# Oracle® OPatch

## User's Guide

F23315-02
October 2020

Oracle OPatch User's Guide,

F23315-02



Primary Author: Oracle Corporation

# Contents

## Glossary

## Index

# Preface

The *OPatch User's Guide* provides information about Oracle's patching solutions to help ensure your Oracle products stay current and secure.



## Audience

This document is intended for administrators who want to set up and manage Oracle software.

## 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 the latest releases of these and other Oracle documentation, check the Oracle Technology Network at:

http://www.oracle.com/technetwork/documentation/index.html#em

Oracle Enterprise Manager also provides extensive Online Help. Click **Help** at the top of any Enterprise Manager page to display the online help window.

## Conventions

The following text conventions are used in this document:

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

1

# Introduction to OPatch and Patching

> **Note:**
>
> This book applies to all Database 12.x, Enterprise Manager 12c/13c, and Fusion Middleware 12.x releases.

OPatch consists of patching utilities that help ensure your Oracle software stays current and secure. The utilities are:

- **OPatch**: A Java-based utility that enables the application and rollback of patches to Oracle software.
- **OPatchauto**: A patch orchestration tool that generates patching instructions specific to your target configuration and then uses OPatch to perform the patching operations without user intervention. Specifically, OPatchAuto can:
  1. Perform pre-patch checks.
  2. Apply the patch
  3. Start and stop the running servers.
  4. Perform post-patch checks.
  5. Roll back patches when patch deinstallation is required.
- **OPatchauto -binary**: A patch application tool that applies a single patch or multiple patches on a selected Oracle home. OPatchauto -binary patches only one Oracle home per session.

These utilities provide you with the flexibility to analyze, troubleshoot, and patch an individual GI (Grid Infrastructure)/RAC (Real Application Cluster) home environments.

For large-scale IT environments, patching individual product (e.g., Fusion Middleware) homes may not be practical since patching large numbers of targets manually is both monotonous and error prone. To maintain and deploy Oracle patches across many targets across your organization, you can use Enterprise Manager Cloud Control's patch automation capability. For more information about Enterprise Manager's patch management solution, see the "Patching Software Deployments" in the *Oracle® Enterprise Manager Lifecycle Management Administrator's Guide*.

This chapter covers the following introductory and overview topics:



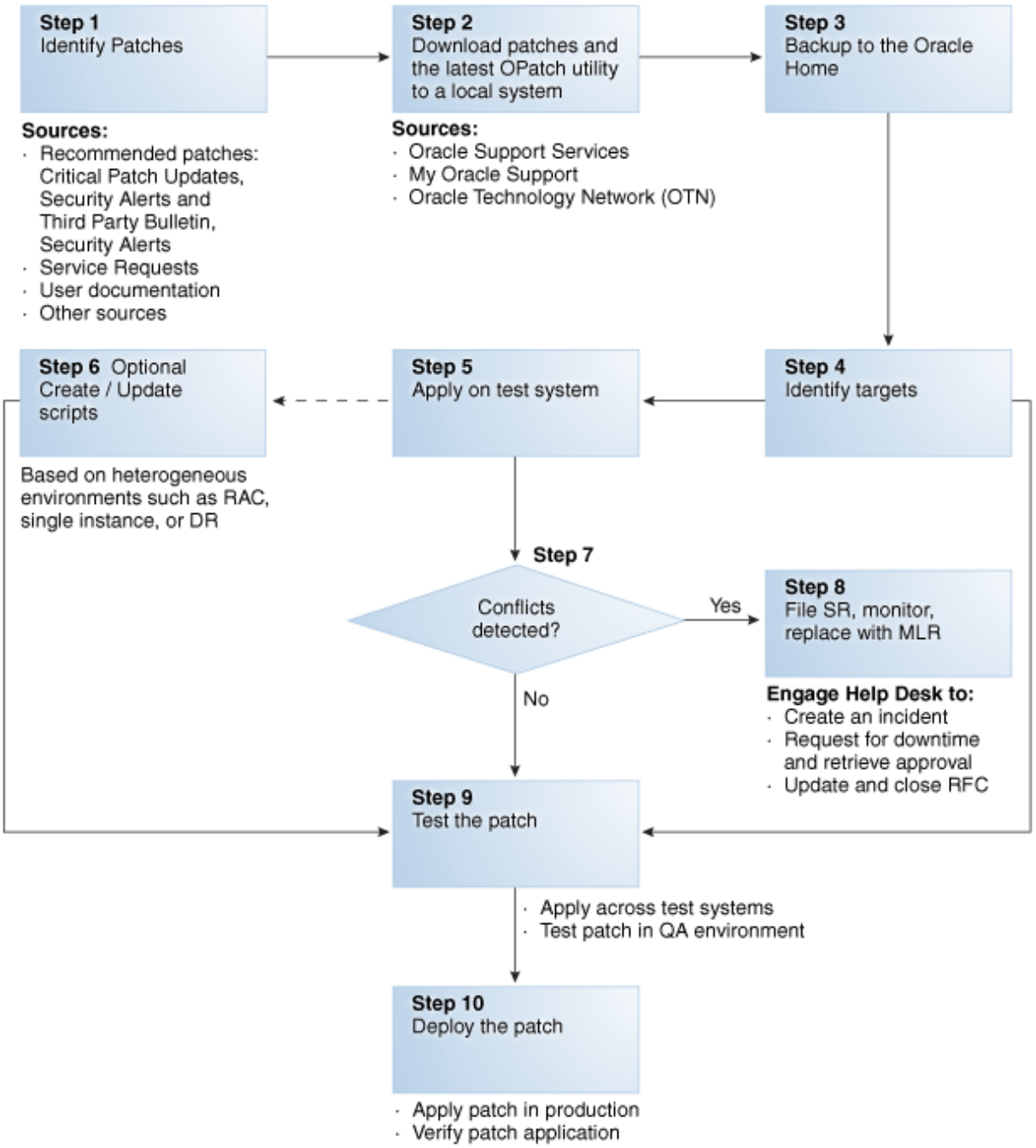
The patch process is not always straight forward as there are numerous factors that determine which software patches you need and how these patches should be

applied. For example, the types of Oracle software installed on each target, software versions, or platforms on which the software is running are just a few.

# Overview of the Patch Process

Regardless of your environment's patching requirements, the basic patching methodology is the same. The normal patching workflow can be broken down into the following nine steps shown in the following figure.

**Figure 1-1 Patch Process Overview - Process Flow**



# Obtaining the Patches You Need

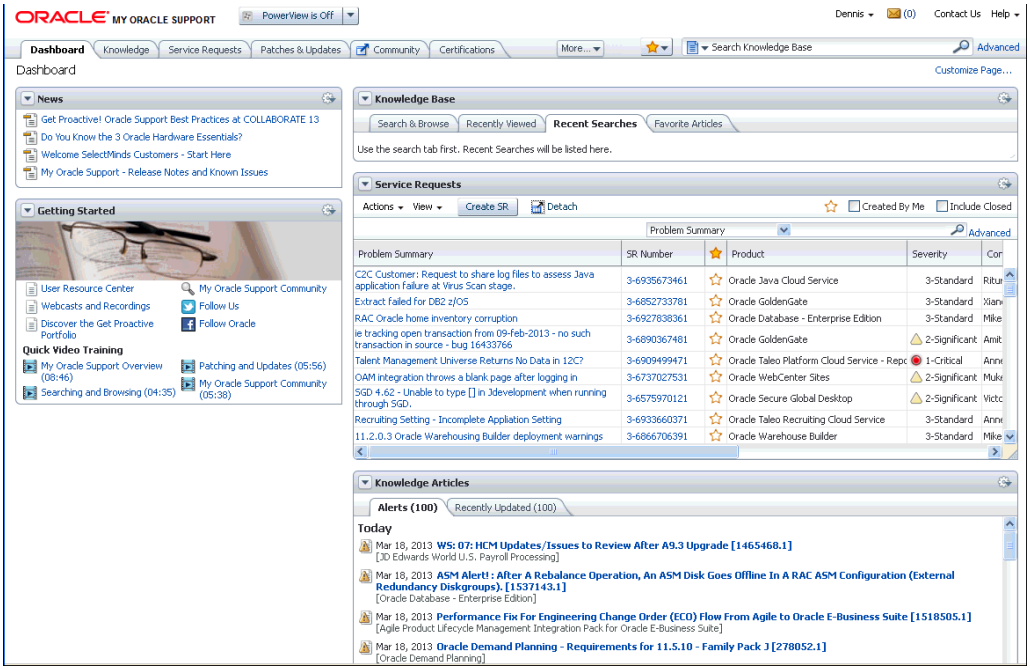
As shown in Figure 1-1, the first step is to determine what patches you need. You may find out about required patches from blogs, Oracle Software Downloads, Service Requests, Knowledge Articles, Oracle documentation, or any number of other sources. However, the single source of truth for patching is the Oracle Support Web site—My Oracle Support (MOS).

`https://support.oracle.com`

From here, you have access to interactive support tools and information that simplify searching for and obtaining the requisite patches for your Oracle environment. You can find complete documentation about MOS at the following location:

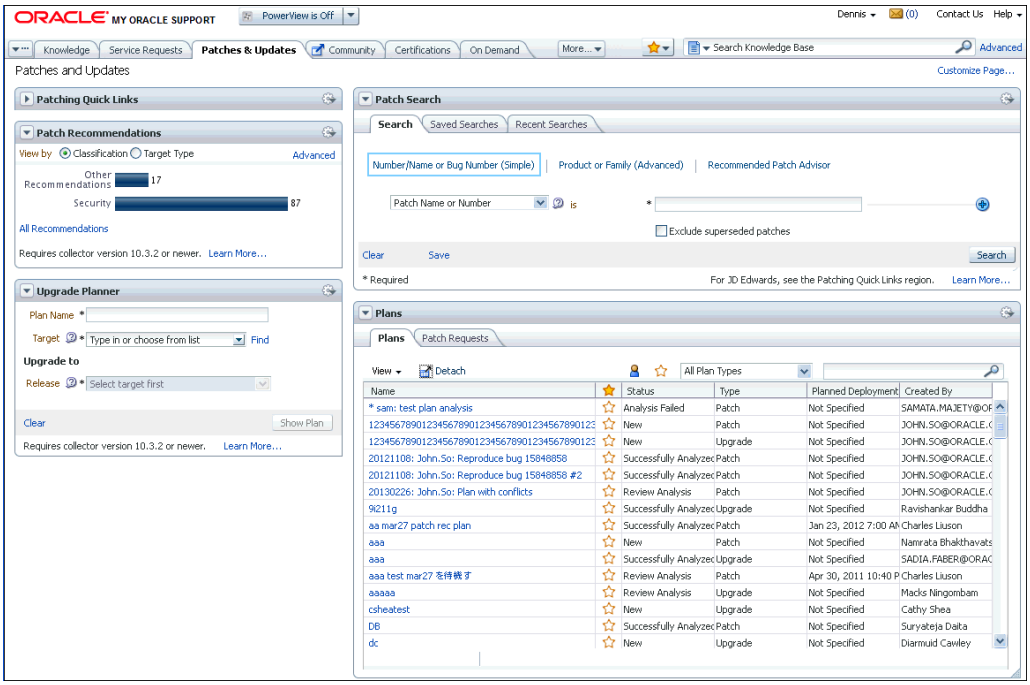
`http://docs.oracle.com/cd/E25290_01/index.htm`

**Figure 1-2 My Oracle Support Main Page**




My Oracle Support contains many features and capabilities that are grouped under tabs across the top of the application. Of primary interest is the **Patches and Updates** tab shown in the following figure.

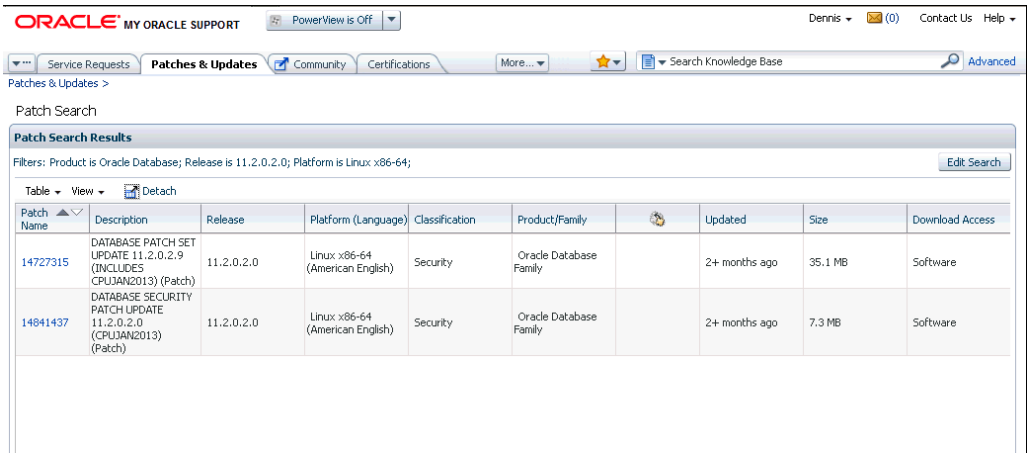
**Figure 1-3 MOS Patches and Updates**



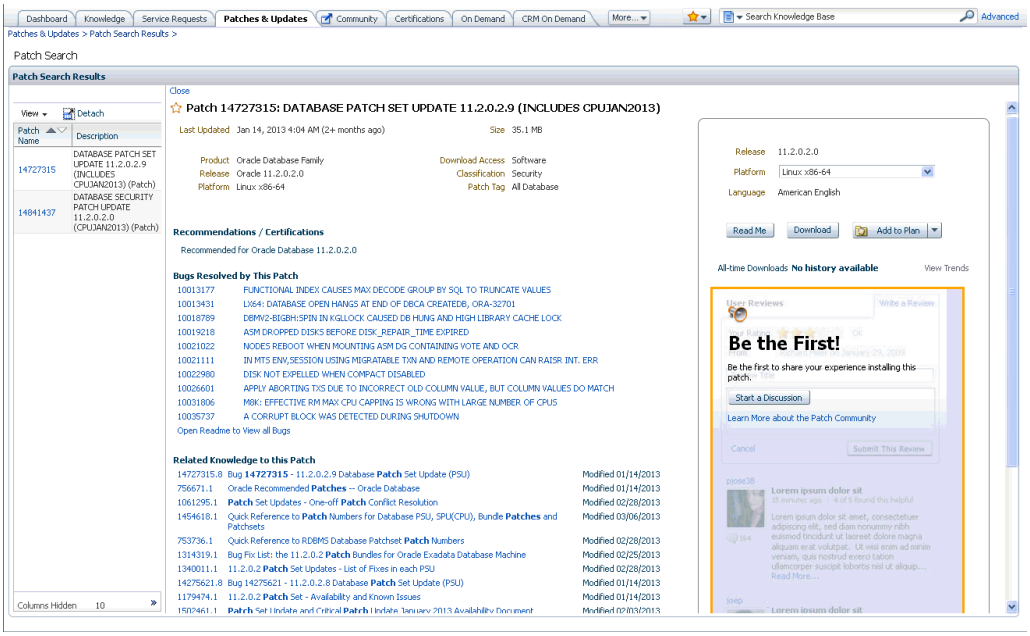

From this page, you can search for the desired patch based on a specific configuration. One particularly useful search feature is the Recommended Patch Advisor. The Recommended Patch Advisor lets you find recommended and mandatory patches for standalone products, product combinations, or products for a product stack. For example, using the Recommended Patch Advisor, you search for patches for the following product:

- Product: Oracle Database
- Release: 11.2.0.2.0
- Platform: Linux x86-64

This search returns the following results:



By clicking on patch 14727315 (PSU) you are taken to the patch page where you can view bugs resolved by this patch, related Knowledge Articles, or view a generic patch README.




From this page, you also complete **Step 2** of the patching workflow—Download the patch to your local system. The following list summarizes sources from which you can obtain patches.

- **Oracle Support Services**: If you are working directly with an Oracle Support engineer, you may be provided with a diagnostic patch or an interim patch.
- **My Oracle Support**: As part of your regular patch maintenance schedule, you can obtain all patches from My Oracle Support:

  `https://support.oracle.com`

  Once you log in, click the **Patches & Updates** tab to begin your patch search. My Oracle Support offers several patch download options and automated tools to help you keep current with patches. See the Patches & Updates Web-based help for more information:

  `http://docs.oracle.com/cd/E25290_01/doc.60/e25224/patchesupdates.htm#CJAGJJGI`

- **Oracle Software Downloads**: Some Oracle software may be distributed through the Oracle Software Downloads:

  `http://www.oracle.com/technetwork/indexes/downloads/index.html`

### Types of Oracle Patches

Oracle regularly makes patches available to upgrade features, enhance security, or fix problems with supported software. The major types of patches are:

- **Interim patches** - contain a single bug fix or a collection of bug fixes provided as required

- **Interim patches for security bug fixes** - contain customer-specific security bug fixes
- **Diagnostic patches** - intended to help diagnose or verify a fix or a collection of bug fixes
- **Bundle Patch Updates (BPUs)** - a cumulative collection of fixes for a specific product or component
- **Patch Set Updates (PSUs)** - a cumulative collection of high impact, low risk, and proven fixes for a specific product or component and Security Patch Updates
- **Security Patch Updates (SPU)** - a cumulative collection of security bug fixes. SPUs were formerly known as Critical Patch Updates (CPU).
- **System Patch** - contains several sub-patches in a format that can be used by OPatchAuto.
- **Merge Label Request (MLR)** - a *merge* of two or more fixes. MLR creation requires a *label* for the new set of merged code and a Patch Set Exception

# Applying the Patch to the Desired Targets

Now that you have the requisite patch, you determine which targets in your environment need to be patched (Step 3 in the patching workflow) and then apply the patch to each target (Step 4). Step 4 is where the OPatch utilities come into play. See Figure 1-1 to view the complete patching workflow.

> **Note:**
>
> Ensure that you have the latest version of OPatch. For more information, see "Obtain the Latest OPatch Utility."

## Manual Patching

Using OPatch, you follow the generic instructions in the patch README. You can view the patch README bundled with the patch or directly from the MOS page for the patch in question.

You are required to read the linked support documentation and fill in the details of your specific configuration before you can implement any of the commands or add them to custom install scripts. Although this method is laborious, it provides you with a great deal of diagnostic capability and control if patch conflicts arise. See Binary Patching Using OPatch .

## Configuration Patching

OPatchAuto performs end-to-end configuration patching. Configuration patching is the process of patching a target based on its configuration. By incorporating the site configuration information into the patch process, OPatchAuto is able to simplify patching tasks by automating most of the steps.

See Concepts of Patch Orchestration Using OPatchAuto for more information.

# Patching with Enterprise Manager

For large-scale patching, Oracle Enterprise Manager Cloud Control integrates OPatch with My Oracle Support (MOS), allows you to view patch recommendations, search patches, and roll out patches from a single user interface. In addition, Enterprise Manager's advanced Patch Plan feature provides you with a complete, end-to-end orchestration of the patching workflow. Automating the selection of deployment procedures and analysis of patch conflicts greatly reduces manual effort required to patch complex IT environments.

For more information, see Patch Management in the *Oracle® Enterprise Manager Lifecycle Management Administrator's Guide*.

# Who Should Use OPatch?

While OPatch is integrated with many Oracle product installations, you may still find that you need to use the OPatch directly. The core OPatch tool is used directly by admins as part of manual patching. OPatchAuto invokes OPatch, so an understanding of core OPatch is useful. OPatch is also useful for conflict detection and resolution as well as troubleshooting.

> **Note:**
>
> Before performing any patch task, *always* read the patch README file for any special patching instructions.

You can use the OPatch utilities if your administrative tasks require you to:

- Report on installed products and patches.
- Apply one or more patches.
- Roll back the application of one or more patches.
- Detect conflicts among incoming patches and between it and previous patches that have been applied. OPatch suggests the best options to resolve a conflict.

# What's Covered in this Guide

This document describes how to use these patching utilities and covers the following topics:

> **Note:**
>
> Recommendation of what tool to use when is stated in the README of the given patch. Always start with the README.

- Binary Patching Using OPatch - Describes the basic functions of the core opatch tool, that applies patches to an Oracle Home.

- Concepts of Patch Orchestration Using OPatchAuto - Recommended for administrators who wish to apply GI-RAC or Exadata patches to GI node in one shot via an orchestration tool.
- Troubleshooting OPatch .
- OPatch Syntax and Commands
- OPatchAuto Syntax and Commands

With these patching tools, you can design and implement a patch plan based on the configuration of your Oracle products.

> **Note:**
>
> Before patching any Oracle product, always check the product documentation for patching instructions.

# OPatch Integration with Other Oracle Software

In addition to Enterprise Manager, many Oracle software products have integrated the OPatch utilities to provide for a seamless and efficient patching task. Depending on the application, the call to the OPatch utility may be transparent, and all patching activity is maintained within the respective application.

These applications listed below have integrated OPatch into their respective environments. Always check the user documentation for any patching instructions before applying a patch.

**Fusion Middleware/Fusion Applications**

Other Oracle products, such as Fusion Middleware and Fusion Applications, integrate OPatch and may require different interaction to apply a particular patch. Refer to the following documentation:

- For patching Fusion MiddleWare see:
  - Summary of the Steps For Using OPatch in a Fusion Middleware Environment in *Oracle® Fusion Middleware Patching Guide.*
  - Preparing to User OPatch Auto in *Oracle® Fusion Middleware Patching Guide.*
- For patching Fusion Applications:
  - Section 3 Using Oracle Fusion Applications Patch Manager in the Oracle® Fusion Applications Patching Guide:`http://docs.oracle.com/cd/E29505_01/fusionapps.1111/e16602/applypatches.htm#CIHBDCBE`

# How to Access the OPatch Utilities

With OPatch integrated in many Oracle products, the utility is automatically installed when you install the respective product (for example, Enterprise Manager). The patching tools are installed in the following directories:

- OPatch - `$ORACLE_HOME/OPatch/opatch`

- OPatchAuto - `$ORACLE_HOME/opatch/opatchauto`

See OPatch Syntax and Commands, and OPatchAuto Syntax and Commands for a complete list of commands and options supported by OPatch utilities.

