Oracle® Fusion Middleware

Using ActiveCache

11*g* Release 1 (10.3.3)

E16517-01

April 2010

This document describes how to use ActiveCache as the caching solution for WebLogic Server applications.

ORACLE®

Oracle Fusion Middleware Using ActiveCache, 11g Release 1 (10.3.3)

E16517-01

Primary Author: Thomas Pfaeffle

Contributing Author: Ballav Bihani, Torkel Dominique, James Kirsch, Adam Leftik, Rosemary Marano, Lenny Phan

Contents

Preface

This preface describes the document accessibility features and conventions used in this guide—*Using ActiveCache*.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

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.

Convention	Meaning
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction and Roadmap

The following sections describe the contents and organization of this guide—*Using ActiveCache*:

- Section 1.1, "Document Scope and Audience"
- Section 1.2, "Guide to This Document"
- Section 1.3, "Related Documentation"
- Section 1.4, "New and Changed Features in This Release"

1.1 Document Scope and Audience

This document is a resource for:

- Application developers who want to develop and configure applications to use the ActiveCache features of Coherence data caches, Coherence*Web session management, and TopLink Grid.
- Administrators who configure, manage, and monitor ActiveCache features, such as Coherence clusters and resources.

1.2 Guide to This Document

- This chapter, Chapter 1, "Introduction and Roadmap," describes the organization of this document.
- Chapter 2, "Overview," provides an overview of ActiveCache features.
- Chapter 3, "Developing Applications for ActiveCache," explains how to use ActiveCache with applications running on WebLogic Server.
- Chapter 4, "Enabling State Session Persistence," describes how to use ActiveCache with Coherence*Web to provide HTTP session state persistence and management.
- Chapter 5, "Accessing and Retrieving Relational Data," describes how to use ActiveCache with TopLink Grid’s relational-to-object mapping capabilities to cache relational data.
- Chapter 6, "An ActiveCache Example," provides the steps for using ActiveCache to cache session information for Web applications deployed across WebLogic Server instances.
- Chapter , "Glossary," describes important, frequently referred to files.

1.3 Related Documentation

For additional information, see the following Coherence and WebLogic Server documents:

Coherence

- *Getting Started for Oracle Coherence*
- *Developer's Guide for Oracle Coherence*
- *Client Guide for Oracle Coherence*
- *Tutorial for Oracle Coherence*
- *User's Guide for Oracle Coherence*Web*

WebLogic Server

- *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*
- *Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server*
- *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*

1.4 New and Changed Features in This Release

ActiveCache is a new feature in this release of WebLogic Server. For a comprehensive listing of other new WebLogic Server features introduced in this release, see *Oracle Fusion Middleware What's New in Oracle WebLogic Server*.

2 Overview

The current release of WebLogic Server includes features that allow deployed applications to easily use Coherence data caches. These features allow WebLogic Server to seamlessly incorporate Coherence*Web for session management and TopLink Grid as an object-to-relational persistence framework. Collectively, these features are referred to as *ActiveCache*. ActiveCache is employed by applications running on WebLogic Server, and combines the following functionality of Coherence clusters, Coherence*Web, and TopLink Grid:

- Reliably stores and distributes serializable Java Objects.
- ActiveCache is fully compatible with the Coherence*Web HTTP Session Management Module. Coherence*Web allows the data cache to enable WebLogic Server memory resources.
- The addition of TopLink Grid gives ActiveCache the ability to scale out JPA applications.
- Manages the life cycle of stored objects.
- Provides serialization options which reduce heap requirements and the computational cost of deserializing session state each time it is accessed.
- Dynamically repartitions data, enabling optimal data distribution.
- Provides near caching, which keeps frequently read data on the heap.
- Provides direct access by applications to data caches. Applications that run on WebLogic Server can use either resource injection or component-based JNDI lookup to directly access data caches.
- Configure Coherence Cluster attributes using the WebLogic Server Administration Console and WLST.
- Monitor cluster-related properties from the WebLogic Server Administration Console. For example, you can use the Administration Console to monitor the cluster size, names and identifiers of cluster members, the license mode, and the cluster version.

ActiveCache provides replicated and distributed caching services that can make an application's data available to all servers in a Coherence data cluster.

ActiveCache gives applications running on WebLogic Server the ability to directly access Coherence data caches. The addition of Coherence data clusters to WebLogic Server instances enables you to create a data tier dedicated to caching application data and storing replicated session state. This is separate from the application tier—the WebLogic Server instances dedicated to running the application. ActiveCache technology allows the application tier to efficiently communicate with the data tier and cache data in it.

See Chapter 3, "Developing Applications for ActiveCache" for more information.

2.1 Accessing Data Caches from WebLogic Server Applications

ActiveCache can be employed for several different combinations of application and data tiers, or *cluster topologies*. For example, one topology is where all WebLogic Server instances employing Coherence (also known as *Coherence nodes*) are configured to store data. The applications and the data caches are collocated, and there is no separate data tier. Each Coherence node can serve requests and cache data.

Another possible topology is where there is a mixture of storage-enabled and storage-disabled ActiveCache nodes. In this topology, the storage-enabled nodes act as a data tier, while the storage-disabled nodes run the applications and serve requests. This topology creates a separate data tier which is isolated from non-cache application faults. All of the nodes in this topology can also be managed and monitored by WebLogic Server tools.

A third topology consists of storage-disabled WebLogic Server nodes and stand-alone Coherence caches. This topology also creates a separate data tier which is isolated from non-cache application faults. However, unlike the first two topologies, the data caches do not incur any costs related to the ActiveCache nodes' use of heap or server startup time.

Applications using ActiveCache can easily access the data cache. ActiveCache provides a `@Resource` annotation that allows a Coherence `NamedCache` cache object to be identified and dynamically injected into a servlet or EJB. As an alternative to resource injection, applications using ActiveCache can use a component-based JNDI tree to look up the `NamedCache`.

Cache services are classloader-scoped, and can be visible at an *application server scope*: caches will be visible globally to all applications deployed on the server, *EAR scope*: caches will be visible to all modules in the EAR, or *WAR scope*: caches will be visible to the individual modules only.

Scoping is determined by where you store the cache configuration file. The cache configuration file is where you define the cache object and cache types, then map cache names and name patterns to the cache types.

Like the cache configuration, cluster nodes are also classloader-scoped. Scoping is determined by where you store the `coherence.jar` and `active-cache.jar` files. Like cache services, cluster nodes can be *application server-scoped*: the entire JVM acts as a single Coherence cluster node, *EAR-scoped*: each application can be a Coherence cluster node, or *WAR-scoped*: each Web module within an application can be a Coherence cluster node.

ActiveCache enables you to display cluster-related properties in the WebLogic Server Administration Console. You can configure or reset some of these properties by using the Administration Console, WLST, or by importing a `tangosol-coherence-override.xml` configuration file.

See Chapter 4, "Enabling State Session Persistence" for more information.

2.2 Adding Session State Persistence and Management

The addition of Coherence*Web to ActiveCache enables you to provide Coherence cluster-based HTTP session state persistence to applications running on WebLogic Server. Coherence*Web enables HTTP session sharing and management across different Web applications, domains and heterogeneous application servers. Session data can be stored in data caches outside of the application server, thus freeing application server heap space and enabling server restarts without losing session data.

See the *User's Guide for Oracle Coherence*Web* for information on using Coherence*Web with WebLogic Server applications.

2.3 Accessing JPA Entities in the Data Cache

TopLink Grid's relational-to-object mapping capabilities allow ActiveCache to cache relational data. The Coherence data caches can cache copies of database queries and result sets. With this feature, database access occurs only when no cached copy of the required data is available in the data cache, or when the application performs a create, update, or delete operation that must be persisted to the database. This added optimization provides improved scalability and performance to the system.

TopLink Grid allows JPA Entity caching. This lets you support very large, shared grid caches that span cluster nodes. Calls for Entities cached in ActiveCache result in a get on the associated data cache. If the data cache does not contain the object, then the database is queried.

TopLink Grid enables you to direct queries to ActiveCache. If the desired query result is not found in the cache, it can be read from the database and then placed in the cache, making it available for subsequent queries. ActiveCache's ability to manage very large numbers of objects increases the likelihood of a result found in the cache, as reads in one cluster member become immediately available to others.

Writing Entities to the database is also made possible by TopLink Grid. Applications can directly write Entities to the database, then put them into the data cache (so that it reflects the database state), or put Entities into the data cache, then have the data cache write them to the database.

See Chapter 5, "Accessing and Retrieving Relational Data" for more information.

3

Developing Applications for ActiveCache

All of the files required by ActiveCache are installed automatically as part of the Oracle WebLogic Server **Typical** (default) configuration. The default root directory for the installation is `C:\Oracle\Middleware`. WebLogic Server is installed in `C:\Oracle\Middleware\wlserver_10.3` and Coherence is installed in `C:\Oracle\Middleware\coherence_3.5`.

The default installation includes all of the files that WebLogic Server needs to work with Coherence, Coherence*Web, and TopLink Grid. For a description of the installed files that are referred to frequently in this book, see the "Glossary".

3.1 Developing Applications to Use ActiveCache—Main Steps

The following steps summarize the procedure for using Coherence caches with applications running on WebLogic Server.

1. Choose the cluster topology on which your applications will run. You can decide to make WebLogic Servers data members of the Coherence cache, or just clients. See "Choose the ActiveCache Deployment Topology" on page 3-2.
2. Specify the configuration for the Coherence caches that your applications will use. See "Create and Configure a Data Cache" on page 3-2.
3. Add code in your Web application to access the Coherence caches. You can use either JNDI lookup or resource injection to access a Coherence `NamedCache` cache object. See "Access the Data Cache from your Application Code" on page 3-3.
4. Store the cache configuration file with the application. Where you store the file depends on how you want the caches to be visible to the deployed applications. See "Locate the Cache Configuration File" on page 3-4.
5. Determine how the cache server will access the cache configuration file when it starts. See "Access the Cache Configuration on Cache Server Startup" on page 3-5.
6. Cluster nodes are classloader-scoped. Where you deploy `coherence.jar` in the classloader hierarchy determines how cluster membership is handled. See "Package Applications and Configure Cluster Nodes" on page 3-6.
7. Adjust preconfigured cluster values for your deployed applications, if necessary. You can use WLST or the WebLogic Server Administration Console to configure some cluster-related values. See "Create and Configure Coherence Clusters" on page 3-9.
8. Start the cache servers. See "Start a Cache Server" on page 3-12.
9. Use one of the several methods to start WebLogic Server. See "Start WebLogic Server" on page 3-14.

10. Monitor the runtime status of the Coherence cluster from the WebLogic Server Administration Console. See "Monitor Coherence Cluster Properties" on page 3-14.

3.2 Choose the ActiveCache Deployment Topology

A cluster is used to harness multiple computers to store and manage data. Usually, this is for reliability and scalability purposes. One of the primary uses of Coherence is to cluster an application's objects and data. In the simplest sense, all of the data that is inserted into Coherence data caches is accessible by all servers in the application cluster that share the same cache configuration.

Two different Coherence cluster topologies can be formed by mixing WebLogic Servers and stand-alone Coherence cache servers. Here, cache servers are defined as Coherence data servers running on JVM instances dedicated to maintaining data (such as serialized session state data).

- In the *In-Process* topology, all WebLogic Servers (employing ActiveCache) in the cluster are *storage-enabled*. In this case, storage-enabled means that these servers will provide cache storage and backup storage. you do not have to create a separate data tier.

  This topology is not recommended for production use. This topology is supported mainly for development and testing. By storing the session data in-process with the application server, this topology is very easy to get up and running quickly for smoke tests, development and testing.

