

## **Oracle® Fusion Middleware**

Administering Zero Downtime Patching Workflows

12c (12.2.1.1.0)

**E69724-01**

June 2016

This document describes how to create workflows to move a domain from an existing Oracle Home to a patched Oracle Home, update to a new Java version, update applications in a domain, or perform a rolling restart without any loss of service.

Copyright © 2016, 2016, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

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

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

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

---

---

# Contents

Preface .....	v
Audience .....	v
Documentation Accessibility .....	v
Related Documents.....	v
Conventions.....	vi
Guide to This Document.....	vi
<b>1 Introduction to Zero Downtime Patching</b>	
1.1 What Is Zero Downtime Patching? .....	1-1
1.2 Types of Patching Workflows.....	1-1
1.3 The Patching Workflow Process.....	1-2
1.4 Reverting an Update .....	1-3
1.5 Rolling Out a Patched Oracle Home: Overview .....	1-3
1.6 Rolling Out a New Java Version: Overview .....	1-5
1.7 Rolling Out Updated Applications: Overview.....	1-6
1.8 In-Memory Session Replication for ZDT Rollouts.....	1-9
<b>2 Preparing for Zero Downtime Patching</b>	
2.1 ZDT Patching Restrictions.....	2-1
2.2 Preparing to Migrate Singleton Services .....	2-2
2.2.1 Creating a JSON File for Migrating Singleton Services .....	2-4
2.3 Preparing to Roll Out a Patched Oracle Home .....	2-6
2.3.1 Creating a Patched Oracle Home Archive Using OPatchAuto .....	2-7
2.3.2 Distributing the Patched Archive to Each Node Using OPatchAuto .....	2-7
2.3.3 Creating a Second Oracle Home .....	2-8
2.3.4 Applying Patches to the Second Oracle Home .....	2-9
2.3.5 Creating an Archive and Distributing It to Each Node .....	2-10
2.4 Preparing to Upgrade to a New Java Version .....	2-10
2.5 Preparing to Update to New Application Versions.....	2-11
2.5.1 The Effects of Staging Modes.....	2-11
2.5.2 Creating an Application Update JSON File.....	2-12

### 3 Configuring and Monitoring Workflows

3.1	Strategies for Rolling Out a Patched Oracle Home.....	3-1
3.2	Starting the Administration Server .....	3-2
3.3	Using OPatchAuto to Initiate, Revert, and Resume Rollouts .....	3-3
3.3.1	Using OPatchAuto to Initiate a Rollout .....	3-3
3.3.2	Using OPatchAuto to Resume a Failed Rollout.....	3-4
3.3.3	Using OPatchAuto to Revert a Rollout .....	3-5
3.4	Using WLST to Initiate and Monitor Workflows.....	3-5
3.4.1	Rolling Out a New Oracle Home .....	3-10
3.4.2	Updating Your Java Version.....	3-10
3.4.3	Updating Both Oracle Home and the Java Version .....	3-11
3.4.4	Rolling Out Updated Applications.....	3-11
3.4.5	Reverting to the Previous Oracle Home, Java Home, or Applications .....	3-12
3.4.6	Initiating a Rolling Restart of Servers or Partitions.....	3-12
3.4.7	Monitoring Workflow Progress .....	3-13
3.4.8	Executing, Reverting, and Resuming Stopped Workflows.....	3-14
3.4.9	Useful WLST Commands for Workflows.....	3-15
3.4.10	Sample WLST Script .....	3-15
3.5	Using the WebLogic Server Administration Console to Create and Monitor Workflows .	3-17
3.5.1	Accessing ZDT Workflow Functions in the WebLogic Server Administration Console.....	3-17
3.5.2	Creating a New Workflow for a Domain, Clusters, or Servers .....	3-17
3.5.3	Monitoring and Managing Workflows .....	3-20
3.5.4	Workflow Statuses .....	3-23
3.5.5	Workflow Logging.....	3-23

---

# Preface

This document, *Administering Zero Downtime Patching Workflows*, describes how to move a domain from an existing Oracle home to a patched Oracle home, update to a new Java version, or update applications in a domain without any loss of service. It describes how to create workflows that methodically apply the changes to the servers in the domain while keeping the domain available. It also describes how to monitor the progress of workflow tasks and revert the domain to its previous state.

## Audience

This document is written for WebLogic Server administrators and operators who are responsible for applying updates to a domain, such as Oracle patches to an Oracle home, new Java versions, or application updates. It is assumed that readers are familiar with the WebLogic Server Administration Console, WebLogic Scripting Tool (WLST), and the operating system and platform on which Oracle WebLogic Server is installed.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

### Access to Oracle Support

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

## Related Documents

For more information, see the following Oracle Fusion Middleware documents:

- *Patching with OPatch*
- *Administering Node Manager for Oracle WebLogic Server*
- *Understanding the WebLogic Scripting Tool*
- *WLST Command Reference for WebLogic Server*
- *Deploying Applications to Oracle WebLogic Server*

- *MBean Reference for Oracle WebLogic Server*

## Conventions

The following text conventions are used in this document:

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

## Guide to This Document

This document is organized as follows:

- [Introduction to Zero Downtime Patching](#), provides an overview of Zero Downtime Patching, including the types of patching workflows that you can create, how the patching workflow proceeds, and how patching is reverted.
- [Preparing for Zero Downtime Patching](#), describes the preliminary steps that must be completed before you can configure a patching workflow.
- [Configuring and Monitoring Workflows](#), describes how to configure a patching workflow that moves a domain to a patched Oracle home, updates the Java version for a domain, updates the applications for a domain, or all three.

---

# Introduction to Zero Downtime Patching

This chapter provides an overview of Zero Downtime Patching, including the types of workflows that you can create, how the patching workflow proceeds, and how patching is reverted.

This chapter includes the following sections:

- [What Is Zero Downtime Patching?](#)
- [Types of Patching Workflows](#)
- [The Patching Workflow Process](#)
- [Reverting an Update](#)
- [Rolling Out a Patched Oracle Home: Overview](#)
- [Rolling Out a New Java Version: Overview](#)
- [Rolling Out Updated Applications: Overview](#)
- [In-Memory Session Replication for ZDT Rollouts](#)

## 1.1 What Is Zero Downtime Patching?

Zero Downtime Patching (ZDT Patching) automates the rollout of out-of-place patching or updates across a domain while allowing your applications to continue servicing requests. After defining your patching strategy, you can use either the WebLogic Scripting Tool (WLST) or the WebLogic Server Administration Console to orchestrate the rollout of updates across some or all of the servers in your domain.

Although WebLogic Server has supported rolling upgrades since version 9.2, the process has always been manual. ZDT Patching automates this process by using workflows that you define. You can patch or update any number of nodes in a domain with little or no manual intervention. Changes are rolled out to one node at a time, allowing a load balancer such as Oracle Traffic Director to redirect incoming traffic to the remaining nodes until the node has been updated.

## 1.2 Types of Patching Workflows

ZDT Patching supports the following tasks. You can create a workflow that performs any one of these tasks. You can also create a workflow that performs any combination of an Oracle home update, Java version update, and application update.

- **Moving servers to a patched Oracle home:** The workflow transitions the Administration Server or clusters or both to another Oracle home that has already been patched using the OPatch utility.

- **Updating to a new Java version:**The workflow updates the Administration Server or clusters or both to use a newly installed Java home.
- **Deploying updated applications:**The workflow deploys updated applications to the selected clusters.
- **Performing a rolling restart of servers:**The workflow sequentially restarts the Administration Server or servers in the selected clusters or both safely, including graceful shutdown of the servers and starting them up again.
- **Performing a rolling restart of partitions:** The workflow sequentially restarts the partitions in a cluster or in the specified resource group, including the graceful shutdown of each partition, one at a time, and starting them up again.

Prior to creating a patching workflow, you must complete the preliminary steps for each of these tasks with the exception of rolling restarts. See [Preparing for Zero Downtime Patching](#), for more information.

## 1.3 The Patching Workflow Process

When you use a ZDT patching workflow to roll out an update, the rollout:

- Systematically works its way through each applicable node
- Identifies the servers on the node that are included in the rollout
- Gracefully shuts down those servers
- When switching to a patched Oracle home:
  - Backs up the existing Oracle home to a backup directory
  - Calls Node Manager to switch the contents of the current Oracle home to the contents of the specified Oracle home
- When updating to a new Java version:
  - Updates all scripts in the domain's Oracle home that contain a reference to Java home to point to the new Java home
  - Updates all scripts in the domain's home directory that contain a reference to Java home to point to the new Java home
- When updating to new application versions:
  - Locates the current directory for each application
  - Moves the current directory for each application to a backup location
  - Moves the directory for the new version of each application to the location of each original application
- Restarts each server once the update has completed on the node

The workflow executes the appropriate steps in order and monitors the success of each step. If a step fails, the workflow may attempt to retry it. If a step cannot be completed successfully, then the workflow reverts each previous step in order. The revert process can be configured to execute automatically or can be initiated manually, as described in [Reverting an Update](#).



## 1.4 Reverting an Update

ZDT patching is also able to revert an update at any point in the process, even after it has completed. Updates can be reverted:

- **Automatically**—When creating a workflow, you can opt to have the update revert automatically if there is a failure. The update will be rolled back from the point of failure, starting with the last successfully completed step.
- **Manually**—While a workflow is in progress, you can stop it and revert the process at any point. The update will then be rolled back, starting with the last successfully completed step.

After a workflow has completed, you can create a workflow to reverse the update that was made. The revert process differs slightly depending on the update. If you are reverting to the previous Oracle home, then you are provided with an option to specify that the process is a rollback. For Java and applications, to revert you can point to the previous version of Java or the application.

For more information about reverting an update, see [Executing, Reverting, and Resuming Stopped Workflows](#).

## 1.5 Rolling Out a Patched Oracle Home: Overview

This section provides an overview of how to roll out a patched Oracle home to all nodes in your domain. Prior to doing the rollout, ensure that the following conditions are met:

- The domain is distributed across all nodes and stored in the same location on all nodes.
- The existing Oracle home is in the same location on all nodes.
- Node Manager is running on all nodes.
- All Managed Servers in all clusters that will be included in the rollout are running.

See [ZDT Patching Restrictions](#), for additional requirements and restrictions. [Figure 1-1](#) shows the sequence of operations that are performed for an Oracle home rollout on each node, regardless of whether you use OPatchAuto, WLST, or the WebLogic Server Administration Console to perform the rollout.

To roll out a patched Oracle home, perform the following tasks:

1. Create and distribute the patched Oracle home archive in either of the following ways:
  - Use OPatchAuto
  - a. Create the patched Oracle home archive.  
For details, see [Creating a Patched Oracle Home Archive Using OPatchAuto](#).
  - b. Distribute the archive to all nodes to which you want to roll out the patched Oracle home.  
For details, see [Distributing the Patched Archive to Each Node Using OPatchAuto](#).

- Manually create and distribute the patched Oracle home archive:
  - a. Use the `copyBinary` command to create an archive of your existing Oracle home.  
For details on this step and the next step, see [Creating a Second Oracle Home](#).
  - b. Use the `pasteBinary` command to create an Oracle home to be patched on a development or test system that has a domain topology similar to your production domain. This gives you an Oracle home that has the same patch level and products as you have on your production system.
  - c. Use OPatch to apply the desired patch or patches to the Oracle home on your development or test system.  
For details, see [Applying Patches to the Second Oracle Home](#), and *Patching with OPatch*.
  - d. Test and verify the patched Oracle home.
  - e. When you are satisfied that the patched Oracle home is stable, use `copyBinary` to create an archive of the patched Oracle home.  
For details on this and the next step, see [Creating an Archive and Distributing It to Each Node](#).
  - f. Distribute this archive to all nodes in your production system.

---

---

**Note:**

There is no need to use `pasteBinary` to create the archive on each node. The rollout process will create the new Oracle home on each node from the archive.