# 2
# Binary Patching Using OPatch

OPatch is a utility that allows you to apply and/or roll back interim patches to Oracle's software. The manual process of applying a patch is called *binary patching*. For binary patching, you can use the OPatch utility to:

- Obtain the Latest OPatch Utility
- Using OPatch
- Applying a Patch Set Update (PSU)
- Patch Conflict Detection and Resolution

> **Note:**
>
> Always refer to the patch README for any special instructions before you apply a patch.
>
> Oracle recommends that you back up the `ORACLE_HOME` before any patch operation. You can back up the `ORACLE_HOME` using your preferred method. You can use any method, such as `zip, cp -r, tar,` and `cpio` to compress the `ORACLE_HOME.`

## Obtain the Latest OPatch Utility

You should use the version of OPatch that supports the ORACLE_HOME release. For example, if you are patching a 12.0.1 Oracle Home, then use OPatch version 12.0.1. If the Oracle Home is version 11.2 , then use OPatch version 11.2.

Oracle recommends that you use the latest released OPatch for 12.1 releases, which is available for download from My Oracle Support (patch 6880880). Select the ARU link for the 12.1.0.1.0 release.

https://updates.oracle.com/download/6880880.html

You can also view the support document "OPatch - Where Can I Find the Latest Version of OPatch?" (Doc ID 224346.1). This document contains a link to the instructional video "Downloading the OPatch Tool from MOS."

## Using OPatch

Before you use the OPatch command and available options, you must check that OPatch prerequisites have been fulfilled.

# Patching Workflow

Using the OPatch utility to patch your product (e.g., Fusion Middleware) home typically consists of the following steps:

1. Setting the ORACLE_HOME Environment Variable
2. Determining What is Installed On Your System
3. Ensuring Patch Application Prerequisites are Met
4. Applying a Patch
5. Running Post-apply Checks

## Setting the ORACLE_HOME Environment Variable

OPatch verifies whether the product home is present. You must ensure that the `ORACLE_HOME` environment variable is set to the product home of the product you are trying to patch. Check your respective vendor documentation for the details on how to set the environment variable.

> **Note:**
>
> Oracle Universal Installer binaries and inventories must be present in the home to be patched.

Other environment variables used include:

- `OPATCH_DEBUG` — Boolean setting that enables additional debugging information that can be used by software developers.
- `PATH` — Product home path information.

> **Note:**
>
> Adding $ORACLE_HOME/OPatch to the $PATH makes it more convenient to execute the OPatch commands from any directory.

## Determining What is Installed On Your System

The next step is to determine what is already installed on your system. For this, you use the `opatch lsinventory` command with either the `patch` or `patch_id` options.

**Example 2-1 `lsinventory` Command**

```
opatch lsinventory
```

For more information about this command, see "`lsinventory`."

## Ensuring Patch Application Prerequisites are Met

After you have determined that your system configuration is appropriate for the patches you wish to apply, it is advisable to view the operations OPatch will perform before performing the patch application to help determine whether all system prerequisites are met before applying the patch. For this, you use the OPatch `-report` option to print all patch application actions OPatch will perform without actually executing the actions.

**Example 2-2 `report` Option**

```
opatch apply -report
```

For more information about the -report option, see "OPatch Syntax and Commands."

## Applying a Patch

Once you have determined that the patch can be applied to your system successfully, you can now use OPatch to apply the patch. For this, you use the OPatch `apply` command.

**Example 2-3 `apply` Command**

```
opatch apply /tmp/patch/12345678
```

For more information about this command, see"`apply`."

## Running Post-apply Checks

After you have applied the patch to your system, you should perform a final check to ensure all patches have been successfully applied. For this, you again use the OPatch `lsinventory` command with either the `patch` or `patch_id` options. For more information about this OPatch command, see "`lsinventory`."

# Applying a Patch Set Update (PSU)

Once you have verified the prerequisite checks, use OPatch to apply a patch:

- Applying a Single Patch
- Apply Multiple Patches Using a Text File

The OPatch utility is located in the `$Oracle_Home/OPatch` directory. You can run it with various commands and options. See OPatch Syntax and Commands, for a complete list of command options available with OPatch.

## Applying a Single Patch

You apply a single patch following the generic patching workflow discussed earlier:

1. Obtain the Latest OPatch Utility
2. Setting the ORACLE_HOME Environment Variable
3. Determining What is Installed On Your System
4. Ensuring Patch Application Prerequisites are Met

5. Applying a Patch
6. Running Post-apply Checks

Once you have downloaded the patch, you can apply it (step 5) using the following command:

```
# opatch apply <patch directory location>/<patch ID>
```

For example:

```
# opatch apply /tmp/patch/12345678
```

## Apply Multiple Patches Using a Text File

You can create a text file containing the location of all patches you need to apply. Use OPatch to reference the file and apply the patches:

1. Create the text file of the patch location. The entry should have each patch location on a separate line:

   ```
   vi patches.txt
   /tmp/patchlocation1/12345678
   /tmp/patchlocation2/12365478
   /scratch/patchlocation3/32165487
   ```

   Save your changes.

2. Apply the patches with OPatch:

   ```
   # opatch napply - -phBaseFile <location of text file>
   ```

   For example:

   ```
   #opatch napply - -phBaseFile /tmp/patches/patches.txt
   ```

# Patch Conflict Detection and Resolution

OPatch detects and reports any conflicts encountered when applying a patch with a previously applied patch. The patch application fails in case of conflicts. You can use the `-force` option of OPatch to override this failure. If you specify `-force`, the installer firsts rolls back any conflicting patches and then proceeds with the installation of the desired patch. Using -force means that you will likely lose some bug fixes, that are causing the conflict, in exchange for getting the incoming patch to be applied.

You may experience a bug conflict and might want to remove the conflicting patch. This process is known as patch rollback. During patch installation, OPatch saves copies of all the files that were replaced by the new patch before the new versions of these files are loaded, and stores them in `$ORACLE_HOME/.patch_storage`. These saved files are called *rollback files* and are key to making patch rollback possible. When you roll back a patch, these rollback files are restored to the system. If you have a complete understanding of the patch rollback process, you should only override the default behavior by using the `-force` flag. To roll back a patch, execute the following command:

```
$ OPatch/opatch rollback -id <Patch_ID>
```

A patch conflict occurs when multiple fixes in different patches touch the same files but have not been tested together as a single entity. OPatch and OPatchAuto help

you avoid these conflicts by identifying these conditions. When patch conflicts occur and you are unable to resolve them using documented support procedures, MOS then becomes the go-to source for technical assistance. Conflict resolution may entail filing a Service Request and obtaining a Merge Label Request (MLR) patch to overcome a patching issue. Once a solution has been found, you use OPatch to apply the fixed patch.

3

# Concepts of Patch Orchestration Using OPatchAuto

OPatchAuto is Oracle's strategic tool for binary and configuration patching. For the supported environments (Fusion Middleware and Grid Infrastructure), OPatchAuto sequences and executes all required steps, on all nodes, for comprehensive patch application. Because OPatchAuto can patch full systems in one invocation, it removes the burden of:

- The physical effort of going host to host and executing commands
- The mental effort of remembering the sequence of commands across the nodes in your system

Your product's patching documentation (Database, Fusion Middleware, Enterprise Manager Cloud Control) explains how to use OPatchAuto to patch your specific product. This book augments those guides, by providing deeper conceptual and reference material for OPatchAuto, in a product independent manner.

## Patch Orchestration Concepts

Applying a patch involves an orchestrated series of steps. As OPatchAuto's name indicates, the customer does not need to understand these steps. They can just apply the patch.

However, the following underlying orchestration concepts are necessary for the customer to learn when the following are true:

- The patching operation has failed and they need to trouble-shoot.
- They see advantage to interleaving their own commands into the patching sequence in order to take advantage of their production system's downtime window. In this case, understanding phases is required.

## OPatchAuto Pre-requisite

- It is recommended to use `bash` shell as the default shell for executing OPatchAuto commands.
- SSH user equivalence is necessary for GI/RAC installation. The steps to create SSH user equivalence can be found in the *Grid Infrastructure Installation Guide.* SSH user equivalence is also mandatory to use OPatchAuto.
- SSH user equivalence must be created for each home owner separately.
- You must have the latest version of OPatch in all the homes of all the nodes of a cluster before you request for patching.

# Phases and Sessions

The conceptual related steps of patching operation is called a phase. Executing all phases leads to a completed patching operation on the target; skipping a phase means the patch is not correctly applied. For example, the phase/sub-phase of applying the bits to an Oracle Home is called offline:binary-patching in OPatchAuto.

Each invocation of `opatchauto apply` generates a new session, whether you are executing all phases in one go (default) or just a sub-phase (advanced).

Phase input to the command is both optional and an advanced feature. However, if the customer wishes to interleave commands between the phases, they will need to invoke OPatchAuto multiple times, specifying the specific phase in the correct sequence so that all phases are executed. Phases are composed of sub-phases. The user may also invoke the tool at the sub-phase granularity. The –help option lists the available phases and sub-phases.

Phases are idem-potent; you may execute them repeatedly. However, the tool will not inform you if you do not follow the correct sequence of invocations. (See ER 21553825.)

The high-level phases follow. They can be specified as (a) book-keeping, (b), life-cycle operations, of (c) configuration change operations

- *Init*: Bookeeping operation to initialize internal state needed for correction patching.
- *Shutdown*: Life-cycle operation that brings down run-time entities to permit patching.
- *Offline*: Configuration change operation to apply the patch content with the system down. Bits application happens here, for instance. So the opatch patch will be recorded to the homes OUI inventory in this phase.
- *Start-up*: Life-cycle operation that brings the shutdown entities back up again.
- *Online*: Configuration change operation apply patch content that requires that the systems be up. If these configuration changes have a system inventory, they will also be recorded to that system's inventory at this point.
- *Finalize*: Book-keeping operation to record that patch operation is complete.

In the product's documentation, you might see sub-phases that include “prepare" and “binary/product" variants. Prepare means "ready materials but do not make a configuration change." Binary operations only change Oracle Homes. Product operations change the configured system, such as domain configurations or database dictionaries.

The specific content of the patch determines precisely what specific Oracle Home and configuration changes occur. Most Fusion Middleware patches, for example, only include offline content changes. But, of course, some include configuration changes as well.

In general, the session is an implicit parameter, set internally to the last session. It is visible to the user in the logs, communicated as a session ID, but there is no requirement for it to be supplied by the user. As a convenience for specifying the rollback parameters, you can specify the session ID. In this case, OPatchAuto knows to query that session for the patch you wish to rollback.

# OPatch Automation (OPatchAuto)

With OPatchAuto, you can automatically patch the typical Grid Infrastructure (GI) and RAC home directories with minimal intervention.

OPatchAuto performs many of the pre-patch checks (see "Using OPatch") as well as the post-patch verification. The power of OPatchAuto lies in its ability to perform end-to-end *configuration patching*. Configuration patching is the process of patching a GI or RAC home based on its configuration. By incorporating the configuration information into the patch process, OPatchAuto streamlines patching tasks by automating most of the steps.

OPatchAuto uses your GI/RAC configuration and, from that information, automatically generates patching instructions specific to your site configuration. OPatchAuto then uses OPatch to implement these instructions and perform the actual application of the patch.

## Supported Patch Format

Beginning with Oracle Database 12*c*, patches have been converted to a *System patch* format in order to support patch automation.

**What is a System Patch?**

A System patch contains several sub-patches whose locations are determined by a file called *bundle.xml* in the top level directory of the patch. The sub-patches are intended for different sub-systems of a system that correspond with the database home organization.

A typical System patch format is organized as follows:

```
<System patch location - directory>
|_____ Readme.txt (or) Readme.html
       bundle.xml
       automation
           |_____ apply_automation.xml
       |_____ rollback_automation.xml
       Sub-patch1
                |_____ etc/config/inventory.xml
                |_____ etc/config/actions.xml
                |_____ files/Subpatch1 'payload'
       Sub-patch2
                |_____ etc/config/inventory.xml
                |_____ etc/config/actions.xml
                |_____ files/Subpatch1 'payload'
```

**Notes:**

- For database releases prior to 12*c*, OPatchAuto is not supported for the released one-off patches. For older releases, you must use OPatch and follow the patch README instructions.
- OPatchAuto and System patches are only supported by Oracle Database 12*c* and above.

**Additional Supported Patch Types/Configurations**

- One ore more one-off patches
- One composite patch
- One system patch / bundle patch
- One system patch / bundle patch and 1 or more one-off patches
- One system patch / bundle patch which has a composite patch in it
- One system patch / bundle patch which has a composite patch in it and 1 or more one-off patches
- One system patch / bundle patch and 1 or more one-off patches and 1 composite patch

# Supported Target Configurations

OPatchAuto can be applied to the following general configurations:

- GI Home Shared
- GI Home Not Shared
- RAC Home Shared
- RAC Home Not Shared
- SIHA and SIDB
- Shard and OGG

## Shared Versus Non-Shared (GI or RAC) Homes

The configuration differences between shared and non-shared Homes come into play when determining the patching mode in which OPatchAuto is used. See Patch Application Modes.

## Patch Application Modes

OPatchAuto supports two modes of patching a GI or RAC Home - Rolling and Non-rolling. When a patching session is started off (on the first node), the stack has to be up and running on this node. This applies to both rolling and non-rolling modes of patching.

**Rolling Mode (Default Mode):** When performing patching in Rolling mode, the ORACLE_HOME processes on a particular node are shut down, the patch is applied, then the node is brought back up again. This process is repeated for each node in the GI or RAC environment until all nodes are patched. This is the most efficient mode of applying an interim patch to an Oracle RAC setup because this results in no downtime. Not all patches can be applied using Rolling mode. Whether or not a patch can be applied in this way is generally specified in the patch metadata. The patch README also specifies whether or not a patch can be applied in Rolling mode. The node (GI Home) from which the `opatchauto` command is executed is considered the LOCAL node and all other nodes are considered REMOTE nodes.

When you begin a rolling mode session, at least 1 remote node has to be up and running.

OPatchAuto applies patches in rolling mode by default.

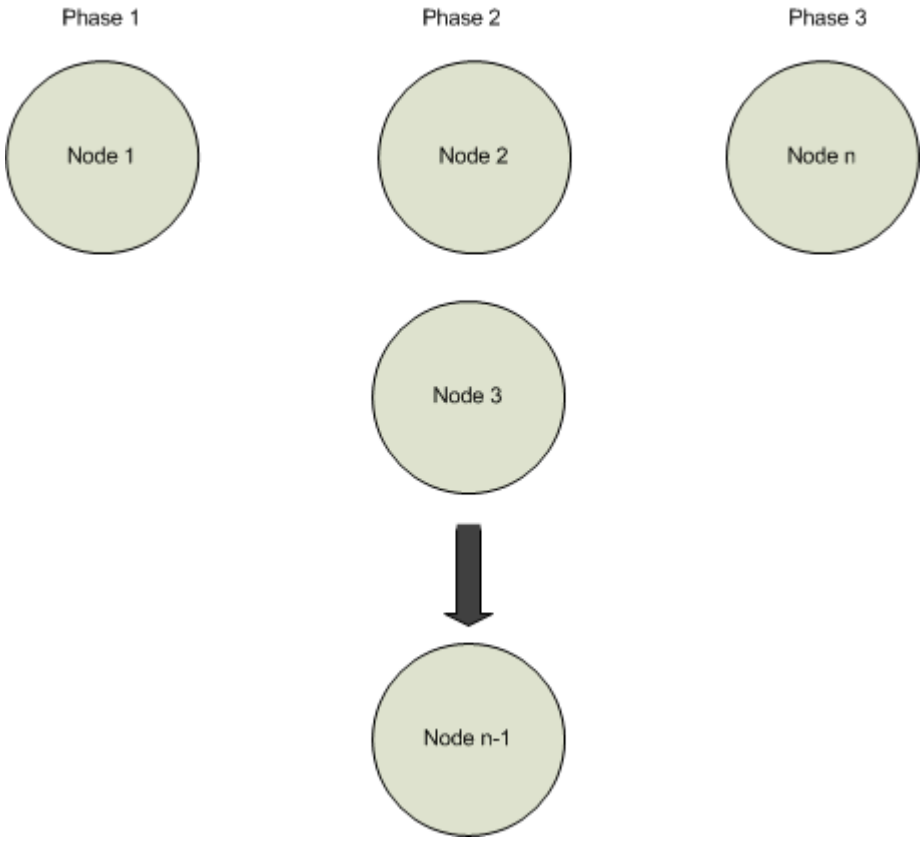
**Non-rolling Mode**: Prior to 12*c*, a non-rolling upgrade was defined as shutting down Oracle processes on all nodes. Beginning with 12c, non-rolling patching requires the GI stack to be up on local node. The patching operation on first and last node have special steps to perform hence the operation needs to be handled separately but not in parallel with other nodes. The non-rolling patching can ve described as three phases:

Beginning with 12c, non-rolling patching occurs in three phases:

1. Patch Node 1
2. Patch Node 2 through n-1
3. Patch Node n

When you start a non-rolling mode session none of the remote nodes can be up and running: Services in all nodes (excluding the first node) must be stopped.

As shown in the following figure, given *n* nodes, you begin the non-rolling patch session by patching a single node, then patch nodes two through *n*-1 in parallel, and finally patch node *n* to finish the patching session.




To run OPatchAuto in non-rolling mode, you must explicitly specify the `-nonrolling` option.

#### Patch Application Mode Conflict

As mentioned earlier, OPatchAuto applies patches in rolling mode by default. If the patch is applied in rolling mode but the patch content is not rollable (content does not support application in rolling mode), OPatchAuto will error out when attempting to run `rootcrs.pl -prepatch`.

## Configuration Support

OPatchAuto supports the following platforms:

- Oracle Solaris on x86-64 (64-bit)
- Linux x86-64
- Oracle Solaris on SPARC (64-bit)
- IBM AIX on POWER Systems (64-bit)
- HP-UX Itanium"
- Linux (32-bit)

OPatchAuto supports shared and non-shared Oracle homes. It supports patching cluster configurations managing mixed versions of the Oracle Database, though it of course only patches those databases with versions matching the input patch content.

> **Note:**
>
> Microsoft Windows is not supported.

# 4
# Multiple Patch Application

OPatchAuto supports the application of multiple patches. Changes to the OPatchAuto utility as well as its use (specific to multiple patch application) are described in this chapter.



## Command Line Support

The following changes to the OPatchAuto command line syntax have been implemented to support the application of multiple patches. A new command line option `-phBaseDir <Parent Directory>` is now available which allows you to specify a 'parent directory' that can hold all the patches to be applied in the session. Accordingly, OPatchAuto will perform all the binary patching and configuration patching/product patching operations on the entire list of these patches.

```
opatchauto apply [ -phBaseDir <patch-location> ] [ -oh <home> ]
                 [ -log <log> ]
                 [ -logLevel <log_priority> ] [ -analyze ]
                 [ -invPtrLoc <inventory.pointer.location> ]
  [ -nonrolling ]
  [ -generatesteps ]
  [ -norestart]
  [ -database <database> ]
  [ -wallet <wallet> ]
```

In addition, command line multiple patch support has been implemented for sharded (horizontally partitioned) databases.

```
opatchauto apply [ -phBaseDir <patch-location> ]
                 [-sdb]
                         [-sidb]
                         [-sid]
                         [-port]
                 [ -log <log> ]
                 [ -logLevel <log_priority> ] [ -analyze ]
                 [ -wallet <wallet> ]
                 [ -host <tns-host> ]
                 [-rolling]
```

## Supported Content of the Patch Base Directory

The following list shows the various types of patches that can be placed in the Patch Base directory.

- 1 or more one-off patches
- 1 Composite patch

- 1 Composite patch and 1 or more one-off patches
- 1 System Patch / bundle patch
- 1 System Patch / bundle patch and 1 or more one-off patches
- 1 System Patch / bundle patch which has a composite patch in it
- 1 System Patch / bundle patch which has a composite patch in it and 1 or more
- 1 System Patch / bundle patch and 1 or more one-off patches and 1 Composite patch (One-off patch)

# -phBaseDir Restrictions

The following table lists the restrictions when using the `-phBaseDir` option.

| Restriction | Failure |
|---|---|
| Patch location and -phBaseDir option are mutually exclusive to each other. | Failed during option validation while starting orchestration. |
| Every single directory in the parent-directory must be a valid patch. | Failed during patch package creation. |
| All patches should be of same version number. | Failed during opatch prereq. |
| The maximum number of system patches allowed is 1. Otherwise patching will fail during bootstrapping. | |

5

# Patching of Grid Infrastructure and RAC DB Environment Using OPatchAuto

OPatchAuto automates patch application to the Grid Infrastructure (GI) cluster, by applying patches to both the GI and the managed Oracle Real Application Cluster (RAC) homes.

Patch orchestration is the automated execution of the patching steps, such as the execution of pre-patch checks, stopping services, applying the binary patches, and starting the services. Patch orchestration for Oracle Database 12*c* applies the patch to the GI/RAC configuration on that machine, including all of its databases. The OPatchAuto patch orchestration utility is available with version 12.1 of the OPatch utility.

This chapter covers the following topics:



> **Note:**
>
> This chapter applies to Oracle Database 12*c* only.

## Configuration Support

OPatchAuto supports the following platforms:

- Oracle Solaris on x86-64 (64-bit)
- Linux x86-64
- Oracle Solaris on SPARC (64-bit)
- IBM AIX on POWER Systems (64-bit)
- HP-UX Itanium"
- Linux (32-bit)

OPatchAuto supports shared and non-shared Oracle homes. It supports patching cluster configurations managing mixed versions of the Oracle Database, though it of course only patches those databases with versions matching the input patch content.

