# Oracle® Fusion Middleware

# Understanding Domain Configuration for Oracle WebLogic Server

14c (14.1.2.0.0)
F61328-01
December 2024

ORACLE

Oracle Fusion Middleware Understanding Domain Configuration for Oracle WebLogic Server, 14c (14.1.2.0.0)

F61328-01

# Contents

## 4 Server Templates

# Preface

This documentation describes the Oracle WebLogic Server domains and how the domains are configured.



## Audience

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

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

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.

## 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.

## Diversity and Inclusion

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

# Related 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 Jakarta Management Applications for Oracle WebLogic Server
- Oracle WebLogic Server MBean Reference
- Monitoring Oracle WebLogic Server with SNMP

For more 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 Remote Console Online Help

### Samples and Tutorials

Oracle provides a variety of code examples and tutorials that show WebLogic Server configuration and API use, and provide practical instructions on how to perform key development tasks. For more information, see Sample Applications and Code Examples in *Understanding Oracle WebLogic Server*.

### New and Changed WebLogic Server Features

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

# Conventions

The following text conventions are used in this document:

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

1

# Understanding Oracle WebLogic Server Domains

Learn about Oracle WebLogic Server domains and their contents, which include an Administration Server and usually one or more Managed Servers.

The following topics provides an overview of WebLogic Server domain, its contents, and certain restrictions associated to the domain configuration:

- What Is a Domain?
  An Oracle WebLogic Server administration **domain** is a logically related group of Oracle WebLogic Server resources.
- 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.
- Contents of a Domain
  An Oracle WebLogic Server domain might consist of an Administration Server, Managed Servers, and the resources and services that Managed Servers and deployed applications require.
- Domain Configuration Restrictions
  In designing your domain configuration, note the restrictions mentioned.
- Domain and Server Name Restrictions
  When naming a domain or a server instance, note the restrictions and considerations mentioned.
- Development, Production, and Secured Production Modes
  You can configure a WebLogic domain to run in one of three modes: development mode, production mode, or secured production mode.

## What Is a Domain?

An Oracle WebLogic Server administration **domain** is a logically related group of Oracle WebLogic Server resources.

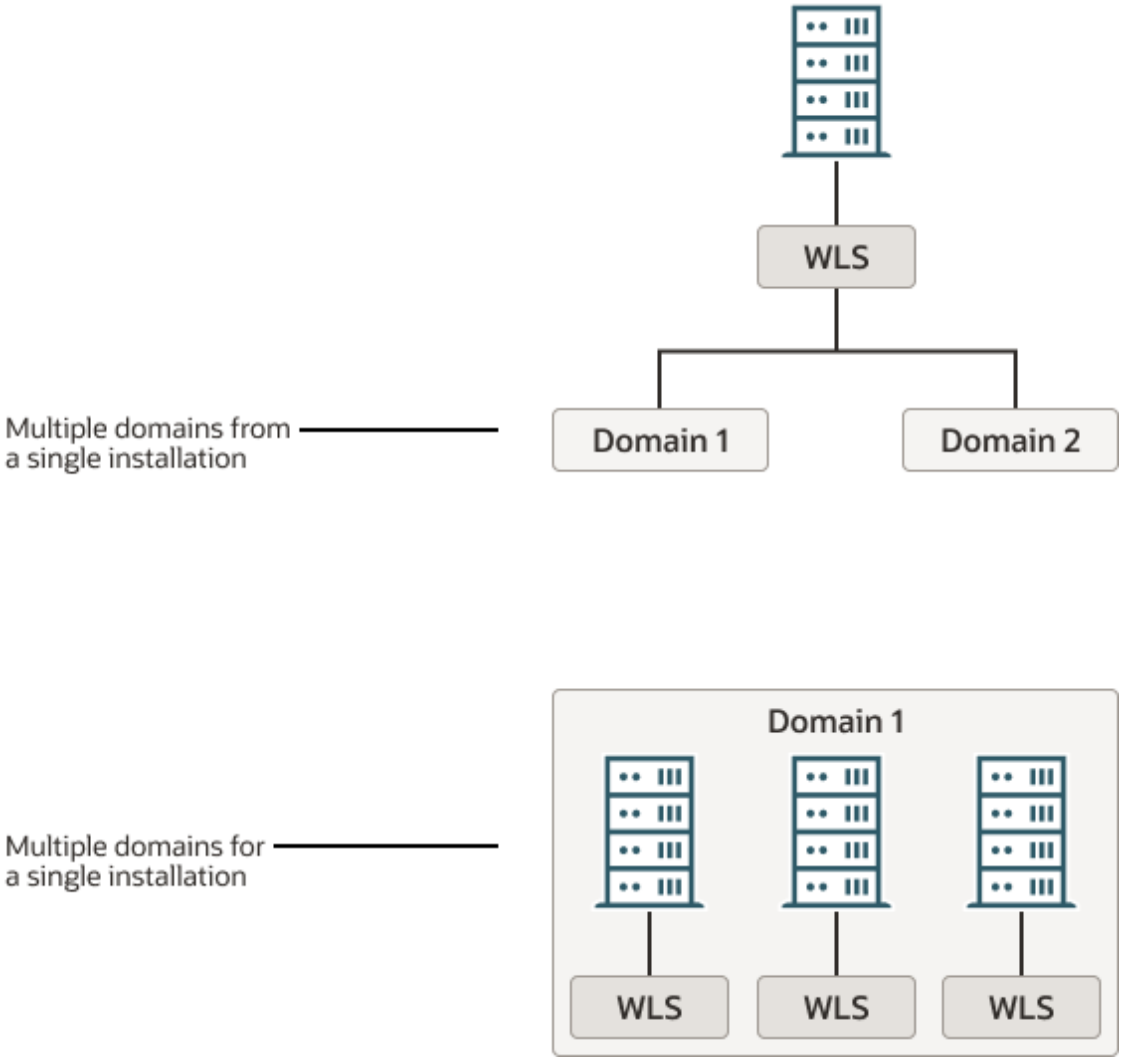
Domains include a special Oracle WebLogic Server instance called the **Administration Server**, which is the central point from which you configure and manage all resources in the domain. Usually, you configure a domain to include additional Oracle WebLogic Server instances called **Managed Servers**. 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.

## 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.

Figure 1-1 shows multiple domains from a single installation and multiple domains for a single installation.

**Figure 1-1 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. A domain can be made up from multiple domain directories spanning multiple physical hosts. They all just need to be configured for the same domain.
- 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.

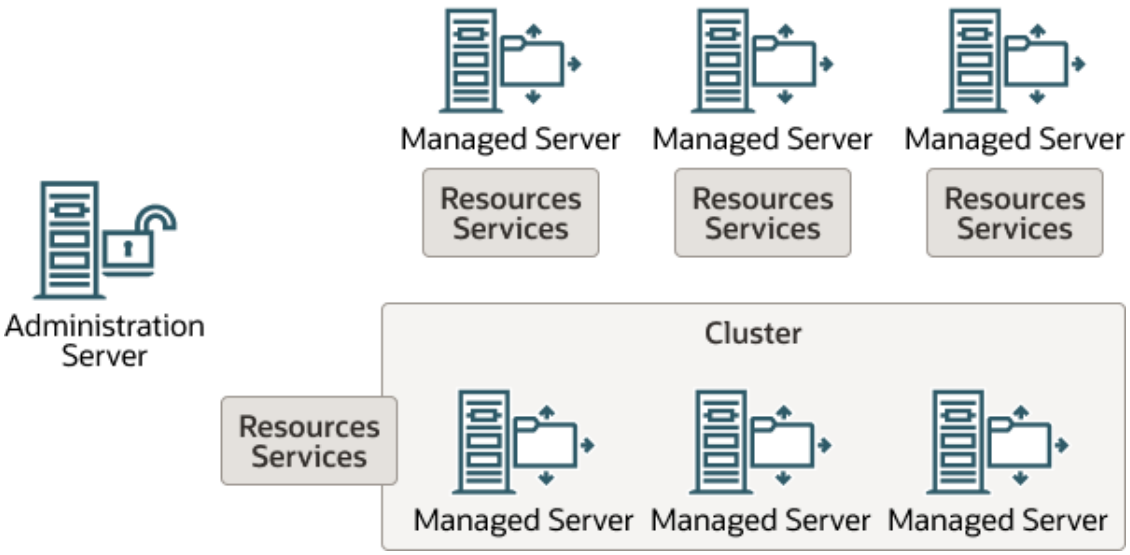
You can create a simple domain that consists of a single server instance. This single instance acts as an Administration Server and hosts the applications that you are developing. The wl_server domain that you can install with Oracle WebLogic Server is an example of this type of domain.