---

---

2. Create a ZDT workflow to roll out the patched Oracle home to your Administration Server. You can do this in any of the following ways:
  - Use OPatchAuto to initiate the rollout and specify Administration Server as the target.  
For details, see [Using OPatchAuto to Initiate a Rollout](#).
  - Use the WLST `rolloutOracleHome` command and specify the Administration Server as the rollout target.  
For details, see [Rolling Out a New Oracle Home](#).
  - In the WebLogic Server Administration Console, click the **ZDT Control** tab and navigate to the **Servers** tab. In the Servers tab, select the Administration Server, and then initiate and configure the workflow. You can also click the **Domain** tab and select the domain to initiate the workflow.  
For details, see [Creating a New Workflow for a Domain, Clusters, or Servers](#).
3. After the workflow completes successfully, create another ZDT workflow to roll out the patched Oracle home to the clusters in your domain. You can do this in any of the following ways:
  - Use OPatchAuto to initiate the rollout and specify a cluster or a comma-separated list of clusters as the rollout target.

For details, see [Using OPatchAuto to Initiate a Rollout](#).

- Use the WLST `rolloutOracleHome` command and specify a comma-separated list of clusters as the rollout target.
- In the WebLogic Server Administration Console, select the **ZDT Control > Clusters** tab, select the clusters to which you want to roll out the Oracle home, and then initiate and configure the workflow.

---



---

**Note:**

You can combine the last two steps into one workflow by either specifying the domain as the target in the `opatchauto` or `rolloutOracleHome` command, or by initiating and configuring the workflow from the **ZDT Control > Domain** tab.

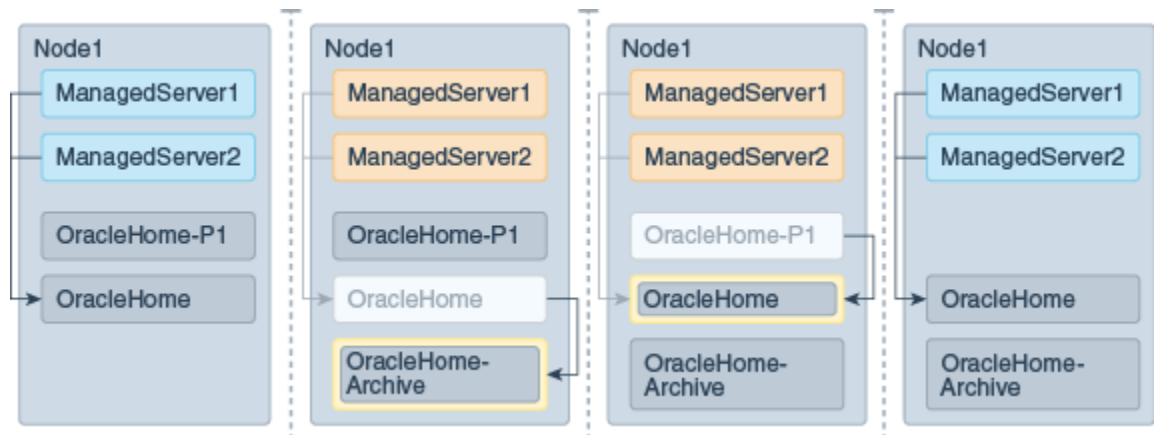
---



---

**Figure 1-1 Oracle Home Rollout Operations**

This figure shows the sequence of operations that are performed for an Oracle home rollout on each node, regardless of whether you use `OPatchAuto`, WLST, or the WebLogic Server Administration Console to perform the rollout.



## 1.6 Rolling Out a New Java Version: Overview

This section provides an overview of how to roll out a new Java version to all nodes in your domain. Prior to doing the rollout, ensure that the following conditions are met:

- The domain is distributed across all nodes and is stored in the same location on all nodes.
- Oracle home must be in the same location on all nodes.
- Node Manager is running on all nodes.
- All Managed Servers in all clusters that will be included in the rollout is running.

See [ZDT Patching Restrictions](#), for additional requirements and restrictions.

To roll out a new Java version:

1. Install the new Java version on all nodes. The full path to this Java home must be the same on all nodes.

For more details, see [Preparing to Upgrade to a New Java Version](#).

2. Create a ZDT workflow to roll out the new Java home to your Administration Server. You can do this in either of the following ways:
  - Use the WLST `rolloutJavaHome` command and specify the Administration Server as the rollout target.  
For details, see [Updating Your Java Version](#).
  - In the WebLogic Server Administration Console, select the **ZDT Control > Servers** tab, select the Administration Server, and then initiate and configure the workflow.  
For details, see [Creating a New Workflow for a Domain, Clusters, or Servers](#).
3. After the workflow completes successfully, create another ZDT workflow to roll out the new Java home to the clusters in your domain. To do this, you can either:
  - Use the WLST `rolloutJavaHome` command and specify a comma-separated list of clusters as the rollout target.
  - In the WebLogic Server Administration Console, select the **ZDT Control > Clusters** tab, select the clusters to which you want to roll out the new Java version, and then initiate and configure the workflow.

---

---

**Note:**

You can combine the last two steps into one workflow by either specifying the domain as the target in the `rolloutJavaHome` command or by initiating and configuring the workflow from the **ZDT Control > Domain** tab.

---

---

## 1.7 Rolling Out Updated Applications: Overview

This section provides an overview of how to roll out new application versions to Managed Server nodes, partitions, or resource groups in your domain.

ZDT now supports WebLogic Server Multitenant to provide application rollout capabilities to both partitions and resource groups. This means that any application deployed to a resource group (partition scoped or global) or to a specific partition can now be updated without affecting other resource groups or partitions within that WebLogic Server instance. A WebLogic Server administrator can update applications defined in the resource group templates that are consumed by many partitions. They can also update applications deployed to a single partition. Partition administrators can update only those applications that are defined in their own partition. For more information about Multitenancy in WebLogic Server, see [Using WebLogic Server Multitenant](#).

Prior to doing the rollout, ensure that the following conditions are met:

- The domain that is being updated is distributed across all nodes and must be stored in the same location on all nodes.
- Oracle Home is in the same location on all nodes.
- Node Manager is running on all nodes.
- All Managed Servers in all clusters that will be included in the rollout is running.

- An instance of the partition that is being updated must be running on more than one cluster.

See [ZDT Patching Restrictions](#), for additional requirements and restrictions. [Figure 1-2](#), [Figure 1-3](#), and [Figure 1-4](#) illustrate the scenario for patching staged, no-stage, and external staged applications, respectively. The patched application source will be moved to the appropriate application source locations for each stage type during the rollout.

To roll out new application versions to your Managed Servers:

1. Place a copy of the updated application directory as follows:

- (Stage mode) Place a copy of each updated application directory on the domain's Administration Server.
- (No-stage mode and external stage mode) Place a copy of each updated application directory on each node that will be affected. The directory must be the same on each node.

For details, see [The Effects of Staging Modes](#).

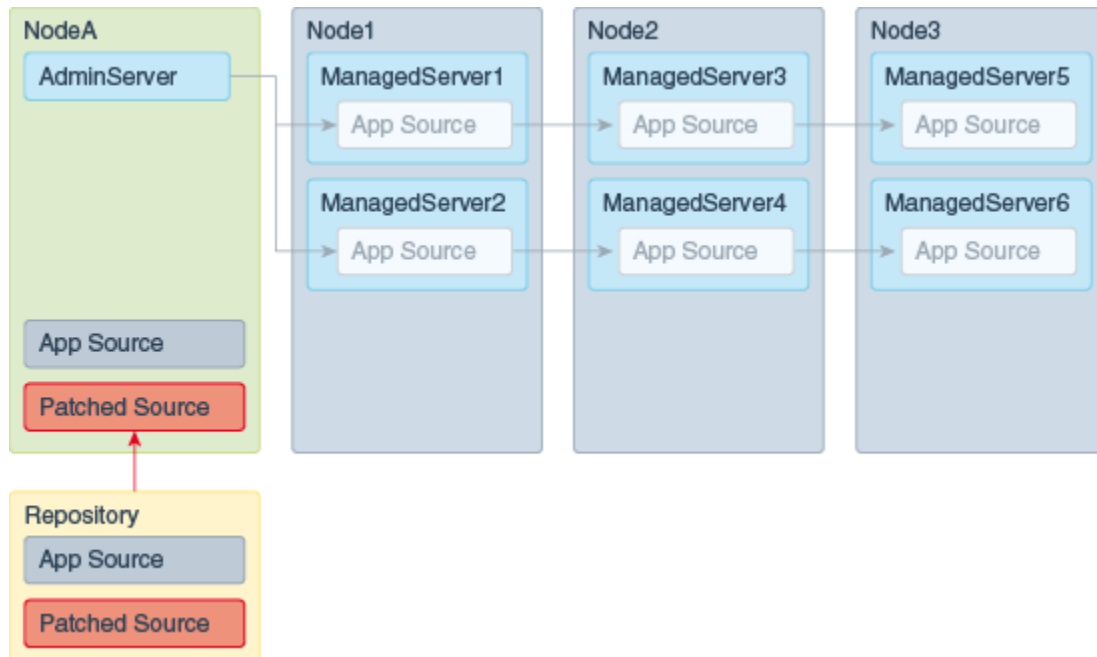
2. Create a JavaScript Object Notation (JSON) file that defines each application name, the partition name (if applicable), the resource group template name (if applicable), the path and file name for each updated application archive, and the path and file to which you want to back up the original application archive.

For details, see [Creating an Application Update JSON File](#).

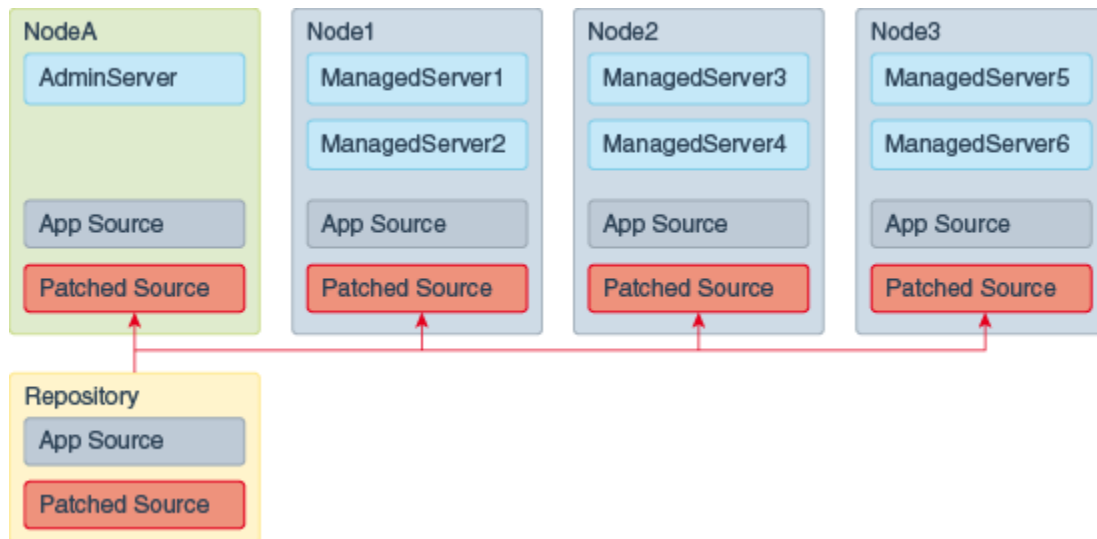
3. Create a ZDT workflow to roll out the new application versions. To do this, you can either:

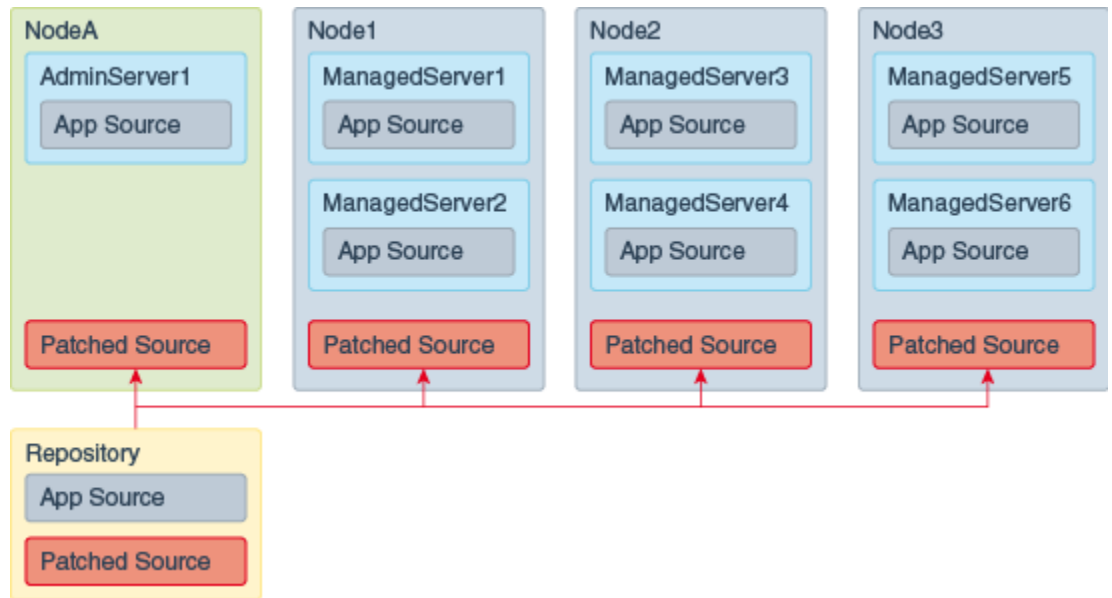
- Use the WLST `rolloutApplications` command and specify a comma-separated list of clusters as the rollout target.
- In the Administration Console, select the **ZDT Control > Clusters** tab, select the Clusters to which you want to roll out the applications, and then initiate and configure the workflow.