> **Note:**
>
> Microsoft Windows is not supported.

# Preparing to Use OPatchAuto

To ensure successful patching, there are several prerequisites you should complete to prepare your environment for running OPatchAuto, such as obtaining the latest version of OPatch, obtaining required patches from My Oracle Support, and backing up the environment.

For more information on preparing your environment, see the following topics:

- Patching Your Environment Using OPatchAuto

  OPatchAuto is installed with the OPatch utility as a part of your installation. OPatchAuto provides several commands that you can use to Patching Your Environment Using OPatchAuto automate the application and roll back of a patch in a single host or multi-host environment.
- Locating and Obtaining the Latest Version of OPatch and OPatchAuto

  Before you run OPatchAuto, find the OPatchAuto utility in the Oracle home and verify that you have the latest version. You must have the latest version of OPatch in all the homes of all the nodes before you request for patching.
- Obtaining Patches Required For Your Installation

  You can search for and download the latest patches for your installation from My Oracle Support.
- Configuring Node Manager to Support Start and Stop Operations

  To ensure that OPatchAuto can properly stop and start your system during patching, you must configure the Node Manager(s) to support the start and stop operations.
- Backup and Recovery Considerations for Patching

  It is highly recommended that you back up the Oracle home before any patch operation. You can back up the Oracle home using your preferred method.

## OPatchAuto Environment Variable

Before you run OPatchAuto, ensure that you set the required ORACLE_HOME environment variable. The ORACLE_HOME environment variable is used to identify the Oracle home you are planning to patch.

# Running OPatchAuto on a Single Node

> **Note:**
>
> - Latest OPatch must be present on all the homes requested for patching on all the nodes.

> **Note:**
>
> The following conditions apply only for the first node, such as when the session is first started on the cluster.

## Node Availability during Patching (Rolling vs. Non-rolling)

In order to start a new patching session, the following conditions must be met.

- The utility must be executed by an operating system (OS) user with root privileges. It must be executed on each node in the cluster if the GI home or Oracle RAC database home is in non-shared storage. The utility should not be run in parallel on the cluster nodes.
- Local node must be **up** for both rolling and non-rolling modes.
- At least one of the remote nodes must be **up** in order to start a rolling mode session.
- All the remote nodes must be **down** in order to start a non-rolling session.
- If the GI cluster is a flex setup, ensure that the first and last node where opatchauto is executed is a hub node and not a leaf node.

## OPatchAuto Apply\Rollback steps

Add the directory containing the opatchauto to the $PATH environment variable. For example:

```
# export PATH=$PATH:<GI_HOME>/OPatch
```

To patch the GI home and all Oracle RAC database homes of the same version:

```
# opatchauto apply <UNZIPPED_PATCH_LOCATION>/<Patch-id>
```

To patch only the GI home:

```
# opatchauto apply <UNZIPPED_PATCH_LOCATION>/<Patch-id> -oh <GI_HOME>
```

To patch one or more Oracle RAC database homes:

```
# export PATH=$PATH: <oracle_home1_path>/OPatch
# opatchauto apply <UNZIPPED_PATCH_LOCATION>/<Patch-id> -oh
      <oracle_home1_path>,<oracle_home2_path>
```

To roll back the patch from the GI home and each Oracle RAC database home:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION>/<Patch-id>
```

To roll back the patch from the GI home:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION>/<Patch-id> -oh <path to GI
home>
```

To roll back the patch from the Oracle RAC database home:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION>/<Patch-id> -oh
<oracle_home1_path>,<oracle_home2_path>
```

## Patching Session Output

The following patching session output examples illustrate successful OPatchAuto apply and rollback sessions.

**Example 5-1 OPatchAuto Apply/Rollback Session in Analyze Mode**

```
--------------------------------Summary--------------------------------
Analysis for applying patches has completed successfully:
Host:myhostq
CRS Home:/scratch/aime_ordb_myhostq/crso1/crshome_crso1
==Following patches were SUCCESSFULLY analyzed to be applied:
Patch: /tmp/patch_gipsu_12024/patch/22191349/21436941
Log: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/cfgtoollogs/opatchauto/core/
opatch/opatch2016-03-08_23-21-21PM_1.log
Patch: /tmp/patch_gipsu_12024/patch/22191349/21948341
Log: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/cfgtoollogs/opatchauto/core/
opatch/opatch2016-03-08_23-21-21PM_1.log
Patch: /tmp/patch_gipsu_12024/patch/22191349/21948344
Log: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/cfgtoollogs/opatchauto/core/
opatch/opatch2016-03-08_23-21-21PM_1.log
Patch: /tmp/patch_gipsu_12024/patch/22191349/21948354
Log: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/cfgtoollogs/opatchauto/core/
opatch/opatch2016-03-08_23-21-21PM_1.log

Host:myhostr
CRS Home:/scratch/aime_ordb_myhostq/crso1/crshome_crso1

==Following patches were SUCCESSFULLY analyzed to be applied:

Patch: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/OPatch/auto/dbtmp/
22191349/21436941
Log: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/cfgtoollogs/opatchauto/core/
opatch/opatch2016-03-08_23-24-56PM_1.log

Patch: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/OPatch/auto/dbtmp/
22191349/21948341

Log: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/cfgtoollogs/opatchauto/core/
opatch/opatch2016-03-08_23-24-56PM_1.log

Patch: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/OPatch/auto/dbtmp/
22191349/21948344

Log: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/cfgtoollogs/opatchauto/core/
opatch/opatch2016-03-08_23-24-56PM_1.log
```

```
Patch: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/OPatch/auto/dbtmp/
22191349/21948354

Log: /scratch/aime_ordb_myhostq/crso1/crshome_crso1/cfgtoollogs/opatchauto/core/
opatch/opatch2016-03-08_23-24-56PM_1.log

OPatchAuto successful.
```

### Example 5-2 OPatchAuto Apply Session

```
--------------------------------Summary--------------------------------
Patching is completed successfully. Please find the summary as follows:
Host:mymachineemq
CRS Home:/scratch/aime_ordb_mymachineemq/crso1/crshome_crso1
Summary:
==Following patches were SUCCESSFULLY applied:
Patch: /tmp/patch_gipsu_12024/patch/22191349/21436941
Log: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-03-08_23-41-38PM_1.log
Patch: /tmp/patch_gipsu_12024/patch/22191349/21948341
Log: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-03-08_23-41-38PM_1.log
Patch: /tmp/patch_gipsu_12024/patch/22191349/21948344
Log: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-03-08_23-41-38PM_1.log
Patch: /tmp/patch_gipsu_12024/patch/22191349/21948354
Log: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-03-08_23-41-38PM_1.log
Host:mymachineemr
CRS Home:/scratch/aime_ordb_mymachineemq/crso1/crshome_crso1
Summary:
==Following patches were SUCCESSFULLY applied:
Patch: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/OPatch/auto/dbtmp/
22191349/21436941
Log: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-03-08_23-59-15PM_1.log
Patch: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/OPatch/auto/dbtmp/
22191349/21948341
Log: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-03-08_23-59-15PM_1.log
Patch: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/OPatch/auto/dbtmp/
22191349/21948344
Log: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-03-08_23-59-15PM_1.log
Patch: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/OPatch/auto/dbtmp/
22191349/21948354
Log: /scratch/aime_ordb_mymachineemq/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-03-08_23-59-15PM_1.log
OPatchAuto successful.
```

### Example 5-3 OPatchAuto Rollback Session

```
--------------------------------Summary--------------------------------
Patching is completed successfully. Please find the summary as follows:
Host:mymachineemm
CRS Home:/scratch/aime_ordb_mymachineemm/crso1/crshome_crso1
Summary:
==Following patches were SUCCESSFULLY rolled back:
Patch: /tmp/patch_gipsu_12019/patch/22191492/17077442
Log: /scratch/aime_ordb_mymachineemm/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-03-08_19-25-46PM_1.log
```

```
Patch: /tmp/patch_gipsu_12019/patch/22191492/17303297
Log: /scratch/aime_ordb_mymachineemm/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-03-08_19-25-46PM_1.log
Patch: /tmp/patch_gipsu_12019/patch/22191492/21951844
Log: /scratch/aime_ordb_mymachineemm/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-03-08_19-25-46PM_1.log
OPatchAuto successful.
```

### Example 5-4 OPatchAuto Apply/Rollback Failure

```
--------------------------Patching Failed---------------------------------
Command execution failed during patching in home: /scratch/aime/app/aime/product/
11.2.0/dbhome_2, host: mymachineemg.
Command failed:  /bin/sh -c 'ORACLE_HOME=/scratch/aime/app/aime/product/
11.2.0/dbhome_2 /scratch/aime/app/aime/product/11.2.0/dbhome_2/bin/srvctl
stop home -o /scratch/aime/app/aime/product/11.2.0/dbhome_2
-n mymachineemg -f -t TRANSACTIONAL -s /
scratch/aime/app/aime/product/11.2.0/dbhome_2/cfgtoollogs/opatchautodb/statfile/
mymachineemg/OracleHome-eca39d53-5b51-4cdf-9c79-ce9d9312d86a_mymachineemg.stat'
Command failure output:
PRCH-1000 : Failed to stop resources running from Oracle home /scratch/aime/app/
aime/product/11.2.0/dbhome_2
PRCH-1029 : One or more resources failed to stop: PRCH-1006 : Failed to stop
Listener
PRCR-1014 : Failed to stop resource ora.LISTENER2.lsnr
PRCR-1065 : Failed to stop resource ora.LISTENER2.lsnr
CRS-5016: Process "/scratch/aime/app/aime/product/11.2.0/dbhome_2/bin/lsnrctl"
spawned by agent "/scratch/aime_ordb_mymachineemg/crso1/crshome_crso1/bin/
oraagent.bin" for action "stop" failed: details at "(:CLSN00010:)"
in "/scratch/aime_ordb_mymachineemg/crso1/crshome_crso1/log/mymachineemg/agent/
crsd/oraagent_aime/oraagent_aime.log"
CRS-2675: Stop of 'ora.LISTENER2.lsnr' on 'mymachineemg' failed

After fixing the cause of failure Run opatchauto resume with session id "J5A3"
]
OPATCHAUTO-68061: The orchestration engine failed.

OPATCHAUTO-68061: The orchestration engine failed with return code 1

OPATCHAUTO-68061: Check the log for more details.
```

## Sample Console Output

### Example 5-5 opatchauto apply -analyze

```
System initialization log file is /scratch/aime_ordb_mymachineelu/crso1/
crshome_crso1/cfgtoollogs/opatchautodb/systemconfig2016-05-05_01-55-58PM.log.

Session log file is /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/
cfgtoollogs/opatchauto/opatchauto2016-05-05_01-56-18PM.log

WARNING: the option -ocmrf is deprecated and no longer needed. OPatch no longer
checks for OCM configuration. It will be removed in a future release.

The id for this session is MDAN
[init:init] Executing OPatchAutoBinaryAction action on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1

Executing OPatch prereq operations to verify patch applicability on CRS
Home........
```

```
[init:init] OPatchAutoBinaryAction action completed on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1 successfully
[init:init] Executing GIRACPrereqAction action on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1

Executing prereq operations before applying on CRS Home........

[init:init] GIRACPrereqAction action completed on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1 successfully

OPatchAuto successful.

--------------------------------Summary--------------------------------
Analysis for applying patches has completed successfully:

Host:mymachineelu

CRS Home:/scratch/aime_ordb_mymachineelu/crso1/crshome_crso1

==Following patches were SUCCESSFULLY analyzed to be applied:

Patch: /tmp/patch_gipsu_12019/patch/22654153/17077442

Log: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-05-05_13-56-25PM_1.log

Patch: /tmp/patch_gipsu_12019/patch/22654153/17303297

Log: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-05-05_13-56-25PM_1.log

Patch: /tmp/patch_gipsu_12019/patch/22654153/22291141

Log: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-05-05_13-56-25PM_1.log
```

### Example 5-6 opatchatuo apply

```
System initialization log file is /scratch/aime_ordb_mymachineelu/crso1/
crshome_crso1/cfgtoollogs/opatchautodb/systemconfig2016-05-05_02-22-02PM.log.

Session log file is /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/
cfgtoollogs/opatchauto/opatchauto2016-05-05_02-22-19PM.log

WARNING: the option -ocmrf is deprecated and no longer needed. OPatch no longer
checks for OCM configuration. It will be removed in a future release.

The id for this session is WLR9

[init:init] Executing OPatchAutoBinaryAction action on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1

Executing OPatch prereq operations to verify patch applicability on CRS
Home........

[init:init] OPatchAutoBinaryAction action completed on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1 successfully

[init:init] Executing GIRACPrereqAction action on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1
```

```
Executing prereq operations before applying on CRS Home........

[init:init] GIRACPrereqAction action completed on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1 successfully

[shutdown:shutdown] Executing GIShutDownAction action on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1

Performing prepatch operations on CRS Home........

Prepatch operation log
file location: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/
crsconfig/crspatch_mymachineelu_2016-05-05_02-22-52PM.log

[shutdown:shutdown] GIShutDownAction action completed on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1 successfully

[offline:binary-patching] Executing OPatchAutoBinaryAction action on home /
scratch/aime_ordb_mymachineelu/crso1/crshome_crso1

Start applying binary patches on CRS Home........

[offline:binary-patching] OPatchAutoBinaryAction action completed on home /
scratch/aime_ordb_mymachineelu/crso1/crshome_crso1 successfully

[startup:startup] Executing GIStartupAction action on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1

Performing postpatch operations on CRS Home........

Postpatch operation log
file location: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/
crsconfig/crspatch_mymachineelu_2016-05-05_02-27-03PM.log

[startup:startup] GIStartupAction action completed on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1 successfully

[finalize:finalize] Executing OracleHomeLSInventoryGrepAction action on home /
scratch/aime_ordb_mymachineelu/crso1/crshome_crso1

Verifying patches applied on CRS Home.
[finalize:finalize] OracleHomeLSInventoryGrepAction action completed on home /
scratch/aime_ordb_mymachineelu/crso1/crshome_crso1 successfully
OPatchAuto successful.

--------------------------------Summary--------------------------------
 Patching is completed successfully. Please find the summary as follows:

Host:mymachineelu
CRS Home:/scratch/aime_ordb_mymachineelu/crso1/crshome_crso1
Summary:

==Following patches were SUCCESSFULLY applied:

Patch: /tmp/patch_gipsu_12019/patch/22654153/17077442

Log: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-05-05_14-23-38PM_1.log

Patch: /tmp/patch_gipsu_12019/patch/22654153/17303297
```

```
Log: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-05-05_14-23-38PM_1.log

Patch: /tmp/patch_gipsu_12019/patch/22654153/22291141

Log: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-05-05_14-23-38PM_1.log
```

### Example 5-7 opatchauto rollback

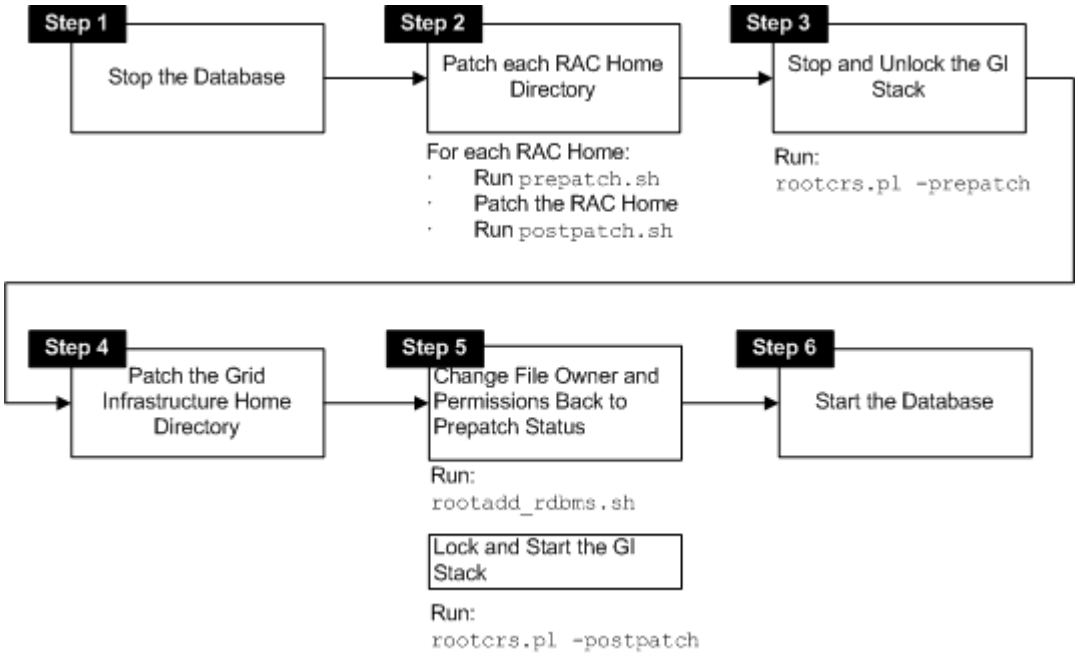
```
System initialization log file is /scratch/aime_ordb_mymachineelu/crso1/
crshome_crso1/cfgtoollogs/opatchautodb/systemconfig2016-05-05_04-34-39PM.log.

Session log file is /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/
cfgtoollogs/opatchauto/opatchauto2016-05-05_04-35-00PM.log
The id for this session is K5BA
[init:init] Executing OPatchAutoBinaryAction action on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1

Executing OPatch prereq operations to verify patch applicability on CRS
Home........

[init:init] OPatchAutoBinaryAction action completed on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1 successfully

[init:init] Executing GIRACPrereqAction action on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1

Executing prereq operations before rolling back on CRS Home........

[init:init] GIRACPrereqAction action completed on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1 successfully

[shutdown:shutdown] Executing GIShutDownAction action on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1

Performing prepatch operations on CRS Home........

Prepatch operation log
file location: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/
crsconfig/crspatch_mymachineelu_2016-05-05_04-35-22PM.log

[shutdown:shutdown] GIShutDownAction action completed on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1 successfull
[offline:binary-patching] Executing OPatchAutoBinaryAction action on home /
scratch/aime_ordb_mymachineelu/crso1/crshome_crso1

Start rolling back binary patches on CRS Home........

[offline:binary-patching] OPatchAutoBinaryAction action completed on home /
scratch/aime_ordb_mymachineelu/crso1/crshome_crso1 successfully
[startup:startup] Executing GIStartupAction action on home /scratch/
aime_ordb_mymachineelu/crso1/crshome_crso1

Performing postpatch operations on CRS Home........

Postpatch operation log
file location: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/
crsconfig/crspatch_mymachineelu_2016-05-05_04-38-59PM.log

[startup:startup] GIStartupAction action completed on home /scratch/
```

```
aime_ordb_mymachineelu/crso1/crshome_crso1 successfully
OPatchAuto successful.

--------------------------------Summary--------------------------------

Patching is completed successfully. Please find the summary as follows:

Host:mymachineelu
CRS Home:/scratch/aime_ordb_mymachineelu/crso1/crshome_crso1
Summary:

==Following patches were SUCCESSFULLY rolled back:

Patch: /tmp/patch_gipsu_12019/patch/22654153/17077442
Log: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-05-05_16-36-04PM_1.log

Patch: /tmp/patch_gipsu_12019/patch/22654153/17303297
Log: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-05-05_16-36-04PM_1.log

Patch: /tmp/patch_gipsu_12019/patch/22654153/22291141
Log: /scratch/aime_ordb_mymachineelu/crso1/crshome_crso1/cfgtoollogs/opatchauto/
core/opatch/opatch2016-05-05_16-36-04PM_1.log
```

# OPatchAuto Apply

When you run OPatchAuto's apply command, numerous operations are performed to implement the complete patch application cycle. These operations vary depending on the environment to be patched. The following environment is representative of the vast majority of patching environments in which OPatchAuto is used. For example, a typical patching environment would be one GI Home managing two RAC Homes. When you run `opatchauto apply`, OPatchAuto will perform the operations shown in Figure 5-1.

**Figure 5-1 Patching with OPatchAuto: Process Flow**



# OPatchAuto: System Reboot Request

Depending on the patch or the home directory configuration, you may encounter a request to reboot the system. After a reboot during the patching process, you need to invoke the `opatchauto` utility again so that it seamlessly continues with rest of the patch application process.

Typically an error message, as shown in the following example, will be displayed when a problem arises.

**Example 5-8 OPatchAuto Console Error**

```
# OPatch/opatchauto apply /scratch/aime/sh/RDBMS_12.1.0.1.0_LINUX.X64_130418/
patches/v2/nosql/gipsu/11111111 -ocmrf /tmp/ocm.rsp
OPatch Automation Tool
Copyright (c) 2013, Oracle Corporation.  All rights reserved.
 ...

CLSRSC-400: A system reboot is required to continue installing.

...

Apply Summary:
Following patch(es) are successfully installed:
GI_HOME=/u01/GI12/app/12.1.0/grid:13852018, 22222222, 123456788
DB_HOME=/scratch/aime/DB12_2/app/aime/product/12.1.0/dbhome_1:13852018, 123456788
DB_HOME=/scratch/aime1/DB12N/app/aime1/product/12.1.0/dbhome_1:13852018,
123456788

opatchauto failed with error code 1.
```

When you receive an error like this, follow the reboot instructions specified in the console. The following example shows a system reboot request issued by the user.

**Example 5-9 Rebooting the System**

```
# OPatch/opatchauto resume -reboot
OPatch Automation Tool
Copyright (c) 2013, Oracle Corporation.  All rights reserved.


OPatchauto version : 12.1.0.1.1
OUI version        : 12.1.0.1.0
Running from       : /u01/GI12/app/12.1.0/grid
Log file location  : /u01/GI12/app/12.1.0/grid/cfgtoollogs/opatch/
opatch2013-05-16_13-36-59PM_1.log

Opatchauto will attempt to resume from reboot patching session. This might take
several minutes...

