	Scripts for building and executing examples. There are two sets of scripts. Scripts with no file extension are bash scripts. Scripts with a `.cmd` file extension are Windows command scripts. The following description refers to the script names without specifying any file extension. • `run`—Runs an example collection • `run-cache-server`—Runs the cache server used for the examples. The command is also used to start a proxy service that is required for extend clients.
`java/classes`	The class files output from a build. This directory will not exist until the build script is executed.
`java/resource/config`	The common Coherence configuration files required by the examples.
`java/resource/<example name>`	If an example has configuration that is required instead of the common configuration, it will have its own directory. The `security` example uses configuration files from `java/resource/security`.
`$COHERENCE_HOME/lib`	Coherence libraries used for compiling and running the examples.
`resource`	The data file used for the contacts `LoaderExample`: `contacts.csv`.

Instructions for Java

Execute the `run` script for each example.

contacts example

1. Start one or more cache servers: `bin/run-cache-server`. Each execution starts a cache server cluster node. To add additional nodes, execute the command in a new command shell.
2. In a new command shell, run with the name of the example: `bin/run contacts`. The `Driver.main` method runs through the features of the example with output going to the command window (`stdout`).

Starting with Coherence 12.1.2, an example of the new Query Language feature was added. This example shows how to configure and use a simple helper class `FilterFactory` using the Coherence `InvocationService`.

security example

The `security` example requires Coherence*Extend, which uses a proxy.

1. Start one or more cache servers: `bin/run-cache-server security`. The cache server also runs a proxy service which allows connections from Coherence*Extend clients.
2. In a new command shell, run with the name of the example: `bin/run security`. The `Driver.main` method runs through the features of the example with output going to the command window (`stdout`).

live events example

1. Start at least two cache servers: `bin/run-cache-server events`. Each execution starts a cache server cluster node. To add additional nodes, execute the command in a new command shell.
2. In a new command shell, run with the name of the example: `bin/run events`. The `Driver.main` method runs through the features of the example with output going to the command window (`stdout`).

Java 8 features example

1. Start a cache server: `bin/run-cache-server`.
2. In a new command shell, run with the name of the example: `bin/run java8`. The `Driver.main` method runs through the features of the example with output going to the command window (`stdout`). Inspect the output and refer to the code at `src/com/tangosol/examples/java8`.

asynchronous features example

1. Start a cache server: `bin/run-cache-server`.
2. In a new command shell, run with the name of the example: `bin/run async`. The `Driver.main` method runs through the features of the example with output going to the command window (`stdout`). Inspect the output and refer to the code at `src/com/tangosol/examples/async`.

federated caching example

1. Start ClusterA using: `bin/run-cache-server federation ClusterA`.
2. In a new command shell, start ClusterB using: `bin/run-cache-server federation ClusterB`.
3. Run the following to start a GUI which connects to ClusterA: `bin/run federation ClusterA`. Use the `cohql` or `console` argument to use CohQL or the console instead of a GUI.
4. Run the following to start a GUI which connects to ClusterB: `bin/run federation ClusterB`. Use the `cohql` or `console` argument to use CohQL or the console instead of a GUI.
5. Add objects to a cluster and observe that the objects are being replicated to the other cluster.
6. Remove objects from a cluster and observe that the objects are being removed from the other cluster.

The example above uses standard TCP connections between clusters. The example can also be configured to use SSL. SSL allows connections between clusters to be encrypted and ensures only authorized clusters can exchange information by using two-way authentication.

> **Note:**
>
> The SSL configuration uses self signed certificates and obvious passwords. You should follow security best practices and refer to the Coherence security documentation to configure this for production environments.

SSL configuration requires:

- Generating keystores for each of the clusters
- Creating SSL certificates for each cluster
- Importing the certificates into a trust store that ensure only authorized members can communicate.

> **Note:**
>
> For windows environments, make sure you use `%JAVA_HOME%\bin\keytool`.

To configure SSL for the federation examples:

1. Generate Keystores for ClusterA and ClusterB

```
cd $COHERENCE_HOME/examples/java/classes

$JAVA_HOME/bin/keytool -genkeypair -dname "cn=ClusterA, ou=Coherence, o=Oracle,
   c=US" -alias ClusterA -keypass password -keystore ClusterA-keystore.jks
   -storepass password

$JAVA_HOME/bin/keytool -genkeypair -dname "cn=ClusterB, ou=Coherence, o=Oracle,
   c=US" -alias ClusterB -keypass password -keystore ClusterB-keystore.jks
   -storepass password
```

2. Export certificates from each store:

```
$JAVA_HOME/bin/keytool -export -alias ClusterA -storepass password -file
   ClusterA.cer -keystore ClusterA-keystore.jks

$JAVA_HOME/bin/keytool -export -alias ClusterB -storepass password -file
   ClusterB.cer -keystore ClusterB-keystore.jks
```

3. Import both certificates into the trust store that defines which clusters can connect.

```
$JAVA_HOME/bin/keytool -import -v -trustcacerts -alias ClusterA -file
   ClusterA.cer -keystore trust.jks -storepass password

$JAVA_HOME/bin/keytool -import -v -trustcacerts -alias ClusterB -file
   ClusterB.cer -keystore trust.jks -storepass password
```

   Enter 'yes' for both of the above to confirm importing the certificates.

4. Validate the entries in the trust store using:

```
$JAVA_HOME/bin/keytool -list -keystore trust.jks -storepass password
```

Once the above has been completed, the `classes` directory contains the following:

- `trust.jks` – keystore containing the ClusterA and ClusterB certificates

- `ClusterA.jks` – keystore containing the ClusterA private key
- `ClusterB.jks` – keystore containing the ClusterB private key

Re-run the examples and set the SSL environment variable in each command prompt window.

`SET SSL=true` (Windows)

`export SSL=true` (Unix)

In the cache server log files, notice that the connection is now `tmbs` (TCP Message Bus over SSL):

```
Connecting to service FederatedPartitionedPofCache at participant ClusterB with
address tmbs://127.0.0.1:56217.39550
```

> **Note:**
>
> Removing a certificate from the trust store disables communication to that member and simulates an unauthorized communication.

When you have completed running the Federation examples with SSL, make sure to unset the SSL environment variable if you are going to run other examples.

persistence example

1. start one or more cache servers: `bin/run-cache-server persistence`
2. In a new command shell, run the persistence example: `bin/run persistence`. The `Driver.main` method will run through the features of the example with output going to the command window (stdout).
3. Start the notification listener: `bin/run persistence notifications`.
4. Run the persistence example: `bin/run persistence`. Output is emitted that indicates that Persistence operations are being completed
5. Use CTRL+C to interrupt the notifications listener.
6. Run the persistence parallel example: `bin/run persistence parallel`. The `Driver.main` method will run through the features of the example with output going to the command window (stdout).
7. Download the JSch library `jsch-0.1.51.jar` or later and extract the contents into the classes directory.
8. Build the archiver example: `bin/build archiver`.
9. Update the `resource/archiver/tangosol-coherence-override.xml` file and modify the third parameter for the custom archiver and replace the `username`, `password` and `path` to the location of a machine running SSH. If you have ssh equivalence setup to your machine, you can omit the password. You may also consider using a system property to hide your password if one is required.
10. Run the archive example: `bin/run archiver`.
11. Inspect the remote SFTP machine to see the archive directory.

How to Run the .NET Examples

This section contains the following topics:

Prerequisites for .NET

To run the examples, you must have Coherence for .NET and Visual Studio 2008 or later. To run the examples, you will also need to build the Java examples. The Java version is required because the Coherence*Extend proxy and cache servers require Java.

Also, the examples depend on Java example classes that must be built before running the proxy and cache server.

Directory Structure for .NET

The directory structure described below is relative to the "`examples`" directory.

Table 6-12 Directory Structure for .NET

Directory Name	Description
`resource`	The data file used for the contacts `LoaderExample`: `contacts.csv`.