**Figure 1-2 Patching Staged Applications**



**Figure 1-3 Patching No-Stage Applications**



**Figure 1-4 Patching External Staged Applications**

## 1.8 In-Memory Session Replication for ZDT Rollouts

For web applications that use in-memory session replication, the in-memory sessions are never replicated or persisted to allow for failover. As a result web applications may lose session state due to a sudden failure of a server or front-end misdirection causing the request to land on a server without the session.

With regard to Zero Downtime (ZDT) rollouts, when you shut down any server that holds the in-memory session, the server waits for that session to complete before shutting down. Because the default value for session timeout is 1 hour, the server may be in the SUSPENDING state for 1 hour or even longer if sessions continue to be used or updated. If you do not wait for the session to complete its life cycle, then the state is lost because in-memory sessions are neither replicated nor persisted for web applications.

If you do not want to wait for an hour or longer, then Oracle recommends that you set the `shutdownTimeout` argument in the `WLST rollout` command to the time (in seconds) that you want the server to wait before shutting down. For information about using the `shutdownTimeout` argument, see [Table 3-1](#).





---

# Preparing for Zero Downtime Patching

Before you configure a patching workflow, you must complete the preliminary steps. These steps include installing and patching a new Oracle home, installing a new Java version, or installing updated applications on each node. There are certain restrictions that you must keep in mind prior to preparing for and creating a ZDT patching workflow.

This chapter includes the following sections:

- [ZDT Patching Restrictions](#)
- [Preparing to Migrate Singleton Services](#)
- [Preparing to Roll Out a Patched Oracle Home](#)
- [Preparing to Upgrade to a New Java Version](#)
- [Preparing to Update to New Application Versions](#)

## 2.1 ZDT Patching Restrictions

Prior to preparing for and creating a ZDT patching workflow, consider the following restrictions:

- The Managed Servers that are included in the workflow must be part of a cluster, and the cluster must span two or more nodes.
- If you want to roll out an update to the Managed Servers without targeting and updating the Administrations Server, then ensure that the Administration Server is on a different node than any of the Managed Servers being updated.
- If you are updating to a patched Oracle home, the current Oracle home must be installed locally on each node that will be included in the workflow. Although it is not required, Oracle also recommends that the Oracle home be in the same location on each node.
- When you are rolling out a new Oracle home using WLST commands, you must specify the path to the JAR archive that contains the Oracle home to roll out. Specifying a local directory is not supported when you are rolling out a new Oracle home. Only if you are rolling back to a previous Oracle home, you can specify the path to the local directory which must be the backup Oracle home directory from the previous rollout that you want to roll back to.
- If Managed Servers on a node belong to different clusters and those clusters share the same Oracle home, then if you include one of those clusters in a workflow, you must also include the other cluster in the workflow. For example, if Node 1 has Managed Server 1 in Cluster 1 and Managed Server 2 in Cluster 2, and both Cluster 1 and Cluster 2 share the same Oracle home, then if you include Cluster 1 in the

workflow, you must also include Cluster 2. This applies to Java home, Oracle home and application update rollouts.

- The domain directory must reside outside of the Oracle home directory.
- (Windows only) When you use the WebLogic Scripting Tool (WLST) to initiate a rollout of a new Oracle home, you cannot run WLST from any Oracle home that will be updated as part of the workflow. Instead, use one of the following options:
  - Run WLST from an Oracle home on a node that will not be included in the workflow. This Oracle home must be the same version as the Oracle home that is being updated on other nodes.
  - Run WLST from another Oracle home that is not part of the domain being updated. This Oracle home must be the same version as the Oracle home that is being updated. It can reside on any node, including the Administration Server node for the domain being updated.
  - Use the WebLogic Server Administration Console to initiate the workflow.
- (Windows only) Windows file locks may pose problems during the ZDT rollout operations. You must attempt to rectify these common file handle lock issues before executing a rollout on Windows to avoid rollout failure:
  - When you deploy an application by using the Administration Console, the Administration Server may hold a lock on the application source file. If this lock is not released, it could prevent subsequent application rollouts from functioning properly. To release the lock, you must log out of the Administration Console anytime after deploying the application and before initiating the rollout.
  - Using the WLST client on the Administration Server will cause the Oracle home directory to be locked. This will cause any rollout on that node, including a domain rollout to fail. To avoid this, use a WLST client installed on a node that is not targeted by the rollout, or initiate the rollout using the Administration Console.
  - Opening command terminals or applications residing in any directory under Oracle home may cause a file lock. As a result, you will be unable to update that particular Oracle home.
  - Any command terminal or application that references the application source file or a JAR file may cause a file lock, making it impossible to update that particular application.

## 2.2 Preparing to Migrate Singleton Services

All ZDT rollouts require a restart of the servers that are included in the rollout. One feature of the rollout is detection and handling of singleton services, such as Java Transaction API (JTA) and Java Messaging Service (JMS). To make these singleton services highly available during the rollout operation, ZDT patching takes advantage of the service migration mechanisms supported by WebLogic Server. For singleton services in your environment, service migration can be configured in either of the following ways:

- For migrating a singleton service that is configured using migratable targets, the service migration is configured as described in Service Migration in *Administering*

*Clusters for Oracle WebLogic Server.* If a service is configured using migratable targets and the migration policy is set to `exactly-once`, then the service automatically migrates during the graceful shutdown of a server. If, however, the migration policy for a service is `manual` or `failure-recovery`, then you must take steps to ensure that the service is migrated safely during server shutdown. To achieve this, you must define the migration properties in the JSON file as described in [Creating a JSON File for Migrating Singleton Services](#).

You must bear in mind the following issues restrictions when migrating singleton services that is configured using migratable targets:

- The data store for JMS servers must reside at a shared location to be used by the members of the cluster, without which the user might experience loss of messages. For more information, see *Using Shared Storage in Fusion Middleware High Availability Guide*.
- The `ClusterMBean` must be configured with the `setServiceActivationRequestResponseTimeout` method and its value must be set depending on the time taken for the migration to succeed.
- The `JNDI NameNotFoundException` is returned during lookup for JMS connection factories and destinations. This is a known limitation. For information about this limitation and its workaround, see note 1556832.1 at [My Oracle Support](#).
- As services migrate during the rollout, the JNDI lookup for JMS connection factories and destinations fail. In such cases of server failure, JMS applications attempt to reconnect to another available server for non-deterministic time till the migration succeeds. For more information about this feature, see *Recovering from a Server Failure in Developing JMS Applications for Oracle WebLogic Server*.
- For migrating a singleton service that is configured using the JMS cluster configuration, the service migration is configured (depending on your cluster type) as described in *Simplified JMS Cluster and High Availability Configuration in Administering JMS Resources for Oracle WebLogic Server*. If a service is configured using the JMS Cluster configuration, then the migration-policy must be set to `Always` to enable the automatic migration of services during the graceful shutdown of a server. If the migration-policy is `On-Failure` or `Off`, then you must take steps to ensure that the service is migrated safely during server shutdown. You must also ensure that the automatic `restart-in-place` option is explicitly disabled when using this simplified HA service migration model.

---



---

**Note:**

ZDT rollout allows you to specify whether a singleton service should be migrated before shutting down during patching. However, during the rollout operation, the user is not allowed to specify the migration of servers on the same machine. This is because, all servers on a machine experience shutdown during a rollout which may cause unavoidable downtime for users. Ensure that you always specify migration of services to a server on a different machine, failing which the rollout might fail.

Service migration involves shutting down one or more singleton services on the first server that is being rolled out. This means that the service is made available on the second server while rollout is in progress. Upon successful completion of the rollout, the services are migrated back to the newly patched first server. Since this process involves restarting of singleton services, the users can expect a brief downtime of services when the service is shut down on the first server and has not fully started on the second server. This would render the service unavailable and applications may experience a brief outage. The period of downtime of services may depend on factors including, hardware (both machine and network) performance, cluster size, the server startup time, and persistent message backlog in case of JMS.

---



---

## 2.2.1 Creating a JSON File for Migrating Singleton Services

To ensure that the singleton service is migrated safely during server shutdown, you must perform the following tasks:

- Create a JSON file to define migration properties for such services, as described in this section
- Configure the rollout to use the JSON file as described in [Configuring and Monitoring Workflows](#).

The JSON file must start with the following line:

```
{"migrations": [
```

Each singleton service migration that you need to migrate is defined using the parameters described in the following table.

Parameter	Description
source	The name of the source server from which the service is to be migrated. This parameter is required.
destination	For migrationType of <code>jms</code> , <code>jta</code> , or <code>all</code> , the name of the destination server to which the service is to be migrated. For migrationType of <code>server</code> , the name of another machine (node) in the domain on which Node Manager is running. This parameter is required if the migrationType is <code>jms</code> , <code>jta</code> , <code>server</code> , or <code>all</code> .

Parameter	Description
<code>migrationType</code>	<p>The type of migration, which can be one of the following types:</p> <ul style="list-style-type: none"> <li><code>jms</code> — Migrate all JMS migratable targets from the source server to the destination server.</li> <li><code>jta</code> — Migrate all JTA services from the source server to the destination server.</li> <li><code>server</code> — Invoke Whole Server Migration to perform a server migration. The destination must be a machine (node) on which Node Manager is running.</li> <li><code>all</code> — Migrate all services (for example, JTA and JMS) from the source server to the destination server.</li> <li><code>none</code> — Disable service migration from the source server. If you specify this type, <code>failback</code> and <code>destination</code> are not needed.</li> </ul>
<code>failback</code>	<p>If set to <code>true</code>, a failback operation is performed. Failback restores a service to its original hosting server, the server on which it was running before the rollout.</p> <p>The default value is <code>false</code> (no failback).</p> <p><b>Note:</b> A JTA service automatically fails back when it is invoked for migration. Therefore, do not use the <code>failback</code> option for JTA services, as it does not apply to them. The rollout fails if you specify the <code>failback</code> option.</p>

The following sample JSON file shows how to define various migration scenarios.

```

{
  "migrations": [
    # Migrate all JMS migratable targets on server1 to server2. Perform a failback
    {
      "source": "server1",
      "destination": "server2",
      "migrationType": "jms",
      "failback": "true"
    },
    # Migrate only JTA services from server1 to server3. Note that JTA migration
    # does not support the failback option, as it is not needed.
    {
      "source": "server1",
      "destination": "server3",
      "migrationType": "jta"
    },
    # Disable all migrations from server2
    {
      "source": "server2",
      "migrationType": "none"
    }
  ]
}

```

```
{  
  
# Migrate all services (for example, JTA and JMS) from server 3 to server1 with  
# no failback  
  "source": "server3",  
  "destination": "server1",  
  "migrationType": "all"  
},  
  
# Use Whole Server Migration to migrate server4 to the node named machine 5 with  
# no failback  
  {  
    "source": "server4",  
    "destination": "machine5",  
    "migrationType": "server"  
  }  
  
}]
```

## 2.3 Preparing to Roll Out a Patched Oracle Home

There are two ways to prepare for rolling out a patched Oracle home to your Managed Servers:

- You can use the OPatchAuto tool to automatically clone your Oracle home, patch it, and create a patched Oracle home archive. You can then use OPatchAuto to distribute the patched Oracle home archive to the nodes in your domain. Oracle recommends using this approach as it is more automated. See these sections for details:
  - [Creating a Patched Oracle Home Archive Using OPatchAuto](#)
  - [Distributing the Patched Archive to Each Node Using OPatchAuto](#)
- You can manually create the second Oracle home, use the OPatch utility to apply patches to it, use the `copyBinary` command to create an archive of the patched Oracle home, and then copy the archive to the nodes in your domain. See these sections for details:
  - [Creating a Second Oracle Home](#)
  - [Applying Patches to the Second Oracle Home](#)
  - [Creating an Archive and Distributing It to Each Node](#)

In both cases, the preparation process does not require you to shut down any of your Managed Servers, so there is no effect on the availability of your applications.