Command "/usr/bin/perl /u01/GI12/app/12.1.0/grid/crs/install/rootcrs.pl -
postpatch" is successfully resumed.
Command "/scratch/aime1/DB12N/app/aime1/product/12.1.0/dbhome_1/bin/srvctl
start home -o /scratch/aime1/DB12N/app/aime1/product/12.1.0/dbhome_1
-n slc00epi -s /scratch/aime1/DB12N/app/aime1/product/12.1.0/dbhome_1/
OracleHome-50b8f1a0-e220-4b8e-98d7-49177979991f.stat " is successfully resumed.
Command "/scratch/aime/DB12_2/app/aime/product/12.1.0/dbhome_1/bin/srvctl
start home -o /scratch/aime/DB12_2/app/aime/product/12.1.0/dbhome_1
-n slc00epi -s /scratch/aime/DB12_2/app/aime/product/12.1.0/dbhome_1/
```

```
OracleHome-58232a10-3130-4930-b588-0c8594cf8c87.stat " is successfully resumed.
Opatchauto was able to resume from the previous reboot patching session and
complete successfully.

opatchauto succeeded.
```

# Using OPatchAuto to Patch a GI/RAC Environment

Applying a patch with OPatchAuto involves a series of steps that must be performed to ensure successful patching.

The following table summarizes the typical steps required to patch your existing GI/RAC environment using OPatchAuto.

**Table 5-1 Using OPatchAuto**

| Task | Description | Documentation |
|---|---|---|
| Acquire patches required for your installation | Log in, search for, and download the patches required for your specific installation.<br>You do not need to worry about whether OPatchAuto supports a particular patch type. If OPatchAuto does not support a particular patch type, you will be notified when you run the tool. | Obtaining the Patches You Need |
| Review the README.txt file for the patch. | Each patch archive includes a README file that contains important information and instructions that must be followed prior to applying your patch. It is important to review the README file because it provides any unique steps or other information specific to the patch. | The README.txt file that is packaged within the patch archive. |
| Check for patch prerequisites. | The OPatchAuto apply -analyze command will identify that the prerequisites for the patch have been met. | If you are patching a single host environment, see Verifying the Prerequisites for Applying a Patch on a Single Host.<br>If you are patching a multi-host environment, see Verifying the Prerequisites for Applying a Patch on Multiple Hosts. |
| Apply the patch. | After you determine the Oracle home to which you need to apply the patch, and you have read the README file, then you should apply the patch with the opatchauto apply command. | If you are patching a multi-host environment, see Applying a Patch on Multiple Hosts Using the Apply Command |
| Verify the patch was applied to the Oracle home successfully. | The OPatch lsinventory command will show what patches have been applied to the Oracle home. . | Using the OPatch lsinventory Command to Verify the Patches Applied to an Oracle Home |

**Table 5-1 (Cont.) Using OPatchAuto**

| Task | Description | Documentation |
|---|---|---|
| Verify that your software runs properly after you apply the patch. | After the patching is complete and your servers are restarted, you should check your product software to verify that the issue has been resolved. | Verifying Your Installation After Applying a Patch |
| Troubleshoot the application of a patch. | If there are problems applying a patch, your first troubleshooting task is to review the log file for the OPatchAuto session. | Troubleshooting a Patch by Viewing the OPatchAuto Log File |
| Roll back the application of a patch | If for some reason the result is not satisfactory, you can use the `opatchauto rollback` command to remove the patch from the Oracle home.<br>If additional assistance is required, go to My Oracle Support (formerly OracleMetaLink). | For a single host environment, see Rolling Back a Patch You Have Applied on a Single Host.<br>For a multi-host environment, see Rolling Back a Patch You Have Applied on Multiple Hosts. |

# Patching a Sharded Database

The session has to be initiated from the catalog host by providing details of the catalog database. It should provide a top level view to the end user about patching a SDB, making it easier to understanding the flow of opatchautoSDB.

Sharding is an application-managed scaling technique using many (hundreds / thousands of) independent databases. With sharding, data is split into multiple databases (shards) with each database holding a subset of data. Shards can be replicated for high availability and scalability.

OPatchAuto supports end-to-end patching of sharded databases across multiple regions, along with the grid infrastructure (GI) that supports the clustered databases/ shards and shards that are managed by Oracle Golden Gate or by Oracle DataGuard.

OPatchAuto supports all sharding and replication methods.

**Supported Configurations:**

- Shard Types: Databases running on GI/SIHA and standalone databases
- Different versions across Data Guards (currently, only version 12.2 is available)
- Multiple shards sharing the same cluster/grid
- Multiple versions of shards in an OGG replicated SDB

## Selectively Patching Subset Entities

A sharded database can span multiple regions and clusters. To manage the patching cycle more efficiently, you may want to scale down the patching effort by patching only specific subset entities. OPatchAuto provides three options for selecting the subset entities:

1. Data Guards

2. Shard Groups

3. Shard Spaces

You can select instances of any one of the above entities. Note: You can only specify a single entity for a given patching cycle. For all of the subset entities, OPatchAuto identifies the targets on the basis of their names:

1. –dg: Comma-separated list of names of the primary database

2. –shardgroup: Comma-separated list of names of the shard groups

3. –shardspace: Comma-separated list of names of the shard spaces

For all of the above options, the Grid homes of the databases will also be patched. The CRS/RAC homes will be patched across all their nodes in a rolling manner. Similarly, the Golden Gate home will also be patched.

## Data Guard

By selecting the `-dg <primaryDB name>` option, only the databases (along with their cluster homes) that belong to the selected Data Guard will be patched. All these databases will be patched in a rolling manner starting with the standby shards and ending with the primary shard.

The entire list of databases in a sharded database can be collected from the catalog database table GSMADMIN_INTERNAL.DATABASE under the column NAME.

## Shard Group

By selecting the `-shardgroup <shardgroup name>` option, the databases of the selected shard group will be patched in a rolling manner. In a Data Guard-replicated configuration, a shard group can host ONLY one member of each Data Guard. Hence, this option is NOT supported in a sharded database that employs Data Guard replication because an entire Data Guard, while being spread across multiple shard groups, needs to be patched together in a defined sequence starting with standby databases and ending with the primary database. Patching individual shard groups poses a major risk of breaking that sequence.

The entire list of shard groups in a sharded database can be collected from the catalog database table GSMADMIN_INTERNAL.SHARD_GROUP under the column NAME.

## Shard Space

By specifying the `-shardspace <shardspace name>` option, the databases of the entire shard space are patched in a rolling manner. These databases could be spread across multiple shard groups, depending upon the configuration of the sharded database.

The entire list of shard spaces in a sharded database can be collected from the catalog database table GSMADMIN_INTERNAL.SHARD_SPACE under the column NAME.

# Sharded Database Command Option

The OPatchAuto *sdb* command option allows you to patch sharded databases. This operation patches all the shards of the Sharded database.

In a sharded database with Data Guard replication, patching a Data Guard involves patching all its standby databases first, followed by its primary database.In Oracle Golden Gate based replication, for a user-defined configuration, the operation involves patching around the shard spaces. For system-managed and composite configuration, the operation involves patching around the shard groups.The values of `-host`, `-port` and `-sid` are used to connect to the catalog database and hence they should form the required connect string for the catalog database.All the database(s) are patched first.Thereafter, the Grid/HAS homes that are host to any of the databases belonging to the sharded database are patched.The catalog database and the GSM's will not be patched. In order to patch these, OPatchAuto needs to be run separately on these databases.

The following syntax illustrates command line usage:

```
opatchauto apply <patch-location>
    -sdb
    -wallet <wallet>
    [ -phBaseDir <patch.base.directory> ]
    [ -logLevel <log_priority> ]
    [ -analyze ]
    [ -host <tns-host> ]
    [ -dg <primary.database.name> ]
    [ -shardgroup <shardgroup> ]
    [ -shardspace <shardspace> ]
    [ -rolling ]
    [ -service <service> ]
    [ -inplace ]
    [ -sid <sid> ]
    [ -port <port> ]
```

**Parameters**

- patch-location

  The patch location.

**Options**

-

- **phBaseDir** <patch.base.directory>

  The location of base patch directory.

- **logLevel** <log_priority>

  The log level (defaults to "INFO").

  Supported values: OFF, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, ALL

- **analyze**

  If this option is selected, the environment will be analysed for suitability of the patch on each home, without affecting the home.

The patch will not be applied or rolled back, and targets will not be shut down.

- **host** <tns-host>

  The tns-host of the catalog database. This should match the 'HOST' used in the network configuration file of the catalog database. The default host is set as the local 'hostname' without appending the domain.

- **wallet** <wallet location> (Refer to Creating Wallet Using OPatchAuto Wallet Tool)

  The location of the wallet file.

  The entries made in the wallet file must satisfy these requirements:

  – It is mandatory to provide the credentials of all home owners of every node that would be patched as part of the session in the wallet. The homes that would be patched on the node also includes GI/SIHA when installed. The list of nodes refers to all hostnames/ip-address listed in GSMADMIN_INTERNAL.SHA_DATABASES.DB_HOST column of the Sharded database catalog.

  – It must also contain the credentials for the database user with 'sysdba' privilege for the catalog database.

  Additionally, the host user provided in the wallet must meet the following requirements:

  – The user must be able to change to root using ‘sudo’ if the node belongs to GI/SIHA environment.

- **dg <primary.database.name>**

  This is used to restrict patching to databases of the selected dataguard. All the standby databases of the dataguard are patched first, followed by its primary database.

- **shardgroup <shardgroup>**

  This is used to restrict patching to databases of the selected shard group.

- **shardspace <shardspace>**

  This is used to restrict patching to databases of the selected shard space.

- **sdb** (Required)

  To signify patching sharded database. Run 'opatchauto <apply|rollback> -sdb -help' to get more help on patching a sharded database.

- **rolling**

  Enables sdb rolling mode where database(s) are patched one after the other.

- **service <service>**

  Service name of the catalog database.

- **inplace**

  This option can be used to perform in place patching through opatchauto. Here opatchauto performs the patching operation on the original Oracle Home, so there will be down time and high availability of services will get affected. The default patching mode from opatchauto is inplace.

- **sid <sid>**

  This option can be used for both Shard patching as well as standalone SIDB patching. In context to standalone SIDB patching it will take the instance name

of the database as its value. In context to shard patching it will take the catalog database name. If -service or -sid is not provided in the commandline, the default sid is used from the environment variable 'ORACLE_SID'.

- **port <port>**

  This signifies the port for connecting to the catalog database. The default port is set as'1521'.

The following examples demonstrate how to use the various OPatchAuto command options when patching a sharded database.

**Example 5-10 Patching a Sharded Database**

```
<CATALOG_DB_HOME>/OPatch/opatchauto apply <patch location> -sdb -wallet <wallet
file location> -sid <sid of catalog db> -port <sid configured port>
```

**Example 5-11 Patching a Data Guard in Sharded Database**

```
<CATALOG_DB_HOME>/OPatch/opatchauto apply <patch location> -sdb -dg
<primary_database_name1,primary_database_name2,...> -wallet <wallet file
location> -sid <sid of catalog db> -port <sid configured port>
```

**Example 5-12 Patching a Shardgroup in a Sharded Database**

```
<CATALOG_DB_HOME>/OPatch/opatchauto apply <patch location> -sdb -shardgroup
<shardgroup_name1,shardgroup_name2,...> -wallet <wallet file location> -sid <sid
of catalog db> -port <sid configured port>
```

**Example 5-13 Patching a Shardspace in a Sharded Database**

```
<CATALOG_DB_HOME>/OPatch/opatchauto apply <patch location> -sdb -shardspace
<shardspace_name1,shardspace_name2,...> -wallet <wallet file location> -sid <sid
of catalog db> -port <sid configured port>
```

**Example 5-14 Listing Wallet Content**

```
<CATALOG_DB_HOME>/OPatch/auto/core/bin/patchingWallet.sh -walletDir
<wallet.location> -list
```

**Example 5-15 Adding a New Host Credential Entry to the Wallet**

```
<CATALOG_DB_HOME>/OPatch/auto/core/bin/patchingWallet.sh -walletDir <wallet
location> -create <username>:<hostname>:ssh
<CATALOG_DB_HOME>/OPatch/auto/core/bin/patchingWallet.sh -walletDir <wallet
location> -create oracle:myhost:ssh
<CATALOG_DB_HOME>/OPatch/auto/core/bin/patchingWallet.sh -walletDir <wallet
location> -create oracle:127.50.50.50:ssh
```

**Example 5-16 Adding a New Catalog Database Credential Entry to the Wallet**

```
<CATALOG_DB_HOME>/OPatch/auto/core/bin/patchingWallet.sh -walletDir <wallet
location> -create <username>:<sid of catalog db>:jdbc
```

## About the Wallet File

Sharded database patching requires credentials to access the targets. The entries made in the wallet file must satisfy the following requirements:

- The wallet file must contain credentials for each home owner in the entire sharding setup. This includes all the shard homes as well as the GI home owners.

- It must also contain the credentials for the database user with ‘sysdba’ privilege for the catalog database.
- The GI home owner must have a privilege to run commands as root using sudo.

**Oracle Wallet for Credential Input**

OPatchAuto accepts credentials, in Oracle wallet format, for accessing run-time entities, such as databases and Admin Servers. A Wallet file contains credentials for the hosts which are part of the cluster that requires patching. You input a wallet on the command line; if you do not supply one, and OPatchAuto needs one, it will prompt you for one on the command line. Successful usage depends on the user possessing both the wallet and the wallet password.

## Creating Wallet Using OPatchAuto Wallet Tool

You can use the command line OPatchAuto wallet tool to generate a wallet file. The wallet file contains credentials for the hosts that are part of the cluster which needs to be patched. The wallet tool works seamlessly during the OPatchAuto patch orchestration process by passing the wallet file path as parameter during patching operations.

The command line tool is as follows:

```
<ORACLE_HOME>/OPatch/auto/core/bin/patchingWallet.[sh|cmd]
[-log log_file] [-log_priority log_priority]
{ -create | -delete | -list } alias1 alias2 ...
```

**Table 5-2 `patchingWallet` Command Options**

| Option | Description |
|---|---|
| -create | (Required) Create secrets for each alias given on the command line. If a given alias already exists in the wallet, its secret is overwritten without warning. |
| -delete | (Required) Delete given aliases from the wallet. Aliases that do not exist in the wallet are ignored. |
| -list | (Required) List aliases defined in the wallet. The secrets associated with the aliases are not displayed. The `alias` command line arguments are ignored. |
| -walletDir | (Optional) The path to the wallet directory. If omitted, the default location, if defined, will be used. If wallet does not exist at the specified location, it will be created when then `-create` option is used. |
| -useStdin | (Optional) When creating aliases, specifies that the passwords should be read from STDIN rather than the console device. Passwords will be read in the order specified by the alias options, with one per line. There will be no prompt. |
| -log | (Optional) Name of the log file. |

**Table 5-2 (Cont.) `patchingWallet` Command Options**

| Option | Description |
|---|---|
| -log_priority | (Optional) The priority setting for the log file. Use a Java Logging Level string or a log4j priority string.<br>Valid Java logging values are off, severe, info, warning, config, fine, finer, finest, and all.<br>Valid log4j priority strings are *debug*, *info*, *warn*, *error*, and *fatal*. The priority string values correspond to the levels defined in the Level class.<br>More information about log4j priority strings can be found at the following Web site: http://logging.apache.org/log4j/docs/api/org/apache/log4j/Level.html. |

## Log Files

Summary on the console shows the location of the following log files that can be accessed for further details about the patching process and troubleshooting:

- Main log file of the sharding session
- Log file for each shard group
- Log file for each dataguard/shard space
- Log file for each individual database
- Log file for each individual grid home

In addition to appearing in the console, these log files will be available on their respective nodes.

# 6
# OPatchAuto -binary

*OPatchAuto -binary* is a tool that applies multiple patches on a selected Oracle home. It can patch only one Oracle home per session. The utility internally performs the prerequisite checks before applying the bits.

**Operational Characteristics**

- *OPatchAuto -binary* assumes that the targets have been already shutdown.
- *OPatchAuto -binary* does not perform any other operation apart from applying the binary bits to the Oracle home.

## Supported Patch Types

*OPatchAuto -binary* can apply multiple patches per session. The selected patches can be any of the following types/configurations:

- One ore more one-off patches
- One composite patch
- One system patch / bundle patch
- One system patch / bundle patch and 1 or more one-off patches
- One system patch / bundle patch which has a composite patch in it
- One system patch / bundle patch which has a composite patch in it and 1 or more one-off patches
- One system patch / bundle patch and 1 or more one-off patches and 1 composite patch
- Multiple composite patches

Note that while OPatchAuto can apply all of the aforementioned patch types, OPatch cannot apply system patches. (OPatch can apply one off and composite patches.) So system patch application is the distinctive value for *opatchauto -binary*.

## Analyze Mode of Operation

When running in *analyze* mode, *OPatchAuto -binary* runs the prerequisite checks on the patches but does not apply the bits. To run *OPatchAuto -binary* in analyze mode, use the `-analyze` command line option. See "Syntax and Commands" for more information.

## Target Types in a SystemPatch

A SystemPatch contains a list of patches inside it and, for each of the patches, a list of applicable target types is defined. By default, *OPatchAuto -binary* applies all the patches inside the SystemPatch to the selected Oracle home. You can specify a

particular target type to have *OPatchAuto -binary* apply only the applicable patches to the selected home.

# Syntax and Commands

The command line syntax for the *opatchauto -binary* utility is as follows.

```
opatchauto [ apply | rollback ] <patch_location> -binary -oh <oracle_home path>
[-phBaseDir <Patch_Base_Directory>]
    [-target_type <target_type>]
    [-id <patchID1, patchID2,..> ]
    [-analyze]
    [-invPtrLoc <oraInst.loc> ]
    [-jre <JRE Location> ]
```

| Option | Description |
|---|---|
| <patch_location> | Path to the location of the Patch. In case of a SystemPatch, this path should contain the *bundle.xml* file. |
| phBaseDir | The location of base patch directory. |
| oh | Path to the Oracle home to be patched. |
| id | Patch IDs that can be specified for rollback sessions. The values will be ignored for apply session. |
| target_type | Specifies the type of the Oracle Home. These are predefined values for every target type. |
| analyze | Run only the prerequisite checks and also apply or roll back the patch in *report mode* (patch application/rollback is not actually carried out). The prerequisite checks are performed, thereby reporting the feasibility of the patch session. |
| invPtrLoc | Used to locate the *oraInst.loc* file when the installation used the `-invPtrLoc` option. This should be the path to the *oraInst.loc* file. |
| jre | Instructs OPatchAuto to use the JRE (Java) from the specified location instead of the default location under Oracle Home. |
| force_conflict | If a conflict exist which prevents the patch from being applied, the -force_conflict flag can be used to force application of the patch. OPatch will remove all the conflicting patches before applying the current patch. It is applicable to only apply operation. |

**Example 6-1 *OPatchAuto -binary* Usage**

```
opatchauto apply <UNZIPPED_PATCH_LOCATION>/<PATCH_ID> -binary  -target_type
cluster -oh <GI_HOME>
```

7

# Out-of-Place Patching

Out-of-place patching, which is based on cloning, is a patching method where patching is performed by creating a copy of oracle home, applying patches to the copied home, and then switching services to the copied home. This patching method helps in reducing the downtime of switching of the services from the existing home to the updated home.

Following are the important operations performed in out-of-place patching:

1. Prepare clone home - A copy of the oracle home is created and patches are applied to the copied home.
2. Switch to clone home - After the copied home is ready, the required services are switched to the copied home.
3. Switch back to original home - When required, the services can be switched back to the original home.

OPatchauto supports out-of-place patching for the following configurations:

- GI RAC
- SIHA SIDB

## GI RAC

Following are the supported sub-configurations of Grid Infrastructure (GI) Real Application Cluster (RAC):

- Non shared GI, non shared RAC
- Shared GI, non shared RAC
- Shared GI, shared RAC
- Non shared GI, shared RAC
- RAC one database
- GI only
- Single RAC home or multiple RAC homes
- Software-only homes

> **Note:**
>
> - This feature is supported from RDBMS version 12.2 and later.
> - For the software-only homes sub-configuration, only the prepare feature is supported. The switch to new home or rollback to original home is not supported.

# Out of Place Patching Method

In the Out of Place Patching method a clone home is created and then the original home is switched to the clone home.

> **Note:**
>
> The Out of Place Patching Method (with explicit outofplace option) is applicable for DB 12.2 to DB 18.

To apply the patches, run the following command:

```
${ORACLE_HOME}/OPatch/opatchauto apply ${PATCH_HOME} -outofplace
```

or

```
${ORACLE_HOME}/OPatch/opatchauto apply ${PATCH_HOME} -outofplace –oh oh1,
oh2
```

Where, *PATCH_HOME* is the location of the patch directory that contains the patch.

The apply session is an active session for all the homes, which are involved in patching, until the session is not completed on all the nodes.

At this stage, OPatchauto generates a default path for the cloned home on the first node and requests you for a confirmation through the interactive mode. You can continue with the default path or choose a custom path for each of the home that needs to be patched.

If you choose a custom path, then ensure that the provided directory path has the required write-permission and is not an existing directory across all the nodes. The custom path is validated by OPatchauto for all the nodes for existence. For all the nodes, OPatchauto finalizes the custom path for the entire OPatchauto OOP session and you would not be prompted again for confirming the path for the subsequent nodes. Once the paths are finalized, if the same path is found in a node (after the first node) then the OOP operation fails and you must delete the directory for the OOP operation to proceed.

You can also use the silent mode, where the interactive option is not displayed. You need to provide a property file containing a map of the original homes to cloned homes. The custom path is validated by OPatchauto for all the nodes for existence. If the path already exists on any node or a path is missing for any of the nodes, then the session fails and you must provide a new path which can be created on all the nodes.

To create a property file, add all the required original and cloned home details and save the file with the *.properties* extension. For example: clone.properties.

Following is an example of the property file:

