

Oracle® Fusion Middleware

Understanding Domain Configuration for Oracle WebLogic
Server 12.1.3

12c (12.1.3)

E41943-04

August 2015

This document describes Oracle WebLogic Server 12.1.3
domains and how they are configured.

Oracle Fusion Middleware Understanding Domain Configuration for Oracle WebLogic Server 12.1.3, 12c (12.1.3)

E41943-04

Copyright © 2007, 2015, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	vii
Documentation Accessibility	vii
Conventions	vii
1 Introduction and Roadmap	
1.1 Document Scope and Audience.....	1-1
1.2 Guide to this Document	1-1
1.3 Related Documentation.....	1-2
1.4 Samples and Tutorials.....	1-2
1.5 New and Changed Features in This Release.....	1-2
2 Understanding Oracle WebLogic Server Domains	
2.1 What Is a Domain?.....	2-1
2.2 Organizing Domains	2-1
2.3 Contents of a Domain.....	2-2
2.3.1 Administration Server.....	2-3
2.3.2 What Happens if the Administration Server Fails?.....	2-3
2.3.3 Managed Servers and Managed Server Clusters	2-4
2.3.4 Resources and Services	2-4
2.4 Domain Restrictions	2-5
2.5 Development and Production Modes.....	2-5
3 Domain Configuration Files	
3.1 Overview of Domain Configuration Files.....	3-1
3.1.1 Editing Configuration Documents	3-2
3.1.2 Security Credentials in Configuration Files.....	3-2
3.1.3 Configuration File Archiving.....	3-3
3.2 Domain Directory Contents	3-3
3.2.1 domain-name	3-3
3.2.2 autodeploy	3-3
3.2.3 bin.....	3-4
3.2.4 config	3-4
3.2.5 config/configCache	3-4
3.2.6 config/diagnostics.....	3-4
3.2.7 config/jdbc	3-4

3.2.8	config/jms.....	3-4
3.2.9	config/lib	3-4
3.2.10	config/nodemanager	3-4
3.2.11	config/security.....	3-4
3.2.12	configArchive	3-5
3.2.13	console-ext	3-5
3.2.14	init-info	3-5
3.2.15	lib.....	3-5
3.2.16	pending	3-5
3.2.17	security	3-5
3.2.18	servers.....	3-6
3.2.19	servers/ <i>server-name</i>	3-6
3.2.20	servers/ <i>server-name</i> /cache	3-6
3.2.21	servers/ <i>server-name</i> /cache/EJBCompilerCache.....	3-6
3.2.22	servers/ <i>server-name</i> /data	3-6
3.2.23	servers/ <i>server-name</i> /data/ldap.....	3-6
3.2.24	servers/ <i>server-name</i> /data/store	3-6
3.2.25	servers/ <i>server-name</i> /logs.....	3-6
3.2.26	servers/ <i>server-name</i> /logs/diagnostic_images	3-6
3.2.27	servers/ <i>server-name</i> /logs/jmsServers	3-7
3.2.28	servers/ <i>server-name</i> /security	3-7
3.2.29	servers/ <i>server-name</i> /tmp.....	3-7
3.2.30	tmp	3-7
3.3	A Server's Root Directory	3-7
3.3.1	Specifying a Server Root Directory	3-7
3.3.2	Server Root Directory for an Administration Server.....	3-8
3.3.3	Server Root Directory for a Managed Server Started with Node Manager	3-8
3.3.4	Server Root Directory for a Managed Server Not Started with Node Manager	3-9

4 Managing Configuration Changes

4.1	Overview of Change Management	4-1
4.1.1	Changes Requiring Server Restart	4-2
4.1.2	Configuration Change Tools	4-2
4.1.3	Configuration Auditing	4-2
4.2	Change Management in the Administration Console.....	4-2
4.2.1	Taking the Configuration Lock From Another Administrator	4-3
4.3	Configuration Change Management Process	4-4
4.3.1	Configuration Locks	4-6
4.3.2	Resolving Change Conflicts	4-7
4.4	Configuration Management State Diagram.....	4-7
4.5	Restricting Configuration Changes	4-8

5 Server Templates

5.1	What are Server Templates?.....	5-1
5.2	Why Do You Use Server Templates?	5-2
5.3	Using Server Templates	5-2
5.3.1	Overriding Attributes	5-2

5.3.2	Using Macros.....	5-4
5.4	Server Templates Example	5-4

Preface

This preface describes the document accessibility features and conventions used in this guide—*Understanding Domain Configuration for Oracle WebLogic Server 12.1.3*.

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.

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.
<i>italic</i>	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.

Introduction and Roadmap

This document describes Oracle WebLogic Server 12.1.3 domains and how they are configured. A domain is the basic administration unit for Oracle WebLogic Server. A domain consists of one or more Oracle WebLogic Server instances (and their associated resources) that you manage with a single Administration Server

The following sections describe the contents and organization of this guide—*Understanding Domain Configuration for Oracle WebLogic Server 12.1.3*.

- [Section 1.1, "Document Scope and Audience"](#)
- [Section 1.2, "Guide to this Document"](#)
- [Section 1.3, "Related Documentation"](#)
- [Section 1.4, "Samples and Tutorials"](#)
- [Section 1.5, "New and Changed Features in This Release"](#)

1.1 Document Scope and Audience

This document is written mainly for Java Platform, Enterprise Edition (Java EE) system architects, application developers, and system administrators who are developing or deploying Web-based applications on one or more Oracle WebLogic Server domains.

The topics in this document are relevant during the design and development phases of a software project. This document does not address production phase administration, monitoring, or performance tuning topics. For links to Oracle WebLogic Server documentation and resources for these topics, see [Section 1.3, "Related Documentation"](#).

It is assumed that the reader is familiar with Java EE, basic concepts of XML, and general networking and application management concepts.

1.2 Guide to this Document

- This chapter, [Chapter 1, "Introduction and Roadmap"](#), introduces the purpose, organization, and context of this guide.
- [Chapter 2, "Understanding Oracle WebLogic Server Domains"](#) introduces Oracle WebLogic Server domains.
- [Chapter 3, "Domain Configuration Files"](#) describes the configuration and directories that maintain the on-disk representation of a domain and its contents.

- [Chapter 4, "Managing Configuration Changes"](#) describes change management features in Oracle WebLogic Server.
- [Chapter 5, "Server Templates"](#) introduces and describes server templates.

1.3 Related Documentation

For information about system administration tasks and the various tools you can use to perform them, see:

- "System Administration" in *Understanding Oracle WebLogic Server*
- *Developing Java EE Management Applications for Oracle WebLogic Server*
- *Oracle WebLogic Server MBean Reference*
- *Monitoring Oracle WebLogic Server with SNMP*

For more detailed information about tools you can use to create and configure Oracle WebLogic Server domains, see:

- *Creating WebLogic Domains Using the Configuration Wizard*
- *Understanding the WebLogic Scripting Tool*
- *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*
- *Command Reference for Oracle WebLogic Server*
- *Oracle WebLogic Server Administration Console Online Help*

1.4 Samples and Tutorials

In addition to this document, Oracle provides code samples that are relevant to domain configuration and administration.

- The WebLogic Scripting Tool (WLST) examples show how to automate the creation of domains using WLST. See "WLST Sample Scripts" in *Understanding the WebLogic Scripting Tool*.
- The Avitek Medical Records Sample Application is an Oracle WebLogic Server sample application suite that concisely demonstrates all aspects of the Java EE platform. To start this sample application, invoke the startWeblogic script in:

```
ORACLE_HOME/user_projects/domains/medrec
```

where *ORACLE_HOME* is the directory you specified as the Oracle Home when you installed Oracle WebLogic Server.

1.5 New and Changed Features in This Release

For a comprehensive listing of the new Oracle WebLogic Server features introduced in this release, see *What's New in Oracle WebLogic Server*.

Understanding Oracle WebLogic Server Domains

This chapter describes Oracle WebLogic Server 12.1.3 domains and their contents, which include an Administration Server and usually one or more Managed Servers.