---

---

**Note:**

If your domain includes Oracle Fusion Middleware products other than Oracle WebLogic Server (such as Oracle SOA Suite or Oracle WebCenter), and you have patched those applications in your Oracle home, if you want to preserve currently active sessions while doing the rollout, ensure that the patched versions are compatible with ZDT patching. For example, the applied patches should have limited changes to session shape and should be backward-compatible with other Oracle Fusion Middleware products that are running in the domain.

---

---

### 2.3.1 Creating a Patched Oracle Home Archive Using OPatchAuto

This section describes how to create a clone of your existing Oracle home, patch it, and create an archive of the patched Oracle home using the OPatchAuto tool. Before you can apply any patches, you must first download them to your *patch\_home* directory using OPatch.

To create a patched Oracle home archive, enter the following commands. You must run the `opatchauto apply` command from the *ORACLE\_HOME* from which you want to create the image. This command creates a clone of your unpatched Oracle home, applies the patches in the specified *patch\_home* directory, and then creates the patched archive.

```
cd ORACLE_HOME/OPatch/auto/core/bin
opatchauto.sh apply patch_home -create-image -image-location path -oop
```

The following table describes the parameters in the `opatchauto apply` command:

Parameter	Description
<i>patch_home</i>	The OPatch <i>\$PATCH_HOME</i> directory where the patches you want to apply are stored
<code>-create-image</code>	Indicates that you want to create an image of the Oracle home directory. The image will include the patches in <i>patch_home</i> .
<code>-image-location path</code>	Specify the full path and file name of the image JAR file to create. For example: <code>-image-location /u01/images/OH-patch1.jar</code>
<code>-oop</code>	Indicates that this is an out-of-place patching archive

### 2.3.2 Distributing the Patched Archive to Each Node Using OPatchAuto

After you create a patched archive, use OPatchAuto to distribute the archive to each node that will be included in the Oracle home patching workflow.

To distribute the archive, use the following commands:

```
cd ORACLE_HOME/OPatch/auto/core/bin
opatchauto.sh apply -plan wls-zdt-push-image -image-location path
-wls-zdt-host adminserver:port -wls-zdt-target target
-wls-zdt-remote-image path -wallet path -walletPassword password
```

The following table describes the parameters in the `opatchauto apply` command:

Parameter	Description
<code>-plan</code>	Indicates the type of operation to be performed by <code>opatchauto apply</code> . For distributing a patched Oracle home for ZDT, always specify <code>wls-zdt-push-image</code> as the value for this parameter.
<code>-image-location path</code>	Specify the full path and file name of the image JAR file to distribute. For example: <code>-image-location /u01/images/OH-patch1.jar</code>

Parameter	Description
<code>-wls-zdt-host</code> <code>adminserver:port</code>	Specify the Administration Server hostname and port number for the domain to which you are distributing the archive. The archive will be distributed to this node.
<code>-wls-zdt-target</code> <code>target</code>	Specify a cluster or a comma-separated list of clusters that will be included in the rollout. The archive will be distributed to all nodes on which these clusters are configured.
<code>-wls-zdt-remote-image</code> <i>path</i>	The full path to the archive file you want to create on each node to be included in the ZDT rollout. This does not have to be the same file name as the original archive. For example: <code>-wls-zdt-remote-image /u01/images/rollout-OH-image.jar</code>
<code>-wallet</code> <i>path</i>	The full path to a wallet directory that was created using <code>configWallet.sh</code> or <code>configWallet.cmd</code> . For example: <code>-wallet \$HOME/wallet</code>
<code>-walletPassword</code> <i>password</i>	The password for the specified wallet, if needed. For example: <code>-walletPassword mypassword</code>

After distributing the patched archive, you are ready to create a workflow that includes patching your Oracle home. See [Configuring and Monitoring Workflows](#).

---



---

**Note:**

If you want to also update your Java version or applications using the same patching workflow, then perform the preparation steps for those upgrades before you create the workflow.

---



---

### 2.3.3 Creating a Second Oracle Home

To manually create a patched Oracle home, you must first create a copy of your existing Oracle home by using the `copyBinary` and `pasteBinary` commands. When using these commands, you must keep in mind that the value of options specified must not contain a space. For example, on Windows, you cannot pass the following as a value to the `-javaHome` option:

```
C:\Program Files\jdk
```

---



---

**Note:**

Oracle recommends that you create and patch the second Oracle home on a nonproduction machine so that you can test the patches you apply, but this is not required. However, you must perform the following steps on the node where you will patch the new Oracle home. The Oracle home on that node must be identical to the Oracle home you are using for your production domain.

---



---

To create the second Oracle home to which you will apply patches:



1. Change to the following directory, where *ORACLE\_HOME* is the Oracle home that you want to patch.

```
cd ORACLE_HOME/oracle_common/bin
```

2. Execute the following command, where *archive* is the full path and file name of the archive file to create, and *oracle\_home* is the full path to your existing Oracle home. Note that *JAVA\_HOME* must be defined as the Java home that was used for your Oracle home installation:

#### UNIX

```
./copyBinary.sh -javaHome $JAVA_HOME -archiveLoc archive -sourceOracleHomeLoc oracle_home
```

#### Windows

```
copyBinary.cmd -javaHome %JAVA_HOME% -archiveLoc archive -sourceOracleHomeLoc oracle_home
```

For example, the following command creates the Oracle home archive *wls1221.jar* in network location */net/oraclehomes/* using the Oracle home located at */u01/oraclehomes/wls1221*:

```
./copyBinary.sh -javaHome $JAVA_HOME -archiveLoc /net/oraclehomes/wls1221.jar -sourceOracleHomeLoc /u01/oraclehomes/wls1221
```

3. Execute the following command to create the second Oracle home, where *archive* is the full path and file name of the archive file you created, and *patch\_home* is the full path to the new Oracle home to which you will apply patches. Note that *JAVA\_HOME* must be defined as the Java home that was used for your original Oracle home installation:

#### UNIX

```
./pasteBinary.sh -javaHome $JAVA_HOME -archiveLoc archive -targetOracleHomeLoc patch_home
```

#### Windows

```
pasteBinary.cmd -javaHome %JAVA_HOME% -archiveLoc archive -targetOracleHomeLoc patch_home
```

For example, the following command creates the Oracle home *wls1221\_patched* in */u01/oraclehomes/* using the archive */net/oraclehomes/wls1221.jar*:

```
./pasteBinary.sh -javaHome $JAVA_HOME -archiveLoc /net/oraclehomes/wls1221.jar -targetOracleHomeLoc /u01/oraclehomes/wls1221_patched
```

## 2.3.4 Applying Patches to the Second Oracle Home

To patch the second Oracle home, use the OPatch tool to apply individual patches, bundle patches, security patch updates, or patch set updates to the second, offline Oracle home. Prior to applying a particular patch or group of patches, ensure that all prerequisite patches have already been applied.

For detailed information about how to prepare for and patch an Oracle home using OPatch, see *Patching with OPatch*.

## 2.3.5 Creating an Archive and Distributing It to Each Node

After you have created the patched Oracle home, use the following steps to create an Oracle home archive and copy it to each node that will be involved in the rollout:

1. Change to the following directory, where *ORACLE\_HOME* is the patched Oracle home that you created.

```
cd ORACLE_HOME/oracle_common/bin
```

2. Execute the following command, where *archive* is the full path and file name of the archive file to create, and *patched\_home* is the full path to the patched Oracle home you created. Note that *JAVA\_HOME* must be defined as the Java home that was used for your current Oracle home installation.

### UNIX

```
./copyBinary.sh -javaHome $JAVA_HOME -archiveLoc archive -sourceOracleHomeLoc patched_home
```

### Windows

```
copyBinary.cmd -javaHome %JAVA_HOME% -archiveLoc archive -sourceOracleHomeLoc patched_home
```

For example, the following command creates the Oracle home archive `wls1221.11.jar` in network location `/net/oraclehomes/` using a patched Oracle home located at `/01/oraclehomes/wls1221_patched`:

```
./copyBinary.sh -javaHome $JAVA_HOME -archiveLoc /net/oraclehomes/wls_1221.11.jar -sourceOracleHomeLoc /u01/oraclehomes/wls1221_patched
```

3. On each node that will be included in the patching workflow, copy the archive file to the parent folder of the Oracle home that you want to replace. For example, if the archive is in network location `/net/oraclehomes/wls_1221.11.jar` and the Oracle home to be replaced is located in `/u01/oraclehomes/wls1221`:

```
cp /net/oraclehomes/wls1221.11.jar /u01/oraclehomes/
```

If you are copying to a large number of nodes, you can use third-party software distribution applications to perform this step.

After completing these steps, you are ready to create a workflow that includes patching your Oracle home. See [Configuring and Monitoring Workflows](#).

---

---

**Note:**

If you want to also update your Java version or applications using the same patching workflow, then perform the preparation steps for those upgrades before you create the workflow.

---

---

## 2.4 Preparing to Upgrade to a New Java Version

Preparation for upgrading to a new version of Java does not require you to shut down Managed Servers, so there will be no interruption to application availability.

To upgrade to a new version of Java:

1. Prior to installing the new Java version, ensure that Node Manager and the Managed Servers are running on all nodes on which you plan to install the new version. This prevents the Java installer from changing the existing Java home path. However, you do not need to have the Node Manager running on the node on which the Administration Server is running.
2. On each node to be included in the upgrade, install the new Java version to the same path on each node. The full path to the new Java version must be the same on each node for the upgrade to be successful.

After copying the new Java version to each node, you are ready to create a workflow that includes upgrading to a new Java home. See [Configuring and Monitoring Workflows](#).

## 2.5 Preparing to Update to New Application Versions

This section describes how to prepare for updating to new applications using a ZDT workflow. It contains the following sections:

- [The Effects of Staging Modes](#)
- [Creating an Application Update JSON File](#)

### 2.5.1 The Effects of Staging Modes

Applications deployed across Managed Servers, partitions, or resource groups can be deployed using one of three staging modes: stage mode, no-stage mode, and external-stage mode. The selected mode indicates how the application will be distributed and kept up-to-date.

How you prepare for an application update workflow depends on the mode you used when you staged the application.

Staging Mode	Required Preparation and Result
Stage	<p>Place a copy of the updated application directory on the domain's Administration Server.</p> <p><b>Result:</b> The workflow will replace the original application directory on the Administration Server and WebLogic Server will copy it to each Managed Server.</p>
No-stage	<p>Place a copy of the updated application directory on each node that will be affected. This directory must be in the same location on each node.</p> <p><b>Result:</b> The workflow will update each node in turn by replacing the existing application directory with the updated application directory, and will move the original application directory to the specified backup location.</p>

Staging Mode	Required Preparation and Result
External stage	<p>Place a copy of the updated application directory on each node that will be affected. This directory must be in the same location on each node.</p> <p><b>Result:</b> The workflow will detect that the application is an external-stage application, figure out the correct path for the stage directory for each Managed Server on the node, copy the updated application to that location, and move the original application to the specified backup location.</p>

For detailed information about the various staging modes, see "Staging Mode Descriptions and Best Practices" in *Deploying Applications to Oracle WebLogic Server*.

