Oracle® Fusion Middleware

Administering Zero Downtime Patching Workflows

12c (12.2.1)

E63852-01

October 2015

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.

Oracle Fusion Middleware Administering Zero Downtime Patching Workflows, 12c (12.2.1)

E63852-01

Contents

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

Guide to This Document

This document is organized as follows:

- Chapter 1, "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.
- Chapter 2, "Preparing for Zero Downtime Patching," describes the preliminary steps that must be completed before you can configure a patching workflow.
- Chapter 3, "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.

1 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 contains the following sections:

1.1 What is Zero Downtime Patching?

WebLogic 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 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 OPatch.

- **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 will sequentially restart the Administration Server or servers in the selected clusters or both safely, including graceful shutdown of the servers and starting them up again.

Prior to creating a patching workflow, preliminary steps must be completed for each of these tasks with the exception of rolling restarts. See Chapter 2, "Preparing for Zero Downtime Patching," for more information.

1.3 Patching Workflow Process

When using 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, 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 the next section.

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 it at any point. The update will then be rolled back, starting with the last successfully completed step.

  After a workflow has completed, you can revert it by creating a workflow to reverse the update that was made. This differs slightly depending on the update you are reverting. If reverting to the previous Oracle Home, you are provided with an option to specify that it is a rollback. For Java and applications, you simply point to the previous version of Java or the application.

For more information on reverting an update, see Section 3.4.7, "Executing, Reverting, and Resuming Stopped Workflows."

1.5 Overview: Rolling Out a Patched Oracle Home

This section provides a high-level overview of how to roll out a patched Oracle Home to all nodes in your domain. Prior to doing the rollout:

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

See Section 2.1, "ZDT Patching Restrictions," for additional requirements and restrictions. Figure 1–1 at the end of this section shows the operations that are performed for an Oracle Home rollout on each node, regardless of whether you use OPatchAuto, WLST, or the Administration Console to perform the rollout.

To roll out a patched Oracle Home:

1. If you want to use OPatchAuto:

   a. Create the patched Oracle Home archive.

      For details, see Section 2.3.1, "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 Section 2.3.2, "Distributing the Patched Archive to Each Node Using OPatchAuto."

   If you want to manually create and distribute the patched Oracle:

   a. Use the `copyBinary` command to create an archive of your existing Oracle Home.

      For details on this step and the next step, see Section 2.3.3, "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 similar domain topology as 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 the Oracle OPatch tool to apply the desired patch or patches to the Oracle Home on your development or test system.

For details, see Section 2.3.4, "Applying Patches to the Second Oracle Home," and *Patching with OPatch*.

d. Test and verify the patched Oracle Home.

e. Once 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 Section 2.3.5, "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 explode the archive on each node. The rollout process will create the new Oracle Home on each node from the archive.

2. If you did not use Node Manager to start your Administration Server, shut down the Administration Server and use Node Manager to start it.

   For details, see Section 3.2, "Using Node Manager to Start the Administration Server."

3. 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 Section 3.3.1, "Using OPatchAuto to Initiate a Rollout."

   - Use the WLST `rolloutOracleHome` command and specify the Administration Server as the rollout target.

     For details, see Section 3.4.1, "Rolling Out a New Oracle Home."

   - In the Administration Console, select the **ZDT Control > Servers** tab, select the Administration Server, and then initiate and configure the workflow.

     For details, see Section 3.5.2, "Creating a New Workflow for a Domain, Clusters or Servers."

4. 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.

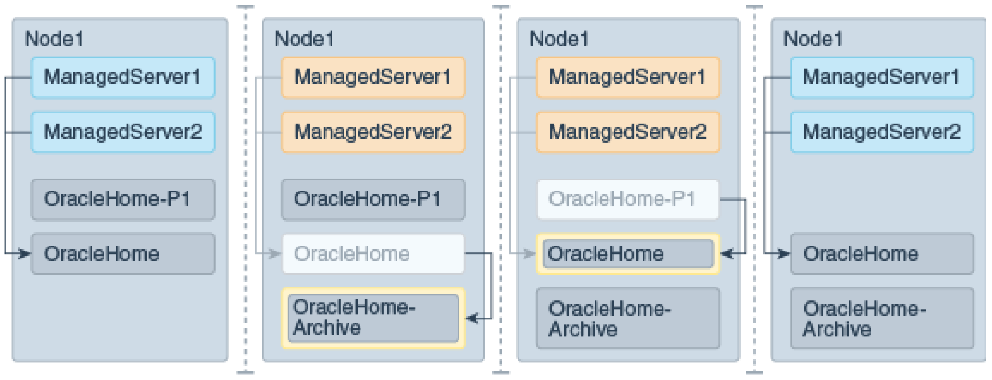
     For details, see Section 3.3.1, "Using OPatchAuto to Initiate a Rollout."