```
/scratch/product/12.2/<orginal_home>=/scratch/product/12.2/<cloned_home>
/scratch/product/db/12.2/<orginal_home>=/scratch/product/db/12.2/<cloned_home>
```

Where,

<*original_home*> is the folder name of the original home.

*<cloned_home>* is the folder name of the clone home to which the original home would be cloned to.

Following are other applicable command-line options:

```
[ -phBaseDir <patch.base.directory> ]
[ -log <log> ]
[ -logLevel <log_priority> ]
[ -analyze ]
[ -invPtrLoc <inventory.pointer.location> ]
[ -force_conflict ]
[ -database <database> ]
[ -nonrolling ]
[ -silent <map.original.to.clone>]
```

> **Note:**
>
> The database switching and SQL operation will be performed only on the last node.

## Switchback to original home

To rollback the applied patches, you need to switch back to original home.

To switch back to the original home, run OPatchauto from the cloned home. OPatchauto switches back all the cloned homes to original homes, which were switched earlier when applying the patch.

To switch back to original home, run the following command:

```
${ORACLE_HOME}/OPatch/opatchauto rollback -switch-clone
```

or

```
${ORACLE_HOME}/OPatch/opatchauto rollback -switch-clone –oh oh1, oh2
```

> **Note:**
>
> - OOP works *only* with the new home and the last home.
> - If you want to switchback to a previous oracle home instance, an instance before applying any of the previous patches, then use the inplace patching option.
> - Do not modify anything in the original home before switchback. After creating the clone home, if there are any changes made to the original home, then OPatchauto switchback would not be allowed.
> - The database switching and SQL operation will be performed only on the last node.

# SIHA SIDB

The SIHA SIDB configuration and procedures are similar to the GI RAC configuration. For information about how apply patches, see GI RAC.

`-nonrolling` option is not applicable to the SIHA SIDB configuration.

# 8
# Inplace Patching

For RDBMS version prior to 19.3, by default it is Inplace Patching. For RDBMS version starting 19.3, user needs to pass Inplace explicitly.

## OPatchAuto Apply\Rollback steps

Add the directory containing the opatchauto to the $PATH environment variable. For example:

```
# export PATH=$PATH:<GI_HOME>/OPatch
```

To patch the GI home and all Oracle RAC database homes of the same version:

```
# opatchauto apply <UNZIPPED_PATCH_LOCATION>/<Patch-id> -inplace
```

To patch only the GI home:

```
# opatchauto apply <UNZIPPED_PATCH_LOCATION>/<Patch-id> -oh <GI_HOME> -
inplace
```

To patch one or more Oracle RAC database homes:

```
# export PATH=$PATH: <oracle_home1_path>/OPatch
# opatchauto apply <UNZIPPED_PATCH_LOCATION>/<Patch-id> -oh
      <oracle_home1_path>,<oracle_home2_path> -inplace
```

To roll back the patch from the GI home and each Oracle RAC database home:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION>/<Patch-id> -inplace
```

To roll back the patch from the GI home:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION>/<Patch-id> -oh <path to GI
home> -inplace
```

To roll back the patch from the Oracle RAC database home:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION>/<Patch-id> -oh
<oracle_home1_path>,<oracle_home2_path> -inplace
```

# 9
# Troubleshooting OPatch

This chapter describes common problems with patching and troubleshooting with OPatch. The following sections are discussed:

- Debugging: Enable Logging and Tracing
- References
- Products and Patch Types Not Supported by OPatch

## Debugging: Enable Logging and Tracing

*Logging* and *tracing* is a common aid for debugging. OPatch maintains logs for all `apply`, `rollback`, and `lsinventory` operations. The log files are located at the following directory:

```
<ORACLE_HOME>/cfgtoollogs/opatch
```

Each log file is tagged with the timestamp of the operation. Log files are named as opatch_<date mm-dd-yyyy>_<time hh-mm-ss>.log.

> **Note:**
>
> A new log file is created each time OPatch is executed.

For example, if a log file is created on May 17th, 2013 at 11.55 PM, it will be named as follows:

```
opatch_05-17-2013_23-55-00.log
```

> **Note:**
>
> You can set OPatch to debug mode by setting the environment variable OPATCH_DEBUG to TRUE.

OPatch also maintains an index of the commands executed with OPatch and the log files associated with it in the `opatch_history.txt` file located in the <ORACLE_HOME>/cfgtoollogs/opatch directory. A sample of the `history.txt` file is as follows:

```
Date & Time : Tue Apr 26 23:00:55 PDT 2013
Oracle Home : /private/oracle/product/11.2.0/db_1/
OPatch Ver. : 12.1.0.1.2
Current Dir : /scratch/oui/OPatch
Command     : lsinventory
Log File    :
```

```
/private/oracle/product/11.2.0/db_1/cfgtoollogs/opatch/
opatch-2013_Apr_26_23-00-55-PDT_Tue.log
```

# References

This section to contain a list of other documentation that may provide support. The main link will point to the "Master Note for OPatch" (Doc ID 293369.1) in MOS.

Patch Set FAQ: See Patchset FAQ (Doc ID 552777.1) for a complete summary of patchset FAQs. This document is available in My Oracle Support.

Information Center: Patching and Maintaining Oracle Database Server/Client Installations (Doc ID 1351428.1 )

# Products and Patch Types Not Supported by OPatch

Oracle now offers a wide range of products, including hardware and operating systems. However, patches required by some products are not currently supported by OPatch.

> **Note:**
>
> If the Oracle product you install (such as Oracle Database or Fusion Middleware) creates an Oracle home directory, then OPatch is provided as part of that installation.

Typically, the types of Oracle products not supported by OPatch include:

- Hardware (for example, firmware updates for Sun servers).
- Operating system (for example, kernel patch updates for Oracle Linux or Oracle Solaris.)
- Java (for example, patch updates for JRE, JDK)
- Software products that do not create an Oracle home directory (for example, Oracle OpenOffice)

10

# Troubleshooting OPatchAuto

This chapter describes common OPatchAuto problems that may occur during usage.

This chapter covers the following:



## OPatchAuto Troubleshooting Architecture

In order for OPatchAuto to fully automate the patching process, it accesses various tools/utilities to carry out different patching phases. The four primary tools/utilities are:

- **OPatch** - Applies patches to product (e.g., Fusion Middleware) homes.
- **rootcrs** - Controls GI Home access by unlocking files so they are patchable, as well as stopping and starting the GI stack.
- **patchgen** - Records the patch level.
- **datapatch** - Applies SQL changes to database instances.

These tools/utilities are accessed during the patching process. Troubleshooting OPatchAuto, therefore, involves diagnosing issues with the individual tools.

## OPatchAuto (Use Cases)

When using OPatchAuto, problems may arise where it is not clear as to how to proceed with the resolution. The following use cases illustrate common patching scenarios where you may encounter such problems and general procedures you can use to resolve the problems.

### OPatch Fails

See Troubleshooting OPatch for more information.

### Rootcrs.pl

The rootcrs.pl script performs the operations necessary to configure the Grid Infrastructure stack on a cluster. During an OPatchAuto session, you may encounter errors stemming from the rootcrs.pl script.

## Rootcrs.pl Prepatch

Command issued by OPatchAuto: `$GRID_HOME/crs/rootcrs.pl -prepatch`

If rootcrs.pl fails, error codes and their associated messages will be generated, as shown in the following example

```
CRS-1159: The cluster cannot be set to rolling patch mode because Grid
Infrastructure is not active on at least one remote node.
```

If the message is not clear, you can obtain additional help by running the **OERR** utility to obtain cause and recommended action information.

### Running OERR

Running OERR for a specific error code will generate both the cause and action for the specified error code.

**Example 10-1 CRS Error**

```
$GRID_HOME/bin/oerr crs 1159

Cause: The cluster could not be set to rolling patch mode because Grid
Infrastructure was not active on any of the remote nodes.

Action: Start Grid Infrastructure on at least one remote node and retry the
'crsctl start rollingpatch' command, or retry patching using the non-rolling
option.
```

**Example 10-2 CLSRSC Error**

```
CLSRSC-400: A system reboot is required to continue installing.

oerr clsrsc 400

Cause: The installation of new drivers requires a node reboot to continue the
install.

Action: Reboot the node and rerun the install or patch configuration step.
```

The following table list the common error codes that you may encounter during a patching session. For an exhaustive list, see the Oracle® Database Error Messages manual.

**Table 10-1 CRS Error Codes**

| Error Code | Console Message |
|---|---|
| 1153 | There was an error setting Grid Infrastructure to rolling patch mode. |
| 1154 | There was an error setting Oracle ASM to rolling patch mode. |
| 1156 | Rejecting the rolling patch mode change because the cluster is in the middle of an upgrade. |
| 1157 | Rejecting the rolling patch mode change because the cluster was forcibly upgraded. |
| 1158 | There was an error setting the cluster to rolling patch mode. |

**Table 10-1 (Cont.) CRS Error Codes**

| Error Code | Console Message |
|---|---|
| 1159 | The cluster cannot be set to rolling patch mode because Grid Infrastructure is not active on at least one remote node. |
| 1162 | Rejecting rolling patch mode change because the patch level is not consistent across all nodes in the cluster. The patch level on nodes <patch_list> is not the same as the expected patch level <patch_level> found on nodes <node_list>. |
| 1163 | There was an error resetting Grid Infrastructure rolling patch mode. |
| 1164 | There was an error resetting Oracle ASM rolling patch mode. |
| 1166 | Rejecting rolling patch mode change because Oracle ASM is in <current_state> state. |
| 1168 | There was an error resetting the cluster rolling patch mode. |
| 1171 | Rejecting rolling patch mode change because the patch level is not consistent across all nodes in the cluster. The patch level on nodes <node_list> is not the same as the patch level <patch_level> found on nodes <node_list>. |
| 1181 | There was an error retrieving the Grid Infrastructure release patch level. |
| 1183 | Grid Infrastructure release patch level is <patch_level> and an incomplete list of patches <patch_list> have been applied on the local node. |
| 1191 | There was an error retrieving the Grid Infrastructure software patch level. |

## Rootcrs Problem Use Cases

### Issue 1: Non-rollable Patch is Applied in Rolling Mode

You have a two-node (node 1 and node 2) configuration and are attempting to apply a non-rollable patch in rolling mode.

> **Note:**
>
> By default, OPatchAuto applies patches in rolling mode.

Because you are applying the patch in rolling mode, you have not shut down all databases and stacks. When OPatchAuto is run, it prints out the stack inventory and updates the binaries as expected.

### Symptom

When `rootcrs.pl -postpatch` (Performs the required steps after the Oracle patching tool (OPatch) is invoked) is run, it fails due to ASM instances on different nodes at different patch levels. In this situation, OPatchAuto (which runs OPatch) fails with a non-zero exit code. However, the patch is left in the GI Home. The stack cannot be brought up.

### Recommended Action

It is important to note that, in this situation, it is not necessary to roll back the patch as it has already been applied to node 1. In general, make sure any attempt to bring up the stack is the very last step performed: Even if the stack fails to come up, the patch has been successfully applied to the node.

Because the patch is non-rollable, to resolve the stack issue:

1. Bring down the stack on all nodes.
2. Patch the remaining nodes by following the manual instructions provided in the patch README.
3. Bring the stack back up on all nodes.

### Issue 2: OPatchAuto Fails to Patch the GI Home

You have a system patch containing sub-patches (P1 and P2). When OPatchAuto apply is run, it will first patch the RAC homes. In this scenario P1 is applied to RAC at time t1, P2 is then applied to RAC at time t2. OPatchAuto attempts to apply sub-patch P2 at time t3 to the GI Home but fails.

### Symptom

OPatchAuto fails with a non-zero exit code. The error message indicates failure occurred when applying sub-patch P2 on the GI Home. Note that the error message will provide you with a log file location. The RAC Home now contains P1 and P2, but the GI Home is missing P2.

### Recommended Action

You need to apply the missing patch to the GI Home. Because the system patch has already been successfully applied to the RAC Home, there is no need to roll back the patch.

1. From the log file, determine what caused patch application to fail for the GI Home.
2. Fix the issue that caused the GI Home patch application to fail.

   When patch application fails for the GI Home, there are three possible causes:

   - `patchgen` -- In this situation, refer to the recommended action specified for patchgen use case. See "Patchgen".

     You will have to manually patch the GI Home. Refer to the patch README for instructions.
   - `opatch -call` command failed. In this situation, an error occurred during OPatch execution. For example, OPatch could not copy a required file.
   - `rootcrs.pl -prepatch` (perform the required steps before OPatch is invoked) fails.

     Regardless of the cause of failure, you must resolve the issue and then manually patch the GI Home.
3. Re-run `opatchauto resume` on the GI Home. OPatchAuto resumes the patch application from where it failed.

# Patchgen

### Issue

When applying a system patch, OPatchAuto fails as a result of error conditions encountered by `patchgen`.

### Symptom

OPatchAuto fails with the STDOUT error message indicating a patching failure due to problems encountered by `patchgen`.

### Recommended Action

1. Determine whether the error message is a result of a `patchgen` error. From the message output, you can determine whether or not it is of patchgen origin by searching for the keyword "patchgen." The following example shows a sample error message generated by patchgen. The keyword "patchgen" and the associated error code is in bold.

2. With the patchgen error code, run the `oerr` command to obtain the cause and recommended action(s) to resolve the specific problem encountered by patchgen. Implement the suggested action. See "Running OERR".

3. When patchgen errors out, it will ask whether or not you want to keep the patch or roll it back. By default, patchgen rolls back the patch. Whether or not the patch is rolled back determines your course of action in the next step.

   - If the patch was not rolled back, run `patchgen` again.

     Despite the error, the patch itself still exists in the GI/RAC home since it was not rolled back.

   - If the patch has been rolled back, you may find that the OPatchAuto has applied the system patch to the RAC Home, but not all sub-patches to the GI Home. At this point, you need to apply only part of the system patch to the GI Home.

     OPatch will tell you via lsinventory, which patches have not been applied. In order to apply specific sub-patches, you must resort to manual patching:

     1. Shut down the stack.

     2. Run `opatch apply` (not OPatchAuto) on the GI Home.

     Refer to the patch README for explicit instructions on applying a patch manually.

**Example 10-3 Patchgen Error Output**

```
$export ORACLE_HOME=/scratch/GI12/product/12.1.0/crs
$/scratch/GI12/product/12.1.0/crs/bin/patchgen commit -pi
13852018 loading the appropriate library for linux
java.lang.UnsatisfiedLinkError:
/scratch/GI12/product/12.1.0/crs/lib/libpatchgensh12.so (libasmclntsh12.so:
cannot open shared object file: No such file or directory)
```

The following table lists possible patchgen error codes.

**Table 10-2 Patchgen Error Codes**

| Error Code | Reason | Debugging Information |
|---|---|---|
| **2** | Internal Error | Generic failure error code. |
| **3** | Internal Error | MS Windows: Resource file read error. |
| **4** | Internal Error | MS Windows: Resource file write failed. |
| **5** | Internal Error | Unix: Open for patch repository failed. |
| **6** | Internal Error | Unix: Normalization of full path libasmclntsh failed. |
| **7** | Internal Error | Unix: Write to patch repository failed. |
| **18** | Internal Error | PGA initialization failed. |
| **19** | Internal Error | Patch iterator init failed. |
| **40** | Syntax Errors, appropriate message would be displayed. | No argument to patchgen.<br>Example: `$]patchgen` |
| **41** | Syntax Errors, appropriate message would be displayed. | No arguments to<br>`patchgen commit/recover`<br>Example: `patchgen commit` |
| **42** | Syntax Errors, appropriate message would be displayed. | -pi patchids are not numbers.<br>Example: `patchgen commit -pi 123d` |
| **43** | Syntax Errors, appropriate message would be displayed. | -rb patchids are not numbers.<br>Example: `patchgen commit -rb 123d` |
| **44** | Syntax Errors, appropriate message would be displayed. | Argument to<br>`patchgen commit/recover`<br>is something other than `-pi` or `-rb`<br>Example: `patchgen recover -random` |
| **45** | Syntax Errors, appropriate message would be displayed. | Patchgen invoked with invalid argument.<br>Example: `patchgen comit -pi` |
| **46** | Loading libpatchgensh12.so failed. | |

# Datapatch

### Issue

You attempt to run OPatchAuto to patch four product (e.g., Fusion Middleware) homes. This patch contains both bits and SQL to update the database. When you run OPatchAuto, it performs two actions:

- Applies bits to the GI/RAC home
- Runs SQL (via the `datapatch` command)

Typically, you run OPatchAuto on each GI/RAC home. With each run, OPatchAuto calls `datapatch` to run the patch SQL. `datapatch` and will do nothing on the first *n-1* nodes (no-op). On the last (n) node, `datapatch` tries to execute the patch SQL.

If `datapatch` fails, you will see an error message. To find out if the error is from datapatch, view the OPatchAuto debug log.

**Symptom**

You see a warning message indicating SQLPatch/datapatch has failed. The warning message was generated when datapatch failed to apply the SQL to the last node.

**Recommended Action**

In general, you can ignore the warning message and then run `datapatch` manually on the last node. Datapatch establishes a connection to the Database and uses Queryable Inventory (http://docs.oracle.com/cd/E16655_01/appdev.121/e17602/d_qopatch.htm) to get information regarding the patch inventory of the Oracle Home. Any issues with establishing a connection to the Oracle database may result in ORA-nnnnn errors that are described under Oracle error codes and have suitable remedial steps listed (http://docs.oracle.com/cd/B28359_01/server.111/b28278/toc.htm). In addition, Queryable Inventory has some expected ORA-nnnnn errors. The list of these errors can be referenced at http://docs.oracle.com/cd/E16655_01/appdev.121/e17602/d_qopatch.htm#CEGIFCHH . For any other issues please contact Oracle Support.

**Rollable VS. Non-Rollable Patches:** Patches are designed to be applied in either rolling mode or non-rolling mode. Depending on whether the patch is rollable or non-rollable determines the course of action.

If a patch is rollable, the patch has no dependency on the SQL script. The database can be brought up without issue. Note that a rollable patch can be applied in either rolling or non-rolling mode.

If, however, the patch is non-rollable, then the patch must first be rolled back. Note that OPatchAuto will prevent you from applying a non-rollable patch in rolling mode.

**Sequence**

1. OPatchAuto succeeds with a warning on `datapatch`/`sqlpatch`.
2. For rollable patches:
   a. Ignore datapatch errors on node 1 - node(n-1).
   b. On the last node (node n), run datapatch again. You can cut and paste this command from the log file.
   c. If you still encounter `datapatch` errors on the last node, call Oracle Support or open a Service Request.
3. For non-rollable patches:
   a. Bring down all databases and stacks manually for all nodes.
   b. Run opatchauto apply on every node.

c. Bring up the stack and databases. Note that the databases must be up in order for datapatch to connect and apply the SQL.

d. Manually run `datapatch` on the last node. Note that if you do not run `datapatch`, the SQL for the patch will not be applied and you will not benefit from the bug fix. In addition, you may encounter incorrect system behavior depending on the changes the SQL is intended to implement.

e. If datapatch continues to fail, you must roll back the patch. Call Oracle Support for assistance or open a Service Request.

# Troubleshooting OPatchAuto

OPatchAuto provides multiple venues to diagnose problems with OPatchAuto operations and patch application issues.

See also: Troubleshooting OPatchAuto for more information.

## Logging and Debugging

There are multiple log files that provide useful information to diagnose operational issues with OPatchAuto and the patching process. To ensure that the log files contain the requisite diagnostic information (such as patch and system configuration details), run OPatchAuto in debug mode.

The following steps detail the typical troubleshooting process:

1. Look at the log files.

   **Log Files on the Local Node**

   The log files will be located in the ORACLE_HOME from which OPatchAuto was run.

   Location: <ORACLE_HOME>/cfgtoollogs/opatchauto

   **Log Files on Remote Nodes**

   The log files will be located in the `<ORACLE_HOME>` of the remote node.

   Location: <ORACLE_HOME on remote node>/cfgtoollogs/opatchauto

   The <ORACLE_HOME> information for the remote node can be found in the main log file of the local node.

   The local console and main log file also contain log information about the remote node. From the console, specific log file information will be available for both the local as well as remote nodes. However, to view detailed log information, you should view the local and remote node log files directly.

2. If there is a failure, what are the suggested steps to follow in order to understand the issue in detail?

   In case of failure, view the logs to determine why patch orchestration has failed. Once resolved, patch orchestration can resume.

3. Determine where patches are staged on the remote node.

   Patches will be copied temporarily to the remote node at the following location:

   `<ORACLE_HOME>/OPatch/auto/dbtmp/<patch_id>/`

4. OPatchAuto generates a system configuration log. The location is displayed on the console.

   For example:

   ```
   <ORACLE_HOME>/cfgtoollogs/opatchautodb/systemconfig*<timestamp>.log
   ```

   This log contains all details of the OPatchAuto flow before patch execution activity starts, such as bootstrapping, identification of GI/SIHA, and user credential check on local node.

5. OPatchAuto interacts with other components such as SRVCTL, Grid utilities like ROOTCRS, OPatch to do patching. Failure can occur with these components also. The following basic checks can be done for each of the components in order for your to isolate where the problem is occurring.

   - SRVCTL: It is a utility provided to get information or alter the state of database homes. OPatchAuto uses it to do few operations like stop home, start home, home status, relocating instance, etc. Which operation opatchauto is trying to perform can be found out from the opatchauto logs. Now if something fails in this area, then srvctl can be directly used to check the status of the database home.