## 2.5.2 Creating an Application Update JSON File

You can update one or more applications in your domain, partition, or resource groups with a single workflow. Application updates are accomplished by creating a JSON file that, for each application, defines:

- The application name (`applicationName`)
- The path and file name for the updated application archive (`patchedLocation`)
- The path and file to which you want to back up the original application archive (`backupLocation`).
- The partition name. This is applicable only if you are updating an application deployed to a partition.
- The resource group template name. This is applicable only if you are updating an application deployed to a resource group.

---

**Note:** Oracle recommends that you avoid using backslash (Windows) while specifying the paths in the JSON file. This is because these paths are interpreted by Java and a backslash may trigger a different character representation.

---

When configuring the workflow either using WLST or the WebLogic Server Administration Console, you specify the file name of the JSON file to use for the update.

The following example shows the structure of a JSON file that is intended to update two applications, *MyApp* and *AnotherApp*, to a new version. You can use a single JSON file to update as many applications as necessary.

```
{ "applications": [
  {
    "applicationName": "MyApp",
    "patchedLocation": "/u01/applications/MyAppv2.war",
    "backupLocation": "/u01/applications/MyAppv1.war"
  },
  {
```

```
"applicationName": "AnotherApp",  
"patchedLocation": "/u01/applications/AnotherAppv2.war",  
"backupLocation": "/u01/applications/AnotherAppv1.war"  
}  
1}
```

After copying the updated application to all required locations and creating the JSON file, you are ready to create a workflow that includes application updates. See [Configuring and Monitoring Workflows](#).



---

## Configuring and Monitoring Workflows

Configure and monitor a patching workflow that moves Managed Servers to a patched Oracle home, updates the Java version on your Managed Servers, updates the applications on your Managed Servers, or any combination of these update tasks. You can use the WebLogic Scripting Tool (WLST) to create and monitor the workflow or you can create and monitor the workflow by using the WebLogic Server Administration Console.

---

**Note:**

Prior to initiating the update process, you must have completed all appropriate preparation steps for the type of update you are doing, as described in [Preparing for Zero Downtime Patching](#).

For Windows-based domains, prior to initiating a workflow to update an Oracle home, on each node, ensure that there are no locked directories or files in the Oracle home being updated, as this can prevent the Oracle home from being moved to the specified backup directory. A directory can be locked by something as simple as having a DOS command window open to that directory. A file can be locked by having it open in an application.

---

This chapter includes the following sections:

- [Strategies for Rolling Out a Patched Oracle Home](#)
- [Starting the Administration Server](#)
- [Using OPatchAuto to Initiate, Revert, and Resume Rollouts](#)
- [Using WLST to Initiate and Monitor Workflows](#)
- [Using the WebLogic Server Administration Console to Create and Monitor Workflows](#)

### 3.1 Strategies for Rolling Out a Patched Oracle Home

When you roll out a new Oracle home using either WLST or the Administration Console, you must ensure that the patched Oracle home is first rolled out to the Administration Server. There are two approaches you can take to do this:

- Use one workflow to roll out the patched Oracle home to the Administration Server, and then use a second workflow to roll out the patched Oracle home to your clusters. Oracle recommends using this approach, but it is not required.

In this scenario:

- If using WLST, you would execute either the `rolloutOracleHome` or `rolloutUpdate` command, and specify the name of the Administration Server as the target. You would then execute `rolloutOracleHome` or `rolloutUpdate` again, and specify cluster targets.
- If using the WebLogic Server Administration Console, you would create one workflow from the **Servers** tab and select your Administration Server as the target. After that workflow completes, you would create a second workflow from the **Clusters** tab and select the clusters to include.
- Use only one workflow to roll out the patched Oracle home to the entire domain. The workflow will automatically roll out the patched Oracle home first before rolling it out to the target clusters.

In this scenario:

- If using WLST, you would execute either the `rolloutOracleHome` or `rolloutUpdate` command, and specify the domain name as the target.
- If using the Administration Console, you would create one workflow from the **Domain** tab.

## 3.2 Starting the Administration Server

If the Administration Server will be included in a workflow, then you can start the Administration server using either the `startWebLogic` script or the Node Manager. The Administration Server will be automatically restarted during the rollout operation if the specified target for the rollout is a domain. However, when the rollout operation restarts the Administration Server, you might experience a brief downtime when you will not be able to connect to either WLST or Administration Console. As a workaround, you must wait and then reconnect when the Administration Server has reached the RUNNING state in order to receive updates on the progress of the rollout operation.

To start the Administration Server before you initiate the rollout operation, you can start the Administration server in one of the following ways:

- Using the `startWebLogic` script

If there is no Node Manager configured for the Administration Server, then you can start the Administration Server by using the `startWebLogic` script. To start the Administration Server using this script, see “Starting an Administration Server with a Startup Script” in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

- Using the Node Manager

If a Node Manager is configured for the Administration Server, then you must start the Administration Server using the Node Manager. To start the Administration Server using the Node Manager, perform the following steps:

1. If the Administration Server is currently running and was started using the `startWebLogic` script in the domain home, use the `stopWebLogic` command to shut it down:

### UNIX

```
cd domain_home/bin
./stopWebLogic.sh
```



**Windows**

```
cd domain_home\bin
stopWebLogic.cmd
```

2. Ensure that Node Manager is running on the host.
3. Start WLST. See "Invoking WLST" in *Understanding the WebLogic Scripting Tool*.
4. Use the nmConnect command to establish a Node Manager session. For example, use the following command to connect to the domain *mydomain* located in `/domains/mydomain` using SSL, where the NodeManager port is *5556*:

```
wls:/myserver/serverConfig> nmConnect('username', 'password', 'localhost',
'5556', 'mydomain', '/domains/mydomain','ssl')
```

5. After successfully connecting, run the nmStart command. For example, use the following command if the Administration Server is called *AdminServer* and the domain is located in `/domains/mydomain`:

```
nmStart('AdminServer', '/domains/mydomain')
```

For more information, see "Starting the Administration Server Using Node Manager" in *Administering Node Manager for Oracle WebLogic Server*.

### 3.3 Using OPatchAuto to Initiate, Revert, and Resume Rollouts

This section describes how to initiate and monitor rollout workflows for applying a patched Oracle home to the nodes in your domain. You can use this approach only if the workflow is applying a patched Oracle home. If you want to include a Java version update or application update in the workflow, then you must use WLST or the Administration Console.

This section contains the following topics:

- [Using OPatchAuto to Initiate a Rollout](#)
- [Using OPatchAuto to Revert a Rollout](#)
- [Using OPatchAuto to Resume a Failed Rollout](#)

---



---

**Note:**

When using OPatchAuto to initiate a workflow, you must use the Administration Console to monitor the progress of the workflow. See [Monitoring and Managing Workflows](#).

---



---

#### 3.3.1 Using OPatchAuto to Initiate a Rollout

Use the following commands to initiate a workflow using OPatchAuto:

```
cd ORACLE_HOME/OPatch/auto/core/bin
opatchauto.sh apply -plan wls-zdt -image-location path
-wls-zdt-host adminserver:port -wls-zdt-target target
-wls-zdt-remote-image path -wallet path -walletPassword password
```

The following table describes the parameters in the `opatchauto apply` command:

Parameter	Description
-plan	Indicates the type of operation to be performed by <code>opatchauto</code> apply. For rolling out a patched Oracle home for ZDT, always specify <code>wls-zdt</code> as the value for this parameter.
-image-location <i>path</i>	Specify the full path and file name of the image JAR file that contains the patched Oracle home to use for the rollout. For example: <code>-image-location /u01/images/rollout-OH-image.jar</code>
-wls-zdt-host <i>adminserver:port</i>	Specify the Administration Server host name and port number for the domain you are rolling out the patched Oracle home to.
-wls-zdt-target <i>target</i>	For <i>target</i> , specify the target for the rollout. This can be: <ul style="list-style-type: none"> <li>• A domain name</li> <li>• A cluster name</li> <li>• A comma-separated list of clusters</li> </ul> When rolling out the archive, you must specify the same target as you specified when you distributed the archive that you are using for the rollout.
-wls-zdt-backup <i>path</i>	The full path to the directory to use for backing up your existing Oracle home. For example: <code>-wls-zdt-backup /u01/images/rollout-OH-image.jar</code>
-wls-zdt-remote-image <i>path</i>	The full path to the archive file that you want to create on each node to be included in the ZDT rollout. This does not have to be the same file name as the original archive. For example: <code>-wls-zdt-remote-image /u01/images/rollout-OH-backup</code>
-wallet <i>path</i>	The full path to a wallet directory that was created using <code>configWallet.sh</code> or <code>configWallet.cmd</code> . For example: <code>-wallet \$HOME/wallet</code>
-walletPassword <i>password</i>	The password for the specified wallet, if needed. For example: <code>-walletPassword mypassword</code>

### 3.3.2 Using OPatchAuto to Resume a Failed Rollout

If an Oracle home rollout failed and you want to resume it, use the following commands:

```
cd ORACLE_HOME/OPatch/auto/core/bin
opatchauto.sh resume -session workflow_id -walletPassword password
```

The following table describes the parameters in the `opatchauto resumecommand`:

Parameter	Description
-session - <i>workflow_id</i>	Specify the workflow ID of the workflow to roll back.

Parameter	Description
<code>-wallet <i>path</i></code>	The full path to a wallet directory that was created using <code>configWallet.sh</code> or <code>configWallet.cmd</code> . For example: <code>-wallet \$HOME/wallet</code>
<code>-walletPassword <i>password</i></code>	The password for the specified wallet, if needed. For example: <code>-walletPassword mypassword</code>

### 3.3.3 Using OPatchAuto to Revert a Rollout

You can use OPatchAuto to revert a rollout that has failed or stopped, as well as to revert a completed workflow. If the rollout has failed or stopped, OPatchAuto will revert it starting with the last successfully completed step. If the rollout has completed, then OPatchAuto will initiate a new rollout, using the backed up Oracle home as the Oracle home to roll out.

Use the following commands to revert a rollout:

```
cd ORACLE_HOME/OPatch/auto/core/bin
opatchauto.sh rollback -session workflow_id -walletPassword password
```

The following table describes the parameters in the `opatchauto rollback` command:

Parameter	Description
<code>-session - <i>workflow_id</i></code>	Specify the workflow ID of the workflow to roll back.
<code>-wallet <i>path</i></code>	The full path to a wallet directory that was created using <code>configWallet.sh</code> or <code>configWallet.cmd</code> . For example: <code>-wallet \$HOME/wallet</code>
<code>-walletPassword <i>password</i></code>	The password for the specified wallet, if needed. For example: <code>-walletPassword mypassword</code>

## 3.4 Using WLST to Initiate and Monitor Workflows

This section describes the WLST commands that you can use to initiate workflows to update your Managed Servers, partitions, or resource groups, and provides sample WLST scripts demonstrating various workflow (rollout) scenarios.

---



---

#### Note:

When using the WLST `rolloutOracleHome` or `rolloutUpdate` commands to initiate a rollout of a new Oracle home for a Windows-based domain, you cannot run WLST from any Oracle home that will be updated as part of the workflow. For more information, see [ZDT Patching Restrictions](#).

---



---

Use the following WLST commands to perform automated rolling updates of your servers. You must execute these commands from the Administration Server for the target domain.

- `rolloutOracleHome` — Rolls out a patched Oracle home to your Managed Servers or reverts your Managed Servers to a previous Oracle home. The patched Oracle home archive that you use in this command can be one that was created either using `opatchauto` or the `copyBinary` and `pasteBinary` commands.
- `rolloutJavaHome` — Updates your Managed Servers to use a new Java version.
- `rolloutUpdate` — Updates your Managed Servers to use a patched Oracle home and a new Java version. The patched Oracle home archive that you use in this command can be one that was created either using `opatchauto` or the `copyBinary` and `pasteBinary` commands.
- `rolloutApplications`—Updates specified applications that are running on your Managed Servers, partitions, or resource groups.

---

---

**Note:**

When specifying paths for Windows in rollout commands, you must use backslashes instead of forward slashes. To avoid unnecessary errors, ensure that the backslashes are escaped. (For example, `C:\\myhome\\files\\apps.json`). For more information, see "Syntax for WLST Commands" in *Understanding the WebLogic Scripting Tool*.