Instructions for .NET

The following sections contain instructions for running the `contacts` and `security` examples.

contacts

1. Start one or more cache servers: `bin/run-cache-server`. The cache server also runs a proxy service which allows connections from Coherence*Extend clients.
2. From Visual Studio, start the `contacts` project without debugging or execute the `contacts.exe` produced from the build in a command shell. The `Driver.Main` method will run through the features of the example with the output going to the command window (stdout).

Starting with Coherence 12.1.2, a new example of the new Query Language feature was integrated. This example shows how configure and use a simple helper class "`FilterFactory`" using the Coherence `InvocationService`.

security

1. Following the java readme.txt instructions, start one or more cache servers: `bin/run-cache-server security`. The cache server also runs a proxy service which allows connections from Coherence*Extend clients.
2. From Visual Studio, start the security project without debugging or execute the `contacts.exe` produced from the build in a command shell. The `Driver.Main` method will run through the features of the example with the output going to the command window (stdout).

How to Run the C++ Examples

This section contains the following topics:

Prerequisites for C++

To build the examples, you must have the appropriate C++ library of Coherence. Also you must have a C++ development environment. To run the examples, you will also need to build the Java examples. The Java version is required because the Coherence*Extend proxy and cache servers require Java. Also, the examples depend on Java example classes that must be built before running the proxy and cache server.

Environment Variable	Description
`$COHERENCE_CPP_HOME`	Make sure that the `COHERENCE_CPP_HOME` environment variable points to the location of the unpacked Coherence C++ installation directory.

The supported C++ compilers are:

- Windows —Microsoft Visual C++ Express/Studio 2008 or later or the equivalent Platform SDK.
- Linux—g++ 4.0
- Mac—g++ 4.0

Directory Structure for C++

The directory structure described below is relative to the `examples` directory.

Table 6-13 Directory Structure for C++

Directory Name	Description
`cpp/bin`	Scripts for building and executing the examples. Scripts with no file extension are bash scripts. Scripts with a `.cmd` file extension are Windows command scripts. The following description refers to the script names without specifying any file extension. • `run`—Runs an example, requires that `java/bin/run-cache-server` be run to start a proxy service.
`cpp`	All example source organized under the `contacts` and `model` directories.
`contact/out`	The object files output from a build. This directory will not exist until the build script is executed.
`resource`	The data file used for the contacts LoaderExample: contacts.csv.
`cpp/contacts`	Contains the `contacts` example sources.
`cpp/security`	Contains the `security` example sources.

Table 6-13 (Cont.) Directory Structure for C++

Directory Name	Description
`cpp/pof`	Contains the `datamodel` sources and any classes that require serialization.
`$COHERENCE_CPP_HOME/include`	Contains the Coherence header files.
`$COHERENCE_CPP_HOME/lib`	Contains the Coherence library.

Instructions for C++

Execute the `run` scripts. There are two parts to running the example. From within new command shells:

contacts example

1. Start one or more cache servers: `bin/run-cache-server`. The cache server also runs a proxy service which allows connections from Coherence*Extend clients.
2. In a new command shell, execute run with the name of the example:

   Running the contacts Example on Windows:

   Type `bin\run.cmd contacts`

   Running the contacts Example on Linux/Mac and Solaris:

   Type `bin/run contacts`

   The `Driver.main` method will run through the features of the example with output going to the command window (`stdout`).

Starting with Coherence 12.1.2, an example of the new Query Language feature was added. This example shows how to configure and use a simple helper class `FilterFactory` using the Coherence `InvocationService`.

security example

1. Start one or more cache servers: `bin/run-cache-server security`. The cache server also runs a proxy service which allows connections from Coherence*Extend clients.
2. In a new command shell, execute `run` with the name of the example:

   Running the security Example on Windows:

   Type `bin\run.cmd security`

   Running the security Example on Linux/Mac and Solaris:

   Type `bin/run security`

   The `Driver.main` method will run through the features of the example with output going to the command window (`stdout`).

Coherence Basic Features Example

The Coherence Basic Features Examples are a collection of examples that demonstrate basic functionality using a simplified contact information tracker application:

This section includes the following topics:

Overview of the Basic Features Examples

The Coherence Basic Features examples include:

- Basic Data Access Example—Getting, putting and removing data from the Coherence Data Grid.
- Loader Example—Loading example data into the Coherence Data Grid.
- Query Example—Querying the Coherence Data Grid including the use of indexes.
- Observer Example—Listening for changes to data in the Coherence Data Grid. See .
- Processor Example—Co-locating data processing with the data itself in the Coherence Data Grid.
- Query Language—How to use the new 3.6 Coherence Query Language.

This example set uses example data represented by these Data Model classes.

Table 6-14 Data Model Classes for the Features Examples

Name	Description
`Address`	Address information
`Contact`	Contact information (includes addresses and phone numbers)
`ContactId`	The key (contact name) to the contact information
`PhoneNumber`	Phone number

This example set also ships with a `contacts.csv` file which is a comma-delimited value file containing sample `Contacts` information.

Running the Example Set

1. Review the following information:
   - How to Build the Examples
   - How to Run the Examples

2. Review the information on the Driver implementation found in Understanding the Features Driver File.

Understanding the Features Driver File

The Driver file has a static `main` method that executes all the Contacts examples in the following order:

1. `LoaderExample`
2. `QueryExample`
3. `QueryLanguageExample`
4. `ObserverExample`
5. `BasicExample`
6. `ProcessorExample`

The Driver file is implemented in each of the three programming languages supported by Coherence.

Language	Implementation Class
Java	`com.tangosol.examples.contacts.Driver` in `java/src`
.NET	Driver in namespace `Tangosol.Examples.Contacts` in `dotnet/src/contacts`
C++	Driver in namespace `coherence::examples` in `cpp/contacts`

Basic Data Access Example

This example shows the most basic data access features of Coherence including getting, putting and removing data.

Java

Implementation Class: `com.tangosol.examples.contacts.BasicExample` in `java/src`

- Associate a `ContactId` with a `Contact` in the cache:

```
cache.put(contactId, contact);
```

- Retrieve the `Contact` associated with a `ContactId` from the cache:

```
contact = (Contact) cache.get(contactId);
```

- Remove mapping of `ContactId` to `Contact` from the cache:

```
cache.remove(contactId);
```

.NET

Implementation Class: `BasicExample` in namespace `Tangosol.Examples.Contacts` in `dotnet/src/contacts`

- Associate a `ContactId` with a `Contact` in the cache:

```
cache.Add(contactId, contact);
```

- Retrieve the `Contact` associated with a `ContactId` from the cache:

```
contact = (Contact)cache[contactId];
```

- Remove mapping of `ContactId` to `Contact` from the cache:

```
cache.Remove(contactId);
```

C++

Implementation Class: `BasicExample` in namespace `coherence::examples` in `cpp/contacts`

- Associate a `ContactId` with a `Contact` in the cache:

```
hCache->put(vContactId, vContact);
```

- Retrieve the `Contact` associated with a `ContactId` from the cache:

```
vContact = cast<Managed<Contact>::View>(hCache->get(vContactId));
```

- Remove mapping of `ContactId` to `Contact` from the cache:

```
hCache->remove(vContactId);
```

Example Output

The example output (due to "Observer Example"):

Example 6-1 Example Output of the Basic Data Access Example

```
entry inserted:
John Nocyefqgqo
Addresses
Home: 1500 Boylston St.
null
Obopnof, NM 88824
US
Work: 8 Yawkey Way
null
Ssedhvmdeq, OR 84217
US
Phone Numbers
work: +11 0 707 3776578
Birth Date: 1971-12-31
entry deleted:
John Nocyefqgqo
Addresses
Home: 1500 Boylston St.
null
Obopnof, NM 88824
US
Work: 8 Yawkey Way
null
Ssedhvmdeq, OR 84217
US
Phone Numbers
work: +11 0 707 3776578
Birth Date: 1971-12-31
```

Loader Example

This example loads contacts into the cache from a file or stream.