  Note: There are different default settings for local storage for Distributed caches on WebLogic Server, depending on whether you are employing Coherence*Web. For WebLogic Server, local storage is enabled by default for Distributed caches. However, when using Coherence*Web on WebLogic Server, local storage is disabled by default. In this case, you must create a separate data tier of stand-alone Coherence caches.

- In the *Out-of-Process* topology, use the stand-alone Coherence cache servers to host the data. Configure the WebLogic Servers to be storage-disabled so they can be used to serve requests. This topology creates a true, separate data tier, and further reduces overhead for the WebLogic Servers that are processing requests.
- The *WebLogic Out-Of-Process* topology is a slight variation on the Out-of-Process topology. In this topology WebLogic Server instances replace storage-enabled cache servers. This enables you to manage the lifecycle of the storage-enabled members. The advantage of this topology is that requests and data are segregated to their own servers. Latency for processing requests is reduced. Both storage-enabled and -disabled servers can be managed by WebLogic Sever management tools.

  Note: For more information on the In-Process and Out-of-Process deployment topologies, see *Deployment Topologies* in the *User's Guide for Oracle Coherence*Web.*

3.3 Create and Configure a Data Cache

ActiveCache can be configured to use any of the cache types supported by Oracle Coherence. An in-depth discussion on Coherence caches and their configuration is beyond the scope of this book. For information on working with Coherence caches and integrating them into your applications, see *Create and Use Coherence Caches* in the *Developer's Guide for Oracle Coherence*.

3.4 Access the Data Cache from your Application Code

Applications that run on WebLogic Server 11gR1 (10.3.3) or later can use ActiveCache to access a data cache. The data cache is represented by the Coherence `NamedCache` cache object. This object is designed to hold resources that are shared among members of a cluster. These resources are managed in memory, and are typically composed of data that is also stored persistently in a database, or data that has been assembled or calculated. Thus, these resources are referred to as *cached*.

Your application can obtain a `NamedCache` either by resource injection or by lookup in a component-scoped JNDI resource tree. The lookup technique can be used in EJBs, servlets, or JSPs. The resource injection technique can be used only by servlets or EJBs.

Note: It is not recommended that you store remote EJB references in Coherence named caches, nor should you store them in Coherence*Web-backed HTTP sessions.

To Obtain the NamedCache by Resource Injection

A `@Resource` annotation can be used in a servlet or an EJB to dynamically inject the `NamedCache`. This annotation cannot be used in a JSP. The name of the cache used in the annotation must be defined in the Coherence cache configuration file.

Example 3–1 illustrates a resource injection of the `NamedCache` `myCache`.

Example 3–1 Obtaining a NamedCache by Resource Injection

```
...
@Resource(mappedName="myCache")
com.tangosol.net.NamedCache nc;
...
```

To Obtain the NamedCache by JNDI Lookup

A component-scoped JNDI tree can be used in EJBs, servlets, or JSPs to reference the `NamedCache`.

To use a component-scoped JNDI lookup, define a `resource-ref` of type `com.tangosol.net.NamedCache` in either the `web.xml` or `ejb-jar.xml` file. Example 3–2 illustrates a `<resource-ref>` stanza that identifies `myCache` as the `NamedCache`.

Note: The `<res-auth>` and `<res-sharing-scope>` elements do not appear in the example. The `<res-auth>` element is ignored because currently no resource sign-on is performed to access data caches. The `<res-sharing-scope>` element is ignored because data caches are sharable by default and this behavior cannot be overridden.

Example 3–2 Defining a NamedCache as resource-ref for JNDI Lookup

```
...
<resource-ref>
```

```
  <res-ref-name>coherence/myCache</res-ref-name>
  <res-type>com.tangosol.net.NamedCache</res-type>
  <mapped-name>MyCache</mapped-name>
</resource-ref>
...
```

3.5 Locate the Cache Configuration File

The location where you store the cache configuration file determines the scope of the caches; that is, the visibility of the caches to the application. There are three options for cache visibility:

- application server-scoped—all deployed applications in a container become part of one cache service. Caches will be visible globally to all applications deployed on the server.
- EAR-scoped—all deployed applications within each EAR become part of one Coherence node. Caches will be visible to all modules in the EAR. For example, this could be a recommended deployment if all of the modules must share the same cache.
- WAR-scoped—each deployed Web application becomes its own Coherence node. Caches will be visible to the individual modules only. For example, this could be a recommended deployment for a stand-alone WAR deployment or stand-alone EJB deployment.

Note: The cache configuration must be consistent for both WebLogic Server instances and Coherence cache servers.

Table 3–1 Storage Locations for Cache Configuration File Based on Cache Scoping

For this cache scoping ...	Store the cache configuration file here	Comments
application server-scope	■ store the cache configuration file in the server's classpath	See the following section, "Access the Cache Configuration on Cache Server Startup" for more information.
application-scoped cache for an EAR file	■ JAR file in the EAR library directory ■ `APP-INF/classes` directory of EAR	Caches defined in `coherence-cache-config.xml` and placed at EAR level can be seen and shared by all modules in the EAR. Caches defined at EAR level will be visible to the individual applications within the EAR only, but they must have unique service names across all the EARs in the application. Also, if you define caches both at the EAR level and at the module level, then the cache, scheme, and service names must be unique across the EAR-level cache configuration and the module cache configuration.
Web-component-scoped cache in an EAR, or a stand-alone WAR deployment	■ JAR file in the `WEB-INF/lib` directory of a WAR file ■ `WEB-INF/classes` directory of a WAR file	Caches defined at module level will be visible to the individual modules only, but they must have unique service names across all the modules in the application. Also, if you define caches both at the EAR level and at the module level, then the cache, scheme, and service names must be unique across the EAR-level cache configuration and the module cache configuration.
stand-alone EJB deployment	■ EJB-JAR file	An EJB module in an EAR cannot have module-scoped caches—they can only be application-scoped.

3.6 Access the Cache Configuration on Cache Server Startup

The cache server must be able to access the cache configuration file on start-up. There are two ways to do this:

- Place the cache configuration file in the server's classpath, or
- Declare the cache configuration file location in the server start-up command with the `tangosol.coherence.cacheconfig` system property. For more information on this property, see the *Developer's Guide for Oracle Coherence*.

  Example 3–3 illustrates the `tangosol.coherence.cacheconfig` system property in a sample startup command.

Example 3–3 Declaring the Cache Configuration File in a Server Startup Command

```
java -server -Xms512m -Xmx512m
-cp <Coherence installation dir>/lib/coherence.jar:<Coherence installation
dir>/lib/coherence-web-spi.war -Dtangosol.coherence.management.remote=true
-Dtangosol.coherence.cacheconfig=WEB-INF/classes/coherence-cache-config.xml
-Dtangosol.coherence.session.localstorage=true com.tangosol.net.DefaultCacheServer
```

If you are working with two (or more) applications, it is possible that they could possibly have two (or more) different cache configurations. In this case, the cache configuration on the cache server must contain the union of these configurations. This allows the applications to be supported in the same cache cluster. Note that this is only valid for the stand-alone cache server topology.

3.7 Package Applications and Configure Cluster Nodes

Coherence cluster nodes are classloader-scoped. Cluster nodes can be *application server-scoped*—the entire JVM acts as a single Coherence cluster node, *EAR-scoped*—each application can be a Coherence cluster node, or *WAR-scoped*—each Web module within an application can be a Coherence cluster node.

The packing and configuration options for these three scenarios are described in the following sections:

- Packaging Applications and Configuring Application Server-Scoped Cluster Nodes
- Packaging Applications and Configuring EAR-Scoped Cluster Nodes
- Packaging Applications and Configuring WAR-Scoped Cluster Nodes

3.7.1 Packaging Applications and Configuring Application Server-Scoped Cluster Nodes

With this configuration, all deployed applications on the WebLogic Server instance that are accessing Coherence caches directly become part of one Coherence node. Caches will be visible to all applications deployed on the server. This configuration produces the smallest number of Coherence nodes in the cluster (one for each WebLogic Server JVM instance).

Since the Coherence library is deployed in the container's classpath, only one copy of the Coherence classes will be loaded into the JVM, thus minimizing resource utilization. On the other hand, since all applications are using the same cluster node, all applications will be affected if one application misbehaves.

To Use Coherence Data Caches with Application Server-Scoped Cluster Nodes

1. Edit your WebLogic Server system classpath to include `coherence.jar` and *`WL_HOME`*`/common/deployable-libraries/active-cache.jar` in the system classpath. The `active-cache.jar` should be referenced only from the `deployable-libraries` folder in the system classpath and should not be copied to any other location.
2. (Optional) If you must configure Coherence cluster properties, create a `CoherenceClusterSystemResourceMBean` and reference it in the `ServerMBean`.

   You can use WLST to reference the MBean. See `createServerScopedCoherenceSystemResource` in Example 3–9.

3.7.2 Packaging Applications and Configuring EAR-Scoped Cluster Nodes

With this configuration, all deployed applications within each EAR become part of one Coherence node. Caches will be visible to all modules in the EAR. For example, this could be a recommended deployment if all the modules must share the same Coherence node. It can also be a recommended configuration if you plan on deploying only one EAR to an application server.

This configuration produces the next smallest number of Coherence nodes in the cluster (one for each deployed EAR). Since the Coherence library is deployed in the application's classpath, only one copy of the Coherence classes is loaded for each EAR.

Since all Web applications in the EAR use the same cluster node, all Web applications in the EAR will be affected if one of them misbehaves. EAR-scoped cluster nodes reduce the deployment effort as no changes to the application server classpath are required.

To Use Coherence Caches with EAR-Scoped Cluster Nodes

1. Use either of the following methods to deploy the `coherence.jar` and `active-cache.jar` files as shared libraries to all of the target servers where the application will be deployed.

   - Use the WebLogic Server Administration Console to deploy `coherence.jar` and `active-cache.jar` as shared libraries. See "Install a Java EE Library" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

     As an alternative to the Administration Console, you can also deploy on the command line. The following are sample deployment commands:

     ```
     java weblogic.Deployer -username <> -password <> -adminurl <> -deploy
     coherence.jar -name coherence -library -targets <>

     java weblogic.Deployer -username <> -password <> -adminurl <> -deploy
     active-cache.jar -name active-cache -library -targets <>
     ```

   - Copy `coherence.jar` and `active-cache.jar` to the EAR's `APP-INF/lib` folder of the application. However, the preferred way is to deploy them as shared libraries.

2. Refer to the `coherence.jar` and `active-cache.jar` files as libraries. Example 3–4 illustrates a sample `weblogic-application.xml` configuration.

Example 3–4 coherence and active-cache JARs Referenced in the weblogic-application.xml File

```
<weblogic-application>
...
  <library-ref>
    <library-name>coherence</library-name>
  </library-ref>
   ...
  <library-ref>
    <library-name>active-cache</library-name>
  </library-ref>
...
</weblogic-application>
```

3. (Optional) If you must configure Coherence cluster properties, create a `CoherenceClusterSystemResourceMBean` and reference it as a `coherence-cluster-ref` element in `weblogic-application.xml` file. This element allows the applications to enroll in the Coherence cluster as specified by the `CoherenceClusterSystemResourceMBean` attributes.

   Example 3–5 illustrates a sample configuration. The `myCoherenceCluster` MBean in the example is of type `CoherenceClusterSystemResourceMBean`.

