|  |
| --- |
| **Ministry of Transportation** |
| **Weblogic Automata** |
| **Documentation** |

|  |
| --- |
| Technical Support Office  2/16/2016  Revision 1.0 – Draft State |

©2016 Technical Support Office. All rights reserved.

This documentation is furnished *AS IS* and is subject to change without notice and therefore should not be construed as a commitment by the Technical Support Office. This documentation may *not* be copied, modified or distributed in any way without the express authorization of the Technical Support Office. This documentation may be used only in connection with the Technical Support Office’s products and services. Use, duplication, reproduction, release, modification, disclosure or transfer of this documentation is restricted in accordance with the Government of Ontario’s policies and regulations. This documentation qualifies as computer software documentation and any use shall be governed solely by these terms. All other use is strictly prohibited. The Technical Support Office assumes no responsibility or liability for any errors or inaccuracies that appear in this documentation.

# Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Owner |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[Revisions 2](#_Toc522307904)

[Introduction 5](#_Toc522307905)

[Documention 5](#_Toc522307906)

[Tasks 6](#_Toc522307907)

[Build 7](#_Toc522307908)

[Config 8](#_Toc522307909)

[Deploy 9](#_Toc522307910)

[Templates 10](#_Toc522307911)

[root 10](#_Toc522307912)

[internal 11](#_Toc522307913)

[wlst 11](#_Toc522307914)

[credential 12](#_Toc522307915)

[edit 12](#_Toc522307916)

[domain 13](#_Toc522307917)

[admin\_server 14](#_Toc522307918)

[machine 15](#_Toc522307919)

[server\_start 16](#_Toc522307920)

[logging 16](#_Toc522307921)

[cluster 19](#_Toc522307922)

[server 19](#_Toc522307923)

[deployment 20](#_Toc522307924)

[data\_source 21](#_Toc522307925)

[property 24](#_Toc522307926)

[security\_realm 24](#_Toc522307927)

[provider 24](#_Toc522307928)

[authentication (DefaultAuthenticator) 25](#_Toc522307929)

[authentication (IPlanetAuthenticator) 26](#_Toc522307930)

[authorization 29](#_Toc522307931)

[role\_mapping 29](#_Toc522307932)

[keystore 29](#_Toc522307933)

[persistent\_store 30](#_Toc522307934)

[target 30](#_Toc522307935)

[messaging 30](#_Toc522307936)

[jms\_server 31](#_Toc522307937)

[jms\_module 32](#_Toc522307938)

[sub\_deployment 33](#_Toc522307939)

[uniform\_distribute\_queue 33](#_Toc522307940)

[connection\_factory 34](#_Toc522307941)

# Introduction

## Documention

This documentation contains manual for the TSO’s in-house **Weblogic Automata** application. It describes the functionality, and uses output and screenshots from version 1.0 of the software.

There are several versions of the *Documentation*, therefore, you should refer to the version appropriate for the variant you are configuring and running.

# Tasks

|  |
| --- |
| Note |
| Weblogic Automata requires a Weblogic domain JSON template before the build, config or deploy command can be invoked. See the Template section for information. |

For a quick reference to the possible commands that can be invoked, run the **weblogic\_automata** application without any parameters or with the **-help** parameter

|  |
| --- |
| weblogic\_automata  // or  weblogic\_automata -help |
| Usage: weblogic\_automata [OPTIONS] [validate|build|config|deploy] <domain\_template.json>  -debug  enable debugging  -noreport  do not display result |

## Build

To build a Weblogic domain, run the **weblogic\_automata** application with the Weblogic domain JSON template you have created.

* To enable debugging message, include the **-debug** parameter
* For a silent run (without any output), include the **-norepor**t parameter

For illustration purpose, below is a sample run of a domain **build** for an application called Audit.

|  |
| --- |
| ./weblogic\_automata build audit.json |
| <ok> Root.WLST.ReturnStatus  <--> Root.Credential.ReturnStatus  <--> Root.Edit.ReturnStatus  <--> Root.Domain.AdminServer.Machine.ReturnStatus  <--> Root.Domain.AdminServer.ServerStart.ReturnStatus  <--> Root.Domain.AdminServer.Logging.ReturnStatus  <--> Root.Domain.AdminServer.ReturnStatus  <--> Root.Domain.Cluster.Servers[0].Machine.ReturnStatus  <--> Root.Domain.Cluster.Servers[0].ServerStart.ReturnStatus  <--> Root.Domain.Cluster.Servers[0].Logging.ReturnStatus  <--> Root.Domain.Cluster.Servers[0].ReturnStatus  <--> Root.Domain.Cluster.Servers[1].Machine.ReturnStatus  <--> Root.Domain.Cluster.Servers[1].ServerStart.ReturnStatus  <--> Root.Domain.Cluster.Servers[1].Logging.ReturnStatus  <--> Root.Domain.Cluster.Servers[1].ReturnStatus  <--> Root.Domain.Cluster.ReturnStatus  <ok> Root.Domain.ReturnStatus  <--> Root.Deployments[0].ReturnStatus  <--> Root.Deployments[1].ReturnStatus  <--> Root.DataSources[0].Properties[0].ReturnStatus  <--> Root.DataSources[0].ReturnStatus  <--> Root.DataSources[1].Properties[0].ReturnStatus  <--> Root.DataSources[1].ReturnStatus  <--> Root.SecurityRealms[0].Provider.Authentications[0].ReturnStatus  <--> Root.SecurityRealms[0].Provider.Authentications[1].ReturnStatus  <--> Root.SecurityRealms[0].Provider.Authorizations[0].ReturnStatus  <--> Root.SecurityRealms[0].Provider.RoleMappings[0].ReturnStatus  <--> Root.SecurityRealms[0].Provider.Auditings[0].ReturnStatus  <--> Root.SecurityRealms[0].Provider.Keystores[0].ReturnStatus  <--> Root.SecurityRealms[0].ReturnStatus  <--> Root.PersistentStores[0].ReturnStatus  <--> Root.PersistentStores[1].ReturnStatus  <--> Root.Messaging.JMSServers[0].ReturnStatus  <--> Root.Messaging.JMSServers[1].ReturnStatus  <--> Root.Messaging.JMSModules[0].SubDeployments[0].ReturnStatus  <--> Root.Messaging.JMSModules[0].UniformDistributedQueues[0].ReturnStatus  <--> Root.Messaging.JMSModules[0].UniformDistributedQueues[1].ReturnStatus  <--> Root.Messaging.JMSModules[0].UniformDistributedQueues[2].ReturnStatus  <--> Root.Messaging.JMSModules[0].UniformDistributedQueues[3].ReturnStatus  <--> Root.Messaging.JMSModules[0].ConnectionFactories[0].ReturnStatus  <--> Root.Messaging.JMSModules[0].ConnectionFactories[1].ReturnStatus  <--> Root.Messaging.JMSModules[0].ReturnStatus  <--> Root.Messaging.ReturnStatus  <ok> Root.ReturnStatus |