   - Use the WLST `rolloutOracleHome` command and specify a comma-separated list of clusters as the rollout target.

   - In the Administration Console, select the **ZDT Control > Clusters** tab, select the Clusters to which you want to rollout 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 > Domains** tab.

Figure 1–1 Oracle Home Rollout Operations



1.6 Overview: Rolling Out a New Java Version

This section provides a high-level overview of how to roll out a new Java version to all nodes in your domain. Prior to doing the rollout:

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

See Section 2.1, "ZDT Patching Restrictions," for additional requirements and restrictions.

To roll out a new Java version:

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

   For more details, see Section 2.4, "Preparing to Upgrade to a New Java Version."

2. If you did not use Node Manager to start your Administration Server, shut down the Administration Server and use Node Manager to start it.

   For details, see Section 3.2, "Using Node Manager to Start the Administration Server."

3. Create a ZDT workflow to roll out the new Java Home to your Administration Server. To do this, you can either:

   - Use the WLST `rolloutJavaHome` command and specify the Administration Server as the rollout target.

     For details, see Section 3.4.2, "Updating Your Java Version."

   - In the Administration Console, select the **ZDT Control > Servers** tab, select the Administration Server, and then initiate and configure the workflow.

For details, see Section 3.5.2, "Creating a New Workflow for a Domain, Clusters or Servers."

4. 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 Administration Console, select the **ZDT Control > Clusters** tab, select the clusters to which you want to rollout the new Java version, and then initiate and configure the workflow.

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

1.7 Overview: Rolling Out Updated Applications

This section provides a high-level overview of how to roll out new application versions to all Managed Server nodes in your domain. Prior to doing the rollout:

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

See Section 2.1, "ZDT Patching Restrictions," for additional requirements and restrictions. Figures 1–2 through 1–4 at the end of this section illustrate the scenario for staged, no-stage, and external staged applications. 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. (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 Section 2.5.1, "Impact of Staging Modes."

2. Create a JSON file that defines each application name, 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 Section 2.5.2, "Creating an Application Update JSON File."

3. If you did not use Node Manager to start your Administration Server, shut down the Administration Server and use Node Manager to start it.

   For details, see Section 3.2, "Using Node Manager to Start the Administration Server."

4. 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 rollout the applications, and then initiate and configure the workflow.