---

---

When you execute one of these WLST commands, the command determines which servers need to be updated and in which order, and creates a patching workflow that will update them safely. This workflow includes:

- Performing a graceful shutdown of Managed Servers, partitions, or resource groups one at a time. This does not include Managed Servers that are currently in ADMIN or STANDBY mode. This includes migration of singleton services if the `migrationProperties` option is included in the rollout command. The ADMIN and STANDBY modes are not supported for rolling out application updates to partitions or resource groups.
- Replacing the Oracle home directory (if applicable)
- Replacing the Java home directory (if applicable)
- Replacing application directories (if applicable)
- Restarting Node Manager on the node
- Restarting the Managed Servers on the node

[Table 3-1](#) describes the parameters available for the `WLSTrollout` commands.

**Table 3-1 Arguments for WLST rollout Commands**

---

**Table 3-1 (Cont.) Arguments for WLST rollout Commands**

Argument	Description
target	<p>Required for all rollout commands.</p> <p>Specifies which Managed Servers, partitions, or resource groups will be included in the update. <code>target</code> can be one of:</p> <p><i>partition_name</i> — Specify a partition name or a comma-separated list of partition names in the <code>rolloutApplications</code> command if you want to roll out application updates to specific partitions. Note that the partitions that you specify as targets must be the same partitions that you specify in the application update JSON file.</p> <p><i>domain_name</i> — Specify a domain name as the target if you want the Administration Server and all Managed Servers in that domain to be updated. You must also specify the domain name as the target in the <code>rolloutApplications</code> command if you want to roll out application updates to resource groups. Additionally, you must also specify the resource group template name in the application update JSON file if you want to roll out application updates to resource groups.</p> <p><i>clusters</i> — Specify a cluster name or a comma-separated list of cluster names if you want to update all Managed Servers in the specified cluster or clusters, but not Managed Servers in other clusters.</p> <p><i>servers</i> — Specify a server name or a comma-separated list of server names if you only want to update those Managed Servers. Note that the servers you specify must still be part of a cluster; they cannot be unclustered servers.</p> <p><b>Note:</b> Typically, you should specify a server target only when updating the Administration Server. Oracle recommends that you not update individual Managed Servers in most cases as sessions may not be preserved and downtime for users may not be avoided. However, one situation in which you can safely specify Managed Server targets is if you have added one or more new Managed Servers and they are not at the same Java version as your other Managed Servers.</p>
rolloutOracleHome	<p>Applies only to and is required for the <code>rolloutOracleHome</code> command. Specifies the location of the Oracle home archive (JAR file) or local Oracle home directory to roll out, thereby replacing the existing Oracle home.</p>
backupOracleHome	<p>Applies only to and is required for the <code>rolloutOracleHome</code> command. Specifies the full path of the directory to which the existing Oracle home will be moved. This effectively renames the original Oracle home. For example, if your original Oracle home is <code>/u01/Oracle_Home</code> and you specify <code>/u01/Oracle_Home_backup</code> for this parameter, <code>/u01/Oracle_Home</code> will be moved (renamed) to <code>/u01/Oracle_Home_backup</code>.</p>
isRollback	<p>Optional. Applies only to the <code>rolloutOracleHome</code> and <code>rolloutUpdate</code> commands.</p>
javaHome	<p>Applies to and is required for the <code>rolloutJavaHome</code> command. Optionally, this argument may be required by the <code>rolloutUpdate</code> command.</p> <p>Specifies the location of the new Java home to use.</p>

**Table 3-1 (Cont.) Arguments for WLST rollout Commands**

<b>Argument</b>	<b>Description</b>
<code>applicationProperties</code>	Applies to and is required for the <code>rolloutApplications</code> command. Optionally, this argument may be required by the <code>rolloutUpdate</code> command. Specifies the full path to the JSON file that defines one or more application names, application archive locations, and application backup locations.

**Table 3-1 (Cont.) Arguments for WLST rollout Commands**

Argument	Description
<i>options</i>	<p>The following options can be included in <code>rollout</code> commands.</p> <ul style="list-style-type: none"> <li>• <code>isDryRun</code> — If <code>TRUE</code>, the workflow operation will be evaluated but not executed. The default is <code>FALSE</code>.</li> <li>• <code>autoRevertOnFailure</code> — If <code>TRUE</code>, the workflow operation should automatically revert on failure. If <code>FALSE</code>, the workflow operation will stop on a failure and you can resume or revert it. The default is <code>TRUE</code>.</li> <li>• <code>isSessionCompatible</code> — This option is applicable to all <code>rollout</code> commands, as it affects rollout time regardless of whether the rollout impacts session handling.</li> </ul> <p>The default is <code>FALSE</code>, which means that the very last server to be updated on each cluster will wait for all existing sessions to complete. This ensures that a compatible server is available in the cluster to handle sessions that must be served by a Managed Server that is still running on the existing version.</p> <p>If set to <code>TRUE</code>, this indicates that the session state in servers is 100% compatible between the existing version and the new version. Therefore, the last Managed Server in the update sequence in a cluster will shut down without waiting for all existing sessions to complete.</p> <p>Oracle recommends that you set this to <code>FALSE</code> unless you are absolutely sure that the session state is identical. This may cause the rollout to take longer due to the wait for session completion.</p> <p><b>Note:</b> Serialization and deserialization in WebLogic Server differs slightly from Java serialization and deserialization. Therefore, additional fields on classes may result in a session being incompatible with servers on the new version, requiring that they be served by a server on the existing version. For example, a <code>User</code> class that adds a field such as <code>Information</code> will cause that session to be incompatible between versions.</p> <ul style="list-style-type: none"> <li>• <code>migrationProperties</code> — The full path to a JSON file that defines singleton service migrations to be performed during the rollout. For more information about this file and service migration, see <a href="#">Preparing to Migrate Singleton Services</a>.</li> <li>• <code>shutdownTimeout</code> — Time (in seconds) WLST waits for a server to shut down gracefully before shutting it down forcefully. The forceful shutdown of servers may cause undesirable consequences, such as loss of session data and loss of in-flight transactions. A value of less than 1 second is ignored.</li> </ul> <p>If <code>isSessionCompatible</code> is set to <code>TRUE</code>, then the <code>shutdownTimeout</code> option defaults to zero, which means that WLST waits forever for the server to shut down gracefully.</p> <p>If <code>isSessionCompatible</code> is set to <code>FALSE</code>, then the user must specify a value for the <code>shutdownTimeout</code> option. Oracle recommends that you specify a value that gives typical applications plenty of time to complete. Because different applications have different behaviors, this value must be decided by the user.</p> <ul style="list-style-type: none"> <li>• <code>DelayBetweenNodes</code> — Use this option to specify the number of seconds to wait between the shutdown of servers on one node and the shutdown of servers on the next node in the workflow. This delay allows for: <ul style="list-style-type: none"> <li>– The servers on the first node to be restarted and join the cluster</li> <li>– The load balancer to evenly distribute traffic</li> <li>– Any slow (lazy) stateful session bean clients to continue making requests before shutdown of the servers on the next node begins</li> </ul> <p>If not specified, this value defaults to 60 seconds. If you are not concerned about the lazy stateful session bean clients, you can include this option and set it to a lower value.</p> </li> <li>• <code>coherenceServiceHATarget</code> — Use this option to specify the High Availability (HA) Status of Coherence services on a managed Coherence server which must be met before the server is shutdown</li> </ul>

You can also use WLST to monitor the progress of a workflow. For more information, see [Monitoring Workflow Progress](#).

### 3.4.1 Rolling Out a New Oracle Home

Use the `rolloutOracleHome` command if you only want to do one of the following tasks:

- Update your Administration Server to use a patched Oracle home.
- Update your entire domain (Administration Server and clustered Managed Servers) to use a patched Oracle home.
- Update clustered Managed Servers to use a patched Oracle home.
- Revert your Administration Server, clustered Managed Servers, or domain to use the previous unpatched Oracle home.

`rolloutOracleHome` has the following syntax:

```
rolloutOracleHome(target, rolloutOracleHome, backupOracleHome, [isRollback],  
[options])
```

This command supports the `isDryRun`, `autoRevertOnFailure`, and `isSessionCompatible` options. For information about these options, see [Using WLST to Initiate and Monitor Workflows](#).

The following example shows how to roll out a new Oracle home to the domain *mydomain*. The JAR file for the patched Oracle home is located at `/net/wls/wls_patched.jar`. The original Oracle home will be moved (renamed) to `/u01/Oracle_Home_backup`. The process will not automatically revert if it fails.

```
connect('adminname', 'adminpassword', 't3://hostname:port')  
domain='/domains/mydomain'  
progress=rolloutOracleHome(domain, '/net/wls/wls_patched.jar',  
'/u01/Oracle_Home_backup', autoRevertOnFailure=FALSE)
```

---

---

**Note:** Specifying a local Oracle home directory in the `rolloutOracleHome` command is not supported when you are rolling out a new Oracle home. For more information, see [ZDT Patching Restrictions](#).

---

---

### 3.4.2 Updating Your Java Version

Use the `rolloutJavaHome` command if you only want to do one of the following tasks:

- Update your Administration Server to use a new Java version.
- Update your entire domain (Administration Server and Managed Servers) to use a new Java version.
- Update your Managed Servers to use a new Java version.
- Revert your Administration Server, Managed Servers, or domain to use the previous Java version.

`rolloutJavaHome` has the following syntax:

```
rolloutJavaHome(target, javaHome, [options])
```



This command supports the `isDryRun` and `autoRevertOnFailure` options. For information about these options, see [Using WLST to Initiate and Monitor Workflows](#).

The following example shows how to roll out a new Java home to clusters `Cluster1`, `Cluster2`, `Cluster3`. The new Java home location is `/u01/jdk1.8.0_50`. The `autoRevertOnFailure` option is not included in this example; therefore, the workflow will automatically revert if the process fails.

```
connect('adminname', 'adminpassword', 't3://hostname:port')
clusters='Cluster1,Cluster2,Cluster3'
progress=rolloutJavaHome(clusters, '/u01/jdk1.8.0_50')
```

### 3.4.3 Updating Both Oracle Home and the Java Version

Use the `rolloutUpdate` command if you only want to do one of the following tasks:

- Update your Administration Server to use both a patched Oracle home and a new Java version.
- Update your entire domain (Administration Server and clustered Managed Servers) to use both a patched Oracle home and a new Java version.
- Update your Managed Servers to use both a patched Oracle home and a new Java version.
- Revert your Administration Server, Managed Servers, or domain to the previous Oracle home and previous Java version.

`rolloutUpdate` has the following syntax:

```
rolloutUpdate(target, rolloutOracleHome, backupOracleHome, [isRollback], [javaHome],
[options])
```

This command supports the `isDryRun`, `autoRevertOnFailure`, and `isSessionCompatible` options. For information about these options, see [Using WLST to Initiate and Monitor Workflows](#).

The following example shows how to roll out a new Oracle home and a new Java home to the Administration Server. The JAR file for the patched Oracle home is located at `/net/wls/wls_patched.jar`. The original Oracle home will be moved (renamed) to `/u01/Oracle_Home_backup`. The new Java home location is `/u01/jdk1.8.0_50`. The `autoRevertOnFailure` option is not included in this example; therefore, the workflow will automatically revert if the process fails.

```
connect('adminname', 'adminpassword', 't3://hostname:port')
server='AdminServer'
progress=rolloutUpdate(server, '/net/wls/wls_patched.jar',
'/u01/Oracle_Home_backup', '/u01/jdk1.8.0_50')
```

### 3.4.4 Rolling Out Updated Applications

Use the `rolloutApplications` command if you want to do one of the following tasks:

- Update your Managed Servers to use a new version of one or more applications.
- Revert your Managed Servers to the previous version of one or more applications.
- Update your partition to use a new version of one or more applications.
- Update your resource group to use a new version of one or more applications.

`rolloutApplications` has the following syntax:

```
rolloutApplications(target, applicationProperties, [options])
```

This command supports the `isDryRun`, `autoRevertOnFailure`, and `isSessionCompatible` options. For information about these options, see [Using WLST to Initiate and Monitor Workflows](#).