     If the execution of opatchauto is done in debug mode then logs for the srvctl failed command will be available in the below location. This log can be analyzed to find the reason of failure and if it is related to system configuration then it needs to be fixed before using opatchauto.

     /tmp/liveoutput_hostname.trc

   - Grid/SIHA Home utility: OPatchAuto uses rootcrs.pl/roothas.pl to stop and start GI/SIHA homes before and after applying the patch. This utility fails in some scenario depending on the system configuration or due to the patch. The log generated by this utility can be found at,

     <GIHome>/cfgtoollogs/crsconfig/crspatch*<timestamp>.log (GI version < 12.2)

     <OracleBase>/crsdata/<host>/crsconfig/crspatch*<timestamp>.log (GI version >= 12.2)

     <OracleBase>/diag/ (GI version >= 12.2)

     This log can be opened to check the reason of failure and if the reason is not familiar then it can be searched over the internet to find the possible cause or fix for it. If still unable to resolve it then the issue can be taken up with the development team for further investigation. Please ensure all the logs are attached along with the details of the initial analysis done on the issue. This will help in saving time for development team by giving a head ups.

   - OPatchCore: OPatchAuto uses opatch core API's for apply/rollback. Logs for the execution of these API's can be found under the below location,

     ```
     <ORACLE_HOME>/cfgtoollogs/opatchauto/
     opatchauto_<timestamp>_binary.log
     ```

     ```
     <ORACLE_HOME>/cfgtoollogs/opatchauto/core/opatch/
     opatch<timestamp>.log
     ```

## Verification

You can verify whether a patching has been performed correctly.

- *Verifying that patching steps have been executed on local and/or remote nodes.*

  If patching has been is executed on the local node, host information will not be available from the console.

  If patching has been executed on a remote node, host information will be available from the console.

- *Verifying that patching has been performed in rolling mode.*

  You can verify that patches have been applied in rolling mode directly from the console. The following sequence of phases occur when patches are applied in rolling mode:

  1. *Init Phase Only*: Both the local and remote nodes will be completed.
  2. Shutdown
  3. Offline
  4. Startup
  5. Online
  6. Finalize

  All of these phases are performed end-to-end on the local node before proceeding to the remote node.

  For multi-node environments, all of these phases performed end-to-end on a given node before moving on to the next node.

- *Verifying that patching has been performed in non-rolling mode*.

  You can verify that patches have been applied in non-rolling mode directly from the console. The following sequence of phases occur when patches are applied in non-rolling mode:

  1. *Init Phase Only*: Both the local and remote nodes will be completed.
  2. Shutdown
  3. Offline
  4. Startup
  5. Online
  6. Finalize

  For Oracle Database 11.2 releases, each phase will be executed in parallel on all nodes.

  For Oracle Database release 12.0 and greater, all of the phases will first be completed end-to-end on the local node. Each phase will then be executed in parallel on *n*-2 nodes (*n* being the number of nodes in the cluster). For the nth node, all phases will be completed end-to-end.

- *Verifying whether patches have been applied or rolled back*.

  – Both the GRID and RAC homes must be in the same state before and after applying/rolling back patches.

    To verify the current status of the GRID and RAC homes, run the following commands:

    ```
    crsctl check status crs
    ```

```
srvctl status database -d <database>
```

– "Using the "opatch lsinv" command user can verify the patches available in the system.

## OHASD FAILURE DURING ROOTCRS POSTPATCH

**Issue**: ohasd failure happens during `rootcrs postpatch` when OPatchAuto is used to apply patches on a 12.1.0.2.0 GI/RAC with OCT/JAN PSU or after multiple apply/ rollback of 12.1.0.2 PSUs.

**Resolution**: Change the following in <crs_home>/./crs/sbs/crswrap.sh.sbs

The workaround involves modifying 2 lines in the ohasd script

```
UID=`/usr/xpg4/bin/id -u`
# Check for root privilege
if [ $UID -eq 0 ];
```

Instead of using UID as a local variable, use anything else (UID1 for instance). This will prevent the following issue:

```
bash-3.2# ./ohasd restart

./ohasd: line 279: UID: readonly variable
```

# Known Issues while Patching

The following patching scenarios illustrate known issues that may be encountered while patching.

## Known Issues: rootcrs.pl

The following issues pertain to rootcrs.pl execution.

### Opatchauto Rollback Fails

**Issue**

`opatchauto rollback` fails in `rootcrs.pl -postpatch` when the `-norestart` option is specified.

**Symptom**

Running OPatchAuto fails in `-norestart` mode during the `rootcrs.pl -postpatch` step when rolling back the October PSU patch.

**Example 10-4 Console Output**

```
Starting CRS ... Failed
Command "/usr/bin/perl /scratch/GI12/app/12.1.0/grid/crs/install/rootcrs.pl
-postpatch -norestart" execution failed:
Died at /scratch/GI12/app/12.1.0/grid/crs/install/crspatch.pm line 851.
```

**Recommended Action:**

A prerequisite one-off patch is required.

## OPatchAuto Fails During Leaf Node Patching

**Issue**

`opatchauto` fails on the leaf node of a Flex cluster fails if the stack on the leaf node is not running.

**Symptom**

Running OPatchAuto fails on the leaf node of a Flex cluster if the stack is not up on the cluster. This occurs in both rolling and nonrolling patching modes. `rootcrs.pl -prepatch` fails with the console message shown in the following example.

**Example 10-5 Console Output**

```
Using configuration parameter file: crs/install/crsconfig_params 2013/09/27
06:00:01 CLSRSC-455: Failed attempt to initiate patch on a Leaf node
```

**Recommended Action:**

Bring up the stack on the leaf node before patching.

# Known Issues: Datapatch

The following issues pertain to datapatch execution.

## Datapatch is Executed on the First Node Instead of the Last Node During Rollback

**Issue**

When running `opatchauto rollback`, SQL changes are rolled back on the first node itself.

**Symptom**

SQL changes are rolled back from the very first node.

**Example 10-6 LOG file output**

```
Output from the command:
2013-10-07_05-16-28 :
SQL Patching tool version 12.1.0.1.0 on Mon Oct  7 05:15:31 2013
Copyright (c) 2012, Oracle.  All rights reserved.
.
Connecting to database...OK
Determining current state...done
The following patches will be rolled back: 17027533
```

**Recommended Action**

Ignore the message if the patch is going to be rolled back from all the nodes. No workaround is available.

# Known Issues: OPatch

The following issue pertains to OPatch execution.

## `opatch napply` failure

**Issue**

OPatchAuto fails during the `opatch napply` step on the CRS home due to active files.

**Symptom**

Opatchauto fails when patching the Grid Home.

**Example 10-7 Log Message**

```
[Sep 19, 2013 6:52:14 PM]    Following executables are active :
                             /u01/app/12.1.0/grid/lib/libclntsh.so.12.1
[Sep 19, 2013 6:52:14 PM]    Prerequisite check
"CheckActiveFilesAndExecutables" failed.
```

**Recommended Action**

Wait a short period of time and then run `opatchauto resume`.

## OPatch fails to rollback July'17 DB PSU

**Issue**: OPatchAuto fails to rollback the DB PSU.

**Symptoms**: Rollback gives the following error : [Apr 4, 2017 2:30:09 PM] [SEVERE] OUI-67073:UtilSession failed: Following patch(es) are inactive and cannot be rolled back.

**Use case**: Following are the pre-condition:

- Apply April 2017 DB PSU and all the overlays mandatory patches
- Apply July 2017 DB PSU

**Step**: Rollback July 2017 DB PSU

**Post-Condition**: OPatch error out as inactive patch cannot be roll back

**Solution**: This has been fixed in OPatch 12.2.0.1.10 for DB Oct PSU. Please upgrade opatch and rerun the command.

# Known Issues: OPatchAuto

The following issues pertain to OPatchAuto execution.

## Known Issues in OPatchAuto 12.1.0.1.7

The following issue pertains to OPatchAuto version 12.1.0.1.7.

### Opatchauto fails when the name of the Real Applicaiton Cluster is all upper-case.

**Issue**: While trying to create the system instance for a GI/RAC setup when the name of the RAC in upper case, OPatchAuto encounters a null pointer exception. (Bug 20858866)

**Symptom**: Because the failure occurs in the early stages of setting up the system, and no operations are performed on the GI/RAC setup, there is no adverse impact.

**Recommended Actions**:

- Use OPatch 12.1.0.1.6 (non-HP)
- Use the OPatch ZIP file specified in the base bug.

## Known Issues in OPatchAuto 12.1.0.1.5

The following issues pertain to OPatchAuto version 12.1.0.1.5 only.

### Software-only Homes

**Issue**: OPatchAuto does not support software-only homes.

**Symptom**: config.sh fails with the following error message:

```
kfod.bin: cannot execute: No such file or directory
```

**Recommended Action**: Follow the instructions in the patch README for manually applying the patch.

### OPatchAuto fails to determine a shared home

**Issue**: OPatchAuto errors out because it cannot find a shared home. This can happen on both shared as well as non-shared homes

**Symptom**: OPatchAuto generates the following error:

```
System Configuration Collection failed:
oracle.osysmodel.driver.crs.productdriver.ProductDriverException: Unable
to determine if "ORACLE HOME" is a shared oracle home.
```

**Recommended Action**: Run the following command (see examples below) as ROOT on the Oracle Home in order to determine the underlying issue. It should be run from the same location where opatchauto was run.

*On a RAC HOME*:

```
su <RAC OWNER> -c "$GRID_HOME/bin/cluvfy comp ssa -t software -s

$DB_HOME -n $NODELIST -display_status"
```

*On a GRID HOME*:

```
su <GRID OWNER> -c "$GRID_HOME/bin/cluvfy comp ssa -t software -s

$GRID_HOME/crs/install -n $NODELIST -display_status"
```

*Example*:

```
su <GRID OWNER> -c "$GRID_HOME/bin/cluvfy comp ssa -t software -s
$GRID_HOME/crs/install -n node1,node2,node3 -display_status"
```

After resolving the underlying issue re-run opatchauto.

## RAC One related Issues:

**Issue**: OPatchAuto fails to detect the status of a RAC One database. Hence, it fails to apply the SQL changes on it.

**Symptom**: OPatchAuto displays the following message from the console:

```
[WARNING] The local database instance 'INST' from 'RAC_HOME' is not
running. SQL changes, if any, will not be applied.
```

**Recommended Action**: Manually run the `datapatch` command on the RAC One database. The exact command will be shown in the opatchauto log file.

## OPatchAuto behavior in '-norestart' mode

**Issue**: When OPatchAuto is run in *-norestart* mode, it still displays the message *Starting CRS ... Successful*

**Symptom**: OPatchAuto displays this message on the console

```
Starting CRS ... Successful
```

**Recommended Action**: Ignore the message. OPatchAuto performs the required operations without actually starting the CRS.

## OPatchAuto fails to apply an incoming subset patch inside a SystemPatch

**Issue**: OPatchAuto fails to apply a system patch if it contains a one-off that is a subset of an existing patch.

**Symptom**: The command `opatch prereq CheckConflictAgainstOH` is reported to have failed.

**Recommended Action**: Roll back the superset patch in the home, apply the system patch and then apply the superset patch again.

## OPatchAuto fails in 'srvctl start home' with error code CRS-2717

**Issue**: OPatchAuto fails to start the RAC home.

**Symptom**: The error message contains the code CRS-2717.

**Recommended Action**: Manually run the pending steps listed in the OPatchAuto log file.

## Failure in creating SystemInstance

**Issue**: OPatchAuto fails to create the system Instance.

**Symptoms**: System Configuration Collection
failure: oracle.osysmodel.driver.crs.productdriver.ProductDriverException:
oracle.ops.mgmt.cluster.ClusterInfoException: PRKC-1094 : Failed to retrieve the active version of crs:

**Recommended Action:** Refer to the Bug 19262534 for available fixes.

## Datapatch is Run on All Nodes

### Issue

Running `opatchauto apply` or `opatchauto rollback` runs datapatch on all the nodes.

### Symptom

On the first node to be patched, the customer will see the message shown in Example 10-8 if RAC databases are configured on that node.

**Example 10-8 Console Output**

```
SQL changes, if any, are applied successfully on the following database(s):
```

### Recommended Action

This message can be ignored. The full information about the SQL changes made by the `datapatch` step can be obtained from the OPatchAuto debug log. It is also possible that `datapatch` might have applied the SQL changes pending from a previous patching session.

## Datapatch Does Not Run When OPatchAuto Resumes

### Issue

`opatchauto resume -reboot` does not run the `datapatch` step.

### Symptom

The `datapatch` step would not be executed by the `opatchauto resume -reboot` command.

### Recommended Action

The `datapatch` step can be manually executed from any one node. All pending changes would also be executed by the next OPatchAuto session's `datapatch` command.

Set the environment variables ORACLE_HOME and ORACLE_SID and execute the following command

```
$ORACLE_HOME/OPatch/datapatch
```

## OPatchAuto Fails without an Error Message

### Issue

The `opatchauto` command fails without any error message or stack trace on the console.

### Symptom

The user sees the following messages in the console and log files.

**Example 10-9 Console Output**

```
Starting CRS ... Failed
```

**Example 10-10 Log Message**

```
Failed to run this command :
/usr/bin/perl $GRID_HOME/crs/install/rootcrs.pl -postpatch

Executing command:
$RAC_HOME/bin/srvctl start home …
```

### Recommended Action

Refer to the `crspatch` log file at this location and make sure the timestamp points to the OPatchAuto execution time:

```
$GRID_HOME/cfgtoollogs/crsconfig/crspatch_<hostname>_<timestamp>.log
```

If this file contains the message *CLSRSC-400: A system reboot is required to continue installing*, follow these steps:

1. Reboot the machine.
2. Run the following command:

   ```
   opatchauto resume -reboot
   ```

# Common Error Symptoms/Conditions

The following are common error conditions.

## Rootcrs.pl Postpatch

Patch scenario where the patch attempt fails when trying to bring up the product stack.

## Patcherr

Patch scenario where the patch attempt fails due to relink failure.

# A
# OPatch Syntax and Commands

This appendix provides a summary of the syntax and command options to use for the `opatch` command. Use these command options to develop your own patch plan.

## OPatch Syntax

The OPatch utility is located in the `$Oracle_Home/OPatch` directory. You can run it with various commands and options. The following string shows the syntax for the OPatch utility:

```
<Path_to_OPatch>/opatch [-help] [-r[eport]][command] [-option]
```

where:

- `[-help]` displays the help message for the `opatch` command.
- `[-report]` prints the actions without executing.
- `[command]` is one of the OPatch commands described in Table A-1.
- `[-option]` is one of the OPatch command options. See each command listed below for a summary of available options.

**Table A-1 OPatch Commands**

| Command | Description |
|---|---|
| `apply` | Installs or applies a patch. |
| `compare` | Compare two files generated by the `opatch lsinventory -xml` command. |
| `lsinventory` | Lists what is currently installed on the system. |
| `lspatches` | Prints a summary of all installed patches. |
| `napply` | Installs *n* number of patches. |
| `nrollback` | Rolls back patches from several product (e.g., Fusion Middleware) homes at the same time. |
| `query` | Queries a given patch for specific details. |
| `rollback` | Removes a patch. |
| `version` | Prints the current version of the patch tool. |
| prereq | Runs patching prerequisite checks on an ORACLE_HOME.. |
| `util` | Invokes specified utilities on an GI/RAC home |

To view additional information for any command, use the following command:

```
<Path_to_OPatch>/opatch command -help
```

If using Perl, use the following command:

```
perl opatch.pl command -help
```

To show the full syntax of the `-help` option, enter `opatch -h` to view the following display:

```
Usage: opatch [ -help ] [ -report ] [ command ]

            command := apply
                       compare
                       lsinventory
                       lspatches
                       napply
                       nrollback
                       rollback
                       query
                       version
                       prereq
                       util

 <global_arguments> := -help       Displays the help message for the command.
                       -report     Print the actions without executing.

 example:
   'opatch -help'
   'opatch -help -fmw'
   'opatch apply -help'
   'opatch compare -help'
   'opatch lsinventory -help'
   'opatch lspatches -help'
   'opatch napply -help'
   'opatch nrollback -help'
   'opatch rollback -help'
   'opatch prereq -help'
   'opatch util -help'
```

## apply

The `apply` command applies an interim patch to an ORACLE_HOME from the current directory. The patch location can specified using the parameter patch_location. This command does not support System Patch.

### Syntax

Use following syntax for this command:

```
opatch apply  [-connectString  <List of connect strings>]
              [-delay <value> ] [ -force ] [ -force_conflict ]
              [-init <parameters for the init script in escaped double
               quotes> [-opatch_init_end] ]
              [-invPtrLoc <Path to oraInst.loc> ]
              [-jre <LOC> ] [-local ] [-local_node <Local node name>]
              [-minimize_downtime ] [-no_bug_superset ] [-no_inventory ]
              [-no_relink] [-no_sysmod] [-ocmrf <Response file location> ]
              [-oh <ORACLE_HOME> ]
              [-post <parameters for the post script in escaped
               quotes> [-opatch_post_end] ]
              [-pre <parameters for the pre script
               in escaped double quotes> [-opatch_pre_end] ]
              [-profile_mask <Name of profile>]
              [-property_file <Path to property file>]
```

```
[-ptlConnect <portal connect string>]
[-ptlPassword <portal password>]
[-ptlSchema <portal schema>]
[-remote_nodes <List of remote nodes (node1,node2)>]
[-retry <value ] [-runSql ]
[silent ] [-sqlScript <path of the sql file>] [-verbose ]
[ <Patch Location> ]
```

### Options

Table A-2 describes the options available for the `apply` command.

**Table A-2 `apply` Command Options**

| Option | Description |
|---|---|
| `delay` | Specifies how many seconds to wait before attempting to lock the inventory in the case of a previous failure. You can use this option only if you specify the `-retry` option. |
| `force` | If a conflict exist which prevents the patch from being applied, the -force flag can be used to apply the patch. OPatch will remove all the conflicting patches before applying the current patch. In case of conflict among the patches to be applied, the non-conflicting patches will be applied. |
| force_conflict | If a conflict exists which prevents the patch from being applied, the -force_conflict flag can be used to apply the patch. OPatch will remove all the conflicting patches before applying the current patch. This will override the 'silent' behavior for conflicts and hence is meaningful only when used with the 'silent' option. |
| `invPtrLoc` | Specifies the location of the `oraInst.loc` file. This option is needed when the `-invPtrLoc` argument was used during installation. Oracle recommends using the default Central Inventory for a platform. |
| `jre` | Instructs OPatch to use JRE (Java) from the specified location instead of the default location under the GI/RAC home directory. |
| `local` | Specifies that the OPatch utility should patch the local node and update the inventory of the local node. It does not propagate the patch or inventory update to other nodes.<br>You can use this option on Oracle Real Application Clusters environments and non-clustered environments. If an entire cluster is shut down before patching, you can use this argument for non-rolling patches. |
| `local_node` | Tells OPatch the local node for this cluster. You can use this option on Oracle Real Application Clusters environments. |
| `minimize_downtime` | Specifies the order of nodes that OPatch should patch.<br>This option only applies to Oracle Real Application Clusters environments. You cannot use it with the `-local` option or a rolling patch. |
| `no_bug_superset` | Specifies to error out if the current patch bugs-to-fix is a superset (or same set) as an installed patch bugs-fixed in the GI/RAC home directory. |

**Table A-2 (Cont.) `apply` Command Options**

| Option | Description |
|---|---|
| `no_relink` | This option does not perform any make operations. You can use it during multiple patch applications and to perform the linking step only once. OPatch does not keep track of the make operations it did not perform. You need to make sure to execute OPatch without this option at the end for compilation. |
| ocmrf | Give OPatch the absolute path to the OCM response file to be used for OCM configuration. -silent must be used in conjunction with -ocmrf if GI/RAC home doesn't have OCM installed and configured. |
| `oh` | Specifies the GI/RAC home directory to use instead of the default. This takes precedence over the environment variable `ORACLE_HOME`. |
| `opatch_post_end` | Marks the end of the `post` option. You use this option with the `post` option. If you do not use this option, everything after `post` until the end of the command is passed into `post`. |
| `opatch_pre_end` | Marks the end of the `pre` options. You use this option with the `pre` option. If you do not use this argument, everything after `pre` until the end of the command is passed into `pre`. |
| `post` | Specifies the parameters to be passed to the `post` script. This script is executed after the patch is applied. You need to enclose the values for this option in double-quotes. |
| `pre` | Specifies the parameters to be passed to the `pre` script. This script is executed before the patch is applied. You need to enclose the values for this option in double-quotes. |
| `property_file` | Specifies the user-defined property file for OPatch to use. The path to the property file should be absolute. This property file takes precedence over the one that OPatch supplies. |
| `report` | Prints the action to the screen without executing it. |
| `retry` | Tells OPatch how many times it should retry when there is an inventory lock failure. |
| `silent` | Suppresses user interaction, and defaults any answers to "yes." |
| `verbose` | Prints additional OPatch output to the screen as well as to the log file. |

> **Note:**
>
> If a patch consists of SQL changes, follow the instructions in the patch README, which is included with the patch to apply the SQL scripts.

## `compare`

The `compare` command allows you to compare the bugs that have been fixed between two product (e.g., Fusion Middleware) homes.This command allows for comparison

between two files generated by the `opatch lsinventory -xml` command. Currently, this command only accepts two files as input.

### Syntax

```
opatch compare     [<file1> <file2>]
```

## lsinventory

The `lsinventory` command lists the inventory for a particular GI/RAC home, or displays all installations that can be found. This command does not have any required options.