Example 3–5 coherence-cluster-ref Element for EAR-Scoped Cluster Nodes

```
<weblogic-application>
...
  <coherence-cluster-ref>
    <coherence-cluster-name>
     myCoherenceCluster
    </coherence-cluster-name>
  </coherence-cluster-ref>
...
</weblogic-application>
```

To Define a Filtering Classloader for Application-Scoped Cluster Nodes

If the `coherence.jar` is placed in the application server classpath, you can still configure an EAR-scoped cluster node by defining a filtering classloader. This is described in the following steps:

1. Place `coherence.jar` in the application classpath.
2. Configure a filtering classloader in the EAR file.

   The filtering classloader is defined in the `<prefer-application-packages>` stanza of the `weblogic-application.xml` file. Example 3–6 illustrates a sample filtering classloader configuration. The `package-name` elements indicate the package names of the classes in the `coherence.jar` and `active-cache.jar`.

Example 3–6 Configuring a Filtering Classloader

```
<weblogic-application>
...
 <prefer-application-packages>
   <package-name>com.tangosol.*</package-name>
   <package-name>weblogic.coherence.service.*</package-name>
   <package-name>com.oracle.coherence.common.*</package-name>
 </prefer-application-packages>
...
</weblogic-application>
```

3.7.3 Packaging Applications and Configuring WAR-Scoped Cluster Nodes

With this configuration, or if only one application wants to use Coherence, each deployed Web application becomes its own Coherence node. Caches will be visible to the individual modules only. For example, this could be a recommended deployment for a stand-alone WAR deployment or stand-alone EJB deployment.

If you are deploying multiple WAR files, note that this configuration produces the largest number of Coherence nodes in the cluster—one for each deployed WAR file that uses `coherence.jar`. It also results in the largest resource utilization of the three configurations—one copy of the Coherence classes are loaded for each deployed WAR. On the other hand, since each deployed Web application is its own cluster node, Web applications are completely isolated from other potentially misbehaving Web applications.

Note: A Web module within an EAR can have a module-scoped Coherence node but an EJB module within an EAR can only have an application-scoped Coherence cluster node.

To Use Coherence Caches with WAR-Scoped Cluster Nodes

1. Use the WebLogic Server Administration Console to deploy `coherence.jar` and `active-cache.jar` as shared libraries to all of the target servers where the application will be deployed. See "Install a Java EE Library" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

   As an alternative to the Administration Console, you can also deploy on the command line. The following are sample deployment commands:

   ```
   java weblogic.Deployer -username <> -password <> -adminurl <> -deploy
   coherence.jar -name coherence -library -targets <>

   java weblogic.Deployer -username <> -password <> -adminurl <> -deploy
   active-cache.jar -name active-cache -library -targets <>
   ```

2. Import `coherence.jar` and `active-cache.jar` as optional packages in the `manifest.mf` file of each module that will be using Coherence.

   As an alternative to using the manifest file, copy `coherence.jar` and `active-cache.jar` to each WAR file's `WEB-INF/lib` directory.

   Example 3–7 illustrates the contents of a sample `manifest.mf` file.

Example 3–7 Referencing coherence and active-cache Jar Files the manifest.mf File

```
Manifest-Version: 1.0
Extension-List: coherence active-cache
coherence-Extension-Name: coherence
active-cache-Extension-Name: active-cache
```

3. (Optional) If you must configure Coherence cluster properties, create a `CoherenceClusterSystemResourceMBean` and reference it as a `coherence-cluster-ref` element in `weblogic.xml` or `weblogic-ejb-jar.xml` file.

   Example 3–8 illustrates a sample configuration for WAR-scoped cluster nodes in the `weblogic.xml` file. The `myCoherenceCluster` MBean is of type `CoherenceClusterSystemResourceMBean`.

Example 3–8 coherence-cluster-ref Element for WAR-Scoped Cluster Nodes

```
<weblogic-web-app>
...
  <coherence-cluster-ref>
    <coherence-cluster-name>
     myCoherenceCluster
    </coherence-cluster-name>
  </coherence-cluster-ref>
...
</weblogic-web-app>
```

3.8 Create and Configure Coherence Clusters

Using WLST or the Administration Console, you can create a Coherence cluster configuration and select WebLogic Server instances or clusters on which the cluster configuration is accessible.

The `createCoherenceClusterMBean.py` WLST script shown in Example 3–9 configures three Coherence clusters, including a server-scoped configuration that gets deployed to the Administration Server (`myserver`).

Example 3–9 createCoherenceClusterMBean.py

```
from java.util import *
from javax.management import *
from java.lang import *
import javax.management.Attribute

"""
This script configures a Coherence Cluster System Resource MBean and deploys it
to the admin server
"""

def createCoherenceSystemResource(wlsTargetNames, coherenceClusterSourceName):

       name = coherenceClusterSourceName
       # start creation
       print 'Creating CoherenceClusterSystemResource with name '+ name
       cohSR = create(name,"CoherenceClusterSystemResource")
       cohBean = cohSR.getCoherenceClusterResource()
       cohCluster = cohBean.getCoherenceClusterParams()
         cohCluster.setUnicastListenAddress("localhost")
         cohCluster.setUnicastListenPort(7001)
       cohCluster.setUnicastPortAutoAdjust(true)
       # you can set up the multicast port or define WKAs
       cohCluster.setMulticastListenAddress("231.1.1.1")
       cohCluster.setMulticastListenPort(8001)
       cohCluster.setTimeToLive(5)

       for wlsTargetName in wlsTargetNames:
          cd("Servers/"+wlsTargetName)
          target = cmo
          cohSR.addTarget(target)
          cd("../..")

def createServerScopedCoherenceSystemResource(wlsTargetNames,
coherenceClusterSourceName):

       name = coherenceClusterSourceName
       # start creation
       print 'Creating CoherenceClusterSystemResource with name '+ name
       cohSR = create(name,"CoherenceClusterSystemResource")
       cohBean = cohSR.getCoherenceClusterResource()
       cohCluster = cohBean.getCoherenceClusterParams()
         cohCluster.setUnicastListenAddress("localhost")
         cohCluster.setUnicastListenPort(7002)
       cohCluster.setUnicastPortAutoAdjust(true)
       # you can set up the multicast port or define WKAs
       cohWKAs = cohCluster.getCoherenceClusterWellKnownAddresses()
       cohWKA = cohWKAs.createCoherenceClusterWellKnownAddress("wka1")
       cohWKA.setName("wka1")
       cohWKA.setListenAddress("localhost")
       cohWKA.setListenPort(9001)

       for wlsTargetName in wlsTargetNames:
          cd("Servers/"+wlsTargetName)
          target = cmo
          cohSR.addTarget(target)
            print cmo
            serverBean = cmo
            serverBean.setCoherenceClusterSystemResource(cohSR)
          cd("../..")
```

```
def createCustomCoherenceSystemResource(wlsTargetNames,
coherenceClusterSourceName, tangosolOverrideFile):

       name = coherenceClusterSourceName
       # start creation
       cohSR = getMBean("/CoherenceClusterSystemResources/"+name)
       if cohSR == None:
         print 'Creating CoherenceClusterSystemResource with name '+ name
         cohSR = create(name,"CoherenceClusterSystemResource")
         cohSR.importCustomClusterConfigurationFile(tangosolOverrideFile)

       for wlsTargetName in wlsTargetNames:
          cd("Servers/"+wlsTargetName)
          target = cmo
          cohSR.addTarget(target)
          cd("../..")

props = System.getProperties()
ADMIN_NAME = props.getProperty("admin.username")
ADMIN_PASSWORD = props.getProperty("admin.password")
ADMIN_HOST = props.getProperty("admin.host")
ADMIN_PORT = props.getProperty("admin.port")
ADMIN_URL = "t3://"+ADMIN_HOST+":"+ADMIN_PORT

TANGOSOL_OVERRIDE = props.getProperty("tangosol-override")

TARGETS = [ 'myserver' ]

print "Starting the script ..."
try :
       connect(ADMIN_NAME, ADMIN_PASSWORD, ADMIN_URL)
       edit()
       startEdit()
       createCoherenceSystemResource(TARGETS, 'cohSystemResource')
       createServerScopedCoherenceSystemResource(TARGETS,
'serverScopedCohSystemResource')
       createCustomCoherenceSystemResource(TARGETS,
'customCohSystemResource',TANGOSOL_OVERRIDE)
       save()
       activate(block="true")
       disconnect()
       print 'Done configuring the Coherence Cluster System Resources'
except:
       dumpStack()
       undo('true','y')
```

For Administration Console procedures, see "Configure Coherence" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

Cluster-related values are stored in a descriptor file in the WebLogic Server configuration repository:

`<`*`domain-home`*`>/config/coherence/CoherenceClusterSystemResourceName/CoherenceClusterSystemResourceName-####-coherence.xml`.

For example, `C:\Oracle\Middleware\user_projects\domains\base_domain\config\coherence\cohSystemResource\cohSystemResource-0759-coherence.xml`.

Alternatively, you can configure properties that are not specified for the cluster by configuring them in a custom configuration file, for example, `tangosol-coherence-override.xml`, shown in Example 3–10.

Example 3–10 tangosol-coherence-override.xml

```
<?xml version='1.0'?>
<!--
This operational configuration override file is set up for use with Coherence in
a development mode.
-->
<coherence xml-override="/tangosol-coherence-override.xml">
 <cluster-config>
  <multicast-listener>
    <time-to-live system-property="tangosol.coherence.ttl">4</time-to-live>
    <join-timeout-milliseconds>3000</join-timeout-milliseconds>
  </multicast-listener>

  <packet-publisher>
   <packet-delivery>
    <timeout-milliseconds>30000</timeout-milliseconds>
   </packet-delivery>
  </packet-publisher>
 </cluster-config>

 <logging-config>
  <severity-level
system-property="tangosol.coherence.log.level">5</severity-level>
  <character-limit
system-property="tangosol.coherence.log.limit">0</character-limit>
 </logging-config>
</coherence>
```

Use WLST to import the custom cluster configuration file (also shown in Example 3–9, see `createCustomCoherenceSystemResource`) or the WebLogic Server Administration Console. See "Import a custom cluster configuration" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

Note: If you specify cluster-related properties by importing a custom configuration file, the properties specified in the file must not be the same properties that were specified using WLST or the WebLogic Server Administration Console.

3.9 Start a Cache Server

A Coherence data node (also known as a Cache Server) is a dedicated JVM that is responsible for storing and managing all cached data. The senior node (which is the first node) in a Coherence data cluster can take several seconds to start; the start up time required by subsequent nodes is minimal. Thus, to optimize performance, you should always start a Coherence data node before starting a WebLogic Server instance. This will ensure that there is minimal (measured in milliseconds) startup time for applications using Coherence. Any additional Web applications that use Coherence are guaranteed not to be the senior data member, so they will have minimal impact on WebLogic Server startup.

Note: Whether you start the cache servers first or the WebLogic Server instances first, depends on the server topology you are employing.

- If you are using In-Process topology (all storage-enabled WebLogic Server instances employing ActiveCache), then it does not matter if you start the cache servers first or WebLogic Server instances first.
- If you are using Out-of-Process topology (storage-disabled WebLogic server instances and stand-alone Coherence cache servers), then start the cache servers first, followed by the WebLogic Server instances.
- If you are using WebLogic Out-of-Process topology, your topology is a mix of storage-enabled and storage-disabled WebLogic Server instances. Start the storage-enabled instances first, followed by the storage-disabled instances.

To Start a Stand-Alone Coherence Data Node

1. Create a script for starting a Coherence data node. The following is a very simple example of a script that starts a storage-enabled cache server to use with ActiveCache. This example assumes that you are using a Sun JVM. See *JVM Tuning* in the *Developer's Guide for Oracle Coherence* for more information.