## Config

Before invoking the **config** task, ensure that the Weblogic admin server is in ***RUNNING*** state. Once this condition is met, run the **weblogic\_automata** application with the **config** command along with the the domain JSON template.

For illustration purpose, below is a sample run of a domain **config** for an application called Audit.

|  |
| --- |
| ./weblogic\_automata config audit.json |
| <ok> Root.WLST.ReturnStatus  <ok> Root.Credential.ReturnStatus  <ok> Root.Edit.ReturnStatus  <ok> Root.Domain.AdminServer.Machine.ReturnStatus  <ok> Root.Domain.AdminServer.ServerStart.ReturnStatus  <ok> Root.Domain.AdminServer.Logging.ReturnStatus  <ok> Root.Domain.AdminServer.ReturnStatus  <ok> Root.Domain.Cluster.Servers[0].Machine.ReturnStatus  <ok> Root.Domain.Cluster.Servers[0].ServerStart.ReturnStatus  <ok> Root.Domain.Cluster.Servers[0].Logging.ReturnStatus  <ok> Root.Domain.Cluster.Servers[0].ReturnStatus  <ok> Root.Domain.Cluster.Servers[1].Machine.ReturnStatus  <ok> Root.Domain.Cluster.Servers[1].ServerStart.ReturnStatus  <ok> Root.Domain.Cluster.Servers[1].Logging.ReturnStatus  <ok> Root.Domain.Cluster.Servers[1].ReturnStatus  <ok> Root.Domain.Cluster.ReturnStatus  <ok> Root.Domain.ReturnStatus  <--> Root.Deployments[0].ReturnStatus  <--> Root.Deployments[1].ReturnStatus  <ok> Root.DataSources[0].Properties[0].ReturnStatus  <ok> Root.DataSources[0].ReturnStatus  <ok> Root.DataSources[1].Properties[0].ReturnStatus  <ok> Root.DataSources[1].ReturnStatus  <ok> Root.SecurityRealms[0].Provider.Authentications[0].ReturnStatus  <ok> Root.SecurityRealms[0].Provider.Authentications[1].ReturnStatus  <ok> Root.SecurityRealms[0].Provider.Authorizations[0].ReturnStatus  <ok> Root.SecurityRealms[0].Provider.RoleMappings[0].ReturnStatus  <ok> Root.SecurityRealms[0].Provider.Auditings[0].ReturnStatus  <ok> Root.SecurityRealms[0].Provider.Keystores[0].ReturnStatus  <ok> Root.SecurityRealms[0].ReturnStatus  <ok> Root.PersistentStores[0].ReturnStatus  <ok> Root.PersistentStores[1].ReturnStatus  <ok> Root.Messaging.JMSServers[0].ReturnStatus  <ok> Root.Messaging.JMSServers[1].ReturnStatus  <ok> Root.Messaging.JMSModules[0].SubDeployments[0].ReturnStatus  <ok> Root.Messaging.JMSModules[0].UniformDistributedQueues[0].ReturnStatus  <ok> Root.Messaging.JMSModules[0].UniformDistributedQueues[1].ReturnStatus  <ok> Root.Messaging.JMSModules[0].UniformDistributedQueues[2].ReturnStatus  <ok> Root.Messaging.JMSModules[0].UniformDistributedQueues[3].ReturnStatus  <ok> Root.Messaging.JMSModules[0].ConnectionFactories[0].ReturnStatus  <ok> Root.Messaging.JMSModules[0].ConnectionFactories[1].ReturnStatus  <ok> Root.Messaging.JMSModules[0].ReturnStatus  <ok> Root.Messaging.ReturnStatus  <ok> Root.ReturnStatus |

## Deploy

To **deploy** the application, run the **weblogic\_automata** application with the **deploy** command along with the the domain JSON template.