# Contents of a Domain

An Oracle WebLogic Server domain might consist of an Administration Server, Managed Servers, and the resources and services that Managed Servers and deployed applications require.

Figure 1-2 shows a production environment that contains an Administration Server, three stand-alone Managed Servers, and a cluster of three Managed Servers.

**Figure 1-2 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.

- Administration Server
- What Happens if the Administration Server Fails?
- Managed Servers and Managed Server Clusters
- Resources and Services

## 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*.

## 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 about re-starting an Administration Server, see Avoiding and Recovering From Server Failure in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

## 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*.

## 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. 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 Mangers 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.

# Domain Configuration Restrictions

In designing your domain configuration, note the restrictions mentioned.

- Each domain requires its own Administration Server for performing management activities. When you use the WebLogic Remote Console or Fusion Middleware Control (FMWC) 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. See Relationship Between Clusters and 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.
- Server instances cannot have the same name as the domain.

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.

# Domain and Server Name Restrictions

When naming a domain or a server instance, note the restrictions and considerations mentioned.

- Use alphanumeric names for domain and server instances.
- Apart from the alphanumeric characters, a valid domain or server name can include only the following characters:
  - Underscore (_)
  - Hyphen (-)
  - Period (.)
- Use a unique name for each server instance. Within a domain, all configuration objects, including server instances, machines, clusters, JDBC connection pools, virtual hosts, and other resource types, must have unique names and must not use the same name as the domain.
- Server names are for identification purposes only. Server names are not used as part of the URL for applications deployed on the server instance. The server name displays in the Remote Console or Fusion Middleware Control. If using WebLogic Server command-line utilities, you use the server name to identify the server instance.
- Once you have created a server instance, you cannot change its name. To use a new name, clone the server instance and provide a new name for the clone.

# Development, Production, and Secured Production Modes

You can configure a WebLogic domain to run in one of three modes: development mode, production mode, or secured 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 more stringent, such as requiring a user name and password to deploy applications and start the Administration Server. In secured production mode, the default authorization and role mapping policies are more restrictive, and the security configuration are set to more secure default values.

As of WebLogic Server 14.1.2.0.0, when you enable production mode, it automatically enables the stricter settings of secured production mode. If you truly want to implement the more moderate default settings of production mode, you must explicitly disable secured production mode.

There are also several differences between the default settings in development mode, production mode, and secured production 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 details about how the domain mode you select determines the default security configuration for your domain, see How Domain Mode Affects the Default Security Configuration in *Securing a Production Environment for Oracle WebLogic Server*. This topic explains how the default settings for SSL, the administrative and listen ports, auditing, logging, deployment of internal applications, the domain configuration lock, and other configuration settings differ depending on the domain mode that is enabled.

> **Note:**
>
> Oracle recommends that you do not enable the Web Services Test Client in production mode. See Using the Web Services Test Client in *Administering Web Services*.

### Modifying Domain Modes

You can configure the domain mode at the time you create the domain, or you can also change the domain mode any time later.

However, you cannot configure domains created prior to 12.2.1.3 to run in secured production mode; only domains running in 12.2.1.3 and later can be configured for secured production mode.

Use one of the following tools to set the domain mode:

- WebLogic Remote Console - see Change the Domain Mode in the *Oracle WebLogic Remote Console Online Help*
- WLST Offline - see the setOption WLST offline command in *WLST Command Reference for Oracle WebLogic Server*
- WLST Online - see Using WLST Online to Update an Existing WebLogic Domain in *Understanding the WebLogic Scripting Tool*
- Configuration Wizard - see Creating a WebLogic Domain in *Creating WebLogic Domains Using the Configuration Wizard*
- Pack/Unpack - see the `server_start_mode` parameter in The Pack Command in *Creating Templates and Domains Using the Pack and Unpack Commands*

- Fusion Middleware Control - see Configure domain security in *Administering Oracle WebLogic Server with Fusion Middleware Control*

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 remains in use.

To enhance the security of your existing production domains, do the following:

1. To change the value of the `-Xverify` flag from `none` to `all`, modify the domain start scripts. This change enables bytecode verification. For more information about bytecode verification, see *Java Language Security and Bytecode Verification* in the *Java Security Overview* at `https://docs.oracle.com/en/java/javase/17/security/java-security-overview1.html#GUID-2EF91196-D468-4D0F-8FDC-DA2BEA165D10`.
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 must 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 start script of the server.

2

# Domain Configuration Files

The domain configuration files are XML documents that Oracle WebLogic Server uses to persist the configuration of a domain.
This chapter includes the following sections:

- Overview of Domain Configuration Files
  Each domain describes its configuration in an XML document that is located in the configuration directory of the domain. At run time, each Oracle WebLogic Server instance in a given domain creates an in-memory representation of the configuration described in this document.
- Domain Directory Contents
  By default, Oracle WebLogic Server creates domain directories under Oracle Middleware *ORACLE_HOME*/user_projects/domains directory. As a best practice, Oracle recommends that you create your domain directories elsewhere.
- 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 path names in the server's configuration.

## Overview of Domain Configuration Files

Each domain describes its configuration in an XML document that is located in the configuration directory of the domain. 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 both Write and Read privileges, for the domain to function properly, even if no changes are made to the configuration after it is created. These privileges are required because WebLogic Server performs its own internal deployments, and configuration files may be rewritten when the server is restarted.
>
> Do not add non-configuration files in the `config` directory or subdirectories. Non-configuration files include log (.log) and lock (.lck) files. Administration Server replicates the `config` directory in all Managed Server instances. Storing non-configuration files in the `config` directory can cause performance issues in the domain.

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. Sometimes, 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 Managing Configuration Changes).

- Editing Configuration Documents
- Security Credentials in Configuration Files
- Configuration File Archiving

## 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 Remote Console, Fusion Middleware Control (FMWC), 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/2.0/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.

## 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*. After you encrypt the credentials, include the encrypted values in your `config.xml` file in elements as shown in Example 2-1:

**Example 2-1 Configuring Encrypted Credentials**

```
<security-configuration>
   <credential-encrypted>{AES256}encypted-value-here</credential-encrypted>
</security-configuration>
<embedded-ldap>
   <credential-encrypted>{AES256}encypted-value-here</credential-encrypted>
</embedded-ldap>
```

## 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`.

In the Remote Console, you can configure archive file retention on the **Environment**: **Domain**: **General** page, **Show Advanced Fields**. 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 2-2.

**Example 2-2 Configuring Archive Files**

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

# Domain Directory Contents

By default, Oracle WebLogic Server creates domain directories under Oracle Middleware *ORACLE_HOME*`/user_projects/domains` directory. As a best practice, Oracle recommends that you create your domain directories elsewhere.

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

The following section describes the contents of the domain directory and its subfolders. Here *domain-name*, *deployment-name*, and *server-name* represent names that you define when you create a domain.

## domain-name

The name of this directory is the name of the domain. The name must conform to the guidance in Domain and Server Name Restrictions.