   ```
   java -server -Xms512m -Xmx512m
   -cp <Coherence installation dir>/lib/coherence.jar:<Coherence installation
   dir>/lib/coherence-web-spi.war -Dtangosol.coherence.management.remote=true
   -Dtangosol.coherence.cacheconfig=WEB-INF/classes/cache_configuration_file
   -Dtangosol.coherence.session.localstorage=true
   com.tangosol.net.DefaultCacheServer
   ```

   In this example, `cache_configuration_file` refers to the cache configuration in the `coherence-cache-config.xml` file. The cache configuration defined for the cache server must be the same as the configuration defined for the application servers which run on the same ActiveCache cluster.

   If you run Coherence*Web for session management, then the cache configuration information should be merged with the session configuration contained in the `session-cache-config.xml` file. Similarly, the cache and session configuration must be consistent across WebLogic Servers and cache servers.

2. Start one or more Coherence data nodes using the script described in the previous step.

To Start a Storage-Enabled or -Disabled WebLogic Server Instance

By default, an ActiveCache-enabled WebLogic Server instance starts in storage-disabled mode.

To start the WebLogic Server instance in storage-enabled mode, include the command line property `-Dtangosol.coherence.session.localstorage=true` in the server startup command.

For more information on working with WebLogic Server through the command line, see the *weblogic.Server Command-Line Reference* chapter in the *Oracle Fusion Middleware Command Reference for Oracle WebLogic Server.*

3.10 Start WebLogic Server

WebLogic Server provides several ways to start and stop server instances. The method that you choose depends on whether you prefer using the Administration Console or a command-line interface, and on whether you are using Node Manager to manage the server's life cycle. For detailed information, see "Starting and Stopping Servers" in *Oracle Fusion Middleware Managing Server Startup and Shutdown for Oracle WebLogic Server*. For a quick reference, see "Starting and Stopping Servers: Quick Reference."

3.11 Monitor Coherence Cluster Properties

The WebLogic Server Administration Console displays run-time monitoring information for Coherence clusters associated with a particular application or module, such as cluster size, members, and version. For more information, see "Monitoring Coherence Clusters" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

4 Enabling State Session Persistence

ActiveCache can use Coherence*Web to provide HTTP session state persistence and management. Coherence*Web is a session management module that uses Coherence for storing and managing session data. 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.
- allows session sharing and management across Web applications, domains, and heterogeneous application servers.
- allows storage of session data outside of the Java EE application server, freeing application server heap space and enabling server restarts without session data loss.
- supports multiple advanced session models (that is, Monolithic, Traditional, and Split Session) which define how the session state is serialized/deserialized in the cluster.
- supports fine-grained session and session attribute scoping by way of pluggable policies.

For information on using Coherence*Web with WebLogic Server applications, see the *User's Guide for Oracle Coherence*Web*.

5 Accessing and Retrieving Relational Data

TopLink Grid marries the standardization and simplicity of application development using the Java Persistence API (JPA) with the scalability and distributed processing power of Oracle's Coherence Data Grid. Developers can leverage their investment in JPA and take advantage of the scalability of Coherence. Standard JPA applications interact directly with their primary data store, typically a relational database, but with TopLink Grid developers can choose to store some or all of their domain model in the Coherence data grid.

To Use TopLink-Grid with Application Server-Scoped Cluster Nodes

If you are using TopLink Grid to store JPA (Java Persistence API) Entities in Coherence caches, follow these steps:

1. Follow the instructions in "Packaging Applications and Configuring Application Server-Scoped Cluster Nodes" on page 3-6 to include `coherence.jar` and `active-cache.jar` in the system classpath.
2. Edit your WebLogic Server system classpath to include `toplink-grid.jar` in the system classpath.

To Use TopLink Grid with EAR-Scoped Cluster Nodes

If you are using TopLink Grid to store JPA (Java Persistence API) Entities in Coherence caches, follow these steps:

1. Follow the instructions in "Packaging Applications and Configuring EAR-Scoped Cluster Nodes" on page 3-6 to deploy the `coherence.jar` and `active-cache.jar` files as shared libraries.
2. Use either of the following methods to deploy `toplink-grid.jar` as a shared library.
   - Use the WebLogic Server Administration Console or the command line to deploy `toplink-grid.jar` as a shared library.

     ```
     java weblogic.Deployer -username <> -password <> -adminurl <> -deploy
     toplink-grid.jar -name toplink-grid -library -targets <>
     ```

     If you deploy `toplink-grid.jar` as a shared library, refer to it in the `weblogic-application.xml` file as a `library-ref`. Example 5–1 illustrates the `toplink-grid.jar` referenced in the `weblogic-application.xml` file.

Example 5–1 Reference to toplink-grid.jar in the weblogic-application.xml File

```
<weblogic-application>
...
```

```
  <library-ref>
    <library-name>coherence</library-name>
  </library-ref>
   ...
  <library-ref>
    <library-name>active-cache</library-name>
  </library-ref>
  <library-ref>
    <library-name>toplink-grid</library-name>
  </library-ref>
...
</weblogic-application>
```

- Copy `toplink-grid.jar` to the application EAR's `APP-INF/lib` folder. However, the preferred way is to deploy it as a shared library.

To Use TopLink Grid with WAR-Scoped Cluster Nodes

If you are using TopLink Grid to store JPA (Java Persistence API) Entities in Coherence caches, follow these steps:

1. Follow the instructions in "Packaging Applications and Configuring WAR-Scoped Cluster Nodes" on page 3-8 to deploy the `coherence.jar` and `active-cache.jar` files.
2. Use the WebLogic Server Administration Console or the command line to deploy `toplink-grid.jar`. The following is a sample command line:

   ```
   java weblogic.Deployer -username <> -password <> -adminurl <> -deploy
   toplink-grid.jar -name toplink-grid -library -targets <>
   ```

3. Import `toplink-grid.jar` as an optional package in the `manifest.mf` file of each module that will be using Coherence. As an alternative, you can copy it to each of the application WAR's `WEB-INF/lib` directories.

   Example 5–2 illustrates a sample manifest file.

Example 5–2   Manifest File with coherence, active-cache, and toplink-grid

```
Manifest-Version: 1.0
Extension-List: coherence active-cache toplink-grid
coherence-Extension-Name: coherence
active-cache-Extension-Name: active-cache
toplink-grid-Extension-Name: toplink-grid
```

6 An ActiveCache Example

The following example demonstrates how to use ActiveCache to cache session information for Web application instances that are deployed across WebLogic Server instances. To do this, you will create a Web application and deploy it to two server instances. The application is a simple counter that stores the current count as a session attribute. Coherence*Web automatically serializes and replicates the attribute across both server instances. A browser is used to access each application instance to demonstrate that the same session attribute is used among the instances.

The Coherence*Web module is included in the default installation of WebLogic Server 11gR1 (10.3.3). For more information, see the *User's Guide for Oracle Coherence*Web*.

6.1 ActiveCache Example—Main Steps

Follow these steps to complete the ActiveCache example.

1. Start a Cache Server
2. Configure and Start the WebLogic Server
3. Create a Machine
4. Create the WebLogic Servers
5. Create a Coherence Cluster
6. Deploy the Shared Library Files
7. Create the Counter Web Application
8. Deploy the Application
9. Start the Node Manager and the WebLogic Servers
10. Verify the Example

6.2 Start a Cache Server

Start a Coherence cache server. Example 6–1 illustrates a sample script to start the cache server. In this example, `tangosol.coherence.clusterport=7777` is the default multicast listen port of a Coherence cluster and `tangosol.coherence.clusteraddress=231.1.1.1` is the default multicast listen address.

Example 6–1 Script to Start the Cache Server

```
setlocal
```

```
    set COHERENCE_HOME=c:\oracle\product\coherence

set COH_OPTS=%COH_OPTS% -server -cp %COHERENCE_HOME%\lib\coherence.jar;%COHERENCE_
HOME%\lib\coherence-web-spi.war;
set COH_OPTS=%COH_OPTS% -Dtangosol.coherence.management.remote=true
-Dtangosol.coherence.cacheconfig=/WEB-INF/classes/session-cache-config.xml
-Dtangosol.coherence.distributed.localstorage=true
-Dtangosol.coherence.clusterport=7777
-Dtangosol.coherence.clusteraddress=231.1.1.1
-Dtangosol.coherence.session.localstorage=true

java %COH_OPTS% -Xms512m -Xmx512m com.tangosol.net.DefaultCacheServer

:exit
```

6.3 Configure and Start the WebLogic Server

This example requires a Coherence Cluster.

1. Run the Oracle WebLogic Configuration Wizard (**Start > All Programs > Oracle WebLogic > WebLogic Server 11gR1 > Tools > Configuration Wizard**) to create a WebLogic domain called `test_domain`.

   Before exiting the wizard, select the **Start Admin Server** check box, and click **Done**. The configuration wizard automatically starts the Administration Server.

2. Start the WebLogic Server Administration Console.

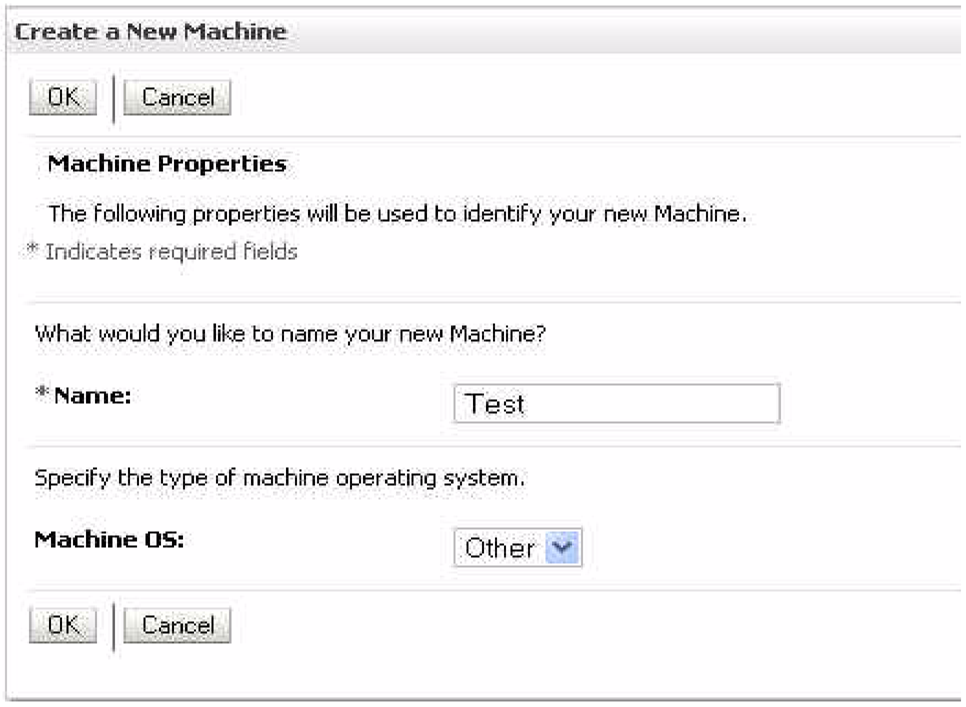
   From the browser, log in to the Oracle WebLogic Server Administration Console using the following URL:`http://`*`hostname`*`:7001/console`. The console starts, and the domain home page displays.

6.4 Create a Machine

Create a Machine on which to host WebLogic Server instances.

From the **Domain Structure** window, select **Environment > Machines**. Click **New**. The **Create a New Machine** page displays. Enter a name for the Machine (in this case, **Test**) and click **OK**.

Figure 6–1 Creating a New Machine



Create a New Machine

Machine Properties

The following properties will be used to identify your new Machine.

* Indicates required fields

What would you like to name your new Machine?