For illustration purpose, below is a sample run of an domain **deploy** for an application called Audit.

|  |
| --- |
| ./weblogic\_automata deploy audit.json |
| <ok> Root.WLST.ReturnStatus  <ok> Root.Credential.ReturnStatus  <--> Root.Edit.ReturnStatus  <--> Root.Domain.AdminServer.Machine.ReturnStatus  <--> Root.Domain.AdminServer.ServerStart.ReturnStatus  <--> Root.Domain.AdminServer.Logging.ReturnStatus  <--> Root.Domain.AdminServer.ReturnStatus  <--> Root.Domain.Cluster.Servers[0].Machine.ReturnStatus  <--> Root.Domain.Cluster.Servers[0].ServerStart.ReturnStatus  <--> Root.Domain.Cluster.Servers[0].Logging.ReturnStatus  <--> Root.Domain.Cluster.Servers[0].ReturnStatus  <--> Root.Domain.Cluster.Servers[1].Machine.ReturnStatus  <--> Root.Domain.Cluster.Servers[1].ServerStart.ReturnStatus  <--> Root.Domain.Cluster.Servers[1].Logging.ReturnStatus  <--> Root.Domain.Cluster.Servers[1].ReturnStatus  <--> Root.Domain.Cluster.ReturnStatus  <--> Root.Domain.ReturnStatus  <ok> Root.Deployments[0].ReturnStatus  <--> Root.Deployments[1].ReturnStatus  <--> Root.DataSources[0].Properties[0].ReturnStatus  <--> Root.DataSources[0].ReturnStatus  <--> Root.DataSources[1].Properties[0].ReturnStatus  <--> Root.DataSources[1].ReturnStatus  <--> Root.SecurityRealms[0].Provider.Authentications[0].ReturnStatus  <--> Root.SecurityRealms[0].Provider.Authentications[1].ReturnStatus  <--> Root.SecurityRealms[0].Provider.Authorizations[0].ReturnStatus  <--> Root.SecurityRealms[0].Provider.RoleMappings[0].ReturnStatus  <--> Root.SecurityRealms[0].Provider.Auditings[0].ReturnStatus  <--> Root.SecurityRealms[0].Provider.Keystores[0].ReturnStatus  <--> Root.SecurityRealms[0].ReturnStatus  <--> Root.PersistentStores[0].ReturnStatus  <--> Root.PersistentStores[1].ReturnStatus  <--> Root.Messaging.JMSServers[0].ReturnStatus  <--> Root.Messaging.JMSServers[1].ReturnStatus  <--> Root.Messaging.JMSModules[0].SubDeployments[0].ReturnStatus  <--> Root.Messaging.JMSModules[0].UniformDistributedQueues[0].ReturnStatus  <--> Root.Messaging.JMSModules[0].UniformDistributedQueues[1].ReturnStatus  <--> Root.Messaging.JMSModules[0].UniformDistributedQueues[2].ReturnStatus  <--> Root.Messaging.JMSModules[0].UniformDistributedQueues[3].ReturnStatus  <--> Root.Messaging.JMSModules[0].ConnectionFactories[0].ReturnStatus  <--> Root.Messaging.JMSModules[0].ConnectionFactories[1].ReturnStatus  <--> Root.Messaging.JMSModules[0].ReturnStatus  <--> Root.Messaging.ReturnStatus  <ok> Root.ReturnStatus |

# Templates

All tasks invoked from **Weblogic Automata** application depends on the the domain JSON template.

## root

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| internal  *(required)* | internal | See the **internal** section for the definition |
| wlst  *(required)* | wlst | See the **wlst** section for the definition |
| credential  *(required)* | credential | See the **credential** section for the definition |
| edit  *(required)* | edit | See the **edit** section for the definition |
| domain | domain | A domain is a collection of WebLogic Server instances that is managed by a single Administration Server  See the **domain** section for the definition |
| deployments | array of deployments | Defines Java EE applications and stand-alone applications.  See the **deployment** section for the definition |
| data\_sources | array of data\_sources | A JDBC data source is an object bound to the JNDI tree that provides database connectivity through a pool of JDBC connections. Applications can look up a data source on the JNDI tree and then borrow a database connection from a data source.  See the **data\_source** section for the definition |
| security\_realms | array of security\_realms | A security realm is a container for the mechanisms -- including users, groups, security roles, security policies, and security providers -- that are used to protect WebLogic resources. You can have multiple security realms in a WebLogic Server domain, but only one can be set as the default (active) realm.  See the **security\_realm** section for the definition |
| persistent\_stores | array of persistent\_stores | A persistent store is a physical repository for storing subsystem data, such as persistent JMS messages. It can be either a JDBC-accessible database or a disk-based file.  See the **persistent\_store** section for the definition |
| messaging | messaging | WebLogic JMS is an enterprise-class messaging system that is tightly integrated into the WebLogic Server platform. It fully supports the JMS Specification and also provides numerous WebLogic JMS Extensions that go above and beyond the standard JMS APIs.  See the **messaging** section for the definition |

## internal

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| timeout  *(optional)* | int (unit in second) | Specifies the default timeout of a command expecting to finish quickly such as changing a value of a parameter within WLS.  A reasonable value is between 3 to 10.  Default value is 5. |
| timeout\_long  *(optional)* | int (unit in second) | Specifies the default timeout of a command expecting to finish within longer duration such as logging in WLS console.  A reasonable value is between 45 to 80.  Default value is 60. |

## wlst

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| env\_vars  *(optional)* | array of strings | Specifies the environment variables to be set before the WLST script is invoked.  For example:  **[** "USER\_MEM\_ARGS=\"-Djava.security.egd=file:/dev/./urandom -XX:MaxPermSize=512M\"" **]**    More than one environment variables can be set which can be separated by a comma. Note that the double quotes have be escaped in order to be considered a valid JSON syntax.  Default value is an empty array of strings. |
| script\_path  *(**required)* | string | Specifies the path to the WLST script to be executed.  For example: "/opt/wl12c/weblogic/wlserver/common/bin/wlst.sh"  Default value is an empty string. |
| scripts\_args  *(**optional)* | array of strings | Specifies the arguments to be appended to the WLST script during execution.  These are the first arguments appended immediately after shell script portion of the startup command.  Separate arguments with a comma.  For example:  "script\_args"**:** **[** "-skipWLSModuleScanning" **]**  Default value is an empty array of strings. |