Figure 1–2 Patching Staged Applications

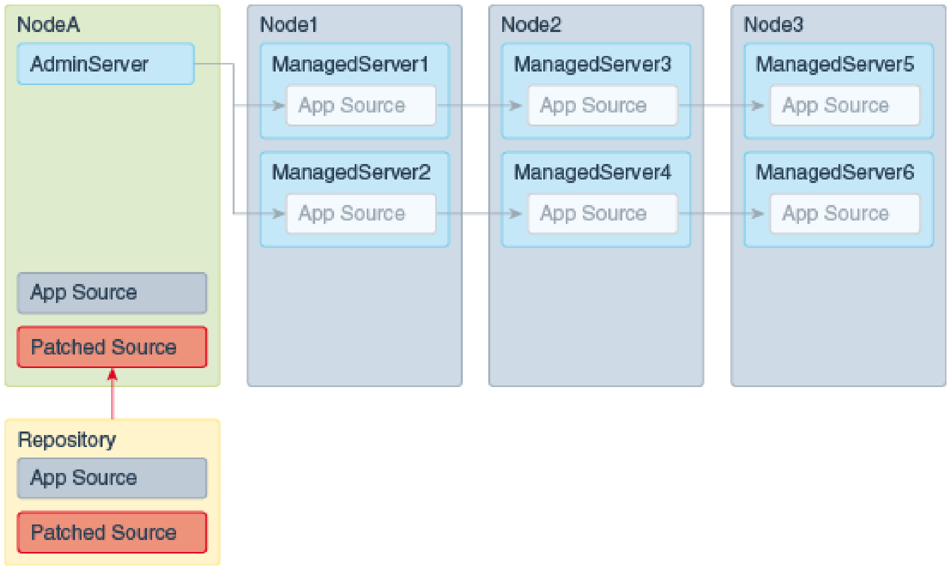


Figure 1–3 Patching No-Stage Applications

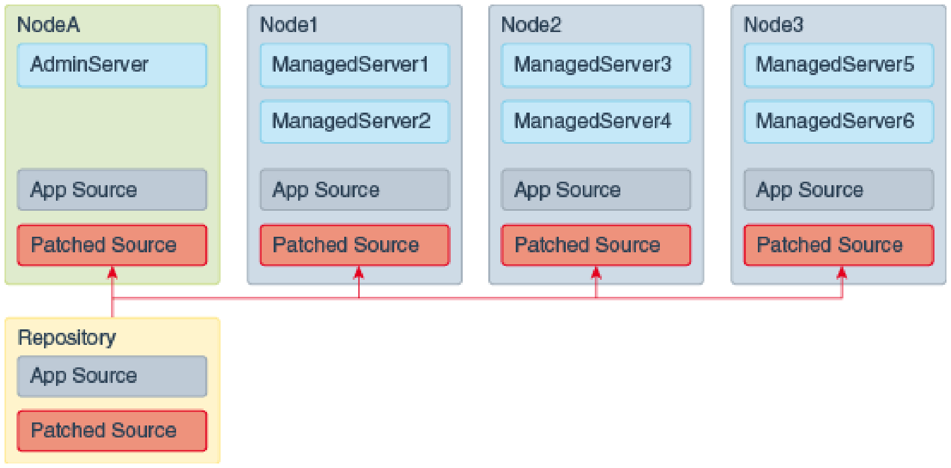
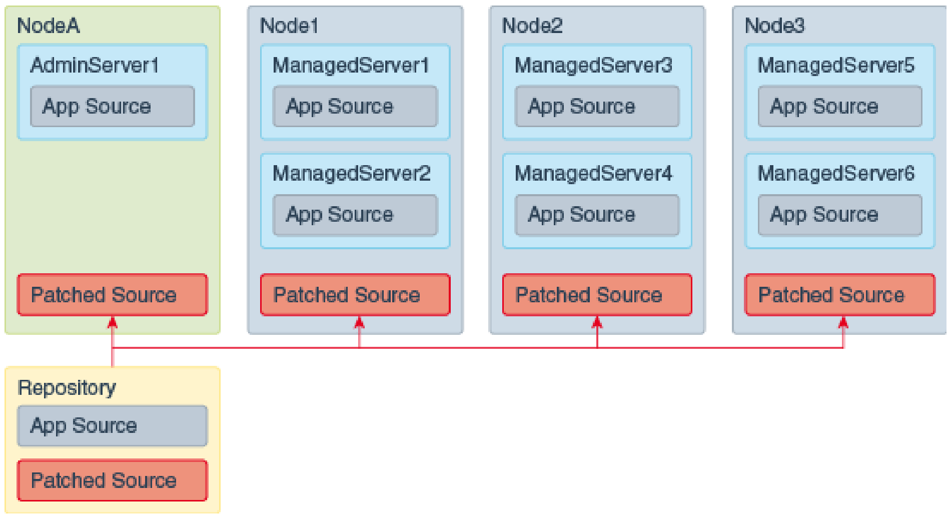


Figure 1–4 Patching External Staged Applications



1.8 In-Memory Session Replication for ZDT Rollouts

For webapps that use in-memory session replication, the in-memory sessions are never replicated or persisted to allow for failover, as a result of which webapps may lose session state due to a server crash or frontend misdirection causing the request to land on a sever 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. Since the default value for session timeout is one hour, the server may be in the SUSPENDING state for one hour or even longer if sessions continue to be utilized or updated. If you do not wait for the session to complete its lifecycle, then the state is lost as in-memory sessions are neither replicated nor persisted for webapps.