This chapter includes the following sections:

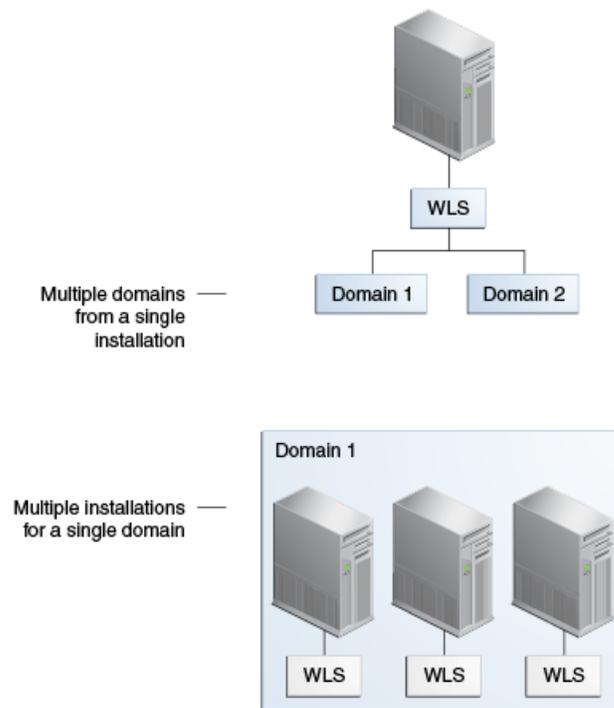
- [Section 2.1, "What Is a Domain?"](#)
- [Section 2.2, "Organizing Domains"](#)
- [Section 2.3, "Contents of a Domain"](#)
- [Section 2.4, "Domain Restrictions"](#)
- [Section 2.5, "Development and Production Modes"](#)

2.1 What Is a Domain?

An Oracle WebLogic Server administration **domain** is a logically related group of Oracle WebLogic Server resources. Domains include a special Oracle WebLogic Server instance called the **Administration Server**, which is the central point from which you configure and manage all resources in the domain. Usually, you configure a domain to include additional Oracle WebLogic Server instances called **Managed Servers**. You deploy Web applications, EJBs, Web services, and other resources onto the Managed Servers and use the Administration Server for configuration and management purposes only.

2.2 Organizing Domains

You can use a single Oracle WebLogic Server installation to create and run multiple domains, or you can use multiple installations to run a single domain. See [Figure 2-1](#).

Figure 2–1 Oracle WebLogic Server Installations and Domains

How you organize your Oracle WebLogic Server installations into domains depends on your business needs. You can define multiple domains based on different system administrators' responsibilities, application boundaries, or geographical locations of the machines on which servers run. Conversely, you might decide to use a single domain to centralize all Oracle WebLogic Server administration activities.

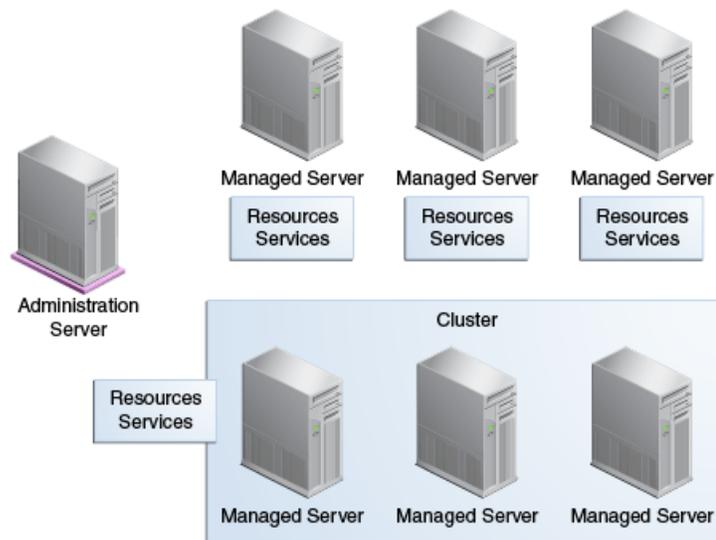
Depending on your particular business needs and system administration practices, you might decide to organize your domains based on criteria such as:

- Logical divisions of applications. For example, you might have one domain devoted to end-user functions such as shopping carts and another domain devoted to back-end accounting applications.
- Physical location. You might establish separate domains for different locations or branches of your business. Each physical location requires its own Oracle WebLogic Server installation.
- Size. You might find that domains organized in small units can be managed more efficiently, perhaps by different system administrators. Contrarily, you might find that maintaining a single domain or a small number of domains makes it easier to maintain a consistent configuration.

For development or test environments, you can create a simple domain that consists of a single server instance. This single instance acts as an Administration Server and hosts the applications that you are developing. The `wl_server` domain that you can install with Oracle WebLogic Server is an example of this type of domain.

2.3 Contents of a Domain

Figure 2–2 shows a production environment that contains an Administration Server, three stand-alone Managed Servers, and a cluster of three Managed Servers.

Figure 2–2 Content of a Domain

Although the scope and purpose of a domain can vary significantly, most Oracle WebLogic Server domains contain the components described in this section.

2.3.1 Administration Server

The Administration Server operates as the central control entity for the configuration of the entire domain. It maintains the domain's configuration documents and distributes changes in the configuration documents to Managed Servers. You can also use the Administration Server as a central location from which to monitor all resources in a domain.

To interact with the Administration Server, you can use any of the administration tools listed in "Summary of System Administration Tools and APIs" in *Understanding Oracle WebLogic Server*. See "System Administration" in *Understanding Oracle WebLogic Server* for information about modifying the domain's configuration.

Each Oracle WebLogic Server domain must have one server instance that acts as the Administration Server.

For more information about the Administration Server and its role in the Oracle WebLogic Server JMX management system, see "System Administration" in *Understanding Oracle WebLogic Server*.

2.3.2 What Happens if the Administration Server Fails?

The failure of an Administration Server does not affect the operation of Managed Servers in the domain but it does prevent you from changing the domain's configuration. If an Administration Server fails because of a hardware or software failure on its host machine, other server instances on the same machine may be similarly affected. However, the failure of an Administration Server itself does not interrupt the operation of Managed Servers in the domain.

If an Administration Server for a domain becomes unavailable while the server instances it manages—clustered or otherwise—are up and running, those Managed Servers continue to run. Periodically, the Managed Servers attempt to reconnect to the Administration Server. If the domain contains clustered server instances, the load

balancing and failover capabilities supported by the domain configuration remain available, even if the Administration Server fails.

You can start a Managed Server even if the Administration Server is not running. In this case, the Managed Server uses a local copy of the domain's configuration files for its starting configuration and then periodically attempts to connect with the Administration Server. When it does connect, it synchronizes its configuration state with that of the Administration Server.

For information on starting a Managed Server without a running Administration Server, see "Managed Server Independence Mode" in *Administering Server Startup and Shutdown for Oracle WebLogic Server*. For information on re-starting an Administration Server, see "Avoiding and Recovering From Server Failure" in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

2.3.3 Managed Servers and Managed Server Clusters

Managed Servers host business applications, application components, Web services, and their associated resources. To optimize performance, Managed Servers maintain a read-only copy of the domain's configuration document. When a Managed Server starts up, it connects to the domain's Administration Server to synchronize its configuration document with the document that the Administration Server maintains.

For production environments that require increased application performance, throughput, or high availability, you can configure two or more Managed Servers to operate as a **cluster**. A cluster is a collection of multiple Oracle WebLogic Server instances running simultaneously and working together to provide increased scalability and reliability. In a cluster, most resources and services are deployed identically to each Managed Server (as opposed to a single Managed Server), enabling failover and load balancing. A single domain can contain multiple Oracle WebLogic Server clusters, as well as multiple Managed Servers that are not configured as clusters. The key difference between clustered and non-clustered Managed Servers is support for failover and load balancing. These features are available only in a cluster of Managed Servers. For more information about the benefits and capabilities of an Oracle WebLogic Server cluster, see "Understanding WebLogic Server Clustering" in *Administering Clusters for Oracle WebLogic Server*.

2.3.4 Resources and Services

In addition to the Administration Server and Managed Servers, a domain also contains the resources and services that Managed Servers and deployed applications require.