## credential

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| username  *(required)* | string | Specifies the username to be used to connect WLST to a WebLogic Server instance  For example: "audit12"  This username needs to be defined in the active WebLogic security realm. Once you are connected, a collection of security policies determine which configuration attributes you are permitted to view or modify. (See "Default Security Policies for MBeans" in the WebLogic Server MBean Reference.)  Default value is an empty string. |
| password  *(required)* | string | Specifies the password of username who is connecting WLST to a WebLogic Server instance  The password needs to be in clear text as hashed password is not supported yet.  For example: "hunter2"  Default value is an empty string. |
| protocol  *(required)* | string | Specifies the protocol to be used to connect WLST to a WebLogic Server instance.  For example: "t3"  Supported protocols are: t3, t3s, http and https. Ensure that the firewall is open before connecting (e.g. use telnet to test first).  Default value is an empty string. |

## edit

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| save\_changes  *(optional)* | bool | Specifies whether to save the changes after execution.  For example: **true**  Default value is false |
| activate\_changes  *(optional)* | bool | Specifies whether to activate the changes after execution.  For example: **true**  Default value is false |

## domain

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the WebLogic domain to be created  For example: "audit"  Default value is false |
| domain\_template  *(required)* | string | Specifies the path to the WebLogic domain template.  For example: "~/weblogic/wlserver/common/templates/wls/wls.jar"  Default value is an empty string. |
| domain\_home  *(required)* | string | Specifies the WebLogic Server home directory.  It is recommended that this directory to be empty; otherwise, errors may occur during creation and/or modification.  For example: "/opt/wl12c/domains/audit"  Default value is an empty string. |
| java\_home  *(required)* | string | Specifies the Java runtime home directory.  For example: "/opt/wl12c/jdk "  Default value is an empty string. |
| production\_mode\_enabled  *(required)* | string | Specifies whether all servers in this domain run in production mode.  In production mode, the security configuration is much more stringent, such as requiring a username and password to deploy applications and start the Administration Server.  There are also several differences between the default settings in production mode and the default settings in domain mode, including:   * The number of threads in an execute queue * Log settings, including the maximum number of files saved during log rotation, the minimum size of the log file that triggers log rotation, and whether logs are rotated at startup * The default for server lifecycle operations timeout * SNMP security level * Servlet reload check periods   For example: "true"  Default value is "false" |
| server\_start\_mode  *(required)* |  | Specifies the mode to use when starting the server for the newly created domain. This value has to be "dev" (development) or "prod" (production).  For example: "prod"  Default value is an empty string. |
| admin\_server | admin\_server | See the **admin\_server** section for the definition |
| cluster | cluster | See the **cluster** section for the definition |

## admin\_server

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the WebLogic Admin server to be created  For example: "auditAdmin"  Default value is an empty string. |
| host  *(required)* | string | Specifies the listen host or IP of the Weblogic Admin server.  For example: "saedev01"  Default value is an empty string. |
| port  *(required)* | string | Specifies the listen port the Weblogic Admin server.  For example: "17120"  Default value is an empty string. |
| type  *(required)* | string | Specifies the type of the WebLogic server. Since this is an admin server, the value here should be "admin"  For example: "admin"  Default value is an empty string. |
| machine  *(optional)* | machine | See the **machine** section for the definition |
| server\_start  *(optional)* | server\_start | See the **server\_start** section for the definition |
| logging  *(optional)* | logging | See the **logging** section for the definition |

## machine

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| build\_mode  *(required)* | string | Specifies the type of build to be performed. This value has to be "offline" or "online". As the name implies, **offline** does not require the WebLogic admin server to be in RUNNING state whereas **online** mode does.  For example: "offline"  Default value is an empty string. |
| name  *(required)* | string | Specifies the name of the WebLogic machine server to be created.  For example: "auditHostAdmin"  Default value is an empty string. |
| host  *(required)* | string | Specifies the listen host or IP of the Weblogic machine server.  For example: "saedev01"  Default value is an empty string. |
| port  *(required)* | string | Specifies the listen port the Weblogic machine server.  For example: "27120"  Default value is an empty string. |
| protocol  *(required)* | string | Specifies the protocol to be used. This value has to be "SSL" (secure HTTP) or "plain" (unsecure HTTP).  For example: "SSL"  Default value is an empty string. |
| type  *(required)* | string | Specifies the type of the WebLogic machine server. This value has to be "UnixMachine" (Linux/UNIX O/S) or "Machine" (Windows O/S).  For example: "UnixMachine"  Default value is an empty string. |

## server\_start

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| arguments  *(required)* | array of strings | Specifies the arguments to use when starting the server.  These are the first arguments appended immediately after java portion of the startup command. For example, you can set Java heap memory or specify any weblogic.Server option.  This property should not be used to specify weblogic.management.username or weblogic.management.password as these values will be ignored during server startup. Instead the username and password properties should be set. This will also enable node manager to properly encrypt these values on the managed server's machine.  Separate arguments with a comma.  For example:  "arguments"**:** **[**"-Djava.security.egd=file:/dev/./urandom"**,**  "-Dweblogic.security.SSL.ignoreHostnameVerification=true"  **]**  Default value is an empty array of strings. |