## 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 Jakarta 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

## 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 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. See Administering Server Startup and Shutdown for Oracle WebLogic Server.

## config

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

## 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.

## config/diagnostics

This directory contains system modules for instrumentation in the WebLogic Diagnostic Framework. See Configuring and Using the Diagnostics Framework for Oracle WebLogic Server.

## config/jdbc

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

## config/jms

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

## config/lib

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

## config/nodemanager

This directory holds information for connection to the Node Manager. See Node Manager Configuration and Log Files in *Administering Node Manager for Oracle WebLogic Server*.

## 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 current realm of the domain. See *Understanding Security for Oracle WebLogic Server*.

## 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-1.jar`, `config-2.jar`, and so on.

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.

## init-info

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

## lib

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

## 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. See Managing Configuration Changes.

## 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

See *Understanding Security for Oracle WebLogic Server*.

## servers

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

For more information about server naming convention, see Domain and Server Name Restrictions.

## servers/*server-name*

This directory is the server directory for the Oracle WebLogic Server instance with the same name as the directory. The name must conform to the guidance in Domain and Server Name Restrictions.

## 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.

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

This directory is a cache for compiled EJBs.

## 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.

## 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.

## 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`.

## 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.

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

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

### 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.

### 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.

### 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.

### tmp

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

## 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 path names 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 creates its own root directory. The server copies data from the domain directory to this root directory and writes 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 has one or more server root directories.

- Specifying a Server Root Directory
- Server Root Directory for an Administration Server
- Server Root Directory for a Managed Server Started with Node Manager
- Server Root Directory for a Managed Server Not Started with Node Manager

## 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 Remote Console on the **Environment**: **Servers**: *myServer*: **Advanced**: **Node Manager** page. For more information, see Configure Startup Arguments for a Managed Server in the *Oracle WebLogic Remote Console Online Help*.

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.

## 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. See Using the weblogic.Server Command Line to Create a Domain in the *Command Reference for Oracle WebLogic Server*.

## 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 Remote Console on the **Environment**: **Servers**: *myServer*: **Advanced**: **Node Manager** page, then the directory you specified is the server root directory.
- If you did not specify a root directory in the WebLogic Remote 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.

## 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.

3

# Managing Configuration Changes

The change management process in Oracle WebLogic Server provides a secure and 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:

- Overview of Change Management
  To provide a secure, predictable means for distributing configuration changes in a domain, WebLogic Server imposes a change management process that loosely resembles a database transaction.
- Change Management in the Remote Console
  After you connect an Administration Server to the WebLogic Remote Console, you are ready to make configuration changes to your domain.
- Configuration Change Management Process
  Configuration changes happen in basically the same way, regardless of the method you choose to use (the WebLogic Remote Console, FMWC, WLST, JMX or REST APIs).
- Configuration Management State Diagram
  The Configuration Management service follows a series of states.
- 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.
- Configuration Overriding
  Configuration overriding lets administrators place configuration information, contained in an XML file, in a known location where running servers identify and load it, overriding aspects of the existing configuration.
- Using Named Concurrent Edit Sessions
  In prior releases, WebLogic Server supported only one active configuration edit session at a time. The system administrator got a global edit lock, made changes, and then activated them. Other administrators could not make changes at the same time without first taking the configuration lock from another administrator, then editing the same active edit session. However, now WebLogic Server enables multiple, named concurrent edit sessions, which allows more than one administrator to make configuration changes in separate edit sessions at the same time.

## Overview of Change Management

To provide a secure, predictable means for distributing configuration changes in a domain, WebLogic Server imposes a change management process that loosely resembles a database transaction.

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 Figure 3-1). 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 cause 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 3-2).

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

- Changes Requiring Server Restart
- Configuration Change Tools
- Configuration Auditing

## 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 Remote 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.

## 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 Remote Console
- Fusion Middleware Control (FMWC)
- WebLogic Scripting Tool

- RESTful management APIs
- JMX APIs

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

For more 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 information about making configuration changes with WLST, see Configuring Existing Domains in *Understanding the WebLogic Scripting Tool*.

## 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*.

# Change Management in the Remote Console

After you connect an Administration Server to the WebLogic Remote Console, you are ready to make configuration changes to your domain.

In the WebLogic Remote Console, it is a generally two step process to apply configuration changes to your domain. For detailed change management information, see Configuring a WebLogic Server Domain in the *Oracle WebLogic Remote Console Online Help*.

# Configuration Change Management Process

Configuration changes happen in basically the same way, regardless of the method you choose to use (the WebLogic Remote Console, FMWC, WLST, JMX or REST APIs).

The following steps describe the process in detail, starting from when you first boot the domain's Administration Server:

> **Note:**
>
> Do not add non-configuration files in the `config` directory or subdirectories. Non-configuration files include log (.log) and lock (.lck) files. Administration Server replicates the `config` directory in all Managed Server instances. Storing non-configuration files in the `config` directory can cause performance issues in the domain.

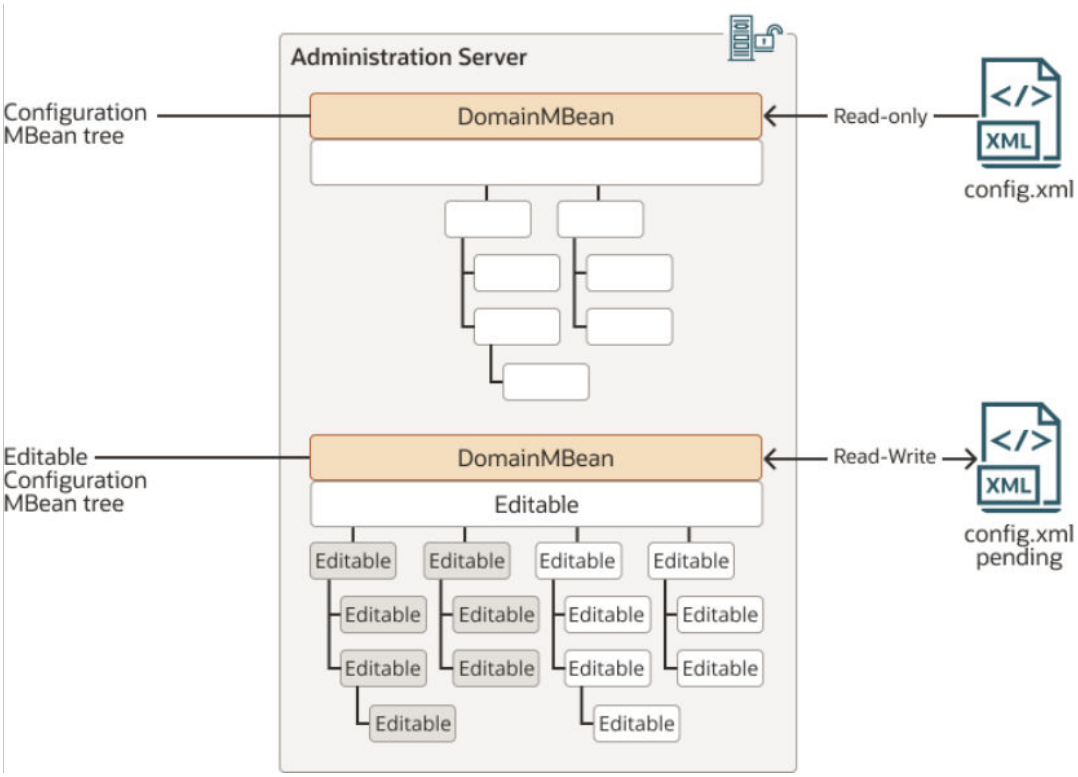
1. When the Administration Server starts, it reads the domain's configuration files, including the `config.xml` file and any subsidiary configuration files referred to by the `config.xml` file and uses the data to instantiate the following MBean trees in memory:
   - A read-only tree of Configuration 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 Remote Console, FMWC, 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 Remote 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 3-1.