Managed Servers can use the following resources:

- Machine definitions that identify a particular, physical piece of hardware. A machine definition is used to associate a computer with the Managed Servers it hosts. This information is used by Node Manager in restarting a failed Managed Server, and by a clustered Managed Server in selecting the best location for storing replicated session data. For more information about Node Manager, see "Node Manager Overview" in the *Administering Node Manager for Oracle WebLogic Server*.
- Network channels that define default ports, protocols, and protocol settings that a Managed Server uses to communicate with clients. After creating a network channel, you can assign it to any number of Managed Servers and clusters in the domain. For more information, see "Configuring Network Resources" in *Administering Server Environments for Oracle WebLogic Server*.
- Virtual hosting, which defines a set of host names to which Oracle WebLogic Server instances (servers) or clusters respond. When you use virtual hosting, you

use DNS to specify one or more host names that map to the IP address of a server or cluster. You also specify which Web applications are served by each virtual host.

Applications can use the following resources and services:

- Security providers, which are modular components that handle specific aspects of security, such as authentication and authorization.
- Resource adapters, which are system libraries specific to Enterprise Information Systems (EIS) and provide connectivity to an EIS.
- Diagnostics and monitoring services.
- JDBC data sources, which enable applications to connect to databases.
- Mail sessions.
- XML entity caches and registry of XML parsers and transformer factories.
- Messaging services such as JMS servers and store-and-forward services.
- Persistent store, which is a physical repository for storing data, such as persistent JMS messages. It can be either a JDBC-accessible database or a disk-based file.
- Startup classes, which are Java programs that you create to provide custom, system-wide services for your applications.
- Work Managers, which determine how an application prioritizes the execution of its work based on rules you define and by monitoring actual run-time performance. You can create Work Managers for entire Oracle WebLogic Server domains or for specific application components.
- Work Contexts, which enable applications to pass properties to a remote context without including the properties in a remote call.

2.4 Domain Restrictions

In designing your domain configuration, note the following restrictions:

- Each domain requires its own Administration Server for performing management activities. When you use the WebLogic Server Administration Console to perform management and monitoring tasks, you can switch back and forth between domains, but in doing so, you are connecting to different Administration Servers.
- All Managed Servers in a cluster must reside in the same domain; you cannot split a cluster over multiple domains.
- All Managed Servers in a domain must run the same version of the Oracle WebLogic Server software. The Administration Server may run either the same version as the Managed Servers in the domain, or a later patch set.

If you have created multiple domains, each domain must reference its own database schema. You cannot share a configured resource or subsystem between domains. For example, if you create a JDBC data source in one domain, you cannot use it with a Managed Server or cluster in another domain. Instead, you must create a similar data source in the second domain. Furthermore, two or more system resources cannot have the same name.

2.5 Development and Production Modes

You can configure a WebLogic domain to run in one of two modes: development mode or production mode. The domain mode determines default settings regarding security and logging. In development mode, the security configuration is more relaxed. You

can start the Administration Server using a boot identity file or deploy an application using the `autodeploy` directory. In production mode, the security configuration is much more stringent, such as requiring a username and password to deploy applications and start the Administration Server.

There are also several differences between the default settings in production mode and the default settings in domain mode, including:

- The number of threads in an execute queue
- Log settings, including the maximum number of files saved during log rotation, the minimum size of the log file that triggers log rotation, and whether logs are rotated at startup
- The default for server lifecycle operations timeout
- SNMP security level
- Servlet reload check periods

For more information on development and production mode default tuning values, see "Development vs. Production Mode Default Tuning Values" in *Tuning Performance of Oracle WebLogic Server*.

You can configure the domain mode at the time you create the domain, and you can also change the domain mode at any time later. For more information about how to select the domain mode using the Configuration Wizard, see "Domain Mode" in *Creating WebLogic Domains Using the Configuration Wizard*.

For information about how to modify a domain from development mode to production mode using the WebLogic Server Administration Console, see "Change to production mode" in the *Oracle WebLogic Server Administration Console Online Help*. (Note that WebLogic Server also supports modifying a domain from production mode to development mode).

Note: Oracle recommends that you do not enable the Web Services Test Client in production mode. For more information about the Web Services Test Client, see "Using the Web Services Test Client" in *Administering Web Services*.

After changing to production mode, Oracle recommends that you do the following to enhance the security of your production domain:

1. Modify the domain start scripts to change the value of the `-Xverify` flag from `none` to `all`. This change enables bytecode verification. For more information on bytecode verification, see "Java Language Security and Bytecode Verification" in the *Java Security Overview* at <http://docs.oracle.com/javase/7/docs/technotes/guides/security/overview/jsoverview.html>.
2. Restart the Administration Server and each Managed Server in your domain. In the command to restart each server instance, include the `-Dweblogic.system.RemoveBootIdentity=true` argument, which removes the boot identity file. The boot identity file only needs to be removed on the first restart after changing the configuration to production mode. You can use either of the following methods:
 - Open a separate command shell and include this argument in the `weblogic.Server` startup command for each server instance.

- Define the `JAVA_OPTIONS` variable to include `-Dweblogic.system.RemoveBootIdentity=true` and then invoke the server's start script.

Note: When you change a domain from development mode to production mode:

- The domain start scripts (and the value of the `-Xverify` flag) do not change.
 - The `boot.properties` file continues to be in use.
-

Domain Configuration Files

This chapter describes the domain configuration files, which are XML documents, that Oracle WebLogic Server 12.1.3 uses to persist the configuration of a domain.

This chapter includes the following sections:

- [Section 3.1, "Overview of Domain Configuration Files"](#)
- [Section 3.2, "Domain Directory Contents"](#)
- [Section 3.3, "A Server's Root Directory"](#)

3.1 Overview of Domain Configuration Files

Each domain describes its configuration in an XML document that is located in the domain's configuration directory. At run time, each Oracle WebLogic Server instance in a given domain creates an in-memory representation of the configuration described in this document.

Note: The domain directory must have Write privileges, as well as Read privileges, for the domain to function properly, even if no changes are made to the configuration after it is created. This is because WebLogic Server performs its own internal deployments, and configuration files may be rewritten when the server is restarted.

The central configuration file for a domain is *DOMAIN_NAME/config/config.xml*. This file specifies the name of the domain and the configuration of each server instance, cluster, resource, and service in the domain. The file includes references to additional XML files that are stored in subdirectories of the *DOMAIN_NAME/config* directory. These included files are used to describe major subsystems of Oracle WebLogic Server.

As a performance optimization, Oracle WebLogic Server does not store most of its default values in the domain's configuration files. In some cases, this optimization prevents XML elements from being written to the configuration files. For example, if you never modify the default logging severity level for a domain while the domain is active, the *config.xml* file does not contain an XML element for the domain's logging configuration.

As an additional performance optimization, each Managed Server maintains a copy of the domain's configuration files. This copy is read-only and can be updated only as part of a change management process (see [Chapter 4, "Managing Configuration Changes"](#)).

3.1.1 Editing Configuration Documents

In most circumstances, you should not use a text editor or other non-Oracle tools to modify a domain's configuration document. Instead, use the WebLogic Server Administration Console, WebLogic Scripting Tool (WLST), or one of the other tools described in "System Administration" in *Understanding Oracle WebLogic Server*.

However, because the Oracle WebLogic Server configuration document is an XML file that conforms to a schema, it is possible to modify them using XSLT or an XML parser application such as Apache Xerces or JDOM. Be sure to test any scripts that you create thoroughly and always make a backup copy of each configuration file before you make any changes to it.

The schemas that define a domain's configuration document are in the following locations:

- <http://xmlns.oracle.com/weblogic/domain/1.0/domain.xsd>
- <http://xmlns.oracle.com/weblogic/security/1.0/security.xsd>
- <http://xmlns.oracle.com/weblogic/weblogic-diagnostics/1.0/weblogic-diagnostics.xsd>
- In JAR files under *WL_HOME*/server/lib/schema, where *WL_HOME* is the directory in which you install Oracle WebLogic Server. Within this directory:
 - The `domain.xsd` document is represented in the `weblogic-domain-binding.jar` under the pathname `META-INF/schemas/schema-0.xsd`.
 - The `security.xsd` document is represented in the `weblogic-domain-binding.jar` under the pathname `META-INF/schemas/schema-1.xsd`.
 - The `weblogic-diagnostics.xsd` document is represented in the `diagnostics-binding.jar` under the pathname `META-INF/schemas/schema-0.xsd`.