## logging

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| log\_file\_name  *(required)* | string | Specifies the name of the file that stores current log messages. Usually it is a computed value based on the name of the parent of this MBean. For example, for a server log, it is SERVER\_NAME.log.  However, if the name of the parent cannot be obtained, the file name is weblogic.log. If you specify a relative pathname, it is interpreted as relative to the server's root directory.  To include a time and date stamp in the file name when the log file is rotated, add **java.text.SimpleDateFormat** variables to the file name. Surround each variable with percentage (%) characters.  For example, if the file name is defined to be:  "myserver\_%yyyy%\_%MM%\_%dd%\_%hh%\_%mm%.log"  ... the log file will be named *myserver\_yyyy\_mm\_dd\_hh\_mm.log*  When the log file is rotated, the rotated file name contains the date stamp. For example, if the log file is rotated for the first time on 2 April, 2003 at 10:05 AM, the log file that contains the old messages will be named myserver\_2003\_04\_02\_10\_05.log00001.  If you do not include a time and date stamp, the rotated log files are numbered in order of creation. For example, *myserver.log00007*.  For example:  "logs/auditAdmin.%yyyy%%MM%%dd%-%k%%mm%.log"  ""    Default value is an empty string. |
| rotation\_type  *(required)* | string | Specifies the criteria for moving old log messages to a separate file.  "none"  Messages accumulate in a single file. You must erase the contents of the file when the size is too large. Note that WebLogic Server sets a threshold size limit of 500 MB before it forces a hard rotation to prevent excessive log file growth.  "bySize"  When the log file reaches the size that you specify in **FileMinSize**, the server renames the file as *SERVER\_NAME.lognnnnn*.  "byTime"  At each time interval that you specify in **TimeSpan**, the server renames the file as *SERVER\_NAME.lognnnnn*.  After the server renames a file, subsequent messages accumulate in a new file with the name that you specified as the log file name.  Default value is an empty string. |
| rotation\_file\_size  *(required)* | string | Specifies the size (1 - 2097150 kilobytes) that triggers the server to move log messages to a separate file. The default is 500 kilobytes. After the log file reaches the specified minimum size, the next time the server checks the file size, it will rename the current log file as *SERVER\_NAME.lognnnnn* and create a new one to store subsequent messages. (Requires that you specify a file rotation type of Size.)  Minimum value: 1  Maximum value: 2097150  For example: "5000"  Default value is an empty string |
| begin\_rotation\_time  *(required)* | string | Determines the start time (hour and minute) for a time-based rotation sequence.  At the time that this value specifies, the server renames the current log file. Thereafter, the server renames the log file at an interval that you specify in File Time Span.  Note that WebLogic Server sets a threshold size limit of 500 MB before it forces a hard rotation to prevent excessive log file growth.  Use the following format: H:mm, where   * H is Hour in day (0-23). * mm is the minute in hour   For example: "23:59"  Default value is an empty string |
| rotation\_interval  *(required)* | string | Specifies the interval (in hours) at which the server saves old log messages to another file. (Requires that you specify a file rotation type of TIME.)  Minimum value: 1  For example: "24"  Default value is an empty string |
| number\_of\_files\_limited *(required)* | string | Indicates whether to limit the number of log files that this server instance creates to store old messages. (Requires that you specify a file rotation type of SIZE or TIME.)  After the server reaches this limit, it deletes the oldest log file and creates a new log file with the latest suffix.  If you do not enable this option, the server creates new files indefinitely and you must clean up these files as you require.  This value has to be "true" or "plain".  For example: "true"  Default value is an empty string. |
| file\_count  *(required)* | string | Specifies the maximum number of log files that the server creates when it rotates the log. This number does not include the file that the server uses to store current messages. (Requires that you enable Number of Files Limited.)  Minimum value: 1  Maximum value: 99999  For example: "125"  Default value is an empty string. |

## cluster

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the WebLogic cluster.  For example: "auditCluster"  Default value is an empty string. |
| messaging\_mode  *(required)* | string | Specifies the messaging type used in the cluster.  Multicast messaging is provided for backwards compatibility.  Unicast, the default, is recommended for new clusters.  This value has to be "unicast" or "multicast".  For example: "unicast"  Default value is an empty string. |
| servers  *(optional)* | array of servers | See the **server** section for the definition |

## server

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the WebLogic Managed server to be created  For example: "auditAS01"  Default value is an empty string. |
| host  *(required)* | string | Specifies the listen host or IP of the Weblogic Managed server.  For example: "saedev01"  Default value is an empty string. |
| port  *(required)* | string | Specifies the listen port the Weblogic admin server.  For example: "7121"  Default value is an empty string. |
| type  *(required)* | string | Specifies the type of the WebLogic server. Since this is a Managed server, the value here should be "managed"  For example: "managed"  Default value is an empty string. |
| machine  *(optional)* | machine | See the **machine** section for the definition |
| server\_start  *(optional)* | server\_start | See the **server\_start** section for the definition |
| logging  *(optional)* | logging | See the **logging** section for the definition |

## deployment

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the WebLogic Managed server to be created  For example: "auditAS01"  Default value is an empty string. |
| path  *(required)* | string | Specifies the path to the source of the deployable unit on the Administration Server.  If the source path is relative, it is resolved relative to InstallDir/app if InstallDir is not null; otherwise, it is resolved relative to the domain root.  Use absolute source path to get a fully resolved value.  For example: "/opt/wl12c/domains/audit/applications/audit/auditApp.ear"  Default value is an empty string |
| stage\_mode  *(required)* | string | Specifies whether an application's deployment plan is copied from a source on the Administration Server to the Managed Server’s staging area during application preparation.  Plan staging mode for an application can only be set the first time the application is deployed. Once the plan staging mode for an application is set, it cannot be changed while the application is configured in the domain. The only way to change the plan staging mode is to undeploy and then redeploy the application.  This attribute overrides the server's plan staging mode.  Valid values:   * "callback-polling" * "nostage" * "stage" * "external\_stage"   For example: "nostage"  Default value is an empty string |
| targets  *(required)* | array of strings | Specifies the WebLogic Server instances and/or clusters to which you want to deploy this application.  More than one target can be set which can be separated by a comma.  For example:  **[** "auditCluster"**]**  Default value is an empty array of strings. |