If you do not want to wait for an hour or longer, it is recommended that you set the `shutdownTimeout` option argument to the time (in seconds) that you want the server to wait before shutting down. For information about using the `shutdownTimeout` option argument, see Table 3–1, " Arguments for WLST rollout Commands".

2

Preparing for Zero Downtime Patching

This chapter describes the preliminary steps to complete before you can configure a patching workflow, including installing and patching a new Oracle Home, installing a new Java version, or installing updated applications on each node. It also describes the restrictions for ZDT Patching.

This chapter contains the following sections:

2.1 ZDT Patching Restrictions

Prior to preparing for and creating a ZDT patching workflow, keep the following restrictions in mind:

- If the Administration Server will be included in the workflow, you must use Node Manager to start it. Do not start the Administration Server using the `startWebLogic` command in the domain's bin directory. For more information, see Section 3.2, "Using Node Manager to Start the Administration Server."
- The Managed Servers that are included in the workflow must be part of a cluster, and the cluster must span two or more nodes.
- The Administration Server must be on a different node than any of the Managed Servers being updated.
- Each node involved in the workflow must have its own Node Manager running, including the node on which the Administration Server resides.
- If 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.
- If on a given node there are Managed Servers that belong to different clusters, and those clusters are sharing the same Oracle Home, if you are including 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, 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.
- Coherence applications (GAR files) are not supported by the WLST `rolloutApplications` command or the Applications function of ZDT in the WebLogic Server Administration Console.
- (Windows only) When using 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 deploying an application 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, including a domain rollout, on that node 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 happens to reference 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 of the features of the rollout is detection and handling of singleton services, such as JTA and JMS. If there are singleton services in your environment, service migration is configured for them as described in "Service Migration" in *Administering Clusters for Oracle WebLogic Server*. If a service is configured for migration the migration policy is `exactly-once`, then the service automatically migrates during a graceful shutdown of a server. If, however, the migration policy for a service is `manual` or `failure-recovery`, you must take steps to ensure that it is migrated safely during server shutdown. To do this:

- 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 Chapter 3, "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 `jms`, `jta`, or `all`, the name of the destination server to which the service is to be migrated. For `migrationType` of `server`, the name of another machine (node) in the domain on which Node Manager is running. This parameter is required if the `migrationType` is `jms`, `jta`, `server`, or `all`.
`migrationType`	The type of migration, which can be one of the following types: ■ `jms`—Migrate all JMS migratable targets from the source server to the destination server. ■ `jta`—Migrate all JTA services from the source server to the destination server. ■ `server`—Invokes Whole Server Migration to perform a server migration. The destination must be a machine (node) on which Node Manager is running. ■ `all`—Migrate all services (for example, JTA and JMS) from the source server to the destination server. ■ `none`—Disable service migration from the source server. If you specify this type, `failback` is not needed.
`failback`	If set to `true`, a fail back is performed. Failback restores a service to its original hosting server, that is, the server on which it was running before the rollout. The default value is `false` (no fail back). **Note:** A JTA service automatically fails back when it is invoked for migration. Therefore, do not use the `failback` option for JTA services, as it does not apply to them.

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

```
    {"migrations":[

# Migrate all JMS migratable targets on server1 to server2. Perform a fail back
# if the operation fails.
    {
    "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"
    }

    ]}
```

Note: ZDT rollouts allows you to specify whether a singleton service should be migrated or simply shutdown during patching. It also provides support for a user to migrate a service to a different server instance on the same machine. However, Oracle recommends that you always specify migration of services to a server on a different machine and never on the same machine. This is because, all servers on a machine experience shutdown during a rollout which may cause unavoidable downtime for users.

2.3 Preparing to Roll Out a Patched Oracle Home

This section describes how to prepare for rolling out a patched Oracle Home to your Managed Servers. There are two ways to do this:

- 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 the following 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 OPatch to apply patches to it, use copyBinary to create an archive of the patched Oracle Home, and then copy the archive to the nodes in your domain. See the following 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 impact on the availability of your applications.