Caution: Do not edit configuration files for a domain that is currently running. Because Oracle WebLogic Server rewrites the files periodically, your changes will be lost. Depending on your platform, you also could cause Oracle WebLogic Server failures.

3.1.2 Security Credentials in Configuration Files

Security credentials for domain security and the embedded LDAP server are stored in the `config.xml` file in encrypted form. If you create your `config.xml` file with a text editor or other non-Oracle tool, you need to locate these credentials, encrypt them, and copy the encrypted credential into your `config.xml` file.

For information about Oracle WebLogic Server's encryption utility, see "encrypt" in the *Command Reference for Oracle WebLogic Server*. Once you have encrypted the credentials, include the encrypted values in your `config.xml` file in elements as in [Example 3-1](#):

Example 3-1 Configuring Encrypted Credentials

```
<security-configuration>
  <credential-encrypted>{3DES}encrypted-value-here</credential-encrypted>
</security-configuration>
<embedded-ldap>
  <credential-encrypted>{3DES}encrypted-value-here</credential-encrypted>
</embedded-ldap>
```

3.1.3 Configuration File Archiving

You can configure Oracle WebLogic Server to make backup copies of the configuration files. This facilitates recovery in cases where configuration changes need to be reversed or the unlikely case that configuration files become corrupted. When the Administration Server starts up, it saves a JAR file named `config-booted.jar` that contains the configuration files. When you make changes to the configuration files, the old files are saved in the `configArchive` directory under the domain directory, in a JAR file with a sequentially-numbered name like `config-1.jar`.

For information on archiving configuration files, see "Archive configuration files" in *Oracle WebLogic Server Administration Console Online Help*. If you want to use WLST to configure Oracle WebLogic Server to make backup copies, set the `ConfigBackupEnabled` attribute in `DomainMBean` to `true` and the `ArchiveConfigurationCount` attribute to the number of configuration archive files that you want to retain, shown in [Example 3-2](#).

Example 3-2 Configuring Archive Files

```
connect()
edit()
startEdit()
cmo.setArchiveConfigurationCount(5)
cmo.setConfigBackupEnabled(true)
activate()
```

3.2 Domain Directory Contents

By default, Oracle WebLogic Server creates domain directories under Oracle Middleware `ORACLE_HOME/user_projects/domains` directory. This section describes the contents of the domain directory and its subfolders. In this section, *domain-name*, *deployment-name*, and *server-name* represent names that you define when you create a domain.

Individual applications in a domain might create additional files and directories in the domain directory.

If you have not yet created a domain, you can see an example of an existing domain directory by looking in `ORACLE_HOME/user_projects/domains/wl_server` where `ORACLE_HOME` is the directory you specified as the Oracle Home when you installed Oracle WebLogic Server.

3.2.1 domain-name

The name of this directory is the name of the domain.

3.2.2 autodeploy

This directory provides a quick way to deploy applications in a development server. When the Oracle WebLogic Server instance is running in development mode, it automatically deploys any applications or modules that you place in this directory.

The files you place in this directory can be Java EE applications, such as:

- An EAR file
- A WAR, EJB JAR, RAR, or CAR archived module
- An exploded archive directory for either an application or a module

3.2.3 bin

This directory contains scripts that are used in the process of starting and stopping the Administration Server and the Managed Servers in the domain. These scripts are generally provided as `.sh` files for UNIX and `.cmd` files for Windows. The `bin` directory can optionally contain other scripts of domain-wide interest, such as scripts to start and stop database management systems, full-text search engine processes, and such. For more information, see *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

3.2.4 config

This directory contains the current configuration and deployment state of the domain. The central domain configuration file, `config.xml`, resides in this directory.

3.2.5 config/configCache

Contains data that is used to optimize performance when validating changes in the domain's configuration documents. This data is internal to Oracle WebLogic Server and does not need to be backed up.

3.2.6 config/diagnostics

This directory contains system modules for instrumentation in the WebLogic Diagnostic Framework. For more information, see *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

3.2.7 config/jdbc

This directory contains system modules for JDBC: global JDBC modules that can be configured directly from JMX (as opposed to JSR-88). For more information, see Database Connectivity for Oracle WebLogic Server.

3.2.8 config/jms

This directory contains system modules for JMS: global JMS modules that can be configured directly from JMX (as opposed to JSR-88). For more information, see Messaging for Oracle WebLogic Server.

3.2.9 config/lib

This directory is not used in the current release of Oracle WebLogic Server.

3.2.10 config/nodemanager

This directory holds information for connection to the Node Manager. For more information, see "Node Manager Configuration and Log Files" in the *Administering Node Manager for Oracle WebLogic Server*.

3.2.11 config/security

This directory contains system modules for the security framework. It contains one security provider configuration extension for each kind of security provider in the domain's current realm. For more information, see *Understanding Security for Oracle WebLogic Server*.

3.2.12 configArchive

This directory contains a set of JAR files that save the domain's configuration state. Just before pending changes to the configuration are activated, the domain's existing configuration state, consisting of the `config.xml` file and the other related configuration files, is saved in a versioned JAR file with a name like `config.jar#1`, `config.jar#2`, and such.

The maximum number of versioned JAR files to be kept is specified by the `archiveConfigurationCount` attribute of `DomainMBean`. Once this maximum number is reached, the oldest conversion archive is deleted before a new one is created.

3.2.13 console-ext

This directory contains extensions to the WebLogic Server Administration Console, which enable you to add content to the WebLogic Server Administration Console, replace content, and change the logos, styles and colors without modifying the files that are installed with Oracle WebLogic Server. For example, you can add content that provides custom monitoring and management facilities for your applications.

3.2.14 init-info

This directory contains files used for WebLogic domain provisioning. You should not modify any files in this directory.

3.2.15 lib

JAR files you put in this directory are made available (within a separate system level classloader) to all Java EE applications running on WebLogic Server instances in the domain. For more information, see "Adding JARs to the Domain `/lib` Directory" in *Developing Applications for Oracle WebLogic Server*.

3.2.16 pending

This directory contains domain configuration files representing configuration changes that have been requested, but not yet activated. Once the configuration changes have been activated, the configuration files are deleted from this directory. For more information, see [Chapter 4, "Managing Configuration Changes"](#).

3.2.17 security

This directory holds those security-related files that are the same for every Oracle WebLogic Server instance in the domain:

- `SerializedSystemIni.dat`

This directory also holds security-related files that are only needed by the domain's Administration Server:

- `DefaultAuthorizerInit.ldift`
- `DefaultAuthenticatorInit.ldift`
- `DefaultRoleMapperInit.ldift`
- `XACMLRoleMapperInit.ldift`

For more information, see *Understanding Security for Oracle WebLogic Server*.

3.2.18 servers

This directory contains one subdirectory for each Oracle WebLogic Server instance in the domain. The subdirectories contain data that is specific to each server instance.

3.2.19 servers/*server-name*

This directory is the server directory for the Oracle WebLogic Server instance with the same name as the directory.

3.2.20 servers/*server-name*/cache

This directory holds directories and files that contain cached data. By "cached" we mean that the data is a copy, possibly in a processed form (compiled, translated, or reformatted), of other data.

3.2.21 servers/*server-name*/cache/EJBCompilerCache

This directory is a cache for compiled EJBs.

3.2.22 servers/*server-name*/data

This directory holds files that maintain persistent per-server state used to run the Oracle WebLogic Server instance, other than security state, as opposed to temporary, cached or historical information. Files in this directory are important data that must be retained as the Oracle WebLogic Server instance is brought up, is brought down, crashes, restarts, or is upgraded to a new version.

3.2.23 servers/*server-name*/data/ldap

This directory holds the embedded LDAP database. The run-time security state for the Oracle WebLogic Server instance is persisted in this directory.