It demonstrates the most effective way of inserting data into a cache using bulk inserts. This will allow for minimizing the number of network roundtrips between the application and the cache.

Java

Implementation Class: `com.tangosol.examples.contacts.LoaderExample` in `java/src`

```
cache.putAll(mapBatch);
```

.NET

Implementation Class: `LoaderExample` in namespace `Tangosol.Examples.Contacts` in `dotnet/src/contacts`

```
cache.InsertAll(dictBatch);
```

C++

Implementation Class: `LoaderExample` in namespace `coherence::examples` in `cpp/contacts`

```
hCache->putAll(hMapBatch);
```

Example Output

Example 6-2 Example Output

```
.........Added 10000 entries to cache
```

Query Example

`QueryExample` runs sample queries for contacts.

The purpose of this example is to show how to create `Extractors` on cache data and how to create a `KeyExtractor` for the cache keys. It also illustrates how to use the indexes to filter the dataset to efficiently create a matching set. Finally, the example demonstrates how to use some of the built-in cache aggregators to do simple computational tasks on the cache data. A subset of the code is shown below.

Java

Implementation Class: `com.tangosol.examples.contacts.QueryExample` in `java/src`

- Add an index to make queries more efficient.

  ```
  cache.addIndex(new ChainedExtractor("getHomeAddress.getState"), /*fOrdered*/ false, /
  *comparator*/ null);
  ```

- Find all contacts who live in Massachusetts.

  ```
  Set setResults = cache.entrySet(new EqualsFilter("getHomeAddress.getState", "MA"));
  ```

- Count contacts who are older than `nAge` for the entire cache dataset.

  ```
  System.out.println("count > " + nAge + ": " + cache.aggregate(new
  GreaterFilter("getAge", nAge), new Count()));
  ```

.NET

Implementation Class: `QueryExample` in namespace `Tangosol.Examples.Contacts` in `dotnet/src/contacts`

- Add an index to make queries more efficient.

```
cache.AddIndex(new ChainedExtractor("getHomeAddress.getState"),/*fOrdered*/ false, /
*comparator*/ null);
```

- Find all contacts who live in Massachusetts.

```
ICacheEntry[] aCacheEntry = cache.GetEntries(new
EqualsFilter("getHomeAddress.getState", "MA"));
```

- Count contacts who are older than `nAge` for the entire cache dataset.

```
Console.WriteLine("count > " + nAge + ": "+ cache.Aggregate(new
GreaterFilter("getAge", nAge), new
Count()));
```

C++

Implementation Class: `QueryExample` in namespace `coherence::examples` in `cpp/contacts`

- Add an index to make queries more efficient.

```
ValueExtractor::View vHomeStateExtractor = ChainedExtractor::create(
ChainedExtractor::createExtractors("getHomeAddress.getState"));
```

- Find all contacts who live in Massachusetts.

```
Object::View voStateName = String::create("MA");
Set::View setResults = hCache->entrySet(
EqualsFilter::create(vHomeStateExtractor, voStateName));
```

- Count contacts who are older than `nAge` for the entire cache dataset.

```
Integer32::View nAge = Integer32::valueOf(58);
Object::View vResult = hCache->aggregate( (Filter::View)
GreaterFilter::create(vAgeExtractor, nAge), Count::create());
std::cout << "count > " << nAge->getValue() << ": " << vResult << std::endl;
```

Example Output

The example output is large due to 10,000 contacts and several queries. A sample of the query for Massachusetts residents:

Example 6-3 Example Output of the Query Example

```
MA Residents
ConverterEntry{Key="John Scqngqda", Value="John Scqngqda
Addresses
Home: 265 Beacon St.
Oaskxm, MA 88259
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, OK 95744
US
Phone Numbers
work: +11 88 903 8991283
home: +11 98 553 5878221
Birth Date: 1960-01-03"}
```

Observer Example

`ObserverExample` demonstrates how to use a `MapListener` to monitor cache events such as when cache data has been inserted, updated, and removed. There is no immediate output when this example is run. The registered listener outputs the entry when it is inserted, updated, and deleted. For an update, it outputs both the old value and the new value. The changes to entries are caused by running the Basic Data Access Example and the Processor Example, so the output happens when those examples are run.

A subset of the code is shown below.

Java

Implementation Class: `com.tangosol.examples.contacts.ObserverExample` in `java/src`

- `ContactChangeListener` is a class that implements the `MapListener` interface.

```
cache.addMapListener(new ContactChangeListener());
```

.NET

Implementation Class: `ObserverExample` in namespace `Tangosol.Examples.Contacts` in `dotnet/src/contacts`

- `ContactChangeListener` is a class that implements the `ICacheListener` interface.

```
cache.AddCacheListener(new ContactChangeListener());
```

C++

Implementation Class: `ObserverExample` in namespace `coherence::examples` in `cpp/contacts`

- `ContactChangeListener` is a class that implements the `MapListener` interface using Coherence `implements` clause.

```
ContactChangeListener::Handle hListener = ContactChangeListener::create();
hCache->addFilterListener(hListener);
```

- Definition of `ContactChangeListener`:

```
class ContactChangeListener
   : public class_spec<ContactChangeListener,
      extends<Object>, implements <MapListener> >
```

Processor Example

`ProcessorExample` demonstrates how to use a processor to modify a set of data in the cache. In the code sample that follows, all Contacts who live in MA will have their work address updated.

Java

Implementation Class: `com.tangosol.examples.contacts.ProcessorExample` in `java/src`

Helper Class: `com.tangosol.examples.contacts.OfficeUpdater` in `java/src`

- Apply the `OfficeUpdater` on all contacts who live in MA. The `OfficeUpdater` is a class that implements the `InvocableMap.EntryProcessor` interface by extending `AbstractProcessor`.

```
cache.invokeAll(new EqualsFilter("getHomeAddress.getState", "MA"), new
OfficeUpdater(addrWork));
```

.NET

Implementation Class: `ProcessorExample` in namespace `Tangosol.Examples.Contacts` in `dotnet/src/contacts`

Helper Class: `OfficeUpdater` in namespace `Tangosol.Examples.Contacts` in `dotnet/src/contacts`

- Apply the `OfficeUpdater` on all contacts who live in MA. The `OfficeUpdater` is a class that implements the `IEntryProcessor` interface by extending `AbstractProcessor`.

```
cache.InvokeAll(new EqualsFilter("getHomeAddress.getState", "MA"), new
OfficeUpdater(addrWork));
```

C++

Implementation Class: `ProcessorExample` in namespace `coherence::examples` in `cpp/contacts`

Helper Class: `OfficeUpdater` in namespace `coherence::examples` in `cpp/contacts`

- The `OfficeUpdater` is a class that extends the `UpdaterProcessor` type.

```
class OfficeUpdater
: public class_spec<OfficeUpdater,
extends<UpdaterProcessor>,
implements<PortableObject> >
```

- Apply the `OfficeUpdater` on all contacts who live in MA.

```
Filter::View vEqualsFilter = EqualsFilter::create(
ChainedExtractor::create(ChainedExtractor::createExtractors(
"getHomeAddress.getState")),
String::create("MA"));
InvocableMap::EntryProcessor::Handle hOffice = OfficeUpdater::create(addrWork);
Map::View vMap = hCache->invokeAll(vEqualsFilter, hOffice);
```

Example Output

The example Output (due to Observer Example) is large due to the number of contacts. A sample of output:

Example 6-4 Example Output of the Processor Example

```
entry updated
old value:
John Keau
Addresses
Home: 443 Beacon St.
Ophvowvw, MA 06539
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, FL 86812
US
Phone Numbers
work: +11 8 919 9456102
home: +11 25 759 588823
```

```
Birth Date: 1968-12-31
new value:
John Keau
Addresses
Home: 443 Beacon St.
Ophvowvw, MA 06539
US
Work: 200 Newbury St.
Yoyodyne, Ltd.
Boston, MA 02116
US
Phone Numbers
work: +11 8 919 9456102
home: +11 25 759 588823
entry updated
old value:
John Lbggblkd
Addresses
Home: 929 Beacon St.
Trwylbmf, MA 50358
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, AZ 19164
US
Phone Numbers
work: +11 60 699 203810
home: +11 34 149 5018157
Birth Date: 1964-01-02
new value:
John Lbggblkd
Addresses
Home: 929 Beacon St.
Trwylbmf, MA 50358
US
Work: 200 Newbury St.
Yoyodyne, Ltd.
Boston, MA 02116
US
Phone Numbers
work: +11 60 699 203810
home: +11 34 149 5018157
Birth Date: 1964-01-02
Birth Date: 1968-12-31
```

Query Language

This example shows how to run sample queries for contacts.