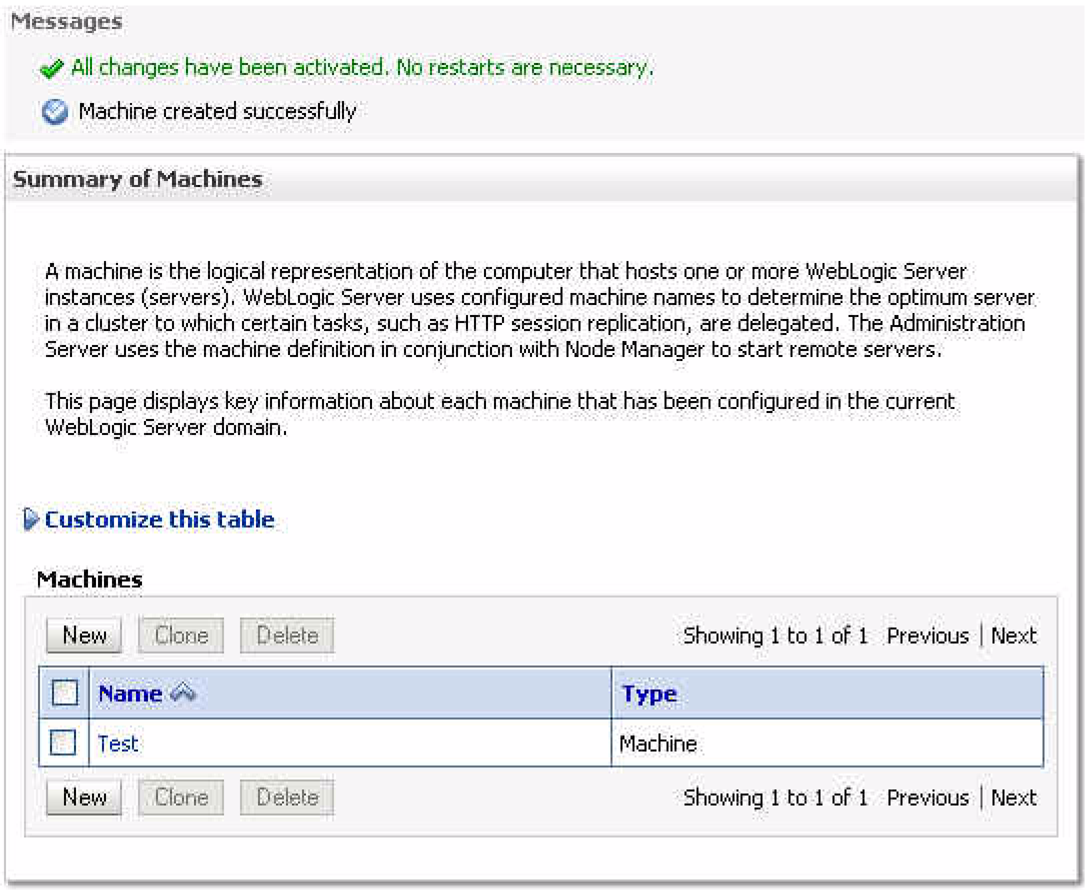
*** Name:** Test

Specify the type of machine operating system.

Machine OS: Other

The **Summary of Machines** page should look similar to Figure 6–2.

Figure 6–2 Summary of Machines


Messages

All changes have been activated. No restarts are necessary.

Machine created successfully

Summary of Machines

A machine is the logical representation of the computer that hosts one or more WebLogic Server instances (servers). WebLogic Server uses configured machine names to determine the optimum server in a cluster to which certain tasks, such as HTTP session replication, are delegated. The Administration Server uses the machine definition in conjunction with Node Manager to start remote servers.

This page displays key information about each machine that has been configured in the current WebLogic Server domain.

	Name	Type
☐	Test	Machine

6.5 Create the WebLogic Servers

Create two server instances associated with the Machine. The application will be deployed to these servers in a later step.

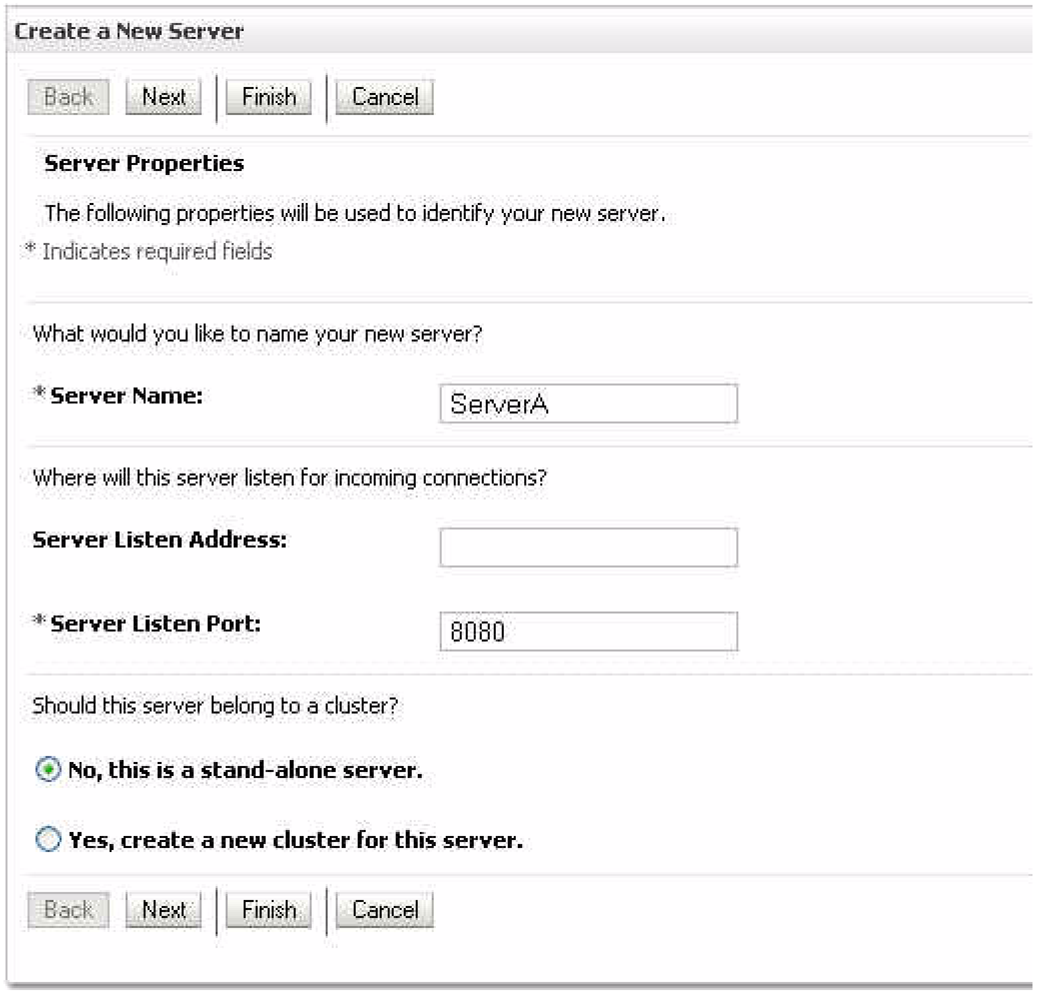
1. Click the name of the Machine in the **Summary of Machines** page to open the **Settings for** <*machine*> page. Click the **Servers** tab then **Add** to create a server.

2. Select **Create a new server and associate it with this machine** in the **Add a Server to Machine** page, and click **Next**.

3. Provide details about the server in the **Add a Server to Machine** page.

   Enter `ServerA` as the **Server Name** and `8080` as the **Server Listen Port**. Enter the appropriate value for the **Server Listen Address**. Click **Finish**.

Figure 6–3 Adding a Server to a Machine



Create a New Server

Server Properties

The following properties will be used to identify your new server.

* Indicates required fields

What would you like to name your new server?

* **Server Name:** ServerA

Where will this server listen for incoming connections?

Server Listen Address:

* **Server Listen Port:** 8080

Should this server belong to a cluster?

[✓ **No, this is a stand-alone server.**]

Yes, create a new cluster for this server.

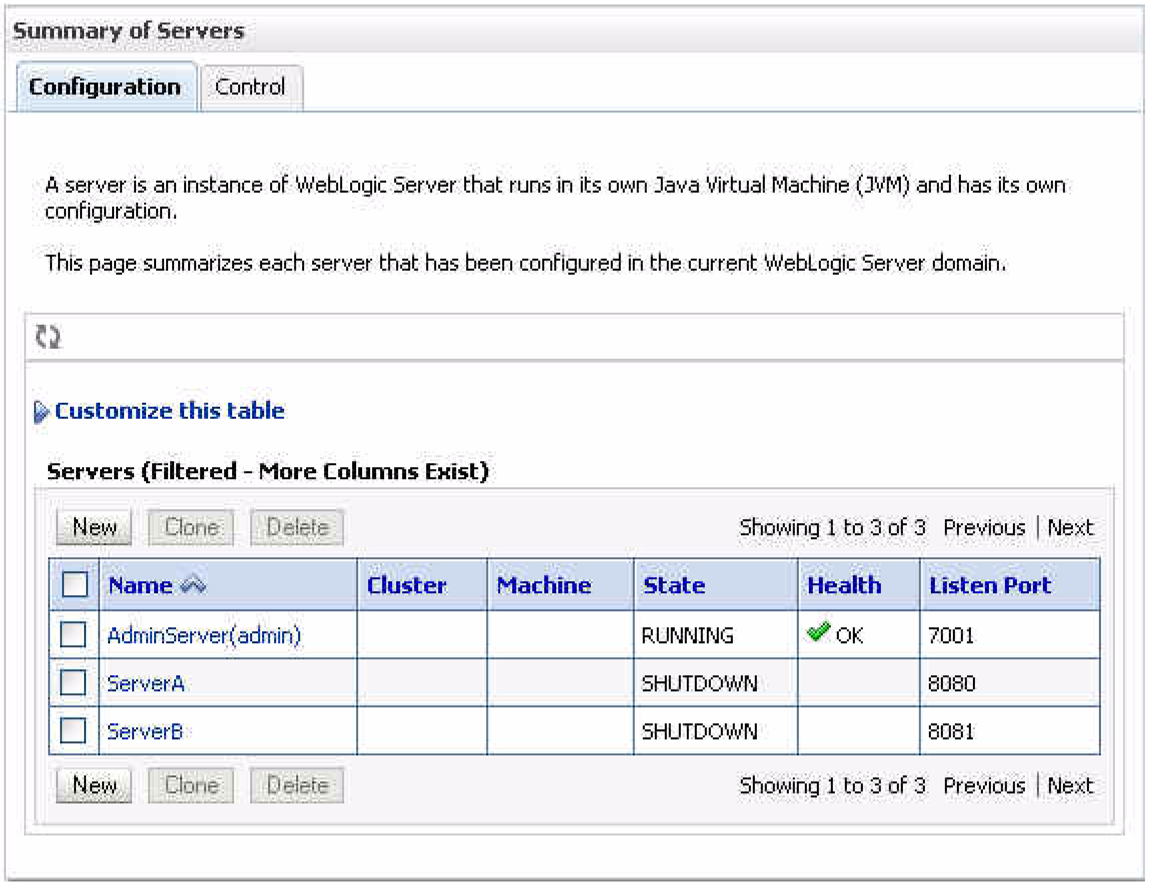
4. When you are returned to the **Settings for *machine*** page, repeat the previous three steps to create a second server.

   Enter `ServerB` as the **Server Name** and `8081` as the **Server Listen Port**. Enter the appropriate value for the **Server Listen Address**. Click **Finish**.

5. Expand **Environment** in the **Domain Structure** menu and click **Servers**.

   The **Summary of Servers** page displays and should be similar to Figure 6–4:

Figure 6–4 Summary of Servers Page



Summary of Servers

Configuration | Control

A server is an instance of WebLogic Server that runs in its own Java Virtual Machine (JVM) and has its own configuration.