3.2.24 servers/*server-name*/data/store

This directory holds WebLogic persistent stores. For each persistent store, there is a subdirectory that holds the files that represent the persistent store. The name of the subdirectory is the name of the persistent store. By convention there is one store named `default`.

3.2.25 servers/*server-name*/logs

This directory holds logs and diagnostic information. This information is historical in nature. It is not crucial to the operation of the server, and can be deleted (while the Oracle WebLogic Server instance is down, at least) without affecting proper operation. However, the information can be quite useful for debugging or auditing purposes and should not be deleted without good reason.

3.2.26 servers/*server-name*/logs/diagnostic_images

This directory holds information created by the Server Image Capture component of the WebLogic Diagnostic Framework. For more information, see *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

3.2.27 `servers/server-name/logs/jmsServers`

This directory contains one subdirectory for each JMS server in the Oracle WebLogic Server instance. Each such subdirectory contains the logs for that JMS server. The name of the subdirectory is the name of the JMS server.

3.2.28 `servers/server-name/security`

This directory holds security-related files that can be or must be different for each Oracle WebLogic Server instance. The file `boot.properties` is an example of a file that resides here because it can differ from one server to the next. This directory also maintains files related to SSL keys.

3.2.29 `servers/server-name/tmp`

This directory holds temporary directories and files that are created while a server instance is running. For example, a JMS paging directory is automatically created here unless another location is specified. Files in this directory must be left alone while the server is running, but may be freely deleted when the server instance is shut down.

3.2.30 `tmp`

This directory stores temporary files used in the change management process. You should not modify any files in this directory.

3.3 A Server's Root Directory

All instances of Oracle WebLogic Server use a root directory to store their working copy of the domain's configuration files, to store run-time data, and to provide the context for any relative pathnames in the server's configuration. An Administration Server always uses the domain directory as its root directory. A Managed Server can use the domain directory but can also use any other directory that you define.

For example, if you start a Managed Server on a computer that does not share a file system with the computer that hosts the Administration Server, the Managed Server will create its own root directory. The server will copy data from the domain directory to this root directory and will write run-time data in this directory.

Note: To create a Managed Server instance in a directory that is outside the domain directory, Oracle recommends using the `pack` command with the `-managed=true` option, or the Configuration Wizard. However, if you create a Managed Server manually by starting it in an empty directory outside the domain directory, you must first copy the `security` directory, and its contents, from the Administration Server directory to your Managed Server directory.

You can specify the path and name of the server root directory for each server instance. You can specify a common server root directory for multiple server instances hosted on a single computer or you can specify a different server root directory for each server. A domain may have one or more server root directories.

3.3.1 Specifying a Server Root Directory

You can specify the path for the server root directory by one of the following means:

- Use the `-Dweblogic.RootDirectory=path` option when starting a Oracle WebLogic Server instance from command line. For example, the following command starts an Oracle WebLogic Server instance and uses `c:\MyServerRootDirectory` as the server root directory:

```
java -Dweblogic.RootDirectory=c:\MyServerRootDirectory weblogic.Server
```

- If you use Node Manager to start an Oracle WebLogic Server instance, you can specify a server root directory with the Root Directory attribute in the WebLogic Server Administration Console on the **Environment > Servers > server_name > Configuration > Server Start** page.

If you do not use one of the above means to specify a server root directory, the path and name of the server root directory depend on whether a server instance is a Managed Server or the Administration Server and whether or not you use Node Manager to start the server instance. These variations are discussed in the next sections.

3.3.2 Server Root Directory for an Administration Server

An Administration Server uses its server root directory as a repository for the domain's configuration data (such as `config.xml`) and security resources (such as the default, embedded LDAP server).

To determine the root directory for an Administration Server, Oracle WebLogic Server does the following:

- If the server's startup command includes the `-Dweblogic.RootDirectory=path` option, then the value of `path` is the server root directory.
- If `-Dweblogic.RootDirectory=path` is not specified, then the working directory is the server root directory.

If Oracle WebLogic Server cannot find a `config.xml` file, then it offers to create one. You can use this method to create a new domain. For more information, see "Using the `weblogic.Server` Command Line to Create a Domain" in the *Command Reference for Oracle WebLogic Server*.

3.3.3 Server Root Directory for a Managed Server Started with Node Manager

If you use the Node Manager to start a Managed Server, the root directory is located on the computer that hosts the Node Manager process. To determine the location of the server's root directory, Oracle WebLogic Server does the following:

- If you specified a root directory in the WebLogic Server Administration Console on the **Environment > Servers > server-name > Configuration > Server Start** page, then the directory you specified is the server root directory.
- If you did not specify a root directory in the WebLogic Server Administration Console, then the server root directory is:

```
ORACLE_HOME\user_projects\domains\domain-name\servers\managed-server-name
```

where `ORACLE_HOME` is the directory in which you installed Oracle WebLogic Server on the Node Manager's host computer.

3.3.4 Server Root Directory for a Managed Server Not Started with Node Manager

If you do not use the Node Manager to start a Managed Server (and therefore use the `java weblogic.Server` command or a script that calls that command), Oracle

WebLogic Server does the following to determine the root directory:

- If the server's startup command includes the `-Dweblogic.RootDirectory=path` option, then the value of *path* is the server's root directory.
- If `-Dweblogic.RootDirectory=path` is not specified, then the working (current) directory is the root directory. For example, if you run the `weblogic.Server` command from `c:\config\MyManagedServer`, then `c:\config\MyManagedServer` is the root directory.

To make it easier to maintain your domain configurations and applications across upgrades of Oracle WebLogic Server software, it is recommended that the server root directory not be the same as the installation directory for the Oracle WebLogic Server software.

Managing Configuration Changes

This chapter describes the change management process in Oracle WebLogic Server 12.1.3 which provides a secure, predictable means for distributing configuration changes in domains. In this process, in-memory changes can be made using editable Configuration MBeans.

This chapter includes the following sections:

- [Section 4.1, "Overview of Change Management"](#)
- [Section 4.2, "Change Management in the Administration Console"](#)
- [Section 4.3, "Configuration Change Management Process"](#)
- [Section 4.4, "Configuration Management State Diagram"](#)
- [Section 4.5, "Restricting Configuration Changes"](#)

4.1 Overview of Change Management

Each domain describes its configuration in an XML document that is located in the domain's configuration directory. At run time, each Oracle WebLogic Server instance in a given domain creates an in-memory representation of the configuration described in this document. The in-memory representation of a domain's configuration is a collection of read-only managed beans (MBeans) called Configuration MBeans.

In addition to the read-only Configuration MBeans, the Administration Server maintains another collection of Configuration MBeans that you can edit (see [Example 4-2](#)). To edit these Configuration MBeans, you obtain a lock using any of the administration tools listed in "Summary of System Administration Tools and APIs" in *Understanding Oracle WebLogic Server*.

While you have the lock on the editable Configuration MBeans, you can save your in-memory changes, which causes the Administration Server to write the changes to a set of pending configuration documents in the domain directory. Oracle WebLogic Server instances do not consume the changes until you activate the changes.

When you activate changes, each server in the domain determines whether it can accept the change. If all servers are able to accept the change, they update their copy of the domain's configuration document. Then they update their working copy of Configuration MBeans and the change is completed (see [Figure 4-3](#)).

Note that Oracle WebLogic Server's change management process applies to changes in domain and server configuration data, not to security or application data.

4.1.1 Changes Requiring Server Restart

Some configuration changes can take effect on the fly, while others require the affected servers to be restarted before they take effect. Configuration changes that can take effect without a server restart are sometimes referred to as *dynamic* changes; configuration changes that require a server restart are sometimes referred to as *non-dynamic* changes. In the WebLogic Server Administration Console, an attribute that requires a server restart for changes to take effect is marked with this icon:



Edits to dynamic configuration attributes become available once they are activated, without restarting the affected server or system resource. Edits to non-dynamic configuration attributes require that the affected servers or system resources be restarted before they become effective.

If an edit is made to a non-dynamic configuration setting, no edits to dynamic configuration settings will take effect until after restart. This is to assure that a batch of updates having a combination of dynamic and non-dynamic attribute edits will not be partially activated.