Note: If your domain includes Fusion Middleware products other than WebLogic Server (such as SOA or 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 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. the patches you want to apply must already have been downloaded to your *patch_home* directory using OPatch.

To create a patched Oracle Home archive, enter the following commands. You must run this 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 this command:

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

2.3.2 Distributing the Patched Archive to Each Node Using OPatchAuto

After creating 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 command:

```
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 this command:

Parameter	Description
`-plan`	Indicates the type of operation to be performed by `opatchauto apply`. For distributing a patched Oracle Home for ZDT, always specify `wls-zdt-push-image` as the value for this parameter.
`-image-location` *`path`*	Specify the full path and file name of the image JAR file to distribute. For example: `-image-location /u01/images/OH-patch1.jar`
`-wls-zdt-host` *`adminserver:port`*	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.
`-wls-zdt-target` *`target`*	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.
`-wls-zdt-remote-image` *`path`*	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: `-wls-zdt-remote-image /u01/images/rollout-OH-image.jar`
`-wallet` *`path`*	The full path to a wallet directory that was created using `configWallet.sh` or `configWallet.cmd`. For example: `-wallet $HOME/wallet`
`-walletPassword` *`password`*	The password for the specified wallet, if needed. For example: `-walletPassword mypassword`

After completing these steps, you are ready to create a workflow that includes patching your Oracle Home. See Chapter 3, "Configuring and Monitoring Workflows."

Note: If you want to also update your Java version or applications using the same patching workflow, perform the preparation steps for those upgrades first 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 using `copyBinary` and `pasteBinary`.

Notes: Oracle recommends that you create and patch the second Oracle Home on a non-production machine so that you can test the patches you apply, but this is not required. The following steps must be performed 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 you want to patch.

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

2. Execute the following command, where *archive* is the full path and filename 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 filename 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

Once 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 you created in the previous section.

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

2. Execute the following command, where *archive* is the full path and filename of the archive file to create, and *patched_home* is the full path to the patched Oracle Home you created in the previous section. 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 Chapter 3, "Configuring and Monitoring Workflows."

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

2.4 Preparing to Upgrade to a New Java Version

This section describes how to prepare for upgrading to a newer version of Java. Preparation 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 the Node Manager and 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.
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 Chapter 3, "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:

- Impact of Staging Modes
- Creating an Application Update JSON File

2.5.1 Impact of Staging Modes

Applications deployed across Managed Servers can be deployed using one of three staging modes, which indicate how the application will be distributed and kept up-to-date. These are stage mode, no-stage mode and external-stage mode.

How you prepare for an application update workflow depends on the mode you used when you staged the application. The following table describes how to prepare for an application update based on the staging mode that you use to deploy the application.

Table 2–1 Preparing for an Application Update Based on Staging Mode

Staging Mode	Required Preparation and Result
Stage	Place a copy of the updated application directory on the domain's Administration Server. **Result:** The workflow will replace the original application directory on the Administration Server and WebLogic Server will copy it to each Managed Server.
No-stage	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. **Result:** 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.
External stage	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. **Result:** 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.

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 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)

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.

Example 2–1 Example JSON File for Updating Multiple Applications