## data\_source

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| jdbc\_name  *(required)* | string | Specifies the unique name that identifies this data source in the WebLogic domain.  For example: "CommonDataSource"  Default value is an empty string |
| jndi\_name  *(required)* | string | Specifies the JNDI path to where this data source is bound. By default, the JNDI name is the name of the data source.  Applications that look up the JNDI path will get a javax.sql.DataSource instance that corresponds to this data source.  For example: "weblogic.CommonDataSource"  Default value is an empty string |
| driver  *(required)* | string | Specifies the full package name of JDBC driver class used to create the physical database connections in the connection pool. (Note that this driver class must be in the classpath of any server to which it is deployed.)  For example: "oracle.jdbc.OracleDriver"  Default value is an empty string |
| two\_phase\_commit  *(required)* | string | Enables a non-XA JDBC connection to emulate participation in distributed transactions using JTA. Select this option only if your application can tolerate heuristic conditions.  With this option, the transaction branch in which the connection is used always returns success for the prepare phase of the transaction. This option offers performance benefits, but also has risks to data in some failure conditions.  Example of possible values: "None"and "TwoPhaseCommit"  For example: "None"  Default value is an empty string |
| username  *(required)* | string | Specifies the database account user name to create database connections.  For example: "RUSAUDIT"  Default value is an empty string |
| password  *(required)* | string | Specifies the database account password to use to create database connections.  For example: "rusauditd"  Default value is an empty string |
| database\_name  *(required)* | string | Specifies the name of the database.  For example: "RUSD"  Default value is an empty string |
| database\_host  *(required)* | string | Specifies the name or IP address of the database server.  For example: "10.77.6.15"  Default value is an empty string |
| database\_port  *(required)* | string | Specifies the port on the database server used to connect to the database.  For example: "1521"  Default value is an empty string |
| url  *(required)* | string | Specifies the URL of the database to connect to. The format of the URL varies by JDBC driver.  The URL is passed to the JDBC driver to create the physical database connections.  For example: "jdbc:oracle:thin:@10.77.6.15:1521:RUSD"  Default value is an empty string |
| test\_table\_name  *(required)* | string | Specifies the name of the database table to use when testing physical database connections. This name is required when you specify a **Test Frequency** and enable **Test Reserved** **Connections**.  The default SQL code used to test a connection is select count(\*) from **TestTableName**  Most database servers optimize this SQL to avoid a table scan, but it is still a good idea to set the **Test Table Name** to the name of a table that is known to have few rows, or even no rows.  If the **Test Table** **Name** begins with SQL, then the rest of the string following that leading token will be taken as a literal SQL statement that will be used to test connections instead of the standard query. For example: SQL BEGIN; Null; END;  For an Oracle database, you can reduce the overhead of connection testing by setting **Test Table Name** to SQL **PINGDATABASE** which uses the **pingDatabase()** method to test the Oracle connection.  For example: "SQL SELECT 1 FROM DUAL"  Default value is an empty string |
| properties  *(required)* | array of **property** | The list of properties passed to the JDBC driver that is used to create physical database connections. For example: server=dbserver1. List each property=value pair on a separate line.  To enable driver-level features, add the driver property and its value to the Properties list. WebLogic Server sets driver-level properties in the Properties list on the driver's ConnectionPoolDataSource object.  **Note:** For security reasons, when WebLogic Server is running in Production mode, you cannot specify database passwords in this properties list. Data source deployment will fail if a password is specified in the properties list. To override this security check, use the command line argument "weblogic.management.allowClearTextPasswords" when starting the server.  See the **property** section for the definition |

## property

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| key  *(required)* | string | Specifies the **property** value of the pair.  For example: "user"  Default value is an empty string |
| value  *(required)* | string | Specifies the **value** of the pair.  For example: "RUSAUDIT"  Default value is an empty string |

## security\_realm

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the security Realm  For example: "myrealm"  Default value is an empty string |
| provider  *(required)* | array of providers | See the **provider** section for the definition |

## provider

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| authentications  *(**optional)* | array of authentications | An Authentication provider allows WebLogic Server to establish trust by validating a user. You must have one Authentication provider in a security realm, and you can configure multiple Authentication providers in a security realm. Different types of Authentication providers are designed to access different data stores, such as LDAP servers or DBMS. You can also configure a Realm Adapter Authentication provider that allows you to work with users and groups from previous releases of WebLogic Server.  See the **authentication** section for the definition |
| authorizations  *(optional)* | array of authorizations | An Authorization provider controls access to WebLogic resources based on user identity or other information. You must have one Authorization provider in a security realm, and you can configure multiple Authorization providers in a security realm.  See the **authentication** section for the definition |
| role\_mappings  *(optional)* | array of role\_mapping | A Role Mapping provider supports dynamic role associations by obtaining a computed set of security roles granted to a requester for a given WebLogic resource. You must have one Role Mapping provider in a security realm, and you can configure multiple Role Mapping providers in a security realm.  See the **role\_mapping** section for the definition |
| auditings  *(optional)* | array of auditing | An Auditing provider collects, stores, and distributes information about operating requests and the outcome of those requests for the purposes of non-repudiation. An Auditing provider can also handle information about configuration changes for auditing purposes. You can configure multiple Auditing providers in a security realm, but none are required. The default security realm does not include an Auditing provider.  See the **auditing** section for the definition |
| keystores  *(optional)* | array of keystore | A keystore is a mechanism designed to create and manage files that store private keys and trusted certificate authorities (CAs). The WebLogic Keystore provider locates instances of keystores. (Note that WebLogic Keystore providers are deprecated.)  See the **keystore** section for the definition |