4.1.2 Configuration Change Tools

As described in "Summary of System Administration Tools and APIs" in *Understanding Oracle WebLogic Server*, you can use a variety of different Oracle WebLogic Server tools to make configuration changes:

- WebLogic Server Administration Console
- WebLogic Scripting Tool
- JMX APIs

Whichever tool you use to make configuration changes, Oracle WebLogic Server handles the change process in basically the same way.

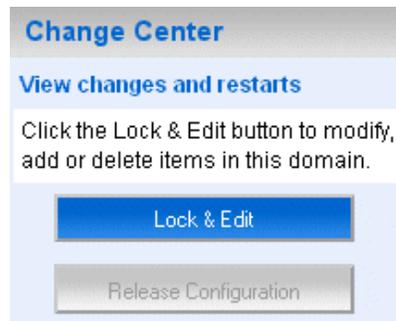
For more detailed information about how configuration changes are carried out through JMX and Configuration MBeans, see "Understanding WebLogic Server MBeans" in *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*. For more detailed information about making configuration changes with WLST, see "Configuring Existing Domains" in *Understanding the WebLogic Scripting Tool*.

4.1.3 Configuration Auditing

Using the WebLogic Auditing provider or another auditing security provider, you can record audit information about changes made to your Oracle WebLogic Server configuration. See "Enabling Configuration Auditing" in *Administering Security for Oracle WebLogic Server*.

4.2 Change Management in the Administration Console

The WebLogic Server Administration Console centralizes the configuration change management process in the Change Center pane:

Figure 4–1 Change Center

If you want to use the WebLogic Server Administration Console to make configuration changes, you must first click the Lock & Edit button in the Change Center. When you click Lock & Edit, you obtain a lock on the editable collection of Configuration MBeans for all servers in the domain (the edit tree).

Note: The domain configuration locking feature is always enabled in production domains. It can be enabled or disabled in development domains. It is disabled by default when you create a new development domain. See "Enable and disable the domain configuration lock" in *Oracle WebLogic Server Administration Console Online Help*.

As you make configuration changes using the WebLogic Server Administration Console, you click Save on the appropriate pages. This does not cause the changes to take effect immediately; instead, when you click Save, you are saving the change to the edit tree and to the `DOMAIN_NAME/pending/config.xml` file and related configuration files. The changes take effect when you click Activate Changes in the Change Center. At that point, the configuration changes are distributed to each of the servers in the domain. If the changes are acceptable to each of the servers, then they take effect. (Note, however, that some changes require a server to be restarted.) If any server cannot accept a change, then all of the changes are rolled back from all of the servers in the domain. The changes are left in a pending state; you can then either edit the pending changes to resolve the problem or revert the pending changes.

For development domains, the WebLogic Server Administration Console has an auto-activate model. Instead of clicking Activate Changes, your changes will be automatically activated when you save them. This is the default behavior in development domains; it cannot be enabled in production domains. You can disable this behavior by deselecting the **Automatically Acquire Lock and Activate Changes** Console preference option.

4.2.1 Taking the Configuration Lock From Another Administrator

In the WebLogic Server Administration Console, an administrator can take a configuration lock from another administrator, make configuration changes, and save these changes. See "Take configuration lock from another administrator" in the *Oracle WebLogic Server Administration Console Online Help*. The administrator you take the lock from will not be able to make any additional changes unless they take the lock back or you submit your changes and release the lock.

If the administrator you take the lock from was using the WebLogic Server Administration Console to make changes, you will inherit any pending changes. If the

other administrator is using another method to make configuration changes, such as WLST, these changes may be lost when you take the lock.

4.3 Configuration Change Management Process

Configuration changes happen in basically the same way, regardless of the JMX tool you choose to use (the WebLogic Server Administration Console, WLST, or JMX APIs). The following steps describe the process in detail, starting from when you first boot the domain's Administration Server:

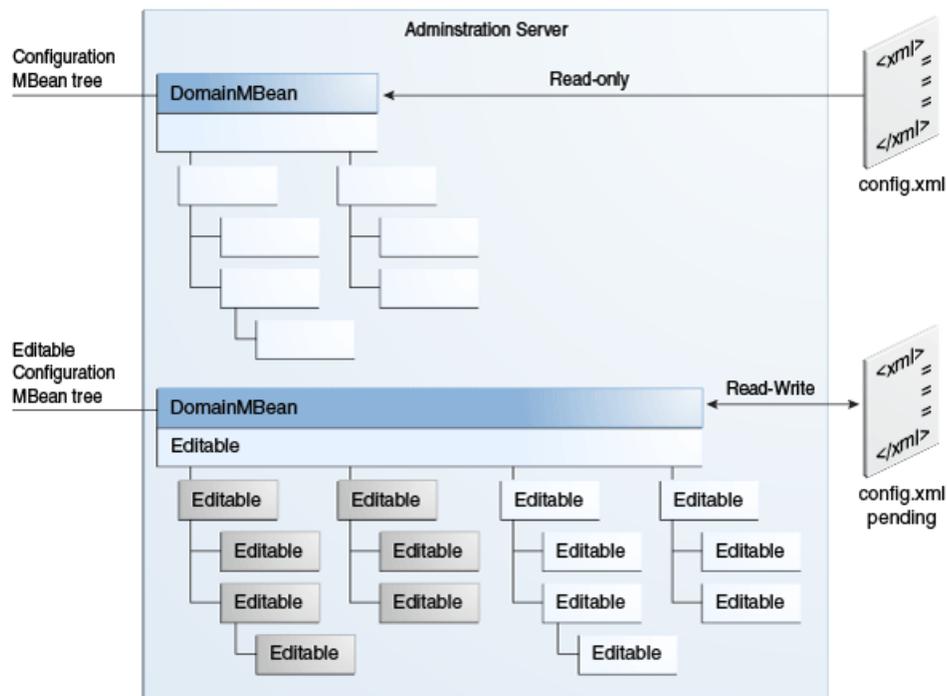
1. When the Administration Server starts, it reads the domain's configuration files, including the `config.xml` file and any subsidiary configuration files referred to by the `config.xml` file and uses the data to instantiate the following MBean trees in memory:
 - A read-only tree of Configuration MBeans that contains the current configuration of resources that are on the Administration Server.
 - An editable tree of all Configuration MBeans for all servers in the domain.

Note: The Administration Server also instantiates a Runtime MBean tree and a DomainRuntime MBean tree, but these are not used for configuration management.

2. To initiate a configuration change, you do the following:
 - a. Obtain a lock on the current configuration.
 - b. Make any changes you desire, using the tool of your choice (the WebLogic Server Administration Console, WLST, the JMX APIs, and such).
 - c. Save your changes to a pending version of the `config.xml` file, using the **Save** button in the WebLogic Server Administration Console; using the WLST `save` command; or using the `save` operation on the `ConfigurationManagerMBean`.
3. The Configuration Manager service saves all data from the edit MBean tree to a separate set of configuration files in a directory named `pending`. See [Figure 4-2](#).

The `pending` directory is immediately below the domain's root directory. For example, if your domain is named `mydomain`, then the default pathname of the `pending config.xml` file is `mydomain/pending/config.xml`.

Figure 4–2 The Administration Server's Pending config.xml File



4. Make additional changes or undo changes that you have already made.
5. When you are ready, activate your changes in the domain, using the **Activate Changes** button in the WebLogic Server Administration Console's Change Center; using the WLST `activate` command; or using the `activate` operation on the `ConfigurationManagerMBean`.

When you activate changes (see [Figure 4–3](#)):

- a. For each server instance in the domain, the Configuration Manager service copies the pending configuration files to a `config` directory in the server's root directory.

If a Managed Server shares its root directory with the Administration Server, `ConfigurationManagerMBean` does not copy the pending configuration files; the Managed Server uses the Administration Server's `config` directory.

- b. Each server instance compares its current configuration with the pending configuration.
- c. Subsystems within each server vote on whether they can consume the new configuration.