```
{"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"
}
]}
```

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 Chapter 3, "Configuring and Monitoring Workflows."

3

Configuring and Monitoring Workflows

This chapter describes how to 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 WLST to create and monitor the workflow or you can create and monitor the workflow via the WebLogic Server Administration Console.

Notes: 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 Chapter 2, "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 contains the following sections:

- Strategies for Rolling Out a Patched Oracle Home
- Using Node Manager to Start the Administration Server
- Using OPatchAuto to Initiate, Revert, and Resume Rollouts
- Using WLST to Initiate and Monitor Workflows
- Using the Administration Console to Create and Monitor Workflows

3.1 Strategies for Rolling Out a Patched Oracle Home

When rolling out a new Oracle Home using either WLST or the Administration Console, the patched Oracle Home must be rolled out to the Administration Server first. There are two approaches you can take to do this:

- Use one workflow to roll out the patched Oracle Home to the Administration Server, 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 `AdminServer` as the target. You would then execute `rolloutOracleHome` or `rolloutUpdate` again, and specify cluster targets.
  – If using the Administration Console you would create one workflow from the **Servers** tab and select your Administration Server as the target. Once 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 Using Node Manager to Start the Administration Server

If the Administration Server will be included in a workflow, you must use Node Manager to start it before initiating the workflow.

Note: If the Administration Server is currently running and was started using Node Manager, there is no need to perform these steps.

To start the Administration Server using Node Manager:

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 Administration Server.
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, 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, 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, you must use WLST or the Administration Console.

Note: When using OPatchAuto to initiate a workflow, you must use the Administration Console to monitor the progress of the workflow. See Section 3.5.3, "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 this command:

Parameter	Description
`-plan`	Indicates the type of operation to be performed by `opatchauto apply`. For rolling out a patched Oracle Home for ZDT, always specify `wls-zdt` as the value for this parameter.
`-image-location` *path*	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: `-image-location /u01/images/rollout-OH-image.jar`
`-wls-zdt-host` *adminserver:port*	Specify the Administration Server hostname and port number for the domain you are rolling out the patched Oracle Home to.
`-wls-zdt-target` *target*	For *target*, specify the target for the rollout. This can be: ■ a domain name ■ a cluster name ■ a comma-separated list of clusters When rolling out the archive, you must specify the same target as you specified when you distributed the archive you are using for the rollout.
`-wls-zdt-backup` *path*	The full path to the directory and file to use for backing up your existing Oracle Home. For example: `-wls-zdt-backup /u01/images/rollout-OH-image.jar`

Parameter	Description
`-wls-zdt-remote-image` *path*	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: `-wls-zdt-remote-image /u01/images/rollout-OH-image.jar`
`-wallet` *path*	The full path to a wallet directory that was created using `configWallet.sh` or `configWallet.cmd`. For example: `-wallet $HOME/wallet`
`-walletPassword` *password*	The password for the specified wallet, if needed. For example: `-walletPassword mypassword`

3.3.2 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, 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 this command:

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

3.3.3 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 this command:

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

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 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 Section 2.1, "ZDT Patching Restrictions."

The following WLST commands are used for performing automated rolling updates of your servers. These commands must be executed 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 you use in this command can be one that was created either using `opatchauto` or `copyBinary` and `pasteBinary`.
- `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 you use in this command can be one that was created either using `opatchauto` or `copyBinary` and `pasteBinary`.
- `rolloutApplications`—Updates specified applications that are running on your Managed Servers.

Note: When specifying paths for Windows in rollout commands, you must use backslashes instead of forward slashes. To avoid undue 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 executing 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 includes:

- graceful shutdown of Managed Servers on a node 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.
- replacing the Oracle Home directory (if applicable)
- replacing the Java Home directory (if applicable)
- replacing application directories (if applicable)
- restarting the Node Manager on the node
- restarting the Managed Servers on the node

Table 3–1 Arguments for WLST rollout Commands

Argument	Description
`target`	Required for all `rollout` commands. Specifies which Managed Servers will be included in the update. `target` can be one of: *domain_name*—Specify a domain name as the target if you want the Administration Server and all Managed Servers in that domain to be updated. *clusters*—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. *servers*—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. **Note:** 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. One situation, for example, 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.
`rolloutOracleHome`	Applies only to and is required for the `rolloutOracleHome` 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.
`backupOracleHome`	Applies only to and is required for the `rolloutOracleHome` 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 /u01/Oracle_Home and you specify /u01/Oracle_Home_backup for this parameter, /u01/Oracle_Home will be moved (renamed) to /u01/Oracle_Home_backup.
`isRollback`	Optional. Applies only to the `rolloutOracleHome` and `rolloutUpdate` commands.
`javaHome`	Applies only to and is required for the `rolloutJavaHome` command. Specifies the location of the new Java home to use.
`applicationProperties`	Applies only to and is required for the `rolloutApplications` command. 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
options	The following options can be included in `rollout` commands. ■ `isDryRun`—If `TRUE`, the workflow operation will be evaluated but not executed. The default is `FALSE`. ■ `autoRevertOnFailure`—If `TRUE`, the workflow operation should automatically revert on failure. If `FALSE`, the workflow operation will stop on a failure and wait for you to resume or revert it. The default is `TRUE`. ■ `isSessionCompatible`—This option is applicable to all rollout commands, as it affects rollout time regardless of whether the rollout impacts session handling. The default is `FALSE`, 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. If set to `TRUE`, 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 set this to `FALSE` unless you are absolutely sure that the session state is practically identical. This may cause the rollout to take longer due to the wait for session completion. **Note:** WebLogic Server serialization/deserialization differs slightly from Java serialization/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. ■ `migrationProperties`—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 Section 2.2, "Preparing to Migrate Singleton Services." ■ `shutdownTimeout`—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 one second is ignored. If `isSessionCompatible` is set to `TRUE`, then the `shutdownTimeout` option defaults to zero, which means WLST waits forever for the server to shut down gracefully. If `isSessionCompatible` is set to `FALSE`, the user has to specify a value for the `shutdownTimeout` option. Oracle recommends that you specify a value that gives typical applications plenty of time to complete. Since different applications have different behaviors, this value must be decided by the user. ■ `DelayBetweenNodes`—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: ■ the servers on the first node to be restarted and join the cluster ■ the load balancer to evenly distribute traffic ■ any slow (lazy) stateful session bean clients to continue making requests before shutdown of the servers on the

You can also use WLST to monitor the progress of a workflow. For more information, see Section 3.4.6, "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.

Example 3–1 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.

Example 3–1 Rolling Out a New Oracle Home to a Domain (UNIX)

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

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.

Example 3–2 shows how to roll out a new Java Home to the cluster Cluster1. The new Java Home location is /u01/jdk1.8.0_50. The `autoRevertOnFailure` option is not included, therefore the workflow will automatically revert if the process fails.

Example 3–2 Rolling Out a New Java Home to Clusters (UNIX)