   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 3-1 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, in the Remote Console, open the Shopping Cart and then click **Commit Changes**; using the WLST `activate` command; or using the `activate` operation on the `ConfigurationManagerMBean`.

When you activate changes (see Figure 3-2):

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.

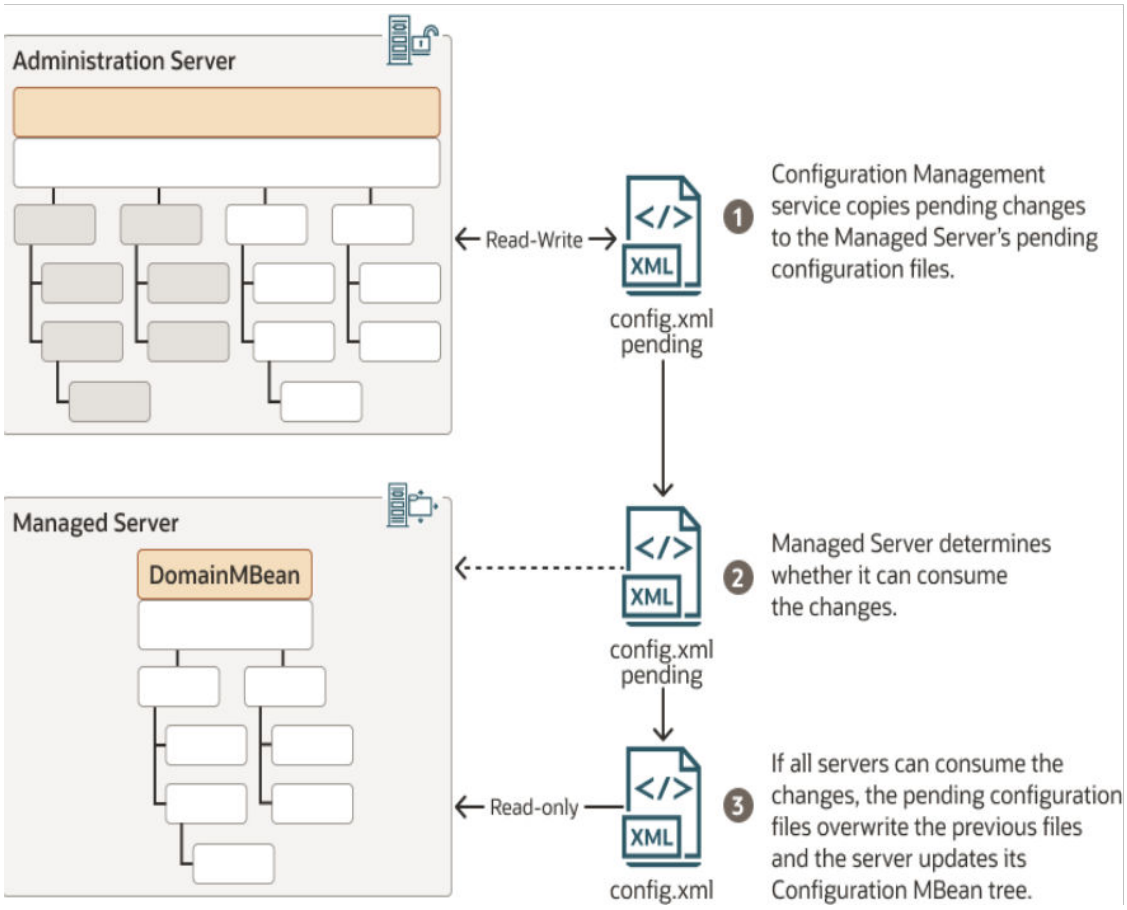
e. Each server instance updates its beans and its read-only Configuration MBean tree according to the changes in the new configuration files. In cases that include one or more changes that require the server to be restarted, this occurs the next time the server is restarted.

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 5.f), upon restart, the Administration Server will recover the pending configuration files saved in step 5.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 3-2 Activating Changes in Managed Servers**




- Configuration Locks
- Resolving Change Conflicts

# Configuration Locks

When you start an edit session, whether you use the WebLogic Remote 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 Remote 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 Remote 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.

## Resolving Change Conflicts

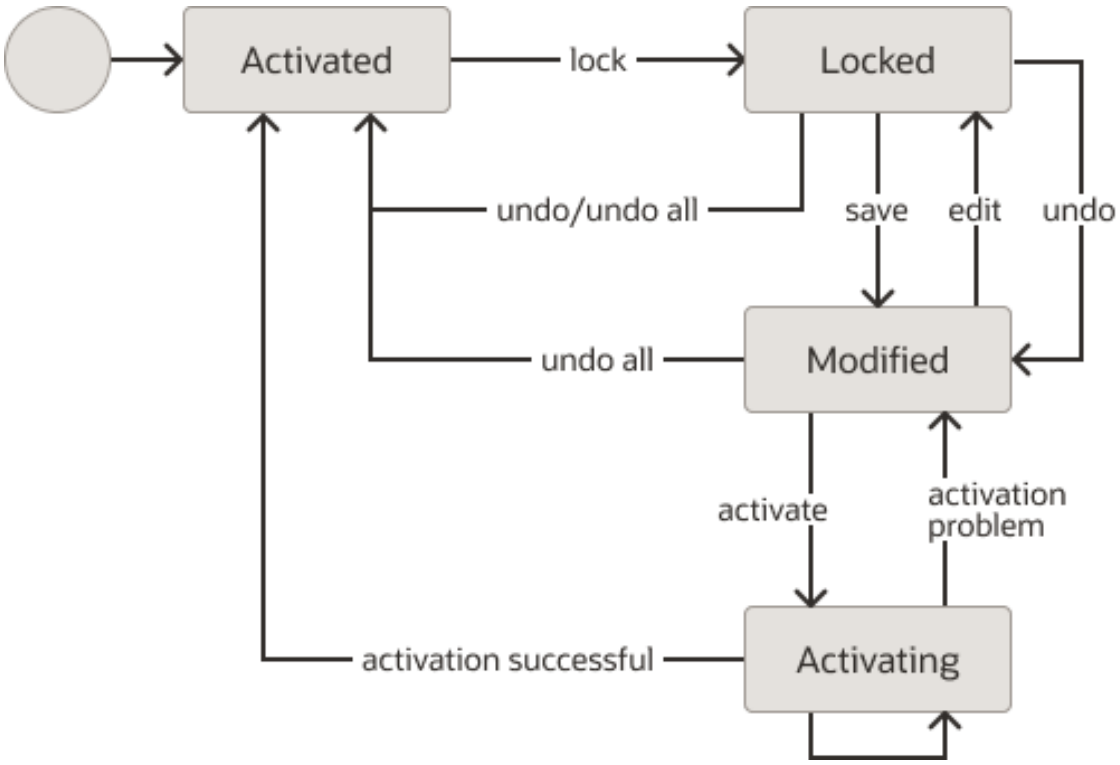
In the event that you have saved more than one change set without activating them and one change would invalidate a prior change, the Change Management service requires you to manually resolve the invalidation before it will save your changes.

# Configuration Management State Diagram

The Configuration Management service follows a series of states.

These states are described in Figure 3-3.

**Figure 3-3 Configuration Management State Diagram**



# 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.

Once you have set your domain to be read-only, you can no longer edit its configuration using the WebLogic Remote 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 must 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*.

# Configuration Overriding

Configuration overriding lets administrators place configuration information, contained in an XML file, in a known location where running servers identify and load it, overriding aspects of the existing configuration.