This page summarizes each server that has been configured in the current WebLogic Server domain.

Customize this table

Servers (Filtered - More Columns Exist)

New | Clone | Delete — Showing 1 to 3 of 3 Previous | Next

	Name	Cluster	Machine	State	Health	Listen Port
☐	AdminServer(admin)			RUNNING	✓ OK	7001
☐	ServerA			SHUTDOWN		8080
☐	ServerB			SHUTDOWN		8081

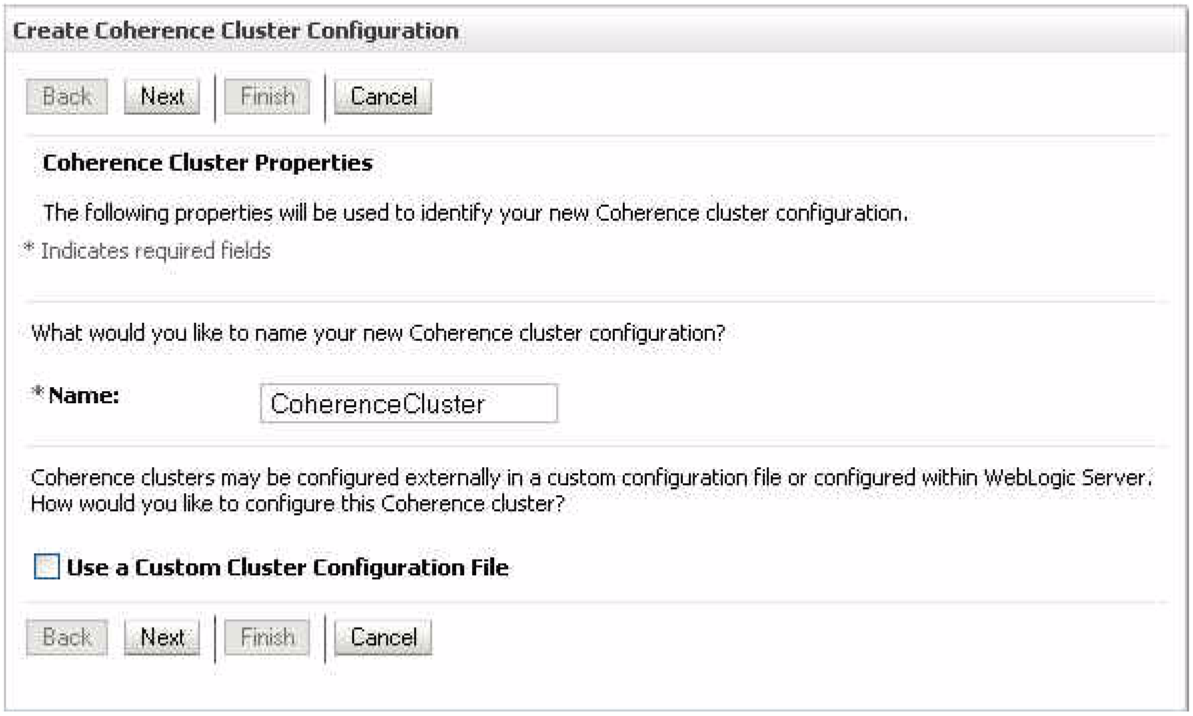
New | Clone | Delete — Showing 1 to 3 of 3 Previous | Next

6.6 Create a Coherence Cluster

Create a Coherence Cluster.

1. Click **Services** in the domain **Structure Window**. Then click **Coherence Clusters**. In the **Summary of Coherence Clusters** page, click **New**. In the **Create Coherence Cluster Configuration** page, enter `CoherenceCluster` in the **Name** field, then click **Next**.

Figure 6–5 Creating a Coherence Cluster



Create Coherence Cluster Configuration

Coherence Cluster Properties

The following properties will be used to identify your new Coherence cluster configuration.

* Indicates required fields

What would you like to name your new Coherence cluster configuration?

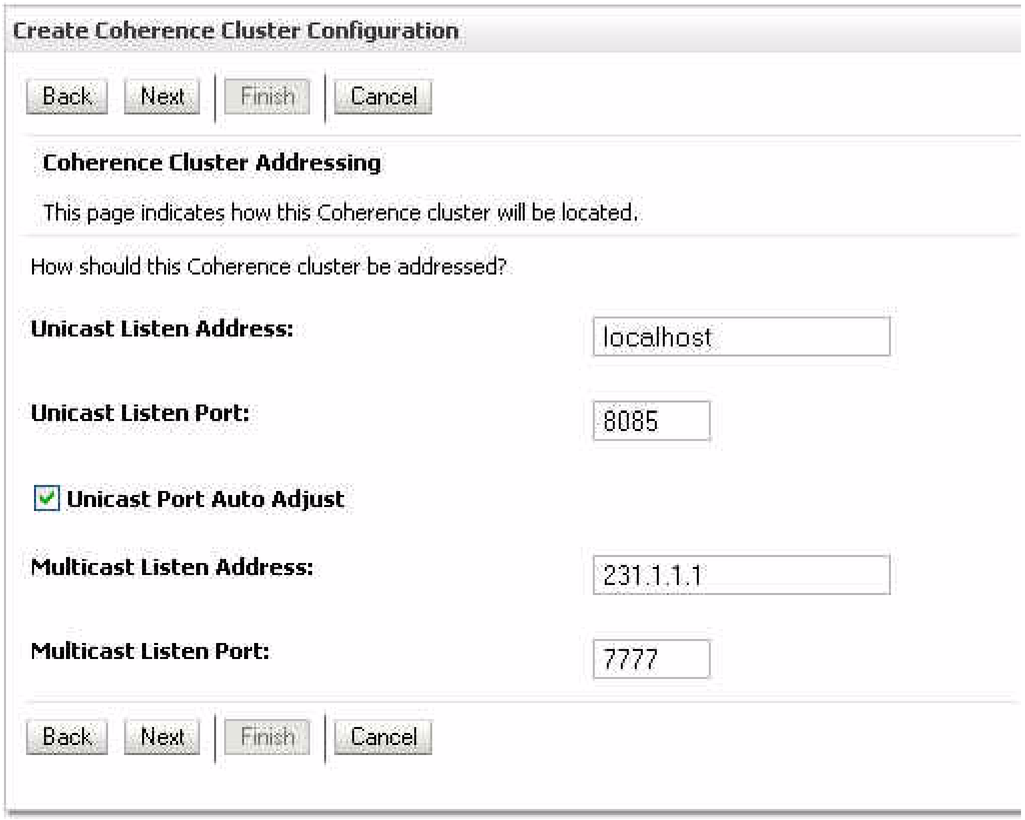
* Name: CoherenceCluster

Coherence clusters may be configured externally in a custom configuration file or configured within WebLogic Server. How would you like to configure this Coherence cluster?

☐ Use a Custom Cluster Configuration File

2. Enter a value such as `8085`, in the **Unicast Listen Port** field. Do not change any of the other values and click **Next**.

Figure 6–6 Specifying a Unicast Listen Port for a Coherence Cluster



Create Coherence Cluster Configuration

Coherence Cluster Addressing

This page indicates how this Coherence cluster will be located.

How should this Coherence cluster be addressed?

Unicast Listen Address: localhost

Unicast Listen Port: 8085

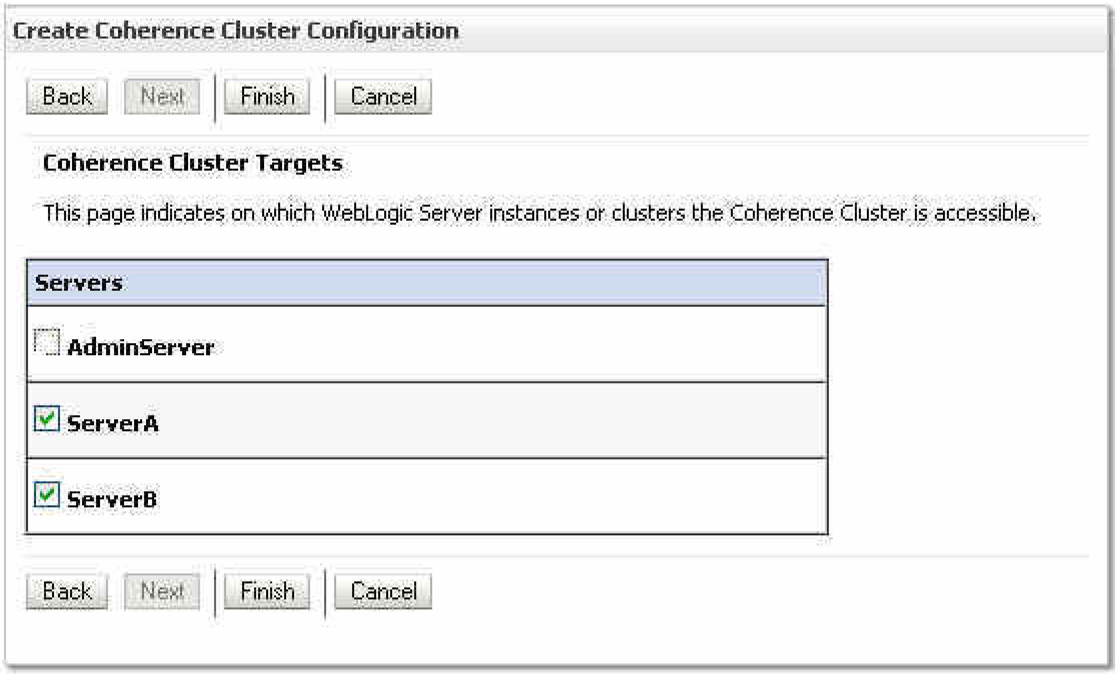
☑ Unicast Port Auto Adjust

Multicast Listen Address: 231.1.1.1

Multicast Listen Port: 7777

3. In the **Coherence Cluster Targets** page, select `ServerA` and `ServerB` as the targets. Click **Finish**.

Figure 6–7 Choosing Coherence Cluster Targets


Create Coherence Cluster Configuration

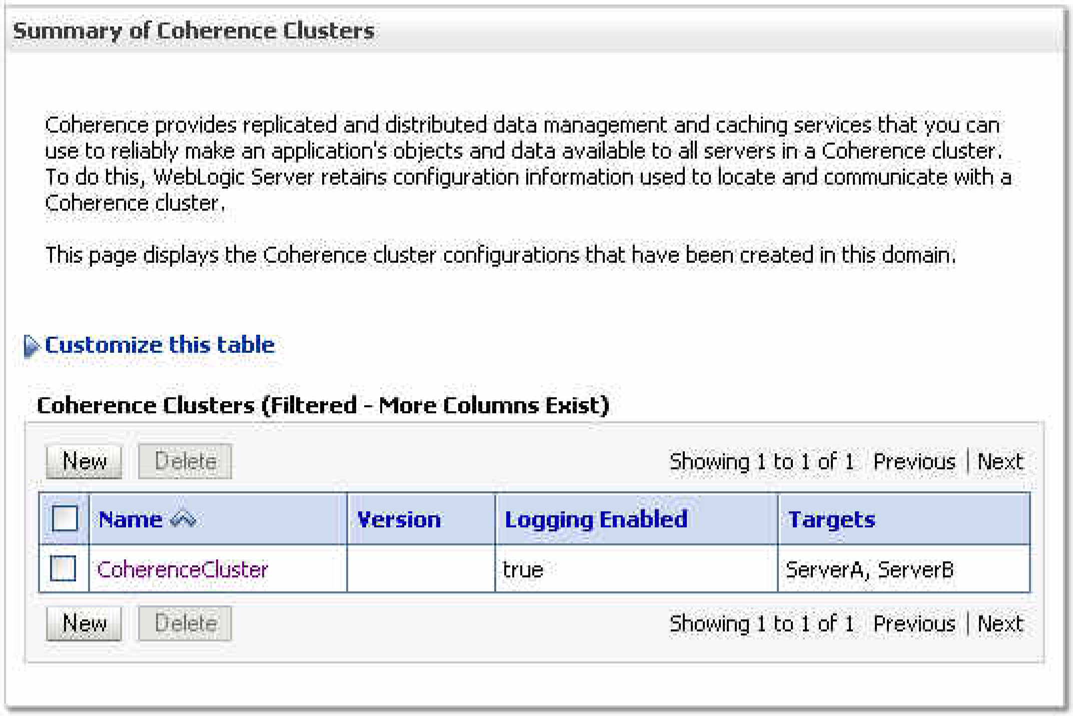
Coherence Cluster Targets

This page indicates on which WebLogic Server instances or clusters the Coherence Cluster is accessible.