The following example shows how to roll out the applications defined in the JSON-formatted application properties file `/u01/scratch/app_update.json` to all clusters `Cluster1`, `Cluster2`, `Cluster3` on a UNIX system.

```
connect('adminname', 'adminpassword', 't3://hostname:port')
clusters='Cluster1,Cluster2,Cluster3'
progress=rolloutApplications(clusters, '/u01/scratch/app_update.json')
```

### 3.4.5 Reverting to the Previous Oracle Home, Java Home, or Applications

After a successful rollout, if you want to roll back to the previous Oracle home, Java home, or application version, you must perform the following two steps to complete the rollback operation:

- Use the `rolloutUpdate` command to roll back to the previous Oracle home and Java home. However, you must keep the following restrictions in mind before you execute the `rolloutUpdate` command to roll back:
  - You must specify the backed up Oracle home as the Oracle home directory to roll out. This directory should be the backup directory from the previous rollout.
  - Once you specify the backup Oracle home directory as the Oracle home directory to roll back to, you must not specify the new Java home in the command. The Java home will be automatically rolled back to the original Java home that was used in the previous Oracle home that you have specified to roll back to.
- Use the `rolloutApplications` command to rollback to the previous application version by specifying the old application archive in the json file. For more information about using this command, see [Rolling Out Updated Applications](#).

The following example shows how to roll back to the previous Oracle home, Java home and applications. In this example, `myDomain` is the name of the domain to roll back to, `/pathto/unpatchedOracleHomeBackup/` is the location of the backup Oracle home directory from the previous rollout, `/pathto/unpatchedOracleHomeBackup1/` is the path of the directory to which the existing Oracle home will be moved. To enable the roll back operation, the `isRollback` parameter must be set to `true` as shown in the example:

```
rolloutUpdate('myDomain', '/pathto/unpatchedOracleHomeBackup/', '/pathto/unpatchedOracleHomeBackup1/', 'true')
```

### 3.4.6 Initiating a Rolling Restart of Servers or Partitions

Use the `rollingRestart` command if you want to do one of the following tasks:

- Initiate a rolling restart of all servers in a domain.

- Initiate a rolling restart of all servers in a specific cluster or clusters.
- Initiate a rolling restart of all partitions in a cluster on a server.

The rolling restart of a partition involves restarting the partitions in a cluster on each server in a sequential manner in such a way that it does not affect other partitions across the cluster or server. Both the WebLogic Server administrator and the partition administrator can perform the rolling restart of partitions on a server.

`rollingRestart` has the following syntax:

```
rolloutRestart(target, [options])
```

The following example shows how to perform a rolling restart of all servers in *Cluster1* and *Cluster2*.

```
connect('adminname', 'adminpassword', 't3://hostname:port')
clusters='Cluster1,Cluster2'
progress=rollingRestart(clusters)
```

### 3.4.7 Monitoring Workflow Progress

Each `rollout` command returns a `WorkflowTaskRuntimeMBean` that you can use to poll the current status of the workflow. To monitor the progress of a rollout, use a `rolloutcommand` in the following format:

```
progress=rollout_command
```

For example, use this command if you are rolling out a new Oracle home:

```
progress=rolloutOracleHome(DomainA, '/net/patched/wls1221p.jar',
'/net/backups/wls1221', autoRevertOnFailure=FALSE)
```

You can then use the methods of the `WorkflowTaskRuntimeMBean` to return information about the workflow. For more information, see `WorkflowTaskRuntimeMBean` in the *MBean Reference for Oracle WebLogic Server*. Here are some examples:

```
progress.getWorkflowId()
```

Returns the ID of the workflow.

```
progress.getProgressString()
'Workflow wf0011 Running: 13/36'
```

Returns a human-readable message containing information about the current workflow progress. In this example, workflow `wf0011` is currently running and has completed 13 of the 36 workflow commands.

```
progress.getStatus()
STARTED
```

Returns the current status of the workflow, which can be `STARTED`, `SUCCESS`, `RETRY`, `REVERTING`, `FAIL`, `REVERTED`, `REVERT_FAIL`, `CANCELED`, or `REVERT_CANCELED`.

The following Python script segment demonstrates one way to use the `progress` object to monitor a workflow and output the progress of a rollout task. Sample output is shown after the script.

```

# Print the starting information
rolloutName = progress.getName()
startTime = progress.getStartTime()
print "Started rollout task \" + rolloutName + "\" at " + str(startTime)

# Check the state every 2 minutes
domainRuntime()
cd('RolloutService/rollout-service/ActiveWorkflows')
cd(progress.getWorkflowId())
while(get('Running')==1):
    progressString = progress.getProgressString()
    print progressString
    time.sleep(120)

# Print the ending information
endTime = progress.getEndTime()
state = progress.getState()
print "rollout \" + rolloutName + "\" finished with state

```

**Output**

```

Started rollout task "Domain1Rollout" at 2014-07-22 07:29:06.528971
Running step 1 of 9
Running step 2 of 9
Running step 3 of 9
Running step 4 of 9
Running step 5 of 9
Running step 6 of 9
Running step 7 of 9
Running step 8 of 9
Running step 9 of 9
rollout "Domain1Rollout" finished with state "SUCCESS" at
2014-07-22 07:47:15.538299

```

### 3.4.8 Executing, Reverting, and Resuming Stopped Workflows

A workflow can stop in either the executing or reverting direction for the following reasons:

- The workflow failed while executing, with the `autoRevertOnFailure` option set to `FALSE`.
- The workflow was manually canceled.
- An unrecoverable error occurred during a revert operation.

When a workflow is stopped, you can resolve any errors manually. You can then set the workflow to continue to execute or revert by using the following methods on the `RolloutServiceRuntimeMBean`:

Method	Description
<code>executeWorkflow(WorkflowTaskRuntimeMBean)</code>	Takes a progress object that is eligible to be resumed and resumes it in the execute direction. If the last successful operation on the workflow was an execute, then the execute will resume with the next execute step. If the last successful operation on the workflow was a revert, then the execute will resume by executing that revert step.

Method	Description
<code>revertWorkflow(WorkflowTaskRuntimeMBean)</code>	Takes a progress object that is eligible to be resumed and resumes it in the revert direction. If the last successful operation on the workflow was an execute, then the revert will resume with that step. If the last successful operation on the workflow was a revert, then the revert will resume by reverting the next step in the revert sequence.
<code>canResume(WorkflowTaskRuntimeMBean)</code>	Returns true if the workflow stopped before it was completed and is not currently running in either direction. A workflow in this state is eligible to be resumed in either the execute or revert direction.

### 3.4.9 Useful WLST Commands for Workflows

This section describes several WLST commands that you may find useful.

- **To get a list of completed workflows:**

```
wls:/domain_name/domainRuntime/RolloutService/rollout-service> completeWfs=
cmo.getCompleteWorkflows()
```

- **To get a list of active workflows:**

```
wls:/domain_name/domainRuntime/RolloutService/rollout-service> activeWfs =
cmo.getActiveWorkflows()
```

- **To look up a workflow by ID and retrieve its status:**

```
wls:/domain_name/domainRuntime/RolloutService/rollout-service>
progress=cmo.getWorkflowTask('workflow_id')
wls:/Domain1221/domainRuntime/RolloutService/rollout-service> progress.getStatus()
```

- **To cancel a running workflow:**

```
wls:/domain_name/domainRuntime/RolloutService/rollout-service>
progress=cmo.getWorkflowTask('workflow_id')
wls:/domain_name/domainRuntime/RolloutService/rollout-service> progress.cancel()
```

- **To delete a completed workflow:**

```
wls:/domain_name/domainRuntime/RolloutService/rollout-service>
cmo.deleteWorkflow('workflow_id')
```

### 3.4.10 Sample WLST Script

This section contains a sample WLST script that illustrates how to perform a rolling restart of all servers in a cluster called `cluster1` with single service migration. In this script, the following arguments are defined:

- `username` — The WebLogic Server administrator user name.
- `password` — The WebLogic Server administrator password.
- `adminURL` — The host name and port number of the domain's Administration Server.

- `target` — The target or targets for the operation. See [Table 3-1](#).
- `options` — The rollout option or options for the operation. See [Table 3-1](#).

The following example shows a sample WLST script for a rollout operation.

```
import sys, socket
import os
import time
from java.util import Date
from java.text import SimpleDateFormat

argUsername = sys.argv[1]
argPassword = sys.argv[2]
argAdminURL = sys.argv[3]
argTarget = sys.argv[4]
argTarget = sys.argv[5]

try:
    connect(argUsername, argPassword, argAdminURL)
    progress = rollingRestart(argTarget, argTarget)
    lastProgressString = ""

    progressString=progress.getProgressString()
    # for testing progressString="12 / 12"
    steps=progressString.split('/')

    while not (steps[0].strip() == steps[1].strip()):
        if not (progressString == lastProgressString):
            print "Completed step " + steps[0].strip() + " of " + steps[1].strip()
            lastProgressString = progressString

            java.lang.Thread.sleep(1000)

            progressString=progress.getProgressString()
            steps=progressString.split('/')
            if(len(steps) == 1):
                print steps[0]
                break;

        if(len(steps) == 2):
            print "Completed step " + steps[0].strip() + " of " + steps[1].strip()

    t = Date()
    endTime=SimplifyDateFormat("hh:mm:ss").format(t)

    print ""
    print "RolloutDirectory task finished at " + endTime
    print ""

    state = progress.getStatus()
    error = progress.getError()

    stateString = '%s' % state
    if stateString != 'SUCCESS':
        #msg = 'State is %s and error is: %s' % (state,error)
        msg = "State is: " + state
        raise(msg)
    elif error is not None:
        msg = "Error not null for state: " + state
        print msg
```

```

        #raise("Error not null for state: %s and error is: %s" + (state,error))
        raise(error)
    except Exception, e:
        e.printStackTrace()
        dumpStack()
        raise("Rollout failed")

exit()

```

To execute this script, save it in a Python (.py) file and then enter commands similar to this. If you are running WLST on Windows, see [ZDT Patching Restrictions](#), for important information about using WLST on Windows.

```

$ORACLE_HOME/oracle_common/common/bin/wlst.sh
/u01/scripts/rollout/RollingRestart.py username password
t3://hostname:port cluster1 "migrationProperties=/u01/json/mig.txt"

```

## 3.5 Using the WebLogic Server Administration Console to Create and Monitor Workflows

This section describes how to create and monitor a patching workflow that rolls out a patched Oracle home, a new Java version, new application versions, or any combination of these tasks. It contains the following sections:

- [Accessing ZDT Workflow Functions in the WebLogic Server Administration Console](#)
- [Creating a New Workflow for a Domain, Clusters, or Servers](#)
- [Monitoring and Managing Workflows](#)
- [Workflow Statuses](#)
- [Workflow Logging](#)

### 3.5.1 Accessing ZDT Workflow Functions in the WebLogic Server Administration Console

To access the ZDT workflow functions in the WebLogic Server Administration Console:

1. In the WebLogic Server Administration Console, click the domain name under **Domain Structure**.
2. On the Settings for *domain\_name* page, select the **ZDT Control** tab.

This displays the four tabs (**Domain**, **Clusters**, **Servers**, and **Workflow Progress**) from which you can manage all workflow-related tasks.

### 3.5.2 Creating a New Workflow for a Domain, Clusters, or Servers

You can create a workflow to roll out an update to all servers in a domain, all servers in one or more clusters, or only selected servers. The workflow can be for rolling out a new Java version, rolling out a new patched Oracle home, rolling out one or more updated applications, or any combination of these. You can also create a patching workflow to roll back to a previous Oracle home, Java home, or application versions, or create a workflow to perform a rolling restart of servers.

Prior to following this procedure, access the ZDT Control tabs as described in [Accessing ZDT Workflow Functions in the WebLogic Server Administration Console](#).

To create a new workflow:

1. Select one of the following tabs:
  - **Domain:** Select this tab if you want to create a workflow for the Administration Server and all clustered servers in the domain.
  - **Clusters:** Select this tab if you want to create a workflow only for servers in specific clusters.
  - **Servers:** Select this tab if you want to create a workflow only for specific servers. Typically, you would select this option only in the following situations:
    - The Administration Server is the only server that will be included in the workflow.
    - A situation exists in which a Managed Server is out-of-sync with other Managed Servers that were already updated. For example, you may have added a new server to a cluster, but that server is using an older version of Java than the other Managed Servers in the cluster.