## authentication (DefaultAuthenticator)

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| mode  *(optional)* | string | Specifies whether adding a new Authentication provider or editing an existing one.  Value must be "add" or "edit"  For example: "add"  Default value is "add" |
| name  *(required)* | string | Specifies the name of the Authentication provider.  For example: "DefaultAuthenticator"  Default value is an empty string |
| control\_flag  *(required)* | string | Returns how the login sequence uses the Authentication provider.  A "REQUIRED" value specifies this LoginModule must succeed. Even if it fails, authentication proceeds down the list of LoginModules for the configured Authentication providers. This setting is the default.  A "REQUISITE" value specifies this LoginModule must succeed. If other Authentication providers are configured and this LoginModule succeeds, authentication proceeds down the list of LoginModules. Otherwise, control is return to the application.  A "SUFFICIENT" value specifies this LoginModule need not succeed. If it does succeed, return control to the application. If it fails and other Authentication providers are configured, authentication proceeds down the LoginModule list.  An "OPTIONAL" value specifies this LoginModule need not succeed. Whether it succeeds or fails, authentication proceeds down the LoginModule list.  For example: "SUFFICIENT"  Default value is an empty string |

## authentication (IPlanetAuthenticator)

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| mode  *(optional)* | string | Specifies whether adding a new Authentication provider or editing an existing one.  Value must be "add" or "edit"  For example: "add"  Default value is "add" |
| name  *(required)* | string | Specifies the name of the Authentication provider.  For example: "DefaultAuthenticator"    Default value is an empty string |
| control\_flag  *(required)* | string | Returns how the login sequence uses the Authentication provider.  A "REQUIRED" value specifies this LoginModule must succeed. Even if it fails, authentication proceeds down the list of LoginModules for the configured Authentication providers. This setting is the default.  A "REQUISITE" value specifies this LoginModule must succeed. If other Authentication providers are configured and this LoginModule succeeds, authentication proceeds down the list of LoginModules. Otherwise, control is return to the application.  A "SUFFICIENT" value specifies this LoginModule need not succeed. If it does succeed, return control to the application. If it fails and other Authentication providers are configured, authentication proceeds down the LoginModule list.  An "OPTIONAL" value specifies this LoginModule need not succeed. Whether it succeeds or fails, authentication proceeds down the LoginModule list.  For example: "SUFFICIENT"  Default value is an empty string |
| type  *(required)* | string | Creates an Authentication provider in this security realm. The new Authentication provider is added to the end of the list of Authentication providers configured in this security realm.  The type of this Authentication provider, for example, "weblogic.security.providers.authentication.IPlanetAuthenticator".  Possible class types are:  **Oracle Internet Directory**  "weblogic.security.providers.authentication.OracleInternetDirectoryAuthenticator"  **Oracle Virtual Directory**  "weblogic.security.providers.authentication.OracleVirtualDirectoryAuthenticator"  **Microsoft AD**  "weblogic.security.providers.authentication.ActiveDirectoryAuthenticator"  **OpenLDAP**  "weblogic.security.providers.authentication.OpenLDAPAuthenticator"  **eDirectory**  "weblogic.security.providers.authentication.NovellAuthenticator"  **SunOne LDAP**  "weblogic.security.providers.authentication.IPlanetAuthenticator"  Default value is an empty string |
| host  *(required)* | string | Specifies the host name or IP address of the LDAP server.  For example: "10.77.30.50"  Default value is an empty string |
| port  *(required)* | string | Specifies the port number on which the LDAP server is listening.  For example: "3899"  Default value is an empty string |
| principal  *(required)* | string | Specifies the Distinguished Name (DN) of the LDAP user that WebLogic Server should use to connect to the LDAP server.  Note that value you enter for principal must be an LDAP administrator who has the privilege to search users and groups in iPlanet. If the LDAP administrator does not have privileges to search iPlanet, an LDAP exception with error code 50 is generated.  For example: "cn=Directory Manager"  Default value is an empty string |
| password  *(required)* | string | Specifies the credential (usually a password) used to connect to the LDAP server.  If this password has not been set, WebLogic Server generates a password at startup, initializes the attribute, and saves the configuration to the config.xml file. If you want to connect to the embedded LDAP server using an external LDAP browser and the embedded LDAP administrator account (cn=Admin), change this attribute from the generated value.  For example: "hunter2"  Default value is an empty string |
| user\_base\_dn *(required)* | string | Specifies the base distinguished name (DN) of the tree in the LDAP directory that contains **users**.  For example: "ou=people,dc=mto,dc=gov,dc=on,dc=ca"  Default value is an empty string |
| group\_base\_dn  *(required)* | string | Specifies the base distinguished name (DN) of the tree in the LDAP directory that contains **groups**.  For example: "ou=groups,dc=mto,dc=gov,dc=on,dc=ca"  Default value is an empty string |

## authorization

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the Authorization provider.  For example: "DefaultAuthorizer"  Default value is an empty string |
| type  *(required)* | string | Specifies the type of Authorization provider to be created.  For example: "weblogic.security.providers.authorization.DefaultAuthorizer"  Default value is an empty string |

## role\_mapping

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the Role Mapping provider.  For example: "DefaultAuthorizer"  Default value is an empty string |
| type  *(required)* | string | Specifies the type of Role Mapping provider to be created.  For example: "weblogic.security.providers.authorization.DefaultRoleMapper"  Default value is an empty string |

## keystore

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the Keystore provider.  For example: "DefaultAuthorizer"  Default value is an empty string |
| type  *(required)* | string | Specifies the type of Keystore provider to be created.  For example: "weblogic.security.providers.pk.DefaultKeyStore"  Default value is an empty string |

## persistent\_store

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the new file store.  For example: "auditAs01FileStore"  Default value is an empty string |
| type  *(required)* | string | Specifies the pathname to the directory on the file system where the file store is kept. This directory must exist on your system.  For example: "/tmp"  Default value is an empty string |
| target  *(required)* | target | Specifies server instances, dynamic clusters, or migratable targets defined in the current domain that are candidates for hosting a file store or JDBC store.  When selecting a dynamic cluster, the file store must be targeted to the same dynamic cluster as the JMS server. When selecting a migratable target, the file store must be targeted it to the same migratable target as the migratable JMS server or SAF agent. As a best practice, a path service should use its own custom store and migratable target.  See the **target** section for the definition |