Servers
☐ AdminServer
☑ ServerA
☑ ServerB

The **Summary of Coherence Clusters** page should look similar to Figure 6–8.

Figure 6–8 Summary of Coherence Clusters


Summary of Coherence Clusters

Coherence provides replicated and distributed data management and caching services that you can use to reliably make an application's objects and data available to all servers in a Coherence cluster. To do this, WebLogic Server retains configuration information used to locate and communicate with a Coherence cluster.

This page displays the Coherence cluster configurations that have been created in this domain.

Coherence Clusters (Filtered - More Columns Exist)

☐	Name	Version	Logging Enabled	Targets
☐	CoherenceCluster		true	ServerA, ServerB

6.7 Deploy the Shared Library Files

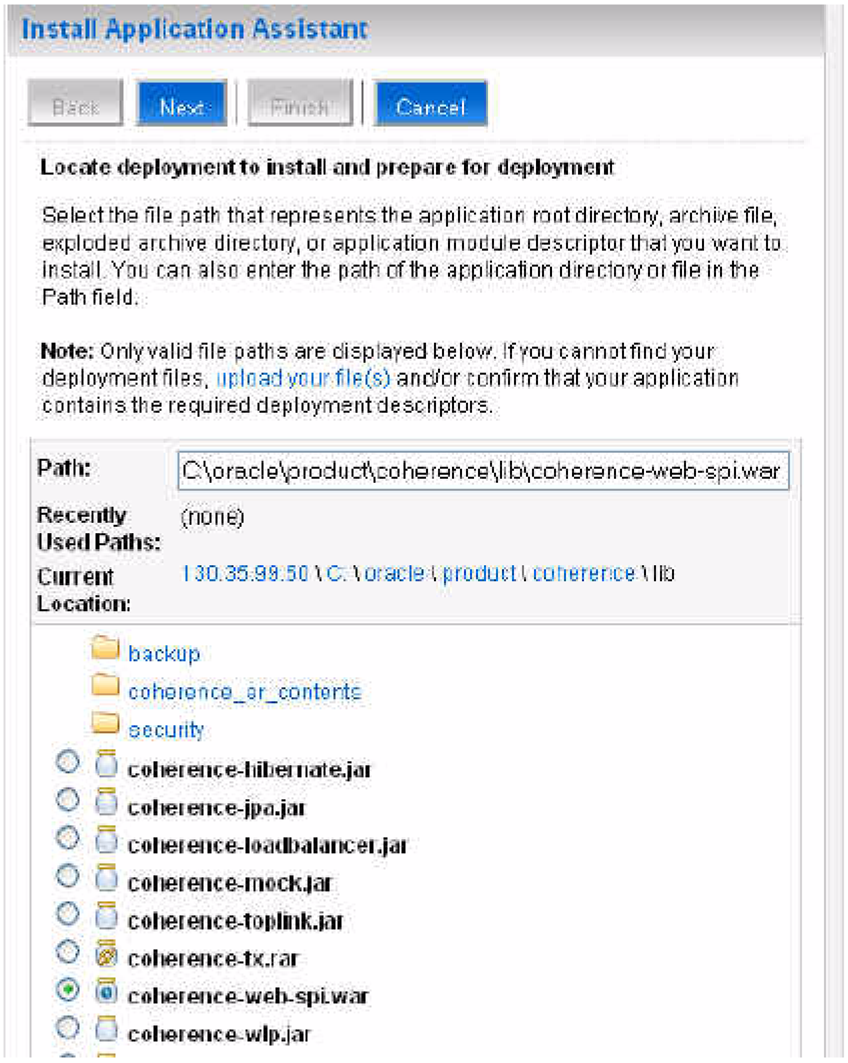
In addition to the `coherence.jar` file, Coherence provides a deployable shared library, `coherence-web-spi.war`, that contains a native plug-in to WebLogic Server's HTTP Session Management interface. Coherence also provides the `active-cache-1.0.jar` file which contains the classes that allow WebLogic Server to interact with Coherence.

You do not have to deploy `coherence.jar` for this example. It will be bundled with the application in a later step.

To deploy the `coherence-web-spi.war` and `active-cache-1.0.jar` files:

1. From the **Domain Structure** menu, click **Deployments**. The **Summary of Deployments** page displays.
2. Click **Install**. The **Install Application Assistant** screen displays.
3. Use the **Install Application Assistant** to deploy `coherence-web-spi.war` as a library to the `ServerA` and `ServerB`.
   a. Locate and select the `coherence-web-spi.war` file. Click **Next**.

Figure 6–9 Selecting the coherence-web-spi.jar File for Deployment



Locate deployment to install and prepare for deployment

Select the file path that represents the application root directory, archive file, exploded archive directory, or application module descriptor that you want to install. You can also enter the path of the application directory or file in the Path field.

Note: Only valid file paths are displayed below. If you cannot find your deployment files, upload your file(s) and/or confirm that your application contains the required deployment descriptors.

Path: C:\oracle\product\coherence\lib\coherence-web-spi.war

Recently Used Paths: (none)

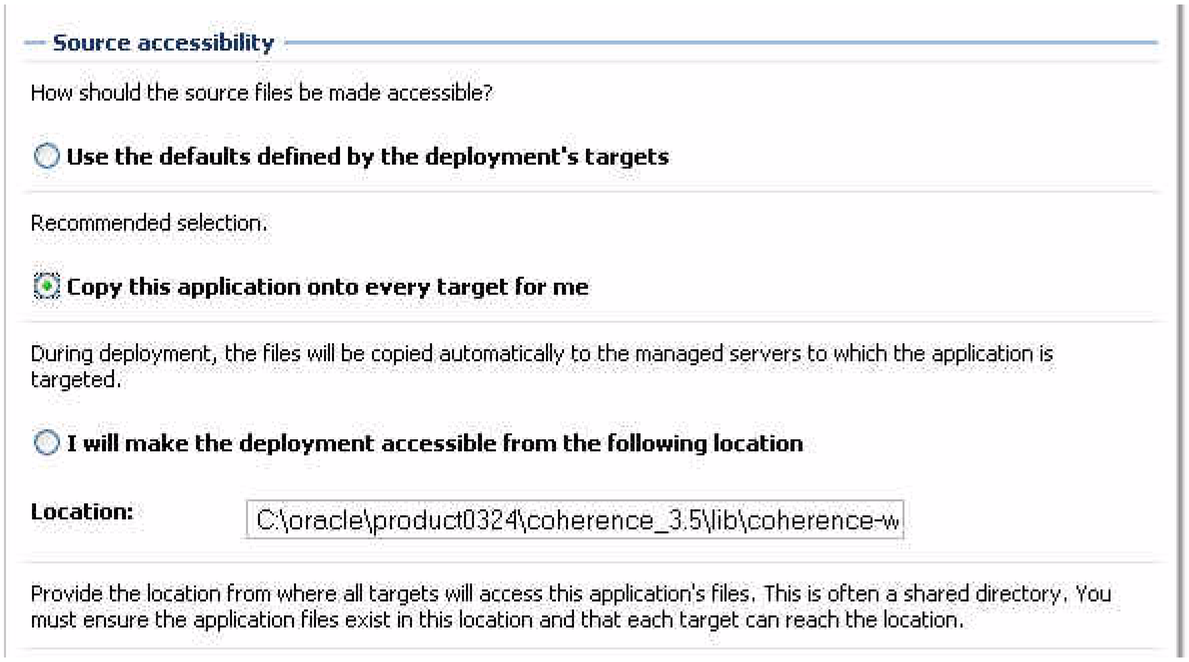
Current Location: 130.35.99.50 \ C: \ oracle \ product \ coherence \ lib

backup
coherence_ar_contents
security
coherence-hibernate.jar
coherence-jpa.jar
coherence-loadbalancer.jar
coherence-mock.jar
coherence-toplink.jar
coherence-tx.rar
[✓ coherence-web-spi.war]
coherence-wlp.jar

b. Select `ServerA` and `ServerB` as the deployment targets (do not deploy `coherence-web-spi.war` to the `AdminServer`). Click **Next**.

c. In the Optional Settings page, select the **Copy this application onto every target for me** option in the **Source accessibility** section.

Figure 6–10 Copying Files to Targets



Source accessibility

How should the source files be made accessible?

Use the defaults defined by the deployment's targets

Recommended selection.

[✓ Copy this application onto every target for me]

During deployment, the files will be copied automatically to the managed servers to which the application is targeted.

I will make the deployment accessible from the following location

Location: C:\oracle\product0324\coherence_3.5\lib\coherence-w

Provide the location from where all targets will access this application's files. This is often a shared directory. You must ensure the application files exist in this location and that each target can reach the location.

d. You can click **Finish** to skip the rest of the steps in the **Install Application Assistant**. The **Summary of Deployments** page displays after the application is deployed.

4. Repeat Steps 1-3 to deploy `active-cache-1.0.jar` to `ServerA` and `ServerB` (do not deploy `active-cache-1.0.jar` to the `AdminServer`).

6.8 Create the Counter Web Application

The Counter Web application is a simple counter implemented as a JSP. The counter is stored as an HTTP session attribute and increments each time the page is accessed.

To create the Counter Web application:

1. Create a standard Web application directory as follows:

```
/
/WEB-INF
```

2. Copy the following code to a text file and save it as `web.xml` in the `/WEB-INF` directory.

```
<?xml version = '1.0' encoding = 'windows-1252'?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
 xmlns="http://java.sun.com/xml/ns/j2ee" version="2.5">
   <description>Empty web.xml file for Web Application</description>
</web-app>
```

3. Create a `weblogic.xml` file in the `/WEB-INF` directory.
   - Add a library references for the `coherence-web-spi.war` file.
   - Reference the Coherence Cluster in a `coherence-cluster-ref` stanza.

   Example 6–2 illustrates a sample `weblogic.xml` file.

Example 6–2 Sample weblogic.xml File

```
<weblogic-web-app xmlns="http://xmlns.oracle.com/weblogic/weblogic-web-app"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-web-app
http://xmlns.oracle.com/weblogic/weblogic-web-app/1.0/weblogic-web-app.xsd">
      <library-ref>
           <library-name>coherence-web-spi</library-name>
           <specification-version>1.0.0.0</specification-version>
           <implementation-version>1.0.0.0</implementation-version>
      </library-ref>
     <coherence-cluster-ref>
        <coherence-cluster-name>CoherenceCluster</coherence-cluster-name>
     </coherence-cluster-ref>
</weblogic-web-app>
```

4. Bundle the `coherence.jar` file with the application: copy `coherence.jar` to the `WEB-INF/lib` directory.

5. Copy the following code for the counter JSP to a text file and save the file as `counter.jsp` in the root of the Web application directory.

```
<html>
    <body>

<h3>
      Counter :
      <%
         Integer counter = new Integer(1);
         HttpSession httpsession = request.getSession(true);
         if (httpsession.isNew()) {
                httpsession.setAttribute("count", counter);
                out.println(counter);
         } else {
                int count = ((Integer)
httpsession.getAttribute("count")).intValue();
                httpsession.setAttribute("count", new Integer(++count));
                out.println(count);
         }
      %>
      </h3>

    </body>
</html>
```

6. Create a `manifest.mf` file in the `META-INF` directory. Add references to the `active-cache` JAR file. Example 6–3 illustrates a sample `manifest.mf` file.