In large clusters or complex distributed environments, such as in cloud platforms, setting and removing configuration properties can be challenging due to network, software, and scaling issues. Situations arise where you might want or need to run your servers in Managed Server Independence Mode (MSI) mode. You can use configuration overriding (also called *situational configuration*) to change settings such as server debugging flags, timeout values, or diagnostics settings, without running the Administration Server. The configuration overrides are targeted to Managed Servers but can also be applied to the Administration Server. When using configuration overriding, there is no synchronization between the Administration Server and Managed Servers. Configuration overriding bypasses the Administration Server as the primary means of distributing configuration information to Managed Servers.

Situational configuration lets you place a file containing configuration changes (overrides) that need to be added to a running WebLogic Server instance. The overrides may have an expiration time to ensure that the configuration changes do not remain in effect longer than necessary or they can be permanent; no expiration time is specified. In an earlier release of WebLogic Server, an overridden domain configuration required an expiration time and was therefore always temporary. However, this limitation is now removed. For more information, see “Temporary Configuration Overriding” in *Understanding Domain Configuration for Oracle WebLogic Server*.

Consider this scenario. In certain environments, you might want to allow for different size clusters. You could have a base configuration for all environments, combined with configuration overrides that provide the appropriate configurations for small, medium, or large clusters. This allows the common, shared configuration to be separate from the cluster-size specific configurations. For example, certain servers are needed only in the large cluster environment. In the small cluster environment, the administrator does not want these servers to be present. The administrator creates a situational configuration file that deletes the servers. At runtime, the situational configuration file is loaded and the unwanted servers are removed from the configuration.

Situational configuration:

- Applies a configuration change to a running system; it does not impact the stored configuration.
- Employs an XML fragment, in a file, which contains values to be changed and context information. The file is used to update configuration information coming from the `config.xml` file or system resource files that exist in the `jdbc`, `jms`, and `diagnostics` subdirectories of the domain `config` directory.
- Can replace configuration element values, add new configuration elements, or delete existing configuration elements.
- Can update dynamic and non-dynamic attributes. However, when changes include non-dynamic modifications that were applied after the server was started, the server will not update a situational configuration change until the next server start.
- May contain an expiration time for the configuration override, but it is not required. Overrides are reverted when the XML file is removed, renamed, or expired.
- Contains an option to *require* that situational configuration files be present, and if not found, then the server startup fails.
- Does not require involvement by the Administration Server and the Managed Servers do not synchronize their values in any way.

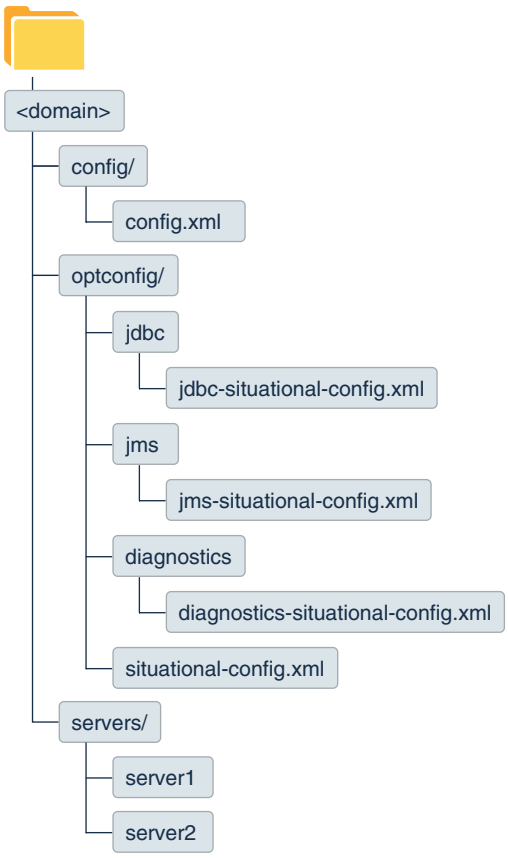
If a value which is currently overridden by a situational configuration is changed in an edit session, that change will *not* take effect until the situational configuration overrides are removed. Situational configuration will log a message in the server's log when it is applied and expired or removed.

- About Situational Configuration Files, Names and Locations
- Loading Situational Configurations
- Monitoring Situational Configuration Changes

## About Situational Configuration Files, Names and Locations

Situational configuration files end with the suffix "`situational-config.xml`" and are domain configuration files, which, by default, reside in an `optconfig` directory. In the case of JDBC, JMS, and diagnostic system resources, they reside in `jdbc`, `jms`, and `diagnostics` subdirectories of `optconfig`.

**Figure 3-4 Domain Configuration File Directories Layout**




Administrators create, update, and delete `situational-config.xml` files. If the domains are located on separate systems, the situational configuration XML files must be copied into each `optconfig` directory.

For JDBC, JMS, and diagnostic system resource files, the situational configuration file name *must* match the system resource file name, then add the `situational-config.xml` suffix. For example, a JMS system resource named *myqueues-jms.xml*, can be overridden only by a file with the name *myqueues-jms-situational-config.xml*.

- Overriding the Default File Location
- File Format
- File Expiration

## Overriding the Default File Location

Using the system property, `weblogic.SituationalConfigPath`, you can override the default `optconfig` directory location for `situational-config.xml` files.

To set the value, use: `export JAVA_OPTIONS=-Dweblogic.SituationalConfigPath=<your sitconfig file path>` and then start the server.

For example:

```
export JAVA_OPTIONS=-Dweblogic.SituationalConfigPath=$optconfig:$domainroot/mydir:/
scratch/myoptconfig
```

The following substitutions are supported:

- `$domainroot` - the domain root directory
- `$optconfig` - the default location for `optconfig`, for example, `$domainroot/optconfig`

You can locate `situational-config.xml` files in server-specific directories under a defined `optconfig` directory. For example, the Administration Server will look for situational configuration files in `$domainroot/optconfig` and `$domainroot/optconfig/admin`. On a Managed Server, `ms1`, the server will look for situational configuration files in `$domainroot/optconfig` and `$domainroot/optconfig/ms1`.

For server-specific system resource situational configuration files, you can specify `jdbc`, `jms`, and `diagnostics` subdirectories in server-specific directories under a defined `optconfig` directory.

The system property can be a colon-separated list of paths, for example: `$optconfig:$domainroot/mydir:/scratch/myoptconfig`. Given this example path, the server will look for (process) situational configuration files in the following order:

```
<domain root>/optconfig
<domain root>/optconfig/<servername>
<domain root>/optconfig/jms
<domain root>/optconfig/jdbc
<domain root>/optconfig/diagnostics

<domain root>/mydir/
<domain root>/mydir/<servername>
<domain root>/mydir/jms
<domain root>/mydir/jdbc
<domain root>/mydir/diagnostics

/scratch/myoptconfig
/scratch/myoptconfig/<servername>
/scratch/myoptconfig/jms
/scratch/myoptconfig/jdbc
/scratch/myoptconfig/diagnostics
```

## File Format

A situational configuration file contains an XML fragment that references the structure of the configuration it is modifying. For example, to modify this server's debug configuration in `config.xml`:

```
<domain .... >
  <name>sitconfigDomain</name>
  <server>
    <name>admin</name>
    <server-debug>
    </server-debug>
  </server>
...
```

The situational configuration XML fragment would look like this:

```
<?xml version='1.0' encoding='UTF-8'?>
<dom:domain
 xmlns:dom="http://xmlns.oracle.com/weblogic/domain"
 xmlns:f="http://xmlns.oracle.com/weblogic/domain-fragment"
 xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
   <s:expiration> 2020-11-29T10:24-08:00 </s:expiration>
```

```
  <dom:name>sitconfigDomain</dom:name>
  <dom:server>
    <dom:name>admin</dom:name>
    <dom:server-debug>
      <dom:debug-jmx-core f:combine-mode="replace">true</dom:debug-jmx-core>
    </dom:server-debug>
  </dom:server>
  </dom:domain>
```

In the following example, `debug-jmx-core`, is *not* replaced (overridden) because part of the path (`<server-debug>`) is missing.

```
<f:domain  xmlns=.... >
   <dom:server>
     <dom:name>admin</dom:name>
      <dom:debug-jmx-core f:combine-mode="replace">true</dom:debug-jmx-core>
   </dom:server>
</f:domain>
```

### Schemas

Situational configuration XML fragments include entries from multiple schemas. The `situational-config` schema provides information on how the situational configuration will be applied; for example, its expiration time. The rest of the fragment contains information on how to modify the in-memory contents of `config.xml` or system resource file.