```
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 use 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.

Example 3–3 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, therefore the workflow will automatically revert if the process fails.

Example 3–3 Rolling Out a New Oracle Home and Java Home to the Administration Server (UNIX)

```
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 only 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 use the previous version of one or more applications.

`rolloutApplications` has the following syntax:

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

This command supports the `isDryRun`, `autoRevertOnFailure`, and `isSessionCompatible` options.

Example 3–4 shows how to roll out the applications defined in the JSON-formatted application properties file /u01/scratch/app_update.json to all servers in Cluster1.

Example 3–4 Rolling Out Application Updates to Clusters (UNIX)

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

3.4.5 Initiating a Rolling Restart of Servers

Use the `rollingRestart` command if you want to initiate a rolling restart of all servers in a domain or all servers in a specific cluster or clusters.

`rollingRestart` has the following syntax:

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

Example 3–5 shows how to perform a rolling restart of all servers in Cluster1 and Cluster2.

Example 3–5 Rolling Restart of Servers in Two Clusters (UNIX)

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

3.4.6 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 rollout command in the following format:

```
progress=rollout_command
```

For example, if 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 `WorkTaskRuntimeMBean` to return information about the workflow. For more information, see `WorkTaskRuntimeMBean` 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 current 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`.

Example Python Script Segment

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.

Example 3–6 Monitoring and Outputting the Progress of a Rollout Task

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

# Check the state every 2 minutes
while not (progress.isComplete()):
  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
```

3.4.7 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 have the opportunity to resolve any errors manually and then continue to execute or revert the workflow. To do so, use the following methods on RolloutService:

Method	Description
`executeWorkflow(WorkflowTaskRuntimeMBean)`	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.
`revertWorkflow(WorkflowTaskRuntimeMBean)`	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.
`canResume(WorkflowTaskRuntimeMBean)`	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.

3.4.8 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')
```

3.4.9 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 hostname 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

Example 3–7   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=SimpleDateFormat("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 running WLST on Windows, see Section 2.1, "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 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 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 Administration Console

To access the ZDT workflow functions in the Administration Console:

1. In the Administration Console, click on 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 Section 3.5.1, "Accessing ZDT Workflow Functions in the 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 want to 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 updating individual Managed Servers, there is no guarantee that sessions will be preserved and downtime will be avoided.

2. If you selected the **Clusters** tab, 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, select the servers to include in the workflow.
3. Click **Patch** to configure the workflow tasks.
4. Select the type of rollout (or rollback) 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

   ```
   /jdks/jdk1.8.0_50
   ```

   Windows

```
C:\jdks\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 or local directory that contains the Oracle Home to change to.