### Syntax

Use the following syntax for this command:

```
opatch lsinventory [-all ] [-all_nodes] [-bugs_fixed <asc | desc> ]
           [-delay <value> ] [-detail ] [-group_by_date ]
           [-inactive]
                   [-invPtrLoc <Path to oraInst.loc> ]
                   [-jre <LOC> ] [-local ]
                   [-oh <ORACLE_HOME> ] [-patch <asc | desc> ]
                   [-patch_id <asc | desc> ]
                   [-ptlConnect  <portal connect string> ]
                   [-ptlPassword <portal password> ]
                   [-ptlSchema <portal schema> ]
                   [-property_file <path to property file>]
           [-retry <value> ] [-translation_patch ]
           [-xml <xmlFile>]
```

The following sections provide examples for the detail, bugs_fixed, and patch desc options. See Table A-3 for descriptions of the command options.

### Options

Table A-3 describes the options available for the `lsinventory` command.

**Table A-3 `lsinventory` Command Options**

| Option | Description |
|---|---|
| all | Reports the name and installation directory for each GI/RAC home directory found. |
| bugs_fixed | Reports bugs fixed by installed patches in a tabular format. Besides the bugs fixed, the report also displays the installed patches, installed times, and bug descriptions.<br>The fixed bugs are sorted per installed patch. Default display is patches in descending order based on installed time and ascending order of bugs within each patch. You can use 'asc' (or) 'desc' with this option to enforce sort order on bugs within each patch.<br>You can use this option with the patch or patch option to obtain sort orders with installed patches. |
| delay | If you specify retry, this option tells OPatch how many seconds it should wait before attempting to lock the inventory again in case of a previous failure. |

**Table A-3 (Cont.) `lsinventory` Command Options**

| Option | Description |
|---|---|
| detail | Reports the installed products and other details. You cannot use this option with the `-all` option. |
| group_by_date | Specifies that OPatch should group all installed patches by the date they were installed in the GI/RAC home. |
| invPtrLoc | Specifies the location of the `oraInst.loc` file. You need this option if you used the `invPtrLoc` option during the installation. Oracle recommends using the default Central Inventory for a platform. |
| jre | Specifies the location of a particular JRE (Java) to use instead of the default location under the GI/RAC home directory. |
| local | Instructs OPatch to display inventory information of the local node only. |
| oh | Specifies the GI/RAC home directory to use instead of the default directory. This takes precedence over the ORACLE_HOME environment variable. |
| patch | Lists the patch IDs installed in the GI/RAC home in ascending (asc) or descending (desc) order, which is the default, based on installed time. |
| patch_id | Lists the patch IDs installed in the GI/RAC home in ascending (asc) or descending (desc) order based on patch numbers. The value defaults to ascending (asc). |
| property_file | Indicates the user-defined property file that OPatch should use. The path to the property should be absolute. This property file takes precedence over the property file that OPatch supplies. |
| retry | Specifies how many times OPatch should retry when there is an inventory lock failure. |
| xml | Generates xml output based on the current GI/RAC home inventory to the specified xml file. |

## `lspatches`

The `lspatches` command prints a summary of all installed patches.

### Syntax

Use the following syntax for the `lspatches` command:

```
opatch lspatches   [-bugs] [-id <patch ID> ]
                   [-invPtrLoc <Path to oraInst.loc> ] [-jre <LOC> ]
                   [-oh <ORACLE_HOME> ]
           [-qfile <file path> ] [-required ] [-verify] <PATCH_ID or
PATCH_LOCATION>
```

### Options

Table A-4 lists the options available for the `lspatches` command.

**Table A-4 `lspatches` Command Options**

| Option | Description |
|---|---|
| `bugs` | Prints out bugs in addition to the summary |
| `id` | This option specifies the patch number. It must be registered in the GI/RAC home inventory. It can be any numeric sequence or combined with language. Example: 11111, 11111/zh_CN. It cannot support multiple patch IDs. |
| `invPtrLoc` | Used to locate the oraInst.loc file. When the installation uses the invPtrLoc flag, the value should indicate the path to oraInst.loc file |
| jre | This option tells OPatch to use JRE (java) from the specified location instead of the default location under GI/RAC home. |
| oh | The GI/RAC home to work on. This takes precedence over the environment variable ORACLE_HOME. |
| qfile | Specifies the relative path to GI/RAC home of the file to determine the latest patch that touches this file. Example:<br>On Linux: admin/rdbms/catcpu.sql On Windows: admin\rdbms\catcpu.sql OPatch can tell which latest patch touches the file catcpu.sql in the GI/RAC home. |
| required | This option will print key metadata only. This includes the following metadata: required components, prereq patches, executables to shutdown and support platforms.<br>This option should be accompanied by either option -id <PATCH_ID> or <PATCH_LOCATION>. |
| verify | This option verifies whether or not the specified patch ID or patch location is registered in the GI/RAC home inventory. In addition, this option validates all patch files in the GI/RAC home. This option should be accompanied by either option -id <PATCH_ID> or <PATCH_LOCATION>.This option doesn't support System Patch. Example: `opatch lspatches -id 111 -verify`<br>`opatch lspatches /scratch/test/111 -verify` |

## napply

This command applies patches to several product (e.g., Fusion Middleware) homes at the same time. This command does not support System Patches.

### Syntax

Use the following syntax for the `napply` command:

```
opatch napply [patch_location] [-id comma-separated list of patch IDs]
                        [ -all_nodes ]
              [-connectString  <List of connect strings>]
                  [-delay <value> ] [ -force ] [ -force_conflict ]
              [-idFile <path of the file that has list of patch IDs ]
                        [-init <parameters for the init script in escaped double
                         quotes> [-opatch_init_end] ]
                        [-invPtrLoc <Path to oraInst.loc> ]
                        [-jre <LOC> ] [ -local ]
                        [ -local_node <Local node name> ]
                        [-minimize_downtime ] [-no_bug_superset ]
                        [-no_inventory ] [-no_relink]
```

```
                      [-no_sysmod]  [-ocmrf <Response file location> ]
                      [-oh <ORACLE_HOME> ]
                      [ -phBaseDir <Path to the directory that contains list
                        of patch directories> ]
                      [ -phBaseFile <Path to the file containing the
                        location of the patches to be applied> ]
                      [-post <parameters for the post script in
                       escaped double quotes> [-opatch_post_end] ]
                      [-pre <parameters for the pre script in
                       escaped double quotes> [-opatch_pre_end] ]
                      [-profile_mask <Name of profile>]
                      [ -property_file <Path to property file> ]
                        [-ptlConnect <portal connect string>]
               [-ptlPassword <portal password>]
                      [-ptlSchema <portal schema>]
                      [ -remote_nodes <List of remote nodes
(node1,node2)> ]                                    [-retry <value> ] [-runSql]
[-silent ]
                      [-skip_subset]
                      [-skip_duplicate]
              [-sqlScript <path of the sql file>]
                      [-verbose ]
```

### Options

Table A-5 lists the options available for this command.

**Table A-5 `napply` Command Options**

| Option | Description |
|---|---|
| `delay` | Specifies how many seconds to wait before attempting to lock the inventory again for a previous failure. You can use this option only if you specify the `retry` option. |
| `force` | If a conflict exist which prevents the patch from being applied, the -force flag can be used to apply the patch. OPatch will remove all the conflicting patches before applying the current patch. In case of conflict among the patches to be applied, the non-conflicting patches will be applied. |
| `force_conflict` | If a conflict exist which prevents the patch from being applied, the -force_conflict flag can be used to apply the patch. OPatch will remove all the conflicting patches before applying the current patch. This will override the 'silent' behavior for conflicts and hence is meaningful only when used with 'silent' option. |
| `id` | Use the 'lsinventory' option to display all patch ids. Each one-off patch is indicated by its id. A comma separated list of patches can be given to select the patches to be applied. For translation patches, the patch id should be of the format <Patch ID>/<Language code>. |
| `idFile` | The input to be given is a file location that contains a list of apply patch ids separated by commas or white spaces. This option cannot to be in conjunction with 'id' option. For translation patches, the patch id should be of the format <Patch ID>/<Language code>. |
| `invPtrLoc` | Specifies the location of the `oraInst.loc` file. The `invPtrLoc` option is needed when this option is used during installation. Oracle recommends the use of the default Central Inventory for a platform. |

**Table A-5 (Cont.) `napply` Command Options**

| Option | Description |
|---|---|
| `jre` | Instructs OPatch to use JRE (Java) from the specified location instead of the default location under the GI/RAC home directory. You cannot specify the `jdk` and `jre` options together. |
| `local` | Specifies that OPatch should patch the local node and update the inventory of the local node. It does not propagate the patch or inventory update to other nodes.<br>You can use this option on Oracle Real Application Clusters environments and non-clustered environments. If an entire cluster is shut down before patching, you can use this option for non-rolling patches. |
| `no_bug_superset` | Specifies to error out if the current patch's bugs-to-fix is a superset (or same set) of an installed patch's bugs-fixed in the GI/RAC home directory. |
| `no_relink` | This option does not perform any `make` operations. You can use it during multiple patch applications and to perform the linking step only once. OPatch does not keep track of the `make` operations it did not perform. You need to make sure to execute OPatch without this option at the end for compilation. |
| ocmrf | Give OPatch the absolute path to the OCM response file to be used for OCM configuration. -silent must be used in conjunction with -ocmrf if GI/RAC home does not have OCM installed and configured. |
| `oh` | Specifies the GI/RAC home directory to use instead of the default. This takes precedence over the environment variable `ORACLE_HOME`. |
| `opatch_post_end` | Marks the end of the `post` option. You use this option with the `post` option. If you do not use this option, everything after `post` until the end of the command is passed into `post`. |
| `opatch_pre_end` | Marks the end of the `pre` options. You use this option with the `pre` option. If you do not use this option, everything after `pre` until the end of the command is passed into `pre`. |
| phBaseDir | Used to specify a directory containing patch directories (or) zip files. |
| `phBaseFile` | If you do not specify `<patch_location>`, use this option to point OPatch to a file containing a list of patches to be n-applied. Each line in the file points to a location of a patch. |
| `post` | Specifies the parameters to be passed to the `post` script. This script is executed after the patch is applied. You need to enclose the values for this option in double-quotes. |
| `pre` | Specifies the parameters to be passed to the `pre` script. This script is executed before the patch is applied. You need to enclose the values for this option in double-quotes. |
| profile_mask | If the patch to be applied specifies WLS patch/patchset as prerequisites, OPatch will read the WLS default patch profile. To have OPatch read non-default patch profile, specify the patch profile name with this option. |

**Table A-5 (Cont.) `napply` Command Options**

| Option | Description |
|---|---|
| `property_file` | Specifies the user-defined property file for OPatch to use. The path to the property file should be absolute. This property file takes precedence over the one that OPatch supplies. |
| `report` | Prints the action to the screen without executing it. |
| `retry` | Tells OPatch how many times it should retry when there is an inventory lock failure. |
| `silent` | Suppresses user interaction, and defaults any answers to "yes." |
| `skip_duplicate` | Skips patches to be applied that are duplicates of other patches installed in the GI/RAC home. Two patches are duplicates if they fix the same set of bugs. |
| `skip_subset` | Skips patches to be applied that are subsets of other patches installed in the GI/RAC home. One patch is a subset of another patch if the former fixes a subset of bugs fixed by the latter.<br>For example, if you used `napply` yesterday for patch A that fixed bugs 1 and 2, you use `napply` today with this option for patch B that fixes bug 1 and patch C that fixes bugs 1, 2, and 3. Then subset patch A is skipped, and patch C then becomes a superset of patch A. |
| `verbose` | Prints additional OPatch output to the screen as well as to the log file. |

**Examples:**

```
'opatch napply <patch_location>' to apply all patches under
  <patch_location> directory

'opatch napply <patch_location> -id 1,2,3' to apply patches
  1, 2, and 3 which are present under <patch_location>
  directory

'opatch napply <patch_location> -skip_subset -skip_duplicate'
  to apply all patches under <patch_location> directory.
  OPatch will skip duplicate patches and subset patches
  (patches under <patch_location> that are subsets of patches
  installed in the Oracle Home)

'opatch napply <patch_location> -id 1,2,3 -skip_subset -skip_duplicate'
  to apply patches 1, 2, and 3 which are under <patch_location>
  directory. OPatch will skip duplicate patches and subset patches
  (patches under <patch_location> that are subsets of patches
  installed in the Oracle Home)

'opatch napply <patch_location> -idfile /tmp/list.txt' where list.txt contains
a list of patch IDs to be applied. The list should be separated by a space or
comma. For example: 1 2 3

'opatch napply <patch_location> -id 1/fr,2/de' to apply patches 1 (french
patch), 1 (german patch) which are present in the <patch_location> directory
```

## nrollback

The `Nrollback` command rolls back patches from several product (e.g., Fusion Middleware) homes at the same time.

### Syntax

Use the following syntax for this command:

```
opatch nrollback -id <comma-separated list of patch IDs>
                      [ -all_nodes ]
              [-connectString  <List of connect strings>]
              [-delay <value>] -id <Comma separated list of patch IDs>
                      [-idFile <file location containing a list of
                       rollback IDs separated by commas or white spaces>
                      [-init <parameters for the init script in escaped double
                       quotes> [-opatch_init_end] ]
                      [-invPtrLoc <Path to oraInst.loc> ]
                      [-jre <LOC> ] [-local]
                      [-local_node <Local node name>]
                      [-no_inventory] [-no_relink] [-no_sysmod]
                      [-oh <ORACLE_HOME> ]
                      [-post <parameters for the post script in
                       escaped double quotes>[ -opatch_post_end] ]
                      [-pre <parameters for the pre script in
                       escaped double quotes> [-opatch_pre_end] ]
                      [-property_file <Path to property file>]
          [-ptlConnect <portal connect string>]
              [-ptlPassword <portal password>]
              [-ptlSchema <portal schema>]
                      [-remote_nodes <List of remote nodes (node1,node2)>]
                      [-retry <value>] [-runSql]  [-silent]
                      [-sqlScript <path of the sql file>]
                      [-verbose]
```

### Options

Table A-6 lists the options available for this command.

**Table A-6 nrollback Command Options**

| Option | Description |
|---|---|
| `delay` | If you use the `retry` option with the `rollback` command, specifies how many seconds OPatch should wait before attempting to lock the inventory again if a previous failure occurs. |
| `id` | Indicates the patch to be rolled back. Use the `lsinventory` option to display all patch identifiers. Each one-off patch is indicated by its ID. To successfully roll back a patch, you must provide the patch identifier. |
| idFile | Use 'lsinventory' option to display all patch ids. Each one-off patch is indicated by its id. To rollback a patch, the id for that patch must be supplied. The input to be given is a file location that contains a list of rollback patch ids separated by commas or white spaces. For translation patches, the patch id should be of the format <Patch ID>/<Language code>. This option cannot to be in conjunction with 'id' option. |

**Table A-6 (Cont.) nrollback Command Options**

| Option | Description |
|---|---|
| `invPtrLoc` | Specifies the location of the `oraInst.loc` file. You need to use this option if you used the `invPtrLoc` option during installation. Oracle recommends the use of the default Central Inventory for a platform. |
| `jre` | Specifies the location of a particular JRE (Java) for OPatch to use instead of the default location under the GI/RAC home directory. |
| `local` | Specifies that OPatch roll back the local node, then updates the inventory of the local node. It does not propagate the patch or inventory update to other nodes.<br>You can use this option on Oracle Real Application Clusters environments and non-clustered environments. If an entire cluster is shut down before patching, you can use this option for non-rolling patches. |
| `no_relink` | This option does not perform any `make` operation in the patch. You can use this option during multiple patch removals and to perform the compilation step only once. |
| `oh` | Specifies the GI/RAC home directory to use instead of the default directory. This takes precedence over the `ORACLE_HOME` environment variable. |
| `opatch_post_end` | Marks the end of the `post` options. Use this option with the `post` option. If you do not use this option, everything after `post` until the end of the command is passed into `post`. |
| `opatch_pre_end` | Marks the end of the `pre` options. Use this option with the `pre` option. If you do not use this option, everything after `pre` until the end of the command is passed into `pre`. |
| post | This option is used to pass parameters to the post script. This script is executed after removal of the patch. The value for this option have to be enclosed in double quotes. The parameters will be common parameters which will be passed to post scripts of all patches being rolled back. This option should be ended by option 'opatch_post_end'. |
| pre | This option is used to pass parameters to the pre script. This script is executed before removal of the patch. The value for this option have to be enclosed in double quotes. The parameters will be common parameters which will be passed to pre scripts of all patches being rolled back. This option should be ended by option 'opatch_pre_end'. |
| `property_file` | Specifies the user-defined property file for OPatch to use. The path to the property file should be absolute. This property file takes precedence over the one that OPatch supplies. |
| `report` | Prints the actions to the screen without executing them. |
| `retry` | Instructs OPatch how many times it should retry when there is an inventory lock failure. |
| `silent` | Suppresses user interaction, and defaults any yes\|no questions to "yes". An Oracle Real Application Clusters setup does not support this option. |
| `verbose` | Prints additional OPatch output to the screen as well as to the log file. |

### Examples

```
'opatch nrollback -id 1,2,3' to roll back patches 1, 2, and 3
  that have been installed in the Oracle Home.

'opatch nrollback -id 1/fr,2/de to rollback patches 1 with language 'fr',
2 with language 'de' that have been installed in the Oracle Home.
```

## rollback

The `rollback` command removes an existing one-off patch from the appropriate GI/RAC home directory indicated by the reference ID. The following syntax is used for this command:

### Syntax

```
opatch rollback -id <ID> [-connectString  <List of connect strings>]
        [-delay <value>]
        [-init <parameters for the init script in escaped double
                 quotes> [-opatch_init_end] ]
                [-invPtrLoc <Path to oraInst.loc> ]
                [-jre <LOC> ] [-local]
        [-local_node <Local node name>] [-no_inventory]
        [-no_relink] [-no_sysmod]
        [-oh <ORACLE_HOME>] [-ph <Patch Location>]
                [-post <parameters for the post script in escaped
                 double quotes>[ -opatch_post_end] ]
        [-pre <parameters for the pre
                 script in escaped double quotes> [-opatch_pre_end] ]
                [-property_file <path to property file>]
                [-ptlConnect <portal connect string>]
                [-ptlPassword <portal password>]
        [-ptlSchema <portal schema>]
                [-remote_nodes <List of remote nodes (node1,node2)>]
        [-retry <value>] [-runSql] [-silent]
        [-sqlScript <path of the sql file>] [-verbose]
        [all_subpatches]
```

### Options

Table A-7 describes the options available for the `rollback` command.

**Table A-7 `rollback` Command Options**

| Option | Description |
|---|---|
| `delay` | If you use the `retry` option with the `rollback` command, specifies how many seconds OPatch should wait before attempting to lock the inventory again if a previous failure occurs. |
| `id` | Indicates the patch to be rolled back. Use the `lsinventory` option to display all patch identifiers. Each one-off patch is indicated by its ID. To successfully roll back a patch, you must provide the patch identifier. |
| `invPtrLoc` | Specifies the location of the `oraInst.loc` file. You need to use this option if you used the `invPtrLoc` option during installation. Oracle recommends the use of the default Central Inventory for a platform. |

**Table A-7 (Cont.) `rollback` Command Options**

| Option | Description |
|---|---|
| `jre` | Specifies the location of a particular JRE (Java) for OPatch to use instead of the default location under the GI/RAC home directory. |
| `local` | Specifies that OPatch roll back the local node, then updates the inventory of the local node. It does not propagate the patch or inventory update to other nodes.<br>You can use this option on Oracle Real Application Clusters environments and non-clustered environments. If an entire cluster is shut down before patching, you can use this option for non-rolling patches. |
| `local_node` | Specifies to OPatch that this is the local node for the cluster to be used for rollback.<br>You can use this option for Oracle Real Application Clusters environments. |
| no_relink | Do not perform the make operations in the patch. This option can be used during multiple patch removals and perform the compilation step only once. |
| `oh` | Specifies the GI/RAC home directory to use instead of the default directory. This takes precedence over the `ORACLE_HOME` environment variable. |
| `opatch_post_end` | Marks the end of the `post` options. Use this option with the `post` option. If you do not use this option, everything after `post` until the end of the command is passed into `post`. |
| `opatch_pre_end` | Marks the end of the `pre` options. Use this option with the `pre` option. If you do not use this option, everything after `pre` until the end of the command is passed into `pre`. |
| `ph` | Specifies the valid patch directory area. Rollback uses the command types found in the patch directory to identify which commands are used for the current operating system. |
| `post` | Specifies the parameters to be passed inside the `post` script. This script executes after the patch is removed. You must enclose the value of this option in double-quotes. |
| `pre` | Specifies the parameters to be passed inside the `pre` script. This script executes before the patch is removed. You must enclose the value of this option in double-quotes. |
| `property_file` | Specifies the user-defined property file for OPatch to use. The path to the property file should be absolute. This property file takes precedence over the one that OPatch supplies. |
| `report` | Prints the actions to the screen without executing them. |
| `retry` | Instructs OPatch how many times it should retry when there is an inventory lock failure. |
| `silent` | Suppresses user interaction, and defaults any yes\|no questions to "yes". An Oracle Real Application Clusters setup does not support this option. |
| sqlScript | This option can be used to specify the custom SQL script to be run by OPatch after patching is completed |
| `verbose` | Prints additional OPatch output to the screen as well as to the log file. |

**Table A-7 (Cont.) `rollback` Command Options**

| Option | Description |
|---|---|
| all_subpatches | This option is valid ONLY for composite patches. It allows the user to rollback all sub-patches of a composite series in one shot. |

## `query`

The `query` command queries a specific patch for specific details. It provides information about the patch and the system being patched.

### Syntax

Use the following syntax for this command:

```
opatch query  [-all] [-is_auto_patch] [-is_translatable_patch]
              [-get_base_bugs] [-get_component] [-get_os] [-get_date]
              [-get_patch_language] [-get_patch_type] [-get_patching_model]
              [-get_product_family] [-has_sql] [-is_online_patch]
              [-is_rolling_patch] [-is_system_patch] [-jre <LOC> ] [-oh <LOC> ]
              [ <patch_location> ]
```

### Options

Table A-8 lists the options available for the `query` command.