The `situational-config.xsd` contains these lines:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
     <!-- expiration time for this file -->
    <xs:element name="expiration" type="xs:string"/>
</xs:schema>
```

`sit-domain-fragment.xsd` is not a “true” schema but it must be referenced in the XML fragment. Use the syntax `f:combine-mode="replace"` to replace (override) values, as shown in the following example:

```
<?xml version='1.0' encoding='UTF-8'?>
<f:domain
 xmlns="http://xmlns.oracle.com/weblogic/domain"
 xmlns:f="http://xmlns.oracle.com/weblogic/sit-domain-fragment"
 xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
   <s:expiration> 2020-07-16T19:20+01:00 </s:expiration>
   <server>
     <name>soa_server1</name>
     <max-message-size f:combine-mode="replace">30000000</max-message-size>
    <server-debug>
      <debug-jmx-core f:combine-mode="replace">true</debug-jmx-core>
    </server-debug>
   </server>
</f:domain>
```

Situational configuration fragments for system resources are similar to the situational configuration files for `config.xml` overrides. For JDBC, JMS, and diagnostic system resource files, the XML situational configuration and namespaces are specific to the system resource type (JMS, JDBC, or WLDF). The rest of the fragment contains information on how to modify the in-memory contents of the system resource. Consider the following example situational configuration files for JMS, JDBC, and WLDF system resources.

- A JMS situational configuration file that overrides the shared configuration element and the expiration policy:

```
<?xml version='1.0' encoding='UTF-8'?>
<jms:weblogic-jms
 xmlns:jms="http://xmlns.oracle.com/weblogic/weblogic-jms"
 xmlns:f="http://xmlns.oracle.com/weblogic/weblogic-jms-fragment"
 xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
 <jms:quota jms:name="MyTemplate.Quota">
    <jms:shared f:combine-mode="replace">true</jms:shared>
 </jms:quota>
  <jms:template jms:name="MyTemplate">
    <jms:delivery-failure-params>
      <jms:expiration-policy f:combine-mode="replace">Discard</jms:expiration-policy>
    </jms:delivery-failure-params>
  </jms:template>
</jms:weblogic-jms>
```

- A JDBC file that adds a user name and password property:

```
<?xml version='1.0' encoding='UTF-8'?>
<jdbc:jdbc-data-source
 xmlns:jdbc="http://xmlns.oracle.com/weblogic/jdbc-data-source"
 xmlns:f="http://xmlns.oracle.com/weblogic/jdbc-data-source-fragment"
 xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
 <jdbc:name>jdbcDataSource</jdbc:name>
   <jdbc:jdbc-driver-params>
     <jdbc:properties>
       <jdbc:property f:combine-mode="replace">
          <jdbc:name>user</jdbc:name>
          <jdbc:value>Jones</jdbc:value>
       </jdbc:property>
       <jdbc:property f:combine-mode="replace">
          <jdbc:name>password</name>
          <jdbc:value>password</value>
       </jdbc:property>
     </jdbc:properties>
   </jdbc:jdbc-driver-params>
</jdbc:jdbc-data-source>
```

- A diagnostics file that enables instrumentation:

```
<?xml version='1.0' encoding='UTF-8'?>
<wldf-resource
  xmlns:wldf="http://xmlns.oracle.com/weblogic/weblogic-diagnostics"
  xmlns:f="http://xmlns.oracle.com/weblogic/weblogic-diagnostics-fragment"
  xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
  <wldf:instrumentation>
    <wldf:enabled f:combine-mode="replace">true</wldf:enabled>
  </wldf:instrumentation>
<wldf:wldf-resource>
```

To add or delete configuration elements, use the syntax, `f:combine-mode="add"` or `f:combine-mode="delete"`, as shown in the following examples.

- The following configuration file fragments demonstrate adding both basic datatypes (String, int, Boolean) and arrays of nested elements. Elements are added at both the domain and nested server layer:

```
<?xml version='1.0' encoding='UTF-8'?>
<dom:domain
 xmlns:dom="http://xmlns.oracle.com/weblogic/domain"
 xmlns:f="http://xmlns.oracle.com/weblogic/domain-fragment"
 xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
```

```
  <s:expiration> 2020-11-30T19:20+01:00 </s:expiration>
  <dom:name>sitconfigDomain</dom:name>
  <dom:notes f:combine-mode="add">Situational Config Domain</dom:notes>
  <dom:archive-configuration-count f:combine-mode="add">2</dom:archive-configuration-
count>
  <dom:startup-class f:combine-mode="add">
    <dom:deployment-order>33</dom:deployment-order>
    <dom:name>AddedStartupClassOne</dom:name>
  </dom:startup-class>
  <dom:startup-class f:combine-mode="add">
    <dom:deployment-order>44</dom:deployment-order>
    <dom:name>AddedStartupClassTwo</dom:name>
  </dom:startup-class>
  <dom:server>
    <dom:name>admin</dom:name>
    <dom:notes f:combine-mode="add">admin server</dom:notes>
    <dom:server-debug>
      <dom:debug-jmx-core f:combine-mode="add">true</dom:debug-jmx-core>
    </dom:server-debug>
    <dom:max-message-size f:combine-mode="add">78787878</dom:max-message-size>
  </dom:server>
</dom:domain>
```

- The following configuration file fragments demonstrate adding a new server to the domain:

```
<?xml version='1.0' encoding='UTF-8'?>
<dom:domain
  xmlns:dom="http://xmlns.oracle.com/weblogic/domain"
 xmlns:f="http://xmlns.oracle.com/weblogic/domain-fragment"
 xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
  <s:expiration> 2020-11-30T19:20+01:00 </s:expiration>
  <dom:name>sitconfigDomain</dom:name>
  <dom:server f:combine-mode="add">
    <dom:name>addedServer</dom:name>
    <dom:notes>addedServer notes</dom:notes>
    <dom:listen-port>7401</dom:listen-port>
    <dom:tgiop-enabled>false</dom:tgiop-enabled>
    <dom:network-access-point>
      <dom:name>AddedNetworkAccessPointOne</dom:name>
      <dom:listen-port>7501</dom:listen-port>
    </dom:network-access-point>
    <dom:network-access-point>
      <dom:name>AddedNetworkAccessPointTwo</dom:name>
      <dom:listen-port>7601</dom:listen-port>
    </dom:network-access-point>
  </dom:server>
</dom:domain>
```

- The following configuration file fragments demonstrate deleting both basic datatypes (String, int, Boolean). Elements are deleted at both the domain and nested server layer:

```
<?xml version='1.0' encoding='UTF-8'?>
<dom:domain
  xmlns:dom="http://xmlns.oracle.com/weblogic/domain"
 xmlns:f="http://xmlns.oracle.com/weblogic/domain-fragment"
 xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
  <s:expiration> 2020-11-30T19:20+01:00 </s:expiration>
  <dom:name>sitconfigDomain</dom:name>
  <dom:configuration-version f:combine-mode="delete"></dom:configuration-version>
  <dom:server>
    <dom:name>MyServerOne</dom:name>
    <dom:listen-port f:combine-mode="delete"></dom:listen-port>
    <dom:network-access-point f:combine-mode="delete"></dom:network-access-point>
   </dom:server>
```