If any subsystem indicates that it cannot consume the changes, the entire activation process is rolled back and the `ConfigurationManagerMBean` emits an exception. You can modify your changes and retry the change activation, or you can abandon your lock, in which case the edit Configuration MBean tree and the pending configuration files are reverted to the configuration in the read-only Configuration MBean tree and configuration files.

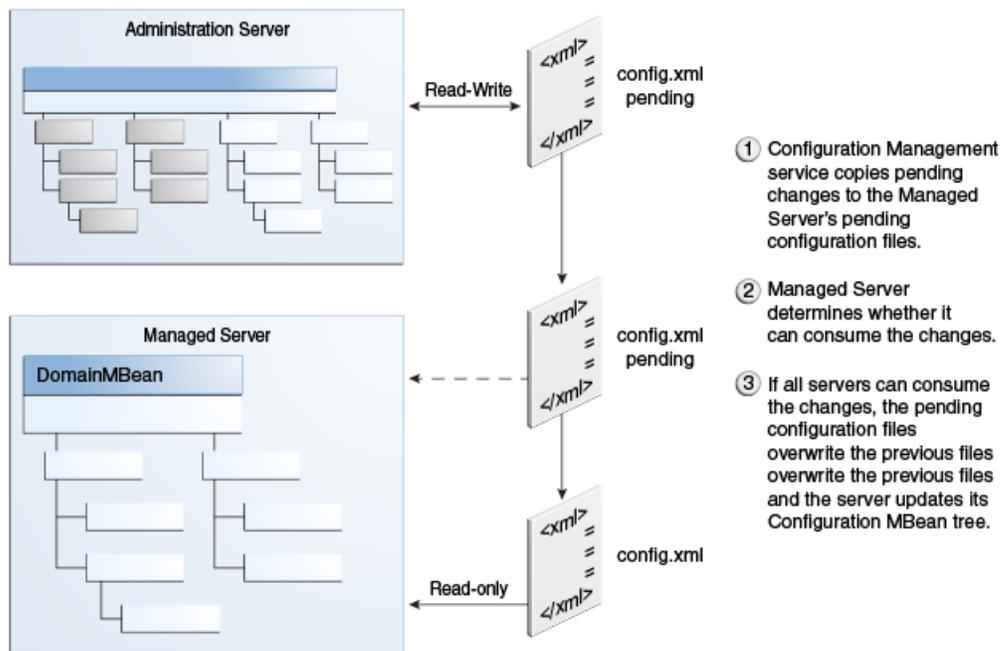
- d. If all subsystems on all servers can consume the change, the Configuration Manager service replaces the read-only configuration files on each server instance in the domain with the pending configuration files.

- e. Each server instance updates its beans and its read-only Configuration MBean tree according to the changes in the new configuration files. In cases that include one or more changes that require the server to be restarted, this occurs the next time the server is restarted.
- f. The pending configuration files are then deleted from the pending directory.

Note: If the Administration Server crashes before completing activation of your changes (step f), upon restart, the Administration Server will recover the pending configuration files saved in step a. In the log file, this process is referred to as `config recovery`.

- 6. You can retain your lock to make additional changes or release it so that others can update the configuration. You can configure a time-out period that causes the Configuration Manager service to abandon a lock.

Figure 4-3 Activating Changes in Managed Servers



4.3.1 Configuration Locks

When you start an edit session, whether you use the WebLogic Server Administration Console, WLST, or the Management APIs, you obtain a lock on the Configuration MBean edit tree.

The configuration change lock does not by itself prevent you from making conflicting configuration edits using the same administrator user account. For example, if you obtain a configuration change lock using the WebLogic Server Administration Console, and then use the WebLogic Scripting Tool with the same user account, you will access the same edit session that you opened in the WebLogic Server Administration Console and you will not be locked out of making changes with the Scripting Tool. You can reduce the risk that such a situation might occur by

maintaining separate administrator user accounts for each person with an administrative role. Similar problems can still occur, however, if you have multiple instances of the same script using the same user account.

To reduce further the risk of this situation, you can obtain an *exclusive* configuration change lock. When you have an exclusive configuration lock, a subsequent attempt to start an edit session by the same owner will wait until the edit session lock is released. To obtain an exclusive configuration lock using WLST, use `true` for the exclusive argument in the `startEdit` command:

```
wls:/mydomain/edit> startEdit(60000, 120000, true)
```

To obtain an exclusive configuration lock using the Management API, use `true` for the exclusive parameter in the `ConfigurationMBean.startEdit` operation:

```
Object [] startEdit(60000, 120000, true)
```

You cannot modify the domain configuration using the compatibility MBean server when either of the following is true:

- there is an existing editing session, or
- there are pending changes saved, but not yet activated from a previous edit session.

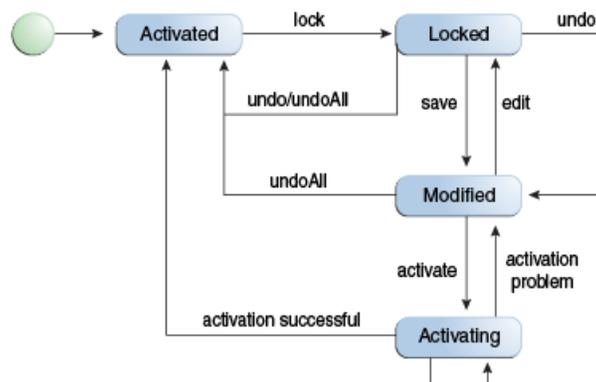
4.3.2 Resolving Change Conflicts

In the event that you have saved more than one change set without activating them and one change would invalidate a prior change, the Change Management service requires you to manually resolve the invalidation before it will save your changes.

4.4 Configuration Management State Diagram

The Configuration Management service follows a series of states, which are described in [Figure 4-4](#).

Figure 4-4 Configuration Management State Diagram



4.5 Restricting Configuration Changes

You can block configuration changes by setting a domain to be read-only. Do this by setting the `EditMBeanServerEnabled` attribute of the `JMXMBean` configuration MBean to `false`, using either WLST or the Management APIs.

Note that once you have set your domain to be read-only, you can no longer edit its configuration using the WebLogic Server Administration Console, WLST online, or the Management APIs. To make the domain editable again, you must either edit the `config.xml` file directly in a text editor, or use WLST offline, and then restart the affected servers.

You should also establish appropriate access controls to the domain's configuration, limiting access to users with the proper security roles. In addition, using the WebLogic Auditing provider or another auditing security provider, you can record audit information about changes made to your Oracle WebLogic Server configuration. See "Enabling Configuration Auditing" in *Administering Security for Oracle WebLogic Server*.

Server Templates

This chapter introduces server templates and describes how to create, configure, and use server templates in WebLogic Server 12.1.3.

This chapter includes the following sections:

- [Section 5.1, "What are Server Templates?"](#)
- [Section 5.2, "Why Do You Use Server Templates?"](#)
- [Section 5.3, "Using Server Templates"](#)
- [Section 5.4, "Server Templates Example"](#)

5.1 What are Server Templates?

A server template contains common, non-default settings and attributes that you can apply to a set of server instances, which then inherit the template configuration. You can define server-specific values for server instances that require unique attributes by overriding the server template in the server instance definition. At runtime, the domain merges the attributes configured in the server template with any server-specific overrides so that all attributes are defined for all server instances.

[Example 5-1](#) shows an example `config.xml` file that uses a server template. In this example, common configuration is defined in the `<server-template>` element. The individual server definitions set server-specific attributes (`name`, `listen-port`) and specify the server template upon which to base additional configuration (`my-cluster-server-template`).

Example 5-1 Example config.xml File Using a Server Template

```
<server>
  <name>my-server-1</name>
  <listen-port>7010</listen-port>
  <server-template>my-cluster-server-template</server-template>
</server>

<server>
  <name>my-server-2</name>
  <listen-port>7020</listen-port>
  <server-template>my-cluster-server-template</server-template>
</server>

<server-template>
  <name>my-cluster-server-template</name>
  <accept-backlog>2000</accept-backlog>
  <cluster>my-cluster</cluster>
```

```
<restart-max>10</restart-max>
<reverse-dns-allowed>true</reverse-dns-allowed>
<startup-timeout>600</startup-timeout>
</server-template>
```

5.2 Why Do You Use Server Templates?

Server templates enable you to easily manage configuration for a group of server instances in one centralized location. You can define common configuration attributes in a server template and then apply the template to other server instances without having to manually configure each one. If you need to define any server-specific attributes, you can easily override the server template at the individual server level.