---

---

**Note:**

Oracle recommends that you not use the **Servers** tab to perform updates to individual Managed Servers unless it is absolutely necessary. When you update individual Managed Servers, there is no guarantee that sessions will be preserved and downtime will be avoided.

---

---

2. If you selected the **Clusters** tab, then select the clusters to include in the workflow. All servers in the selected clusters will be included in the workflow.  
If you selected the **Servers** tab, then select the servers to include in the workflow.
3. Click **Patch** to configure the workflow tasks.
4. Select the type of rollout (or rollback) that you want to perform:
  - **Java home:** Select if you only want to change to another Java version.
  - **Oracle home:** Select if you only want to roll out a new Oracle home or roll back to a previous Oracle home.
  - **Application:** Select if you only want to roll out one or more updated applications or roll back to one or more previous application versions.
  - **All Combinations:** Select if you want to roll out or roll back any combination of Java home, Oracle home, and application updates.
  - **Rolling Restart:** Select if you want to perform a rolling restart of the selected targets.
5. Click **Next**.

The displayed fields and options depend on the type of rollout or rollback you are performing.



6. If you are changing the Java home, in the **Java Home** field, enter the full path to the Java home to change to. For example:

**UNIX**

```
/jdk1.8.0_50
```

**Windows**

```
C:\jdk1.8.0_50
```

7. If you are rolling out a new Oracle home or rolling back to a previous Oracle home:
  - a. In **Rollout Oracle Home**, enter the full path to the JAR archive. *Only* if you are rolling back to a previous Oracle home, you can specify the path to the local directory that contains the Oracle home to roll back to. For more information about rolling back to a previous Oracle home, see [Reverting to the Previous Oracle Home, Java Home, or Applications](#).
  - b. In **Backup Oracle Home**, enter the full path to the directory in which you want to back up the current Oracle home. For example, if your original Oracle home is `/u01/Oracle_Home` and you specify `/u01/Oracle_Home_backup` for this field, then `/u01/Oracle_Home` will be moved (renamed) to `/u01/Oracle_Home_backup`.
  - c. If you are rolling back to a previous Oracle home, select the **Rollback** check box.
8. If you are rolling out one or more new application versions, in the **Application Properties** field, enter the full path to a JSON-formatted text file that contains the information needed to upgrade the applications. For more information about creating this file, see [Creating an Application Update JSON File](#).
9. If you only want to evaluate the patching workflow before executing it, then select the **Dry Run** check box.
10. If you want to migrate singleton services, such as JTA or JMS, during the rollout, then in the **Migration Properties** field, enter the full path to a JSON-formatted file that contains the migration information. For more information about creating this file, see [Preparing to Migrate Singleton Services](#).
11. By default, the **Auto Revert on Failure** check box is already selected. This will cause the patching operation to automatically revert everything if there is a failure while the workflow is executing. If you clear this check box, then the patching operation will not automatically revert if there is a failure; the operation will stop and wait for you to resume it or revert it.
12. The **Session Compatibility** option determines whether or not the very last server being updated on a cluster will wait for sessions to complete on that server.
  - If not selected, the last server in a cluster waits for sessions to complete. This ensures that a compatible server is available in the cluster to handle sessions that must be served by a Managed Server that is still running on the existing version.
  - If selected, this indicates that the session state in servers is 100% compatible between the existing version and the new version. Therefore, the last Managed

Server in the update sequence in a cluster will shut down without waiting for all existing sessions to complete.

Oracle recommends that you not select this option unless you are absolutely sure that the session state is identical. This may cause the rollout to take longer due to the wait for session completion. The default session timeout value is 1 hour.

---

---

**Note:**

Serialization and deserialization in WebLogic Server differs slightly from Java serialization and deserialization. Therefore, additional fields on classes may result in a session being incompatible with servers on the new version, requiring that they be served by a server on the existing version. For example, a User class that adds a field such as Information will cause that session to be incompatible between versions.

---

---

13. Click **Finish** to initiate the patching workflow.

The workflow will be added to the Workflow Progress table.

After the workflow has started, you can monitor and manage its progress from the Workflow Progress page as described in [Monitoring and Managing Workflows](#).

### 3.5.3 Monitoring and Managing Workflows

This section describes how to monitor and manage the progress of all running or completed workflows.

Prior to following this procedure, if you have not already done so, access the ZDT patching tabs as described in [Accessing ZDT Workflow Functions in the WebLogic Server Administration Console](#).

To monitor and manage workflows, select the **Workflow Progress** tab. This page contains two tables:

- The Workflows in Progress table shows all workflows that are not yet completed (active); that is, they are in an executing, reverting, stopped, canceled, or failed state. Depending on its status, you can perform various actions on an active workflow:
  - You can **Cancel** any workflow that is in a STARTED, REVERTING, or RETRY state.

To cancel one or more workflows, select the check box for each workflow that you want to cancel, and then click **Cancel**. You can then revert the workflow by clicking **Revert** or resume it by clicking **Execute**.
  - You can **Execute** any workflow that is in a CANCELED, REVERT\_CANCELED, FAIL, or REVERT\_FAIL state.

To execute one or more stopped (canceled) workflows, select the check box for each workflow that you want to resume, and then click **Execute**. The workflow will continue executing, starting with the step after the last successfully completed step.
  - You can **Revert** any workflow that is in a CANCELED, REVERT\_CANCELED, FAIL, or REVERT\_FAIL state.

To revert one or more stopped (canceled) workflows, select the check box for each workflow that you want to revert, and then click **Revert**. The workflow will revert, starting with the last successfully completed step.

- You can **Delete** any workflow that is in a CANCELED, REVERT\_CANCELED, FAIL or REVERT\_FAIL state. You can delete only one workflow at a time.

To delete a workflow, select the check box for each workflow that you want to delete, and then click **Delete**.

- The Completed Workflows table shows all workflows that have completed. This table is sorted based on when the workflow completed, with the most recently completed workflow at the top of the table.

To delete completed workflows, select one or more of them and click **Delete**.

From these tables, you can also view additional details about the status of a workflow. To do so, click the workflow ID in the Workflow ID column. For more information, see [Viewing Workflow Details](#). For information about workflow statuses, see [Workflow Statuses](#).

### 3.5.3.1 Viewing Workflow Details

This section describes how to view the details of an active or completed workflow, and also describes the information that is displayed for the workflow.

To view the details for a workflow, click the workflow ID (for example, *wf00071*) in the Workflow ID column of either the Workflows in Progress or Completed Workflows table on the **Workflow Progress** page. A page is displayed with the information described in the following table.

Field	Description
Workflow ID	The workflow that was automatically assigned when you created it
Type	The type of workflow, which can be: <ul style="list-style-type: none"> <li>• <b>rolloutJavaHome</b>: You are rolling out or rolling back to a different Java home version.</li> <li>• <b>rolloutOracleHome</b>: You are rolling out or rolling back to a different Oracle home.</li> <li>• <b>rolloutApplications</b>: You are rolling out one or more new application versions or rolling back to one or more previous application versions.</li> <li>• <b>rolloutUpdate</b>: You are rolling out or rolling back to any combination of Java home, Oracle home, or application version.</li> </ul>
Target	The servers to which the workflow is targeted, which can be: <ul style="list-style-type: none"> <li>• <b>Domain</b>: The workflow is targeted to all eligible servers in the domain, including the Administration Server.</li> <li>• <b>Comma-separated list of cluster names</b>: The workflow is targeted to all eligible servers in the listed clusters.</li> <li>• <b>Comma-separated list of servers</b>: The workflow is targeted only to those servers that are listed.</li> </ul>
Status	The current status of the workflow. For more information, see <a href="#">Workflow Statuses</a> .

Field	Description
Can Resume	Indicates whether or not the workflow can be resumed or reverted. If <b>false</b> , you will not be able to use the <b>Execute</b> or <b>Revert</b> functions on the workflow.
# of Completed Commands	The number of workflow commands that have currently been completed
# of Total Commands	The total number of commands in the workflow that need to be executed to complete the workflow
Progress String	A detailed message about the progress of the workflow, such as: <code>Workflow wf0008 finished successfully. 36 steps completed.</code>
Next Execute Step	If the workflow is still active and is not reverting, then this field shows the next command that will be executed after the current command completes.
Next Revert Step	If the workflow is still active and is reverting, this field shows the next command that will be executed in the revert process after the current command completes.
Begin Time	The time at which the workflow was started
End Time	If the workflow has completed, then this field displays the time of completion.
Exception	If the workflow failed, then this field displays the exception that occurred when it failed.
Advanced	Click the arrow to expand the Advanced section, which shows all steps that have been executed for the workflow up to the current time. If the workflow has completed, then this section lists all commands that were completed by the workflow.

### 3.5.3.2 Viewing Server Status

From the **Servers** page, you can view the current status of all your servers before and after running a workflow and while workflows are in progress. When you click the **Servers** tab, you can view the workflow-related information about each server in your domain. For information about the columns in the **Servers** table and additional columns that you can add to the table, see *View Server Patching Status* in *Administration Console Online Help*.

When a workflow is running, you can monitor and refresh the information on this page to get up-to-date status for each server.

### 3.5.3.3 Viewing Cluster Status

From the **Clusters** page, you can view information about all clusters in your domain before and after running a workflow and while workflows are in progress. For information about the columns in the **Clusters** table and additional columns that you can add to the table, see *View Cluster Patching Status* in *Administration Console Online Help*.

When a workflow is running, you can monitor and refresh the information on this page to get up-to-date information for each cluster.

### 3.5.4 Workflow Statuses

An active workflow can have any of the following statuses:

Status	Description
STARTED	The workflow has started and is currently running.
RESUME	A stopped workflow has been resumed.
REVERTING	A workflow that failed or was stopped is reverting.
FAIL	The workflow has failed to execute completely. This status appears only if the <b>Auto Revert on Failure</b> option was not configured for the workflow when it was started.
REVERTED	A workflow that was either automatically or manually reverted has successfully completed the revert process.
REVERT_FAIL	A workflow that was either automatically or manually reverted failed to revert successfully.
CANCELED	The workflow was canceled (paused).
REVERT_CANCELED	A workflow that was either automatically or manually reverted was canceled (paused).

### 3.5.5 Workflow Logging

A rollout consists of a series of steps. Each step logs a message to the Administration Server log when it starts and when it finishes. Messages are also logged if a step reverts, fails, or retries. The Administration Server log is located at

`domain_home/servers/AdminServer/logs`

#### 3.5.5.1 Filtering the Log File

The workflow ID is included in every log message related to a given workflow. Use the workflow ID to filter the Administration Server log file for messages related to a given workflow. If you initiated the workflow from the WebLogic Server Administration Console, then you can get the workflow ID from the **ZDT Control > Workflow Progress** tab. From WLST, you can use the following command to get the workflow ID:

```
progress.getWorkflowId()
```

To filter the log file, enter the following command:

```
fgrep wf0001 domain_home/servers/AdminServer/logs/  
AdminServer.log
```

#### 3.5.5.2 Log Message Format

The log messages for ZDT patching are formatted as follows:

<b>Message Type</b>	<b>Message Format</b>
A step begins executing.	Workflow <i>workflowId</i> is executing <i>step name</i> on <i>target</i> .
A step is complete.	Workflow <i>workflowId</i> successfully completed <i>step name</i> on <i>target</i> .
A step is being reverted.	Workflow <i>workflowId</i> is reverting <i>step name</i> on <i>target</i> .
A step has successfully reverted.	Workflow <i>workflowId</i> successfully completed revert of <i>step name</i> on <i>target</i> .
A step is being retried.	Workflow <i>workflowId</i> is retrying <i>step name</i> on <i>target</i> .
A step could not be completed successfully.	Workflow <i>workflowId</i> failed to complete <i>step name</i> on <i>target</i> .
A step could not be completed successfully due to an exception.	Workflow <i>workflowId</i> failed to complete <i>step name</i> on <i>target</i> due to error <i>exception</i> .
A step could not be reverted successfully.	Workflow <i>workflowId</i> failed to revert <i>step name</i> on <i>target</i> .
A step could not be reverted successfully due to an exception.	Workflow <i>workflowId</i> failed to revert <i>step name</i> on <i>target</i> due to error <i>exception</i> .

---