## target

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the target.  For example: "auditAs01"  Default value is an empty string |
| type  *(required)* | string | Specifies the type of the target. Possible values are: "Servers" or "Clusters"  For example: "Servers"  Default value is an empty string |

## messaging

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| jms\_servers  *(required)* | array of jms\_servers | JMS servers act as management containers for the queues and topics in JMS modules that are targeted to them.  See the **jms\_server** section for the definition |
| jms\_modules  *(required)* | array of jms\_modules | JMS system resources are configured and stored as modules similar to standard J2EE modules. Such resources include queues, topics, connection factories, templates, destination keys, quota, distributed queues, distributed topics, foreign servers, and JMS store-and-forward (SAF) parameters. You can administratively configure and manage JMS system modules as global system resources.  See the **jms\_module** section for the definition |

## jms\_server

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the JMS server.  For example: "auditAs01JMSServer"  Default value is an empty string |
| persistent\_store  *(required)* | persistent\_store | The file or database in which this JMS server stores persistent messages. If unspecified, the JMS server uses the default persistent store that is configured on each targeted WebLogic Server instance. If the JMS server has a store configured, then the configured store is used to store persistent messages.  The disk-based file store or JDBC-accessible database store that you specify must be targeted to the same server, cluster, or migratable target instance as this JMS server. Multiple services on the same WebLogic Server instance, including multiple JMS servers, may share the same persistent store. Each service's persistent data will be kept apart.  If you specify a PersistentStore, the deprecated Store field must not be set. If neither the PersistentStore field nor the Store field are set, the JMS server supports persistent messaging using the default persistent store for the targeted WebLogic Server instance.  See the **persistent\_store** section for the definition |
| target  *(required)* | target | Specifies server instances, dynamic clusters, or migratable targets defined in the current domain that are candidates for hosting a file store or JDBC store.  When selecting a dynamic cluster, the file store must be targeted to the same dynamic cluster as the JMS server. When selecting a migratable target, the file store must be targeted it to the same migratable target as the migratable JMS server or SAF agent. As a best practice, a path service should use its own custom store and migratable target.  See the **target** section for the definition |

## jms\_module

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the System Module.  For example: "auditSystemModule"  Default value is an empty string |
| sub\_deployments | array of sub\_deployments | A subdeployment is a mechanism by which JMS module resources (such as queues, topics, and connection factories) are grouped and targeted to a server resource (such as JMS servers, server instances, or cluster).  See the **sub\_deployment** section for the definition |
| targets  *(required)* | array of targets | Specifies server instances, dynamic clusters, or migratable targets defined in the current domain that are candidates for hosting a file store or JDBC store.  When selecting a dynamic cluster, the file store must be targeted to the same dynamic cluster as the JMS server. When selecting a migratable target, the file store must be targeted it to the same migratable target as the migratable JMS server or SAF agent. As a best practice, a path service should use its own custom store and migratable target.  See the **target** section for the definition |
| uniform\_distribute\_queues  *(required)* | array of uniform\_distribute\_queues | Defines a set of queues that are distributed on multiple JMS servers, but which are accessible as a single, logical queue to JMS clients.  Distributed queues can help with message load balancing and distribution, and have many of the same properties as standalone queues.  See the **uniform\_distribute\_queue** section for the definition |
| connection\_factories  *(required)* | array of connection\_factories | Defines a set of connection configuration parameters that are used to create connections for JMS clients.  Connection factories can configure properties of the connections returned to the JMS client, and also provide configurable options for default delivery, transaction, and message flow control parameters.  See the **connection\_factory** section for the definition |

## sub\_deployment

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the subdeployment  For example: "auditSystemModule"  Default value is an empty string |
| targets  *(required)* | array of targets | Specifies server instances, dynamic clusters, or migratable targets defined in the current domain that are candidates for hosting a file store or JDBC store.  When selecting a dynamic cluster, the file store must be targeted to the same dynamic cluster as the JMS server. When selecting a migratable target, the file store must be targeted it to the same migratable target as the migratable JMS server or SAF agent. As a best practice, a path service should use its own custom store and migratable target.  See the **target** section for the definition |

## uniform\_distribute\_queue

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the new destination.  For example: "auditLogQueue"  Default value is an empty string |
| jndi\_name  *(required)* | string | Specifies the JNDI name to use to look up the new destination.  For example: "weblogic.jms.auditLogQueue"  Default value is an empty string |
| sub\_deployment\_name  *(required)* | string | Specifies the subdeployment to be used.  A subdeployment is a mechanism by which JMS resources are grouped and targeted to a server instance, cluster, or SAF agent. You can also reconfigure subdeployment targets later by using the parent module's subdeployment management page.  For example: "auditGroup"  Default value is an empty string |

## connection\_factory

|  |  |  |
| --- | --- | --- |
| Key | Type | Example |
| name  *(required)* | string | Specifies the name of the new destination.  For example: "emailConnectionFactory"  Default value is an empty string |
| jndi\_name  *(required)* | string | Specifies the JNDI name to use to look up the new destination.  For example: "common.jms.emailConnectionFactory"  Default value is an empty string |
| default\_target\_enabled  *(required)* | string | Specifies whether this JMS resource defaults to the parent module's targeting or uses the subdeployment targeting mechanism.  When set to true, this resource implicitly inherits the targeting of its parent module. When set to false, this resource gets targeted based its subdeployment's targets, if one is specified.  Possible values are "true" or "false".  For example: "true"  Default value is an empty string |