Java

Implementation Class: `com.tangosol.examples.query.QueryExample` in `java/src`

- Add indexes to make queries more efficient.

```
cache.addIndex(ff.createExtractor("age"), /*fOrdered*/ true, /*comparator*/ null);
cache.addIndex(ff.createExtractor("homeAddress.state"), /*fOrdered*/ false, /
*comparator*/ null);
```

- Find all contacts who live in Massachusetts.

```
Set setResults = cache.entrySet(ff.createFilter("homeAddress.state = 'MA'"));
```

- Count contacts who are older than `nAge` for the entire cache dataset.

```
final int nAge = 58;
Object[] aEnv = new Object[] {new Integer(nAge)};
System.out.println("count > " + nAge + ": " + cache.aggregate(ff.createFilter("age
> ?1", aEnv), new
Count()));
```

.NET

Implementation Class: `SimpleQueryExample` in namespace `Tangosol.Examples.Query` in `dotnet/src/query`

- Add indexes to make queries more efficient.

```
cache.AddIndex(ff.CreateExtractor("age"), /*fOrdered*/ true, /*comparator*/ null);
cache.AddIndex(ff.CreateExtractor("homeAddress.state"), /*fOrdered*/ false, /
*comparator*/ null);
```

- Find all contacts who live in Massachusetts.

```
ICollection results = cache.GetEntries(ff.CreateFilter("homeAddress.state = 'MA'"));
```

- Count contacts who are older than `age` for the entire cache dataset.

```
const int age = 58;
object[] env = new object[] { age };
results = cache.GetEntries(ff.CreateFilter("age > ?1", env));
```

C++

Implementation Class: `SimpleQueryExample` in namespace `coherence::examples` in `cpp/query`

- Add indexes to make queries more efficient.

```
hCache->addIndex(hff->createExtractor("age"), /*fOrdered*/ true, /*vComparator*/
NULL);
hCache->addIndex(hff->createExtractor("homeAddress.state"), /*fOrdered*/ false, /
*vComparator*/ NULL);
```

- Find all contacts who live in Massachusetts.

```
Set::View setResults = hCache->entrySet(hff->createFilter("homeAddress.state is
'MA'"));
s
```

- Count contacts who are older than `nAge` for the entire cache dataset.

```
Integer32::View nAge = Integer32::valueOf(58);
ObjectArray::Handle haEnv = ObjectArray::create(1);
haEnv[0] = nAge;
HashMap::Handle hbinds = HashMap::create();
hbinds->put(String::create("nAge"), nAge);
setResults = hCache->entrySet(hff->createFilter("age > ?1", haEnv));
```

Example Output

The example output (due to Query Example):

Example 6-5 Example Output of the Query Language Example

```
MA Residents
ConverterCacheEntry{Key="John Wmbltik", Value="John Wmbltik
Addresses
Home: 785 Beacon St.
Vpmji, MA 34400
US
Work: 200 Newbury St.
Yoyodyne, Ltd.
Boston, MA 02116
US
Phone Numbers
work: +11 62 133 6144503
home: +11 17 238 6189757
Birth Date: 1/1/1968 12:00:00 AM"}
ConverterCacheEntry{Key="John Dtpx", Value="John Dtpx
Addresses
Home: 673 Beacon St.
Mvblms, MA 25889
US
Work: 200 Newbury St.
Yoyodyne, Ltd.
Boston, MA 02116
US
Phone Numbers
work: +11 89 900 8436918
home: +11 32 686 9582798
Birth Date: 1/3/1960 12:00:00 AM"}
.
.
.
count > 58 : 496
```

Data Generator

Implementation Class: `com.tangosol.examples.contacts.DataGenerator` in `java/src`

The `DataGenerator` has a static main method that generates random Contact information and stores the results in a comma separated value file. This class was used to generate the `contacts.csv` that is packaged with the `contacts` examples and is included in case more sample data is needed. The class is implemented only in Java.

Coherence Security Examples

The Coherence security examples are a collection of examples that show how to use the security features of Coherence in order to provide access control. The examples are simplified to show only the security features of Coherence. They are not examples of security best practices.
This section contains the following topics:

- Overview of the Coherence Security Examples
- This Example Set
- Password Example

- Access Control Example
- Password Identity Transformer
- Password Identity Asserter
- Entitled Cache Service
- Entitled Invocation Service
- Entitled Named Cache
- Security Example Helper

Overview of the Coherence Security Examples

The security examples include:

- Password Example—Shows how a Coherence Proxy can require a password to access a cache.
- Access Control Example—Shows simplified role based access control.
- Password Identity Transformer—Creates a custom security token that contains the required password and then adds a list of Principal names.
- Password Identity Asserter—Asserts that the security token contains the required password and then constructs a Subject based on a list of Principal names.
- Entitled Cache Service—Wraps a cache service for access control.
- Entitled Invocation Service—Wraps an invocation service for access control.
- Entitled Named Cache—Wraps a named cache for access control.

This Example Set

The Coherence security example set gets a cache reference that requires a password an attempts cache and invocation service operations that require different roles.

This section includes the following topics:

- Running the Security Example Set
- Understanding the Security Driver File

Running the Security Example Set

1. Review the following information:
   - How to Build the Examples
   - How to Run the Examples
2. Review the information on the security Driver implementation found in the next section.

Understanding the Security Driver File

Has a static `main` method that executes all the security examples in the following order:

1. `PasswordExample`
2. `AccessControlExample.accessCache()`
3. `AccessControlExample.accessInvocationService()`

Is implemented in each of the three programming languages supported by Coherence:

Language	Implementation Class
Java	`com.tangosol.examples.security.Driver` in `java/src`
.NET	Driver in namespace `Tangosol.Examples.Security` in `dotnet/src/security`
C++	Driver in namespace `coherence::examples` in `cpp/security`

Please refer to this example set's `example.zip` file for more details on each of the examples outlined below.

Password Example

This example shows how a Coherence Proxy can require a password to get a reference to a cache.

Java

Implementation Class: `com.tangosol.examples.security.PasswordExample` in `java/src`

The code logs in to get a `Subject`, and then tries to get a cache reference running in the context of the Subject.

The Password Identity Transformer will generate a security token that contains the password. The Password Identity Asserter (running in the proxy) will validate the security token to enforce the password. The token generation and validation occurs automatically when a connection to the proxy is made.

.NET

Implementation Class: `PasswordExample` in namespace `Tangosol.Example.Security` in `dotnet/src/security`

The code logs in to get a `Principal`, and then tries to get a cache reference running in the context of the `Principal` by making the `Principal` the Thread's current principal.

The Password Identity Transformer will generate a security token that contains the password. The Password Identity Asserter (running in the proxy) will validate the security token to enforce the password. The token generation and validation occurs automatically when a connection to the proxy is made.

C++

Implementation Class: `AccessExample` in namespace `coherence::examples` in `cpp/security`

The code logs in to get a `Subject`, and then tries to get a cache reference running in the context of the `Subject`.

The Password Identity Transformer will generate a security token that contains the password. The Password Identity Asserter (running in the proxy) will validate the security token to enforce the password. The token generation and validation occurs automatically when a connection to the proxy is made.

Example Output

Example 6-6 Example Output of the Password Example

```
------password example begins------
------password example succeeded------
------password example completed------
```

Access Control Example

This example shows simplified role-based access control.