Example 6–3 Sample manifest.mf File

```
Extension-List: active-cache
active-cache-Extension-Name: active-cache
active-cache-Specification-Version: 1.0
```

```
active-cache-Implmenentation-Version: 1.0
```

7. The Web application directory should appear as follows:

```
/
/counter.jsp
/META-INF/manifest.mf
/WEB-INF/web.xml
/WEB-INF/weblogic.xml
/WEB-INF/lib/coherence.jar
```

8. ZIP or JAR the Web application directory and save the file as `counter.war`.

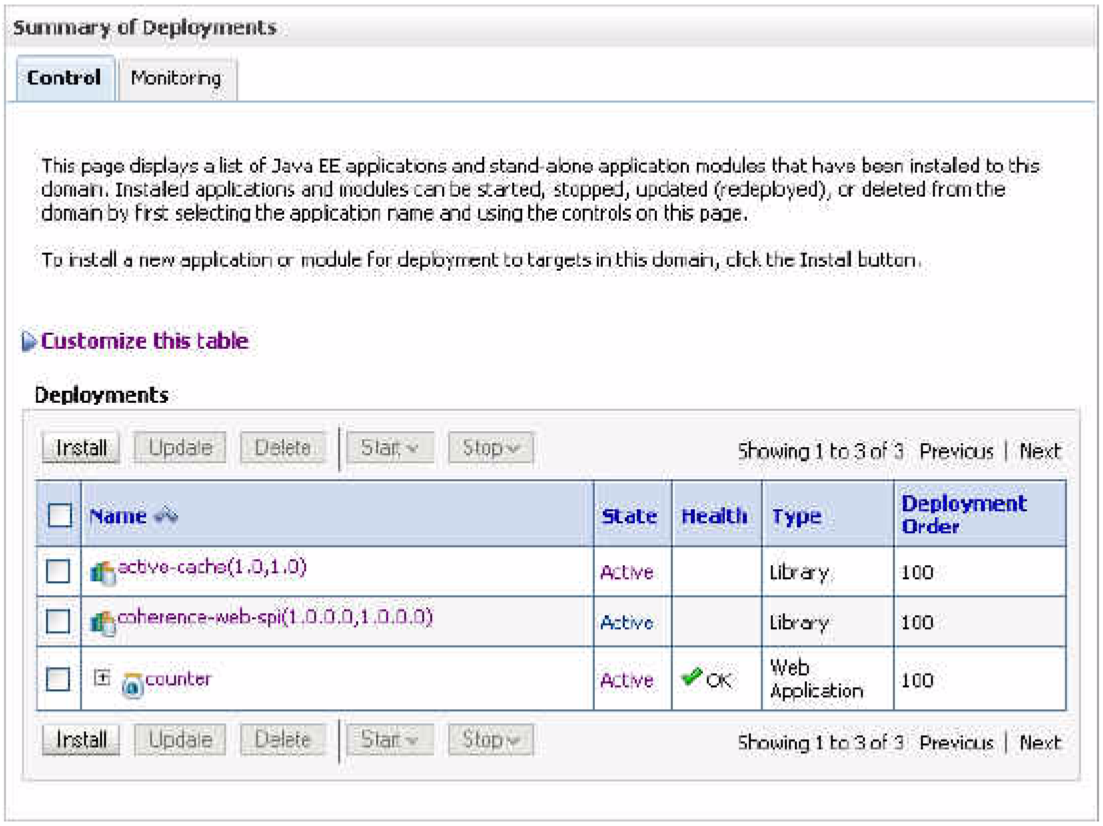
6.9 Deploy the Application

To deploy the `counter.war` application:

1. Open the **Summary of Deployments** page by clicking **Deployments** in the **Domain Structure** menu in the Oracle WebLogic Server Administration Console.
2. Click **Install**. The **Install Application Assistant** screen displays.
3. Use the **Install Application Assistant** to deploy `counter.war` to `ServerA` and `ServerB`. In the **Optional Settings** page, select the **Copy this application onto every target for me** option in the **Source accessibility** section.

   The **Summary of Deployments** page displays after the application is deployed. Figure 6–11 illustrates the deployments table with the counter Web application.

Figure 6–11 Deployments Window Showing the Deployed Application and Libraries



This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

To install a new application or module for deployment to targets in this domain, click the Install button.

Deployments

Showing 1 to 3 of 3 Previous | Next

	Name	State	Health	Type	Deployment Order
☐	active-cache(1.0,1.0)	Active		Library	100
☐	coherence-web-spi(1.0.0.0,1.0.0.0)	Active		Library	100
☐	counter	Active	✓ OK	Web Application	100

Showing 1 to 3 of 3 Previous | Next

6.10 Start the Node Manager and the WebLogic Servers

Start the Node Manager then start the WebLogic Servers from the WebLogic Server Administration Console. The Node Manager is a Java utility that runs as a separate process from Oracle WebLogic Server, and enables you to perform common operations for a Managed Server, regardless of its location with respect to its Administration Server.

1. Start the Node Manager from **Start** > **All Programs** > **Oracle Fusion Middleware** > **WebLogic Server10.3** > **Tools** > **Node Manager**.
2. Click **Environment** > **Servers** in the domain **Structure Window**. From the **Summary of Servers** screen in the WebLogic Server Administration Console, click the **Control** tab and start both server instances.

6.11 Verify the Example

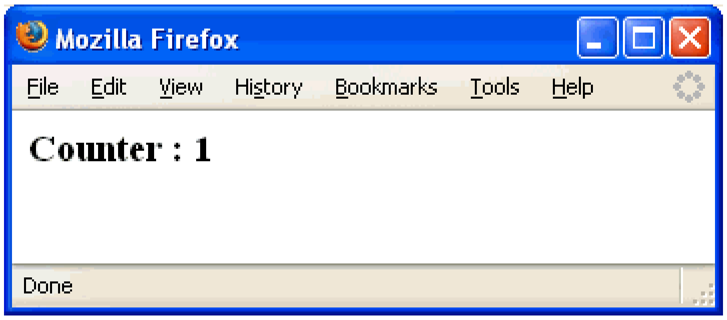
To verify the example:

1. Open a browser and access the `ServerA` counter instance using the following URL:

   `http://`*`host`*`:8080/counter/counter.jsp`

   The counter page displays and the counter is set to 1 as follows:

   Figure 6–12 Counter Page with Counter Set to 1

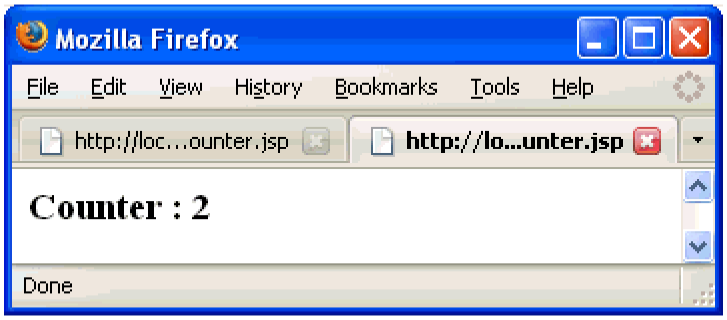
   Counter : 1

2. In a new browser (or new browser tab), access the `ServerB` counter instance using the following URL:

   `http://`*`host`*`:8081/counter/counter.jsp`

   The counter page displays and the counter increments to 2 based on the session data.

   Figure 6–13 Counter Page with Counter Set to 2

   Counter : 2

3. If you refresh the page, the counter increments to 3. Return to the original browser (or browser tab), refresh the instance and the counter displays 4.

7 Glossary

active-cache.jar

This file contains the Coherence integration classes for WebLogic. The file is installed regardless of whether you install Coherence. The default installation directory for this file is: `oracle/Middleware/wlserver_10.3/common/deployable-libraries`.

coherence.jar

The main development and run-time library for Coherence. The default installation directory for this file is `oracle\Middleware\coherence_3.5\lib`. The JAR contains a set of default XML configuration files that provide a default setup that allows Coherence to be used out-of-box with minimal changes. Among the several default configuration files, the most important in this book are coherence-cache-config.xml and tangosol-coherence.xml.

Coherence cluster

A group of Coherence nodes that share a group address which allows them to communicate. Coherence clusters consist of nodes formed by applications, modules, or application servers (WebLogic Server instances or cache servers). There can be many data caches within a single Coherence node.

coherence-cache-config.xml

This file is used to specify the various types of caches which can be used within a cluster. This file is typically referred to as the cache configuration deployment descriptor. The DTD for this file is `cache-config.dtd` file.

coherence-eclipselink.jar

This JAR contains the Coherence integration for EclipseLink. This JAR is required on the classpath if you are integrating your EclipseLink JPA application with Oracle Coherence.

coherence-web-spi.war

Contains a native plugin to WebLogic Server's HTTP Session Management interface. The WAR file provides the necessary support files to enable HTTP session management for applications running on WebLogic Server. The session state can be managed in the various caching topologies available in Coherence. This WAR is installed in the *`COHERENCE_HOME`*`/lib` directory.

Coherence node

Any application or server process that is running the Coherence software is called a Coherence node.

A single server process (WLS instance or cache server), WLS application (EAR), or application module (Web application) can be a Coherence node.

There can be many data caches within a single Coherence node.

eclipselink.jar

This JAR contains the EclipseLink persistence framework. EclipseLink supports virtually any type of data source, including relational databases, XML, or EIS systems.

session-cache-config.xml

Provides the configuration for the Coherence caches used by the Coherence*Web SPI. This file is located inside the `coherence-web-spi.war` file under the `WEB-INF\classes` directory. Any cache configuration change should defined in `session-cache-config.xml`, and then repackaged inside `coherence-web-spi.war`.

tangosol-coherence.xml

This files provides operational and run-time settings and is used to create and configure cluster, communication, and data management services. This file is typically referred to as the operational deployment descriptor. The DTD for this file is the `coherence.dtd` file.

You can override the settings in the default `tangosol-coherence.xml` file by creating a `tangosol-coherence-override.xml` file and placing it in the classpath at run time. The override file and the operational deployment descriptor have the same structure, however all of the elements in the override file are optional. The override file includes only the elements that are being changed. Any missing elements are loaded from the `tangosol-coherence.xml` file.

weblogic.xml

The WebLogic-specific deployment descriptor file that defines how named resources in the `web.xml` file are mapped to resources residing elsewhere in WebLogic Server. This file is also used to define JSP and HTTP session attributes.

weblogic-application.xml

The WebLogic Server-specific deployment descriptor extension for the `application.xml` deployment descriptor from Sun Microsystems. This is where you configure features such as shared Java EE libraries referenced in the application and EJB caching. The file is located in the `META-INF` subdirectory of the application archive.

weblogic-ejb-jar.xml

The WebLogic Server-specific deployment descriptor extension for the `ejb-jar.xml` deployment descriptor from Sun Microsystems.

WebLogic Server cluster

A group of WebLogic Server instances that work to provide scalability and high-availability for applications.

WebLogic Server node

One WebLogic Server instance hosting one or more Coherence nodes.