```
  <dom:server>
    <dom:name>MyServerTwo</dom:name>
    <dom:listen-port f:combine-mode="delete"></dom:listen-port>
  </dom:server>
  <dom:server>
    <dom:name>ms1</dom:name>
    <dom:accept-backlog f:combine-mode="delete"></dom:accept-backlog>
    <dom:ignore-sessions-during-shutdown f:combine-mode="delete"></dom:ignore-
sessions-during-shutdown>
  </dom:server>
</dom:domain>
```

### XML Namespaces

The following table summarizes the XML namespaces used by situational configuration files.

| xmlns Type | Example | Description |
|---|---|---|
| Domain | `xmlns:dom="http://xmlns.oracle.com/weblogic/domain"` | Used for domain situational `config.xml` overrides to identify domain elements, for example, `<dom:domain>`. |
| Domain Fragment | `xmlns:f="http://xmlns.oracle.com/weblogic/domain-fragment"` | Used for domain `config.xml` overrides to identify domain fragment elements, for example, `f:combine-mode="replace"` inside a domain element. |
| Situational Configuration | `xmlns:s="http://xmlns.oracle.com/weblogic/situational-config"` | Used for domain config.xml and system resource overrides to identify situational configuration elements, for example, `<s:expiration-time>`. |
| JMS | `xmlns:jms="http://xmlns.oracle.com/weblogic/weblogic-jms"` | Used to identify JMS system resource elements, for example, `<jms:template name="JMSTemplate1">`. |
| JDBC | `xmlns:jdbc="http://xmlns.oracle.com/weblogic/jdbc-data-source"` | Used to identify JDBC system resource elements, for example, `<jdbc:jdbc-driver-params>`. |
| Diagnostics | `xmlns:wldf="http://xmlns.oracle.com/weblogic/weblogic-diagnostics"` | Used to identify WLDF system resource elements, for example, `<wldf:watch-notification>`. |
| JMS System Resource Fragment | `xmlns:f="http://xmlns.oracle.com/weblogic/weblogic-jms-fragment"` | Used for JMS system resource overrides to identify JMS system resource fragment elements, for example, `f:combine-mode="replace"` inside a JMS system resource element. |
| JDBS System Resource Fragment | `xmlns:f="http://xmlns.oracle.com/weblogic/jdbc-data-source-fragment"` | Used for JDBC system resource overrides to identify JDBC system resource fragment elements, for example, `f:combine-mode="replace"` inside a JDBC system resource element. |
| WLDF System Resource Fragment | `xmlns:f="http://xmlns.oracle.com/weblogic/weblogic-diagnostics-fragment"` | Used for WLDF system resource overrides to identify WLDF system resource fragment elements, for example, `f:combine-mode="replace"` inside a WLDF system resource element. |

## File Expiration

If desired, you can set an explicit expiration time in the XML fragment file by referencing the `situational-config.xsd` schema and then setting the expiration time element. The expiration time is either a long (system time) or a W3C complaint String.

For example:

```
<s:expiration> 2020-11-29T10:24-08:00 </s:expiration>
     11/29/2020 @ 10:24 am (PDT)

<s:expiration> 2020-11-29T10:24 </s:expiration>
     11/29/2020 @ 10:24 am (local timezone)

<s:expiration> 2020-11-29T10:24:15 </s:expiration>
     11/29/2020 @ 10:24:15 am (local timezone)
```

The situational configuration values will be in place until either the file is removed or the system time exceeds the expiration time. After that happens, the file and its values will be unloaded and the settings will return to their previous values. If the situational configuration file has a timeout value which is earlier than the current time (the file has expired), the contents of the file will be ignored.

To enable a *permanent* situational configuration file with no expiration, specify either no expiration element or an expiration element value of minus one (`-1`).

- An example permanent situational configuration file with no expiration element:

```
<?xml version='1.0' encoding='UTF-8'?>
<domain
  xmlns:dom="http://xmlns.oracle.com/weblogic/domain"
 xmlns:f="http://xmlns.oracle.com/weblogic/domain-fragment"
 xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
  <name>sitconfigDomain</name>
  <server>
    <name>admin</name>
    <server-debug>
      <debug-jmx-core f:combine-mode="replace">true</debug-jmx-core>
    </server-debug>
  </server>
</domain>
```

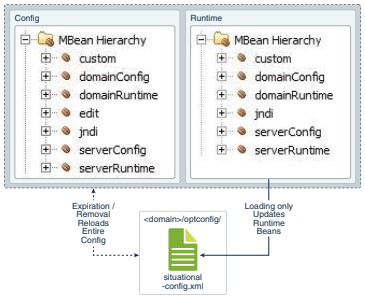
- An example permanent situational configuration file with an expiration of -1:

```
<?xml version='1.0' encoding='UTF-8'?>
<dom:domain
  xmlns:dom="http://xmlns.oracle.com/weblogic/domain"
 xmlns:f="http://xmlns.oracle.com/weblogic/domain-fragment"
 xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
  <s:expiration>-1</s:expiration>
  <dom:name>sitconfigDomain</dom:name>
    <dom:server>
      <dom:name>admin</dom:name>
      <dom:server-debug>
        <dom:debug-jmx-core f:combine-mode="replace">true</dom:debug-jmx-core>
      </dom:server-debug>
    </dom:server>
  </dom:domain>
```

# Loading Situational Configurations

The situational configuration file exists when a server is started. Situational configurations are loaded into the runtime hierarchy, not the edit configuration hierarchy. Administrators can see the effective configuration (base configuration + situational configuration overrides) by looking at the runtime tree. The configuration edit tree shows only the base values before the situational configuration was applied or after the situational configuration is removed.

**Figure 3-5 Loading and Removal of Situational Configuration Files**




### Accessing the Edit and Runtime Configuration Using WLST

Given:

- A `config.xml` with a domain `Notes` attribute, "My Domain"
- A situational configuration file that overrides the domain `Notes` attribute with the value, "My New Domain"

Using WLST, you can access the values in the runtime and edit configurations as follows:

```
# runtime configuration
serverConfig()
wls:/mydomain/serverConfig/> print cmo.getNotes()
My New Domain
# edit configuration
wls:/mydomain/serverConfig/> edit()
wls:/mydomain/edit/> print cmo.getNotes()
My Domain
```

- Processing Multiple Fragments
- Using Wildcards

# Processing Multiple Fragments

Each situational configuration XML fragment is stored in its own `situational-config.xml` file; only one fragment per file is allowed. If multiple fragments are available, there will be multiple situational configuration files, each named uniquely. For example:

```
wildcard-situational-config.xml
example1-situational-config.xml
foobar-situational-config.xml
```

Situational configuration files are loaded based on their modification date, such that the oldest file is loaded first, then subsequent files are processed (loaded).

## Using Wildcards

In XML fragments, you can use wildcards to make them more flexible, such as, to process the same set of configuration settings on multiple resources.

For example, to override identical debug settings on all servers in a cluster:

```
<f:domain
  xmlns=.... >
    <dom:server>
     <dom:name>'*'</dom:name>
     <dom:max-message-size f:combine-mode="replace">30000000</dom:max-message-size>
      <dom:server-debug>
               <dom:debug-jmx-core f:combine-mode="replace">true</dom:debug-jmx-core>
      </dom:server-debug>
   </dom:server>
</f:domain>
```

Only one wildcard is supported, `'*'` (an asterisk, escaped within single quotation marks). It matches everything.

Follow these rules when using wildcards:

- Use wildcards in elements only.
  - Wildcards let you replace a specific element with an asterisk (*) and have it match any element within the same set of tags.
  - The wildcard can match any element. For example, if you change the server name, `admin`, to *, any settings in the fragment which were targeted for `admin` would now be applied to all servers. For example: `<name>'*'</name>`
- Tags cannot contain wildcards. The following is not supported: `<'*'>admin</'*'>`
- Elements which are to be modified (overridden) cannot have wildcards. The following are not supported:

```
<dom:debug-jmx-core f:combine-mode='*'>'*'</dom:debug-jmx-core>
<dom:debug-jmx-core f:combine-mode="replace">'*'</dom:debug-jmx-core>
```

- A wildcard replaces a whole word (all or nothing); there are no partial matches. For example, if the string is LKS, * will match; L* or LK* will *not* match and generate an error.
- You can use wildcards to add or delete elements. However, because only asterisk ('*') is supported, the add or delete applies to all MBeans. For example, if used with a Server MBean, this action would delete all the servers in the domain. Consider the following examples.
  - The following configuration file fragments demonstrate the use of a wildcard to add basic datatype and container elements to all servers in the domain.