Java

Implementation Class: `com.tangosol.examples.security.AccessControlExample` in `java/src`

The code logs in to get a `Subject` with a user-id with a particular role, gets a cache reference running in the context of the `Subject`, and then tries cache operations. Depending on the role granted to the user, the cache operation is allowed or denied.

Someone with a `writer` role is allowed to put and get. Someone with a `reader` role can get but not put. Someone with a `writer` role cannot destroy a cache. Someone with an `admin` role can destroy a cache.

Then a user with a particular role tries to use the invocation service. A `reader` is not allowed to invoke, but a `writer` is allowed.

Note that once the cache or invocation service reference is created in the context of a `Subject`, that identity is permanently associated with that reference. Any use of that cache or service reference is on behalf of that identity.

The Password Identity Transformer will generate a security token that contains the password, the user-id, and the roles. The Password Identity Asserter (running in the proxy) will validate the security token to enforce the password, and construct a `Subject` with the proper user-id and roles.

The production and assertion of the security token happens automatically.

See the Entitled Cache Service, Entitled Invocation Service, and Entitled Named Cache code for the implementation of access control.

.NET

Implementation Class: `AccessControlExample` in namespace `Tangosol.Example.Security` in `dotnet/src/security`

The code logs in to get a `Principal` with a user-id with a particular role, gets a cache reference running in the context of the `Principal`, and then tries cache operations. Depending on the role granted to the user, the cache operation is allowed or denied.

Someone with a `writer` role is allowed to put and get. Someone with a `reader` role can get but not put. Someone with a `writer` role cannot destroy a cache. Someone with an `admin` role can destroy a cache.

Then a user with a particular role tries to use the invocation service. A `reader` is not allowed to invoke, but a `writer` is allowed.

Note that once the cache or invocation service reference is created in the context of a `Principal`, that identity is permanently associated with that reference. Any use of that cache or service reference is on behalf of that identity.

The Password Identity Transformer will generate a security token that contains the password, the user-id, and the roles. The Password Identity Asserter (running in the proxy) will validate the security token to enforce the password, and construct a `Subject` with the proper user-id and roles.

The production and assertion of the security token happens automatically.

See the Entitled Cache Service, Entitled Invocation Service, and Entitled Named Cache code for the implementation of access control.

C++

Implementation Class: `AccessControlExample` in namespace `coherence::examples` in `cpp/security`

The code logs in to get a `Subject` with a user-id with a particular role, gets a cache reference running in the context of the `Subject`, and then tries cache operations. Depending on the role granted to the user, the cache operation is allowed or denied.

Someone with a `writer` role is allowed to put and get. Someone with a `reader` role can get but not put. Someone with a `writer` role cannot destroy a cache. Someone with an `admin` role can destroy a cache.

Then a user with a particular role tries to use the invocation service. A `reader` is not allowed to invoke, but a `writer` is allowed.

Note that once the cache or invocation service reference is created in the context of a Subject, that identity is permanently associated with that reference. Any use of that cache or service reference is on behalf of that identity.

The Password Identity Transformer will generate a security token that contains the password, the user-id, and the roles. The Password Identity Asserter (running in the proxy) will validate the security token to enforce the password, and construct a `Subject` with the proper user-id and roles.

The production and assertion of the security token happens automatically.

See the Entitled Cache Service, Entitled Invocation Service, and Entitled Named Cache code for the implementation of access control.

Example Output

Example 6-7 Example Output of the Access Control Example

```
------cache read/write example begins------
      Success: read and write allowed
------cache read/write example complete------
------cache read only example begins------
      Success: read allowed
------cache read only example complete------
------cache destroy with read only access example begins------
------cache destroy with read only access example complete------
------cache destroy example begins------
      Success: Correctly allowed to destroy the cache
------cache destroy example complete------
```

```
------MapListener access control example begins------
------MapListener access control example ends------
------InvocationService read/write access control example begins------
      Success: Correctly allowed to use the invocation service
------InvocationService read/write access control example complete------
------InvocationService read-only access control example begins------
------InvocationService access control example completed------
```

Password Identity Transformer

This example shows how an `IdentityTransformer` produces a security token from an identity.

Java

Implementation Class: `com.tangosol.examples.security.PasswordIdentityTransformer` in `java/src`

The code produces a security token that is an array of strings. The first string is the password. The second string is the user-id and subsequent strings are the user's roles. Arrays of strings will be serialized by Coherence*Extend without requiring a custom serializer.

This class will be invoked automatically when the Extend client connects to a proxy or a channel is opened in an existing connection.

.NET

Implementation Class: `PasswordIdentityTransformer` in namespace `Tangosol.Example.Security` in `dotnet/src/security`

The code produces a security token that is an array of strings. The first string is the password. The second string is the user-id and subsequent strings are the user's roles. Arrays of strings will be serialized by Coherence*Extend without requiring a custom serializer.

This class will be invoked automatically when the Extend client connects to a proxy or a channel is opened in an existing connection.

C++

Implementation Class: `PasswordIdentityTranfromer` in namespace `coherence::examples` in `cpp/security`

The code produces a security token that is an array of strings. The first string is the password. The second string is the user-id and subsequent strings are the user's roles. Arrays of strings will be serialized by Coherence*Extend without requiring a custom serializer.

This class will be invoked automatically when the Extend client connects to a proxy or a channel is opened in an existing connection.

Password Identity Asserter

This example shows how an `IdentityAsserter` validates a security token and produces a `Subject` from a list of principal names.

Java

Implementation Class: `com.tangosol.examples.security.PasswordIdentityAsserter` in `java/src`

The code processes a security token that should be an array of strings. The first string must be the password. Subsequent strings are principals. Any failure processing the token results in a `SecurityException` that will deny access to the proxy.

.NET

Implementation Class: none

The `IdentityAsserter` runs only on the proxy (in Java), so it does not run in the .NET client. Therefore, there is no `PasswordIdentityAsserter` for .NET.

C++

Implementation Class: none

The `PasswordIdentityAsserter` runs only on the proxy (in Java), so it does not run in the C++ client. Therefore there is no `PasswordIdentityAsserter` for C++.

Entitled Cache Service

This example shows how a remote cache service can be wrapped to provide access control.

Java

Implementation Class: `com.tangosol.examples.security.EntitledCachService` in `java/src`

The code instantiates an Entitled Named Cache that provides access control for cache operations. The code also provides access control for the cache service methods `release` and `destroy`. The access control check is delegated to the Security Example Helper.

This class will be instantiated automatically when the cache service is started on the proxy.

.NET

There is no .NET implementation. The class runs only on the proxy in Java.

C++

There is no C++ implementation. The class runs only on the proxy in Java.

Entitled Invocation Service

This example shows how a remote invocation service can be wrapped to provide access control.

Java

Implementation Class: `com.tangosol.examples.security.EntitledInvocationService` in `java/src`

The code provides access control for the invocation service methods. The access control check is delegated to the Security Example Helper.

This class will be instantiated automatically when the invocation service is started on the proxy.

.NET

There is no .NET implementation. The class runs only on the proxy in Java.

C++

There is no C++ implementation. The class runs only on the proxy in Java.

Entitled Named Cache

This example shows how a remote named cache can be wrapped to provide access control.

Java

Implementation Class: `com.tangosol.examples.security.EntitledNamedCache` in `java/src`

The code provides access control for the `NamedCache` methods. The access control check is delegated to the Security Example Helper.

This class will be instantiated automatically when the cache service is started on the proxy.

.NET

There is no .NET implementation. The class runs only on the proxy in Java.

C++

There is no C++ implementation. The class runs only on the proxy in Java.

Security Example Helper

This example is a helper class for authentication and access control.

Java

Implementation Class: `com.tangosol.examples.security.SecurityExampleHelper` in `java/src`

The code simulates authentication. For the sake of simplicity, it creates a `Subject`. A real implementation would do platform- and company-specific authentication. The login also does simple mapping of user names to roles.

The `checkAccess` method checks that the operation is allowed by the user's role.