    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, /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 is **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 Section 2.5.2, "Creating an Application Update JSON File."

9. If you only want to evaluate the patching workflow before executing it, select the **Dry Run** check box.

10. If you want to migrate singleton services, such as JTA or JMS, during the rollout, 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 Section 2.2, "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, 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 practically identical. This may cause the rollout to take longer due to the wait for session completion. The default session timeout value is one hour.

Note: WebLogic Server serialization/deserialization differs slightly from Java serialization/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.

Once the workflow has started, you can monitor and manage its progress from the Workflow Progress page as described in Section 3.5.3, "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 Section 3.5.1, "Accessing ZDT Workflow Functions in the 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, 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 (cancelled) workflows, select the check box for each workflow that you want to resume, 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 (cancelled) workflows, select the check box for each workflow that you want to revert, 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 active workflow at a time.

To delete a workflow, select the check box for each workflow that you want to revert, then click **Revert**. The workflow will revert, starting with the last successfully completed step.

- 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 on the workflow ID in the Workflow ID column. For more information, see Section 3.5.3.1, "Viewing Workflow Details." For information about workflow statuses, see Section 3.5.4, "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 on 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 Table 3–2.

Table 3–2 Workflow Progress Details Fields

Field	Description
Workflow ID	The workflow that was automatically assigned when you created it.
Type	The type of workflow, which can be: ■ **rolloutJavaHome**—You are rolling out or rolling back to a different Java Home version. ■ **rolloutOracleHome**—You are rolling out or rolling back to a different Oracle Home. ■ **rolloutApplications**—You are rolling out one or more new application versions or rolling back to one or more previous application versions. ■ **rolloutUpdate**—You are rolling out or rolling back to any combination of Java Home, Oracle Home, or application version.
Target	The servers to which the workflow is targeted, which can be: ■ **Domain**—The workflow is targeted to all eligible servers in the domain, including the Administration Server. ■ **Comma-separated list of cluster names**—The workflow is targeted to all eligible servers in the listed clusters. ■ **Comma-separated list of servers**—The workflow is targeted only to those servers that are listed.
Status	The current status of the workflow. See Section 3.5.4, "Workflow Statuses," for more information.
Can Resume	Indicates whether or not the workflow can be resumed or reverted. If **false**, you will not be able to use the **Execute** or **Revert** 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.

Table 3–2 (Cont.) Workflow Progress Details Fields

Field	Description
Progress String	A detailed message about the progress of the workflow, such as: Workflow wf0008 finished successfully. 36 steps completed.
Next Execute Step	If the workflow is still active and is not reverting, this field shows the next command that will be executed once 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 once the current command completes.
Begin Time	The time at which the workflow was started.
End Time	If the workflow has completed, this field displays the time of completion.
Exception	If the workflow failed, 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, 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 also 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 .

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 also 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 .

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 statuses listed in the following table.

Table 3–3 Workflow Statuses for Active Workflows

Status	Description
STARTED	The workflow has started and is currently running.
RETRY	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 **Auto Revert on Failure** option was not configured for the workflow when it was started.

Table 3–3 (Cont.) Workflow Statuses for Active Workflows

Status	Description
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, making it easy to filter the Administration Server log file for messages related to a given workflow. If you initiated the workflow from the Administration Console, 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 domain_home/servers/AdminServer/logs/AdminServer.log
```

3.5.5.2 Log Message Format

The following table shows the format of log messages for ZDT patching.

Table 3–4 Log Messages Format for ZDT Patching

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

Table 3–4 (Cont.) Log Messages Format for ZDT Patching

Message Type	Message Format
A step could not be reverted successfully	`Workflow <workflowId> failed to revert <step name> on <target>.`
A step could not be reverted successfully due to an exception	`Workflow <workflowId> failed to revert <step name> on <target> due to error <exception>.`