If you need to update an attribute across all server instances, you can simply change the value in the server template and the new value takes effect in all of the server instances that use that server template. You do not need to write a WLST script to update all the servers in the domain.

Server templates also help with performance and startup time. Defining common attributes once in the server template, instead of setting the same attribute in each server instance definition, reduces the size of the `config.xml` file.

You also use server templates to create dynamic clusters. For more information, see "Dynamic Clusters" in *Administering Clusters for Oracle WebLogic Server*.

5.3 Using Server Templates

You create server templates using WebLogic Scripting Tool (WLST) or the WebLogic Server Administration Console. [Example 5-2](#) demonstrates using WLST. For information about using the WebLogic Server Administration Console, see "Create server templates" in the *Oracle WebLogic Server Administration Console Online Help*. To configure server templates, you can use any of the administration tools listed in "Summary of System Administration Tools and APIs" in *Understanding Oracle WebLogic Server*.

If needed, you can define server-specific attributes or override any server template value directly in the individual server definition. For example, some server-specific attributes, such as `Notes`, `ConsensusProcessIdentifier`, `InterfaceAddress`, and so on, are more useful when defined at the individual server level.

5.3.1 Overriding Attributes

A server template contains different types of attributes, including primitive datatypes, singleton elements, and arrays of configuration elements. To override a server template value, you define the attribute in the individual server definition.

WebLogic Server determines which attribute value to use based on the following override hierarchy:

- If you do not define an attribute in the individual server configuration or in the server template, then the default value is used.
- If you define an attribute in the server template, then the server template overrides the default value. The server template value is used.
- If you define an attribute in the individual server configuration, then the server configuration overrides the server template value and the default value. The individual server configuration value is used.

Note: Once you override a value in the individual server configuration, this value always takes precedence over the value in the server template. To remove the overridden value and use the value in the server template, use the `unset` attribute command.

At runtime, the domain merges the attributes defined in the server template and individual server definitions to return the desired configuration.

A server instance can override attributes from a server template, but the overrides for each type of attribute differ. For primitive datatypes, the server configuration value overrides the value in the server template. For example, if the server template defines the listen port to be 7101, but you set an explicit listen port value in the server configuration, the server instance value overrides the server template listen port value.

A server instance also overrides values for attributes contained within a child singleton element. For example, if the server template defines a common root directory value in the `<server-start>` child element, but you set an explicit root directory value in the `<server-start>` element in the server configuration, the server instance overrides the server template root directory value.

For child array configuration elements, the server instance cannot override individual child elements or attributes within a child element within an array. The server configuration can only override the entire array of configuration elements. For example, if a server template defines an array of `NetworkAccessPoint` elements, and the server instance configuration defines an array of `NetworkAccessPoints`, the server instance `NetworkAccessPoints` override the values in the server template.

The following example demonstrates overriding `NetworkAccessPoints` in a server instance by using WLST, and includes inline comments.

```
#
#A server template named Cluster Template defines a NetworkAccessPoint named
#AccessPoint-1. The server instance s1 references Cluster Template as its
#template. A ls() command on s1 shows the following:
#

wls:/mydomain/edit/Servers/s1/NetworkAccessPoints !> ls

drw- AccessPoint-1

#
#Server instance s1 then defines a NetworkAccessPoint at the server instance
#level.
#

wls:/mydomain/edit/Servers/s1 !>
cmo.createNetworkAccessPoint("AccessPoint-2")

#
#This results in an override of the NetworkAccessPoints from the Cluster Template.
#AccessPoint-1 is no longer present in s1 and only AccessPoint-2 is defined in s1.
#

wls:/mydomain/edit/Servers/s1/NetworkAccessPoints !> ls

drw- AccessPoint-2
```

5.3.2 Using Macros

Server templates contain common attributes and provide the same values for each server instance that inherits configuration from the template. In some cases, you may want to define a standard value for an attribute, but the value must be slightly different for each server instance. You can define the server-specific attribute value in each server instance, but if the attribute follows a pattern, you may be able to use a macro.

You can define a macro for any string attribute in a server template. Macros are case sensitive and cannot be used for integers or references to other configuration elements. The valid macros available for use in server templates are:

- `${id}`: Instance ID of the dynamically created server; this ID starts at 1.
- `${serverName}`: The name of the server to which this element belongs.
- `${clusterName}`: The name of the cluster to which this element belongs. If this element does not belong to a cluster, then an empty string is substituted.
- `${domainName}`: The name of the domain to which this element belongs.
- `${system-property-name}`: If this is not one of the predefined macro names listed previously, then it is evaluated as a system property, and the value is returned. If the system property does not exist, then an empty string is substituted.

5.4 Server Templates Example

[Example 5-2](#) demonstrates using WLST to create two server instances that use the same server template. The example includes inline comments and describes how to:

1. Create a server template and name it `my-cluster-server-template`.
2. Set all of the common attributes in `my-cluster-server-template`.
3. Create server instances `my-server1` and `my-server2`, and assign the `my-cluster-server-template` to each one.
4. Set server-specific attributes.

Example 5-2 Using Server Templates with WLST

```
#
# This example demonstrates the WLST commands needed to create a server template
# (my-cluster-server-template) and server instances (my-server-1, my-server-2)
# that use the server template. The creation of the cluster (my-cluster) and
# machines (PrimaryMachine, SecondaryMachine) are not included in this example;
# they must be created before running this script. To keep this example
# simple, error handling was omitted.
#
connect()
edit()
startEdit()
#
# Create the server template and set the common attributes that are used in all
# servers. Attributes such as Accept Backlog, Auto Restart, etc. are the same
# across all cluster members.
#
serverTemplate=cmo.createServerTemplate("my-cluster-server-template")
serverTemplate.setAcceptBacklog(2000)
serverTemplate.setAutoRestart(true)
serverTemplate.setRestartMax(10)
```

```

serverTemplate.setReverseDNSAllowed(true)
serverTemplate.setStagingMode("external_stage")
serverTemplate.setStartupTimeout(600)
myCluster=cmo.lookupCluster("my-cluster")
serverTemplate.setCluster(myCluster)
#
# Create the individual server instances, set the server specific attributes, and
# set the template so the server uses the common attributes from the template.
# Attributes such as cluster weight, listen port, and machine are server-specific
# so they are not included in the server template.
#
server1=cmo.createServer("my-server-1")
server1.setClusterWeight(70)
server1.setListenPort(7010)
primary=cmo.lookupMachine("PrimaryMachine")
server1.setMachine(primary)
server1.setServerTemplate(serverTemplate)
#
server2=cmo.createServer("my-server-2")
server2.setClusterWeight(90)
server2.setListenPort(7020)
secondary=cmo.lookupMachine("SecondaryMachine")
server2.setMachine(secondary)
server2.setServerTemplate(serverTemplate)
#
# activate the changes
#
activate()

```

The resulting config.xml file is:

```

<server>
  <name>my-server-1</name>
  <cluster-weight>70</cluster-weight>
  <listen-port>7010</listen-port>
  <machine>PrimaryMachine</machine>
  <server-template>my-cluster-server-template</server-template>
</server>

<server>
  <name>my-server-2</name>
  <cluster-weight>90</cluster-weight>
  <listen-port>7020</listen-port>
  <machine>SecondaryMachine</machine>
  <server-template>my-cluster-server-template</server-template>
</server>

<server-template>
  <name>my-cluster-server-template</name>
  <accept-backlog>2000</accept-backlog>
  <auto-restart>true</auto-restart>
  <cluster>my-cluster</cluster>
  <restart-max>10</restart-max>
  <reverse-dns-allowed>true</reverse-dns-allowed>
  <staging-mode>external_stage</staging-mode>
  <startup-timeout>600</startup-timeout>
</server-template>

```