.NET

Implementation Class: `SecurityExampleHelper` in namespace `Tangosol.Example.Security` in `dotnet/src/security`

The code simulates authentication. For the sake of simplicity, it creates a `Principal`. A real implementation would do platform- and company-specific authentication. The login also does simple mapping of user names to roles.

C++

Implementation Class: `SecurityExampleHelper` in namespace `coherence::examples` in `cpp/security`

The code simulates authentication. For the sake of simplicity, it creates a `Subject`. A real implementation would do platform- and company-specific authentication. The login also does simple mapping of user names to roles.

Coherence Live Events Example

The Coherence Live Events examples illustrate the various event types and how they can be consumed, including `EntryEvents`, `EntryProcessorEvents` and `TransferEvents`. The Live Events Examples are available only in the Java programming language, as they are executed on the storage-enabled members of the partitioned service.

This section includes the following topics:

Overview of the Coherence Live Events Example

The Coherence Live Events examples include:

- EventsExamples—Illustrates various features within Live Events.
- TimedTraceInterceptor—Provides timings between pre- and post-commit events for different types of events.
- CantankerousInterceptor—Responds with runtime exceptions at either pre- or post-commit time, based on the type of key being inserted.
- RedistributionInterceptor—Logs partition events when enabled.
- RedistributionInvocable—Defines three actionable states that will be executed on various members of the cluster. The states are enable logging performed by the `RedistributionInterceptor`, disable logging, or terminate the JVM that the invocable (`RedistributionInvocable`) is executed on.
- LazyProcessor—Creates a superficial delay between the processing of events.

This Example Set

The live events example set illustrates: how to measure the elapsed time between pre- and post-events which are inserted into a results cache; the semantics of throwing exceptions in pre- and post-commit events, and how partition redistribution events can be logged.

This section includes the following topics:

Running the Live Events Example Set

1. Review the following information:

- How to Build the Examples
- How to Run the Examples

2. Review the information on the Live Events Driver implementation found in the next section.

Understanding the Live Events Driver File

Has a static `main` method that executes all the Live Events examples in the following order:

1. Timed Events Example
2. Veto Events Example
3. Partition Transfer Events Example

Is implemented only in the Java programming language:

Language	Implementation Class
Java	`com.tangosol.examples.events.Driver` in `java/src`

EventsExamples

Implementation Class: `com.tangosol.examples.events.EventsExamples` in `java/src`

The `EventsExamples` class illustrates various features within Live Events. This includes:

- Providing mean elapsed times split by event type.
- Illustrating the different semantics in throwing exceptions in pre-events compared to post-events.
- Illustrating logging of partition movement when enabled.

The `EventsExamples` class defines these inner classes:

- EventsTimingExample
- VetodEventsExample
- RedistributionEventsExample

EventsTimingExample

The `EventsTimingExample` inner class is a catalyst for action to be performed by TimedTraceInterceptor. This illustrates how the elapsed time between pre- and post-events can be measured which are inserted into a results cache. The entries inserted into the results cache are displayed by using the `stdout` of the process executing this class.

The example output:

Example 6-8 Example Output of the EventsTimingExample

```
Received stats [memberId=1, eventType=INSERTED, sample=6] = EventStats[name = INSERTED,
sampleMean = 0.357616ms, mean = 0.613750ms]
Received stats [memberId=8, eventType=INSERTED, sample=1] = EventStats[name = INSERTED,
sampleMean = 0.890652ms, mean = 0.890652ms]
Received stats [memberId=1, eventType=UPDATED, sample=2] = EventStats[name = UPDATED,
sampleMean = 0.607513ms, mean = 0.920558ms]
Received stats [memberId=8, eventType=UPDATED, sample=1] = EventStats[name = UPDATED,
sampleMean = 0.729151ms, mean = 0.729151ms]
Received stats [memberId=1, eventType=EXECUTED, sample=6] = EventStats[name = EXECUTED,
```

```
sampleMean = 4.143700ms, mean = 9.267525ms]
Received stats [memberId=8, eventType=EXECUTED, sample=1] = EventStats[name = EXECUTED,
sampleMean = 2.621131ms, mean = 2.621131ms]
Received stats [memberId=1, eventType=REMOVED, sample=6] = EventStats[name = REMOVED,
sampleMean = 3.481549ms, mean = 6.704784ms]
```

VetodEventsExample

The `VetodEventsExample` inner class is a catalyst for action to be performed by CantankerousInterceptor. This illustrates the semantics of throwing exceptions in pre- and post-events. The exceptions that are expected to only be logged are inserted into a results cache. The entries inserted into the results cache are displayed by using the `stdout` of the process executing this class.

The example output:

Example 6-9 Example Output of the VetodEventsExample

```
Received event [memberId=1, eventType=NON_VETO, count=51] = Objection falls on deaf
ears! value = value: 11
Received event [memberId=1, eventType=NON_VETO, count=52] = Objection falls on deaf
ears! value = value: 22
Received event [memberId=1, eventType=NON_VETO, count=53] = Objection falls on deaf
ears! value = value: 33
Received event [memberId=1, eventType=NON_VETO, count=54] = Objection falls on deaf
ears! value = value: 44
Received event [memberId=1, eventType=NON_VETO, count=55] = Objection falls on deaf
ears! value = value: 55
Received event [memberId=1, eventType=NON_VETO, count=56] = Objection falls on deaf
ears! value = value: 66
Received event [memberId=1, eventType=NON_VETO, count=57] = Objection falls on deaf
ears! value = value: 77
Received event [memberId=1, eventType=NON_VETO, count=58] = Objection falls on deaf
ears! value = value: 88
Received event [memberId=1, eventType=NON_VETO, count=59] = Objection falls on deaf
ears! value = value: 99
Number of veto'd events: 5
Received event [memberId=1, eventType=NON_VETO, count=60] = Objection falls on deaf
ears! value = value: 110
```

RedistributionEventsExample

The `RedistributionEventsExample` inner class is a catalyst for action to be performed by the RedistributionInterceptor class. This illustrates how partition redistribution events can be logged, by enabling logging in the `RedistributionInterceptor` and killing a member thus inducing partition redistribution.

The example output:

Example 6-10 Output of the RedistributionEventsExample

```
Choosing to kill member Member(Id=1, Timestamp=2019-09-09 10:44:21.5,
Address=127.0.0.1:62920, MachineId=10131, Location=machine:localhost,process:39588,
Role=CoherenceServer)
```

TimedTraceInterceptor

Implementation Class: `com.tangosol.examples.events.TimedTraceInterceptor` in `java/src`

The `TimedTraceInterceptor` class provides timings between pre- and post-commit events for each type of event; that is, inserts, updates, removes, and entry processor execution. These timings are collected and averaged at a sample rate defined by parameter `cSample`. Additionally they are output to the log at the same time. This implementation does maintain a strong reference to the each binary key however this is removed upon receiving the post-commit event for the same key.

The interceptor implements the `EventInterceptor` interface. The `@Interceptor` annotation provides the unique name of the interceptor with the `identifier` attribute and the order in which it should be executed (`Order.HIGH`) with the `order` attribute.

The interceptor also contains a protected `EventTimer` inner-class. This class times the elapsed time for each event it is notified of. The interceptor tracks the time between a pre- and post-commit event for each entry and the respective event types (`INSERT`, `UPDATE`, `REMOVE`). The timings are sent to the Coherence log in batches displaying sample and cumulative statistics.

As the generic argument is `com.tangosol.net.events.partition.cache.Event`, you will only get events that are consumers of that event, that is, `EntryEvent` and `EntryProcessorEvent`, without specifying any filtering.

CantankerousInterceptor

Implementation Class: `com.tangosol.examples.events.CantankerousInterceptor` in `java/src`

The `CantankerousInterceptor` class is an `EventInterceptor` implementation that is argumentative in nature, hence the event of inserting certain keys will result in runtime exceptions at either pre- or post-commit phases.