```
<?xml version='1.0' encoding='UTF-8'?>
<dom:domain
  xmlns:dom="http://xmlns.oracle.com/weblogic/domain"
 xmlns:f="http://xmlns.oracle.com/weblogic/domain-fragment"
 xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
  <s:expiration> 2020-11-30T19:20+01:00 </s:expiration>
  <dom:name>sitconfigDomain</dom:name>
  <dom:server>
    <dom:name>'*'</dom:name>
```

```
    <dom:notes f:combine-mode="add">wildcard notes</dom:notes>
    <dom:server-debug>
      <dom:debug-jmx-core f:combine-mode="add">true</dom:debug-jmx-core>
    </dom:server-debug>
    <dom:cluster-weight f:combine-mode="add">49</dom:cluster-weight>
      <dom:network-access-point f:combine-mode="add">
        <dom:name>AddedNetworkAccessPoint</dom:name>
        <dom:enabled>false</dom:enabled>
        <dom:listen-port>7501</dom:listen-port>
      </dom:network-access-point>
   </dom:server>
</dom:domain>
```

- The following configuration file fragments demonstrate the use of a wildcard to delete basic datatypes from all servers in the domain.

```
<?xml version='1.0' encoding='UTF-8'?>
<dom:domain
  xmlns:dom="http://xmlns.oracle.com/weblogic/domain"
 xmlns:f="http://xmlns.oracle.com/weblogic/domain-fragment"
 xmlns:s="http://xmlns.oracle.com/weblogic/situational-config" >
  <s:expiration> 2020-11-30T19:20+01:00 </s:expiration>
  <dom:name>sitconfigDomain</dom:name>
  <dom:server>
    <dom:name>'*'</dom:name>
    <dom:accept-backlog f:combine-mode="delete"></dom:accept-backlog>
    <dom:server-debug>
      <dom:debug-jms-back-end f:combine-mode="delete"></dom:debug-jms-back-end>
    </dom:server-debug>
  </dom:server>
</dom:domain>
```

**Wildcard Overrides**

If a file has both wildcard and non-wildcard entries for a value, the first one which matches prevails.

For example:

```
<dom:server>
   <dom:name>server2</dom:name>
    <dom:max-message-size f:combine-mode="replace">666666</dom:max-message-size>
</dom:server>
 <dom:server>
    <dom:name>'*'</dom:name>
     <dom:max-message-size f:combine-mode="replace">30000000</dom:max-message-size>
 </dom:server>
 <dom:server>
    <dom:name>server1</dom:name>
     <dom:max-message-size f:combine-mode="replace">9999999</dom:max-message-size>
 </dom:server>
```

In this example, `server2` has a `max-message-size` of `666666` but `server1` has a value of `30000000` (because the * is above `server1` in the file).

# Monitoring Situational Configuration Changes

Use the `ServerRuntimeMBean.isInSitConfigState()` method to check if a situational configuration is currently in place. For example:

```
ServerRuntimeMBean runtimeMBean = runtimeService.getServerRuntime();
    boolean setting = runtimeMBean.isInSitConfigState();
```

To control how frequently the server polls for situational configuration changes (new, removed, or changed `situational-config.xml` files), use `ServerMBean.setSitConfigPollInterval(int)` to configure the polling period (in seconds).

For example, to set the polling rate:

```
DomainMBean domainMBean = domainService.getDomainConfiguration();
ServerMBean serverMBean = domainMBean.lookupServer(adminServerName);
serverMBean.setSitConfigPollingInterval(value);
```

### Require Situational Configuration

By default, applying a configuration override is optional. Under certain conditions, you might want to *require* a configuration override, for example, to prevent booting with the default large cluster configuration on a small pod. The `SitConfigRequired` attribute on the Server MBean controls whether a situational configuration is required or not. This configuration attribute causes the server to *not* start if no override files are applied. The default is to start the server and not fail.

```
/**
 * Sets whether situational config files are required. If required, then WebLogic Server
will
 * fail to boot if no situational config files are present. The default is to not
require situational config files.
 * @param isRequired - true if situational config files are required, false otherwise.
 * @default false
 * @dynamic
 * @include-api for-public-api
 */
void setSitConfigRequired(boolean isRequired);

/**
 * Returns whether situational config files are required and WebLogic Server should fail
to boot if situational config files are not present.
 * @return  true if situational config files are required, false otherwise.
 * @dynamic
 * @include-api for-public-api
 */
int isSitConfigRequired();
```

# Using Named Concurrent Edit Sessions

In prior releases, WebLogic Server supported only one active configuration edit session at a time. The system administrator got a global edit lock, made changes, and then activated them. Other administrators could not make changes at the same time without first taking the configuration lock from another administrator, then editing the same active edit session. However, now WebLogic Server enables multiple, named concurrent edit sessions, which allows more than one administrator to make configuration changes in separate edit sessions at the same time.

The concurrent edit sessions are typically useful when multiple administrators work in different parts of the system. Also, when configuring a system takes a long time because of the serial execution of configuration commands, a single administrator can open multiple named edit sessions. This saves time by running the configuration edit sessions in parallel.

4

# Server Templates

Use server templates to easily manage configuration for a group of server instances in one centralized location. Learn how to create, configure, and use server templates in WebLogic Server.

This chapter includes the following sections:

- 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.
- 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.
- Using Server Templates
  You create server templates using WebLogic Scripting Tool (WLST), Fusion Middleware Control (FMWC), or the WebLogic Remote Console.
- Server Templates Example
  Examine an example that demonstrates using WLST to create two server instances that use the same server template.

## 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 4-1 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 4-1 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>
```

# 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.

You can use server templates in one of the following two ways:

- As the basis for any configured managed server
- As the basis for dynamic servers in a dynamic cluster

**Use Server Templates as the Basis for Any Configured Managed Server**

A server template serves as a way to simplify server configuration. The managed server inherits the configuration attributes from the server template it references except for attributes that are manually configured in the server configuration. You can override any configuration attribute in this case.

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 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.

**Use Server Templates as the Basis for Dynamic Servers in a Dynamic Cluster**

You also use server templates to create dynamic clusters. You can define a specific server template to use as the basis for all dynamic servers in a dynamic cluster.

A dynamic cluster contains multiple dynamic servers that are based on a single-server template. This server template is the basis for all the servers in the cluster so that you do not need to manually configure each server when expanding the cluster. When you use a server template to create a dynamic cluster and specify the number of server instances you want, WebLogic Server inherits and calculates necessary values from the server template and then uses these calculated values for the attributes. You cannot change individual server configuration for dynamic servers except for system properties that you would specify when starting the server. To know more about how to create, configure, and use dynamic clusters in WebLogic Server, see Dynamic Clusters in *Administering Clusters for Oracle WebLogic Server*.

# Using Server Templates

You create server templates using WebLogic Scripting Tool (WLST), Fusion Middleware Control (FMWC), or the WebLogic Remote Console.

To create server templates using WLST, see Example 4-2. To create server templates using the WebLogic Remote Console, see Create a Server Template in the *Oracle WebLogic Remote 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*.

- Overriding Attributes
- Using Macros

## 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
```

## 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.

# Server Templates Example

Examine an example that demonstrates using WLST to create two server instances that use the same server template.

Example 4-2 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 4-2 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>
```