If the exception is thrown at pre-commit time, then a rollback occurs and the exception is propagated to the client. If the exception occurs at post-commit time, then a log event is recorded. The keys used for the exceptions are `VETO` and `NON-VETO`. `INSERTING` and `UPDATING` are events that can be vetoed, whereas `INSERTED` and `UPDATED` events cannot be vetoed.

RedistributionInterceptor

Implementation Class: `com.tangosol.examples.events.RedistributionInterceptor` in `java/src`

The `RedistributionInterceptor` class is an `EventInterceptor` that logs partition activity when enabled. Logging can be enabled by setting the `RedistributionInvocable.ENABLED` constant. See RedistributionInvocable.

RedistributionInvocable

Implementation Class: `com.tangosol.examples.pof.RedistributionInvocable` in `java/src`

The `RedistributionInvocable` class defines three actionable states that will be executed on various members of the cluster. For this example, define the states as follows:

- `DISABLE`: Disable the logging performed by the `RedistributionInterceptor` event interceptor.
- `ENABLE`: Enable the logging performed by the `RedistributionInterceptor` event interceptor.
- `KILL`: Terminate the JVM that this invocable (`RedistributionInvocable`) is executed on.

LazyProcessor

Implementation Class: `com.tangosol.examples.pof.LazyProcessor` in `java/src`

The `LazyProcessor` class creates a superficial delay between the processing of events. The class specifies the number of milliseconds this processor should sleep between processing events. This class will be used by the EventsTimingExample subclass in the EventsExamples class.

Coherence Java 8 Features Example

The Coherence Java 8 Features examples illustrate how to use Coherence with features that are available in Java 8. The examples demonstrate using Streams, Lambda, and default methods that were introduced in the `Map` interface. The features are organized as three separate examples; however, these features often build on each other and are not mutually exclusive.
This section includes the following topics:

- This Example Set
- Streams
- Lambda
- Map Default Method

This Example Set

The Coherence Java 8 features example illustrates: how to use the Java streams when querying and processing cache entries; how Lambda features can be used to simplify common Coherence tasks and how to query and process cache entries using new default methods from the `Map` interface that have been overridden in the Coherence `InvocableMap` interface.

This section includes the following topics:

- Running the Java 8 Features Example Set
- Understanding the Java 8 Driver File

Running the Java 8 Features Example Set

1. Review the following information:
   - How to Build the Examples
   - How to Run the Examples
2. Review the information on the Java 8 `Driver` implementation found in the next section.

Understanding the Java 8 Driver File

Has a static `main` method that executes all the Java 8 examples in the following order:

1. Streams
2. Lambda
3. Map Default Method

Is implemented only in the Java programming language:

Language	Implementation Class
Java	`com.tangosol.examples.java8.Driver` in `java/src`

Streams

Implementation Class: `com.tangosol.examples.java8.StreamsExample` in `java/src`.

The `StreamsExample` class perform multiple queries of the Contact cache using the `Stream` API and also makes use of Lambda expressions. The results of the queries are printed to the console. The class also uses the Coherence `RemoteCollector` interface which extends the standard Java `Collector` interface and adds support for serialization in order to process stream elements that are distributed.

Lambda

Implementation Class: `com.tangosol.examples.java8.LambdaExample` in `java/src`.

The `LambdaExample` class uses lambda expressions to add a listener for the Contact cache and update a contact using an entry processor. Lastly a lambda expression is used to query the Contact cache using the Coherence `Filters` API.

Map Default Method

Implementation Class: `com.tangosol.examples.java8.MapDefaultMethdodExample` in `java/src`.

The `MapDefaultMethodExample` class performs multiple queries of the Contact cache and updates several cache entries using default methods that have been added to the `Map` interface. Note that Coherence overrides the default methods in the `InvocableMap` interface. Note also that the example uses lambda expressions when querying the cache.

Coherence Asynchronous Features Example

These Asynchronous examples illustrate how to perform asynchronous data grid operations using the `AsyncNamedCache` API. The examples also uses the `java.util.concurrent.CompletableFuture` API, which is used to check if an operations is complete, to wait for its completion, and to retrieve the result of the operation.
This section includes the following topics:

This Example Set

The Coherence asynchronous features example illustrates: how to asynchronously get and put data in a cache; how to asynchronously process cache entries; how to asynchronously aggregate cache entries.

This section includes the following topics:

- Running the Asynchronous Features Example Set
- Understanding the Asynchronous Driver File

Running the Asynchronous Features Example Set

1. Review the following information:
   - How to Build the Examples
   - How to Run the Examples
2. Review the information on the asynchronous `Driver` implementation found in the next section.

Understanding the Asynchronous Driver File

Has a static `main` method that executes all the asynchronous examples in the following order:

1. Data Access Example
2. Processor Example
3. Aggregator Example

Is implemented only in the Java programming language:

Language	Implementation Class
Java	`com.tangosol.examples.async.Driver` in `java/src`

Asynchronous Data Access

Implementation Class: `com.tangosol.examples.async.DataAccessExample` in `java/src`.

The `DataAccessExample` class uses the `AsyncNamedCache` API to get an instance of the Contact cache. The class creates a new contact and uses the `AsyncNamedCache` instance to put the contact in the cache and then gets the contact from the cache. The contact is changed and then put back into the cache.

Asynchronous Entry Processor

Implementation Class: `com.tangosol.examples.async.ProcessorExample` in `java/src`.

The `ProcessorExample` class uses the `AsyncNamedCache` API to get an instance of the Contact cache. The `AsyncNamedCache` instance is used to query the cache and execute an entry processor which changes the set of contact names to uppercase. The entry processors are then used to change the names back to lower case.

Asynchronous Aggregator

Implementation Class: `com.tangosol.examples.async.AggregatorExample` in `java/src`.

The `AggregatorExample` class uses the `AsyncNamedCache` API to get an instance of the Contact cache. The `AsyncNamedCache` instance is used to query the cache and execute an aggregation on a set of contacts based on age.

Coherence Federated Caching Example

The federated caching example starts two clusters: ClusterA and ClusterB. The clusters are configured in a federation topology and cached data is actively synchronized between the two clusters.

This section includes the following topics:

This Example Set

The Coherence federated caching example set illustrates: federation cluster participant configuration; an active-active replication topology configuration; a federated cache service configuration; SSL configuration to secure communication between cluster participants.

This section includes the following topics:

Running the Federated Caching Example Set

1. Review the following information:
   - How to Build the Examples
   - How to Run the Examples
2. Review the information on the federated caching `Driver` implementation found in the next section.

Understanding the Federated Caching Driver File

Has a static `main` method that:

1. Starts two clusters.
2. Starts either a GUI application, CohQL, or console for each cluster.

Is implemented only in the Java programming language:

Language	Implementation Class
Java	`com.tangosol.examples.federation.Driver` in `java/src`

Federation Configuration

The federation example demonstrates configuration. Inspect the `resource/federation/examples-cache-configure.xml` file for an example of federated cache configuration. Inspect the `resource/federation/tangosol-coherence-override.xml` file for details about how to configure federation participants and replication topologies.

Coherence Persistence Example

The persistence example demonstrate saving and recovering cache data from disk. The examples exercise many different persistence operations programmatically. Persistence operations can also be performed using the `PersistenceCoordinatorMBean` MBean and using CohQL commands.
This section includes the following topics:

This Example Set

The Coherence persistence example set illustrates: how to save and recover a cache snapshot; how to register for persistence notifications how to perform persistence operations in parallel; how to create and use a custom archiver.

This section includes the following topics:

Running the Persistence Example Set

1. Review the following information:
   - How to Build the Examples
   - How to Run the Examples
2. Review the information on the persistence `Driver` implementation found in the next section.

Understanding the Persistence Driver File

Has a static `main` method that executes the persistence examples depending on the arguments that are entered.

1. Basic Snapshot Example
2. Persistence Notifications Example
3. Persistence Operations in Parallel Example
4. Custom Archiver Example

Is implemented only in the Java programming language:

Language	Implementation Class
Java	`com.tangosol.examples.async.Driver` in `java/src`

Basic Snapshot Operations

Implementation Class: `com.tangosol.examples.persistence.BasicSnapshotOperations` in `java/src`

The `BasicSnapshotOperations` class demonstrates how to use persistence snapshots to the save and recover the contents of a cache. It uses the contacts example to populate a cache and then programmatically performs persistence operations as follows:

- A snapshot of the contacts cache is created. The location for persistence files is `java/persistence-data`.
- A list of available snapshots is discovered.
- The contacts cache is cleared of all data.
- The cache contents are recovered from the snapshot.
- The size of cache is reported.
- The snapshot of the contacts cache is removed.
- A list of available snapshots verifies that the snapshot has been removed

Persistence Notifications

Implementation Class: `com.tangosol.examples.persistence.NotificationWatcher` in `java/src`

The `NotificationWatcher` class demonstrates how to monitor notifications from persistence operations. The basic snapshot operations example is run with each operation being monitored. The class creates and registers a persistence notification listener on the contacts cache service. The notifications are then used to monitor the amount of time it takes for persistence operations to be performed.

Persistence Operations in Parallel

Implementation Class: `com.tangosol.examples.persistence.ParallelSnapshotOperations` in `java/src`

The `ParallelSnapshotOperations` class demonstrates how to call snapshot operations for multiple partitioned cache services in parallel. The basic snapshot operations example is run and two instance of the contact cache service are created. The persistence operations are then performed for each cache service.

Archiving Snapshots with a Custom Archiver

Implementation Class: `com.tangosol.examples.archiver.SFTPSnapshotArchiver` in `java/src`

The `SFTPSnapshotArchiver` class is a custom implementation of a snapshot archiver that uses JSch library from JCraft (`http://www.jcraft.com/jsch/`) to create an archiver that archives snapshots to a remote server using secure FTP. The SFTPSnapshotArchiver class extends the `AbstractSnapshotArchiver` class. To run this example, the JSch library must be downloaded and the remote server must support SSH.

The example `Driver` file performs the same operations as the basic snapshot operations example, but also includes archive operations. The persistence operations are performed as follows:

- A snapshot of the contacts cache is created. The location for persistence files is `java/persistence-data`.
- A list of available snapshots is discovered.
- The snapshot of the contacts cache is archived using SFTP.
- The snapshot of the contacts cache is removed.
- The contacts cache is cleared of all data.
- The size of cache is reported.
- The archived snapshot is retrieved using SFTP.
- A list of available snapshots is discovered.
- The cache contents are recovered from the snapshot.
- The size of cache is reported.
- The snapshot and the archived snapshot are both removed.

Coherence REST Examples

The Coherence REST examples shows how to create a basic web-based application that uses the Coherence REST API. The example uses the Grizzly HTTP server to receive client HTTP requests.

The example client is built using several JavaScript libraries and also the Angular JS framework. For complete documentation about Coherence REST, see Using Coherence REST in *Developing Remote Clients for Oracle Coherence*.

Unlike the other Coherence examples, the Coherence REST examples uses Apache Maven to build and run the examples. Maven is the preferred approach when using Coherence REST and facilitates managing all library dependencies. The REST examples are organized in a standard Maven directory structure in the *`COHERENCE_HOME`*`/examples/rest/` directory.

- `/src/main/java` – Directory for Java source files
- `/src/main/resources` – Directory for Coherence configuration files.
- `/src/main/resources/web` – Directory for static HTML pages and JavaScript files.

This section includes the following topics:

This Example Set

- Illustrates how create configure and deploy Coherence REST using the Grizzly HTTP server.
- Illustrates how to build a basic JavaScript client that use the Coherence REST APIs.
- Illustrates how to query, create, update and remove cache entries using standard REST API's in Coherence.
- Illustrates how to use a custom entry processors
- Illustrates how to use composite keys and shows the use of a `KeyConverter` class.
- Illustrates how use server-sent events to be notified of cache events.
- Illustrates how to use pass-through for native JSON objects and static binary objects.

Building and Running the Example

The examples are built and run using Maven 3.2.5 or above and require a browser that supports AngularJS 1.4.1 or above.

To build and run the examples:

1. Include the `coherence.jar` and `coherence-rest.jar` libraries in the local Maven repository.

```
mvn install:install-file -Dfile=COHERENCE_HOME/lib/coherence.jar
   -DpomFile=COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence
   /12.2.1/coherence.12.2.1.pom

mvn install:install-file -Dfile=COHERENCE_HOME/lib/coherence-rest.jar
   -DpomFile=COHERENCE_HOME/plugins/maven/com/oracle/coherence/coherence-rest
   /12.2.1/coherence-rest.12.2.1.pom
```

> **Note:**
>
> You may need to specify the path to your `settings.xml` file to download the required dependencies. For example:
>
> ```
> mvn -s /path/to/settings.xml ...
> ```
>
> If you do not have a settings file and you are using a proxy server for internet access, you can utilize the sample `settings.xml` provided in the base directory. You can modify the file to add your proxy server settings.

2. Issue the following to build the REST examples:

```
mvn clean compile
```

3. Start a cache server and HTTP proxy:

```
mvn exec:exec -DhttpProxy
```

The application starts and the home page automatically loads in the default browser. If the home page does not load in the default browser, then navigate to the following URL:

```
http://127.0.0.1:8080/application/index.html
```

> **Note:**
>
> the HTTP server listens on all IP Addresses but you can change the address and port that the application runs on by passing the following to the `mvn exec:exec` command:
>
> ```
> mvn exec:exec -DhttpProxy -Dhttp.address=x.x.x.x -Dhttp.port=7777
> ```

4. Optionally, start additional cache servers (without an HTTP server):

   ```
   mvn exec:exec -DcacheServer
   ```

Products Page

Implementation: *COHERENCE_HOME*`\examples\rest\src\main\resources\web\js\products.js`

The Products page shows how to query, create, update, remove or populate default products using standard REST API's in Coherence. The page also makes use of custom entry processors to increase product prices and receive additional quantities of an item.

Departments Page

Implementation:
COHERENCE_HOME`\examples\rest\src\main\resources\web\js\departments.js`

The Department page shows how to query, create, update, remove or populate default departments using standard REST API's in Coherence.

Contacts Page

Implementation: *COHERENCE_HOME*`\examples\rest\src\main\resources\web\js\contacts.js`

The Contacts page shows how to query, create, update, remove or populate default contacts using standard REST API's in Coherence. The example has composite keys and shows the use of a `KeyConverter` class to work with these keys. Lastly, the example shows how to sort queries that are returned from REST calls.

Server-Sent Events

Implementation: *COHERENCE_HOME*`\examples\rest\src\main\resources\web\js\sse.js`

The server-sent events page listens for events from the Coherence REST API's. Click **Start Listening** to register a listener for the respective cache. Start a new instance of the application and modify the respective caches. Switch back to the original instance of the application to view the updated statistics.

> **Note:**
>
> Internet Explorer does not support server-sent events.

JSON Pass-Through Page

Implementation:
COHERENCE_HOME\examples\rest\src\main\resources\web\js\json.js

The JSON Pass-through page shows how a cache can store and retrieve native JSON objects. The objects are serialized in the cache using POF and JSON attribute ordering is preserved. The page also shows how native JSON objects in a cache can be processed and aggregated like any other value object.

Binary Pass-Through Page

Implementation:
COHERENCE_HOME\examples\rest\src\main\resources\web\js\static.js

The Binary Pass-through page shows how a cache can store and delete static binary content (such as a graphic). The example makes use of the `PassThroughResourceConfig` resource, which supports pass-through access to caches.

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`
`examples`	This directory contains a set of examples that demonstrate many Coherence features and how to use the Coherence API. See Running the Coherence Examples.
`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 Java VisualVM. The Maven plug-ins are used to integrate Coherence as part of a Maven build process. See Integration with Maven. The Java VisualVM plug-in provides Coherence monitoring. See Using the Coherence-JVisualVM Plug-In in *Managing Oracle Coherence*.