**Table A-8 query Command Options**

| Option | Description |
|---|---|
| `all` | Retrieves all information about a patch. This is equivalent to setting all available options. |
| is_auto_patch | This option says 'true' if the patch is auto-enabled, 'false' otherwise. This command doesn't support System Patch. |
| is_system_patch | This option says 'true' if the patch is a System Patch, 'false' otherwise. |
| is_translatable_patch | This option says 'true' if the patch is translatable, 'false' otherwise. This option doesn't support System Patch. |
| `get_base_bug` | Retrieves bugs fixed by the patch. |
| `get_component` | Retrieves components the patch affects. |
| `get_date` | Retrieves the patch creation date and time. |
| get_os | Get platforms for which this patch could be applied. This option does not support System Patch. |
| get_patch_language | Get the language supported by the patch. This option doesn't support System Patch. |
| get_product_family | Get the product family to which the patch belongs. This option does not support System Patch. |
| `is_online_patch` | Indicates true if the patch is an online patch. Otherwise, the option is false. |
| `is_rolling_patch` | Indicates true if the patch is a rolling patch. Otherwise, the option is false. |

**Table A-8 (Cont.) query Command Options**

| Option | Description |
|---|---|
| jre | This option tells OPatch to use JRE (java) from the specified location instead of the default location under GI/RAC home. |
| oh | Specifies the GI/RAC home directory to use instead of the default directory. This takes precedence over the ORACLE_HOME environment variable. |

## version

The version command shows the current version number of the OPatch utility.

### Syntax

The following syntax is used for this command:

```
opatch version [-all] [-invPtrLoc <Path to oraInst.loc>]
               [-jre <LOC>] [-oh <ORACLE_HOME>]
               [-v2c <5-digit version> -oui_loc <Custom OUI Location>
                -ph <Patch Location> -ohs <list of Oracle Homes
                separated by commas]
               [-help] [-h]
```

**Table A-9 version Command Options**

| Option | Description |
|---|---|
| all | This option displays versions of OPatch for all product (e.g., Fusion Middleware) homes registered in the Central Inventory. |
| invPtrLoc | Used to locate the oraInst.loc file. When the installation used the invPtrLoc flag. This should be the path to the oraInst.loc file. |
| jre | This option tells OPatch to use JRE (java) from the specified location instead of the default location under GI/RAC home. |
| oh | The GI/RAC home to work on. This takes precedence over the environment variable ORACLE_HOME |
| v2c | The standard 5-digit version to compare. If this option is specified with an valid version which made by no more than 5 numbers separated by '.', those product (e.g., Fusion Middleware) homes with valid version will be break up to two parts, one is those product (e.g., Fusion Middleware) homes which have opatch version higher or equal to the value of this option, and the other is those with lower version |
| ph | The patch location from where the Minimum OPatch Version (if defined) will be picked. If a valid patch location is provided, will take the required minimum opatch version from the patch and set it as the standard OPatch version to be compared to. |
| ohs | List of product (e.g., Fusion Middleware) homes to be considered. Please provide them separated by commas |
| help | Display valid options can be attached to this operation. |

# prereq

This operation runs the prerequisite checks on an ORACLE_HOME. This command does not support System Patches.

### Syntax

```
opatch prereq <command> [-id <Comma separated list of patch IDs>]
            [-invPtrLoc <Path to oraInst.loc> ]
            [-jre <LOC>] [-local_node <Local node name>]
           [-oh <ORACLE_HOME> ]
    [-ph <Path to the single patch location>]
           [-phBaseDir <Path to the dir containing all patches>]
           [-phBaseFile <Path to the file containing the
                         location of the patches to be applied>]
    [-property_file <Path to property file>]
                  [-remote_nodes <List of remote nodes
(node1,node2)>]                                    [-sid <Comma separated list of
database SIDs>]
           [-connectString  <List of connect strings>]
```

### Commands

The `prereq` command executes commands that check for the prerequisite conditions shown in the table.

**Table A-10 `prereq` Commands**

| Command | Description |
|---|---|
| `CheckActiveFilesAndExecutables` | Check if there are any file(s) that are active, which are touched by the patch to be applied or rolled back. |
| CheckActiveServices | Check for the services that are active. Note: Applicable for Windows platforms only. |
| CheckApplicable | Check for the presence of the required components in the ORACLE_HOME and check if all the actions of the given patch(es) are applicable. |
| CheckApplicableProduct | Check if the patch is applicable for the given GI/RAC home. If the patch is marked for stand-alone homes, then it can not be applied on normal OUI-based home and vice versa. Also, a patch can be marked as a hybrid patch, where it is applicable for both homes. |
| CheckCentralInventoryForOH | Check if the given ORACLE_HOME is registered in the central inventory specified by the oraInst.loc file. |
| CheckCentralInventoryForRWSession | Check if a RW (read-write) session can be created for the given central inventory. |
| CheckCentralInventoryLocation | Validate the Central Inventory location. Check if it has the correct directory structure and has the inventory.xml with read permissions. |
| CheckComponents | Check for the presence of the required components in the ORACLE_HOME. |
| CheckConflictAgainstOH | Check if there are any conflicts between the patch(es) to be applied and the patch(es) in the OH. |

**Table A-10 (Cont.) `prereq` Commands**

| Command | Description |
|---|---|
| CheckConflictAgainstOHWithDetail | Check if there are any conflicts between the patch(es) to be applied and the patch(es) in the OH, by giving out the detailed information about the conflicts/supersets. |
| CheckConflictAmongPatches | Check if there are any conflicts among the patch(es) to be applied. |
| CheckConflictAmongPatchesWithDetail | Check if there are any conflicts among the patch(es) to be applied, by giving out the detailed information about the conflicts/supersets. |
| CheckFileVersions | Check if the copy actions of Fusion Applications patch(es) have at least one or more file version(s) greater than the version(s) installed in the GI/RAC home. |
| CheckFusionAppsCompatible | Check if OUI for the GI/RAC home supports patching of Fusion applications. |
| CheckForIdenticalPatchInOracleHome | Check if the given list of patch(es) are identical with respect to the patch(es) installed in the GI/RAC home. |
| CheckForInputValues | Check if the input values provided to OPatch are enough for OPatch to proceed further. |
| CheckForNoOpPatches | Check if any of the patch(es) provided by the user are no-op patches. A no-op patch cannot be applied to the GI/RAC home and can be skipped. This prerequisite will fail for no-op. patches. |
| CheckIfOHLockedForPatching | Check if the ORACLE_HOME is locked for patching by any previous unsuccessful OPatch Session. |
| CheckInstalledOneOffs | Check if all the patches provided by the user to rollback are present in the given GI/RAC home. |
| CheckMinimumOPatchVersion | Check if all the patches provided by the user satisfy the requirement of minimum OPatch version for the OPatch currently being used. |
| CheckOneOffSuperset | Check if the given input Fusion Applications patch list are all candidates for one-off (or) singleton supersets. This prereq does not do any checks among the input patch list. |
| CheckOracleHome | Check if the given ORACLE_HOME is valid. Check if it has the inventory files with proper permissions. |
| CheckOraInstLocation | Check if the oraInst.loc file is proper and has the read permissions. |
| CheckOUILocation | Check the ORACLE_HOME for the presence of OUI. |
| CheckOUIVersionCompatible | Check if the OUI in the ORACLE_HOME is compatible for the OPatch. |
| CheckPatchApplicableOnCurrentPlatform | Check if the given patch(es) is applicable on the current platform. |
| CheckPatchApplyDependents | Check if all the patch(es) required by the patch(es) currently being installed is present in the GI/RAC home or not. |
| CheckPatchRollbackDependents | Check if there are any patch(es) in the GI/RAC home that are depending on the patch(es) being currently rolled back. |
| CheckPatchShipHome | Check if the given patch to be applied has the proper structure and has the correct permissions for all the files. |

**Table A-10 (Cont.) `prereq` Commands**

| Command | Description |
|---|---|
| CheckRemoteCommandInvocable | Check if commands can be invoked on the remote machines. |
| CheckRemoteCopyAndRemove | Check if files can be copied to and removed from the remote machines. |
| CheckRequiredLibs | Check if all the required OUI libraries are present in the given ORACLE_HOME. |
| CheckRollbackable | Check if the given patch(es) can be rolled back from the ORACLE_HOME. |
| CheckSystemCommandAvailable | Check if all the commands required for applying or rolling back the given patch are present in the system. |
| CheckSystemSpace | Check if enough system space is available for the patch(es) to be applied. |
| CheckUserAdminPrivilege | Check is the user is 'root'.<br>Note: OPatch should not be invoked by 'root', if so then this check fails. |
| CheckPatchingModel | Check if the patching model of all incoming patch(es) is compatible with that of the GI/RAC home. |

**Table A-11 `prereq` Options**

| Option | Description |
|---|---|
| `connectString` | This option can be used to specify the list of database instances and remote nodes. The value for this option is specified as per the following syntax "SID:User:Passwd:Node". The SID is a must, others can be ignored, OPatch takes default values for it.Example: oracle:dba:dba:mymachine,oracle1:::NOTE: If the system is not part of RAC setup, then to specify just the local node, provide the node name as empty string. This option cannot be used along with 'sid' option. |
| id | This option can be used to specify the patch IDs of all the patches that are to be rolled back from the given GI/RAC home. |
| invPtrLoc | Used to locate the oraInst.loc file. Needed when the installation used the -invPtrLoc flag. This should be the path to the oraInst.loc file. |
| jre | This option tells OPatch to use JRE (java) from the specified location instead of the default location under GI/RAC home. |
| local_node | This option can be used to specify to OPatch the local node name to be used for RAC mode application of the patch. |
| oh | The GI/RAC home to work on. This takes precedence over the environment variable ORACLE_HOME. |
| ph | This option can be used to specify the path to the patch location. Example: /tmp/101010 |
| phBaseDir | This option can be used to specify the path to the base directory where all the patches to be applied are kept. Note: The directory should contain only non-duplicate patches in zipped or unzipped format. |

**Table A-11 (Cont.) `prereq` Options**

| Option | Description |
|---|---|
| phBaseFile | This option can be used to specify complete path to the file containing the location of the patches to be applied. |
| property_file | The user defined property file for OPatch to use. The path to the property file should be absolute. This property file takes precedence over the one that is supplied with OPatch. |
| sid | This option can be used to specify the SIDs of the database instances. This option can be used only for local system operations. |

## util

The `util` command invokes the chosen utilities on an ORACLE_HOME. This command does not support System Patches.

### Syntax

```
opatch util [ -help ] [ COMMAND ]
```

Run `opatch util [ COMMAND ] -help` to get help on a specific command.

**Table A-12 `util` Commands**

| Option | Description |
|---|---|
| `CheckMinimumOpatchVersion` | Check if a patch is compatible with the given OPatch version. |
| CheckComponents | Check if the given patch is suitable for the product (e.g., Fusion Middleware) homes registered in the Central Inventory by components check. |
| Cleanup | Remove the backup for restore area of the given patch or for all the patches. |
| DisableOnlinePatch | Disable and remove the specified online patch(es) on the given database instances. |
| EnableOnlinePatch | Install and enable the specified online patch(es) on the given database instances. |
| GetPatchLevel | Return the patching level on Local Grid Home. |
| InstallOCM | Install and configure OCM. |
| LoadXML | Prompt for path/name of the XML file, then check if the XML is correct. |
| SaveConfigurationSnapshot | Save configuration snapshot of current GI home to specified file. ORACLE_HOME shall point to GI home. Default snapshot file is ORACLE_HOME/cfgtoollogs/opatch/sysconfig/configData.txt |
| UpdateOPatchVersion | Update the version of OPatch in the inventory of GI/RAC home. |
| Verify | Using the defined ORACLE_HOME and given patch location via -ph, the program will check to make sure the patch was applied to the ORACLE_HOME.Example: 'opatch util verify -ph /tmp/patchLoc' |

# B

# OPatchAuto Syntax and Commands

This appendix provides a comprehensive listing and description of all OPatchAuto commands.

> **Note:**
>
> OPatchAuto commands must be run from the GI Home.

## OPatchAuto Commands

The OPatchAuto commands are run from the product home out of the standard `OPatch` directory.

Example:

```
$PRODUCT_HOME/OPatch/opatchauto apply <PATH_TO_PATCH_DIRECTORY>
```

where <PATH_TO_PATCH_DIRECTORY> is the full path to local staging area where you have downloaded your patches.

OPatchAuto consists of four primary commands:

- apply
- resume
- rollback
- version

And one global argument:

- -help

### apply

Apply a System Patch to a product home. User specified the patch location or the current directory will be taken as the patch location.

**Important**: OPatchAuto must be run from the product home as a root user.

**Syntax**

```
opatchauto apply  <patch-location>
    [ -phBaseDir <patch.base.directory> ]
    [ -oh <home> ]
    [ -log <log> ]
    [ -logLevel <log_priority> ]
    [ -binary ]
    [ -analyze ]
```

```
    [ -invPtrLoc <inventory.pointer.location> ]
    [ -host <host> ]
    [ -wallet <wallet> ]
    [ -rolling ]
    [ -database <database> ]
    [ -generatesteps ]
    [ -norestart ]
    [ -ocmrf ocmrf> ]
    [ -sdb ]
       [ -shardgroup <shardgroup.name> ]       [ -shardspace
<shardspace.name> ]       [ -dg <dataguard.name> ]
      [ -nonrolling ]
```

### Parameters

**patch-location**: The patch location.

### Options

The following table describes the options available for the `apply` command.

**Table B-1 `apply` Command Options**

| Option | Description |
|---|---|
| phBaseDir | The location of base patch directory. |
| oh | The location of the oracle home. |
| log | The log location. |
| logLevel | The log level (defaults to "INFO").<br>Supported values: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, ALL, OFF |
| binary | Forces execution of *-phases offline:binary-patching*. |
| analyze | If this option is selected, the environment will be analyzed for suitability of the patch on each home without affecting the home. The patch will not be applied or rolled back and targets will not be shut down. |
| invPtrLoc | The central inventory pointer file location. |
| host | The remote host or *host:port*. |
| wallet | The location of the wallet file. |
| rolling | Enables sharded database (*sdb*) rolling mode where database(s) are patched one after the other. |
| database | List of databases to be patched. |
| generatesteps | Enables generation of steps. |
| norestart | The no restart option during execution. |
| ocmrf | Location of the Oracle Configuration Manager response file (ocmrf). |
| sdb | To signify patching sharded database.<br>For more information, run `opatchauto <apply\|rollback> -sdb -help` to get more help on patching a sharded database. |
| shardgroup | This is used to restrict patching to databases of the selected shard group. |
| shardspace | This is used to restrict patching to databases of the selected shard pace. |

**Table B-1 (Cont.) `apply` Command Options**

| Option | Description |
|---|---|
| dg | This is used to restrict patching to databases of the selected DataGuard. All the standby databases of the DataGuard are patched first, followed by its primary database. |
| nonrolling | Enables non-rolling mode. |
| force_conflict | If a conflict exist which prevents the patch from being applied, the -force_conflict flag can be used to force application of the patch. OPatch will remove all the conflicting patches before applying the current patch. It is applicable to only apply operation. |

**Notes:**

If `opatchauto apply` is run and encounters an individual patch within a patch set that cannot be installed, that patch will be skipped and OPatchAuto will continue with the installation of the next patch in the sequence.

If `opatchauto apply` is run and encounters an individual patch that is identical (same patch ID and Unique Patch Identifier (UPI)) to a patch already installed in the product home, OPatchAuto perform the following based on specific patch conditions:

- If the individual patch was created later than the product home patch, OPatchAuto installs the individual patch.
- If the individual patch's creation date is the same as the product home patch, OPatchAuto will skip installing the individual patch.
- If the individual patch was created before the product home patch, an error will be generated.

This `analyze` option simulates an `OPatchAuto apply` session by running all prerequisite checks, when possible, without making changes to the system (either bits or configurations). Because the `analyze` command does not modify the system, it will perform the following checks:

- Run SQL sync in analyze mode.
- Validate all pre and post processing steps making sure the command is present and executable.

## resume

This operation resumes a previous patching session.

**Important**: OPatchAuto must be run from the GI Home as a root user.

### Syntax

```
opatchauto resume [ -log <log> ] [ -logLevel <log_priority> ]
     [ -password <password> ]
     [ -walletPassword <wallet.password> ]
     [ -session <session> ]
```

### Options

The following table describes the options available for the `resume` command.

**Table B-2 `resume` Command Options**

| Option | Description |
|---|---|
| log | The log location. |
| logLevel | The log level (defaults to "INFO"). |

## rollback

This operation rolls back a patch.

**Important**: OPatchAuto must be run from the product home as a root user.

### Syntax

```
opatchauto rollback [ <patch-location> ]
    [ -phBaseDir <patch.base.directory> ]
    [ -oh <home> ]
    [ -log <log> ]
    [ -logLevel <log_priority> ]
    [ -binary ]
    [ -analyze ]
    [ -invPtrLoc <inventory.pointer.location> ]
    [ -host <host> ]
    [ -wallet <wallet> ]
    [ -rolling ]
    [ -database <database> ]
    [ -generatesteps ]
    [ -norestart ]
    [ -sdb ]
    [ -nonrolling ]
```

### Parameters

**patch-location**: The patch location.

### Options

The following table describes the options available for the `rollback` command.

**Table B-3 `rollback` Command Options**

| Option | Description |
|---|---|
| phBaseDir | The location of base patch directory. |
| oh | The location of the oracle home. |
| log | The log location. |
| logLevel | The log level (defaults to "INFO").<br>Supported values: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, ALL, OFF |
| binary | Forces execution of *-phases offline:binary-patching.* |
| analyze | If this option is selected, the environment will be analyzed for suitability of the patch on each home without affecting the home. The patch will not be applied or rolled back and targets will not be shut down. |

**Table B-3 (Cont.) `rollback` Command Options**

| Option | Description |
|---|---|
| invPtrLoc | The central inventory pointer file location. |
| host | The remote host or *host:port*. |
| wallet | The location of the wallet file. |
| rolling | Enables sharded database (*sdb*) rolling mode where database(s) are patched one after the other. |
| database | List of databases to be patched. |
| generatesteps | Enables generation of steps. |
| norestart | The no restart option during execution. |
| sdb | To signify patching sharded database.<br>For more information, run `opatchauto <apply\|rollback> -sdb -help` to get more help on patching a sharded database. |
| nonrolling | Enables non-rolling mode. |

## help

You can view online help for any command by specifying the `-help` or `-h` option.

For example:

```
opatchauto -help
opatchauto -version
opatchauto apply -help
opatchauto resume -help
opatchauto rollback -help
```

## version

Print the version of the OPatch utility, dependent OPlan version, and the osysmodel version.

**Important**: OPatchAuto must be run from the GI Home as a root user.

### Syntax

```
<GI_HOME>OPatch/opatchauto version
```

# Glossary

**Bundle Patch**

A bundle patch is a cumulative collection of fixes for a specific product or component. A patch of this type is released as needed depending on the product's requirements. You may also know a bundle patch as: maintenance pack, service pack, MLRs, cumulative patch, or update release.

**Diagnostic Patch**

A diagnostic patch is designed to help diagnose or verify a fix or collection of bug fixes. You may also know a diagnostic patch as: test patch, Fix Verification Binary (FVB) or e-fix.

**Interim Patch**

An interim patch provides a single bug fix, a collection of bug fixes, or a customer-specific security fix. They generally address specific bugs for a particular customer, and generally should not be applied unless instructed by Oracle Support to do so. You may also know an interim patch as: security one-off, exception release, x-fix, PSE, MLR, or hotfix.

**MLR**

Merge Label Request. A bundle of patches fixing several bugs.

**Patch Set**

The main way in which Oracle provides bug fixes in between releases. Oracle bundles a number of fixes, test them thoroughly together, and package them together for easy download and installation. They generally do not include new functionality and do not require a new certification. All of the fixes in the patch set have been tested and are certified to work with each other.

**Patch Set Update**

A collection of proactive, stabilizing cumulative patches for a particular product version (base release or patch set). PSUs are cummulative and include all of the security fixes from SPU patches (formerly known as CPU),plus additional fixes.

**Security Patch Update**

A security patch update is a cumulative collection of security-related bug fixes. Generally, security patch updates are release regularly. The security patch update was previously known as Critical Patch Update or CPU.

**Singleton**

A patch with one bug fix.

**Patch**

A patch is a piece of code/software designed to fix problems with the existing code/ software. This includes fixing security vulnerabilities and other bugs, and improving the usability or performance.

**Patch Conflict**

If a patch makes different changes to the same section of code that another OPatch modifies, then these two patches conflict, and only one of them can be installed (unless a merge or overlay patch is available).

**Superset Patch**

If a particular patch to be applied contains all of the fixes included in an already installed patch, plus additional fixes, then the patch with more fixes is a superset patch, and there is no conflict.

**Combination Conflict**

If a patch to be installed conflicts with more than one already installed patch, this is considered a combination conflict. In this case, OPatch will remove all conflicting patches then apply only the new patch.

**Critical Patch Update**

Critical Patch Updates (CPUs) are the primary means of releasing security fixes for Oracle products. CPUs are cumulative with respect to prior CPUs and generall contain only security fixes.

### Merge Patch

A merge patch is one where multiple conflicting patches are combined into one integrated patch.

### Overlay Patch

When an interim patch conflicts with a PSU, patch conflict resolution is achieved by providing a new patch that coexists with (and requires) the PSU patch. The new patch overlays the PSU, and the PSU is a pre-requisite for the overlay patch.

### Shared/Non-shared (GI or RAC) Home

In a shared GI/RAC home, all nodes in the cluster use the same physical copy of the software. This simplifies configuration and management of many database operations because there is a single Home location rather than separate Homes on each node. When a GI Home or RAC Home is shared, individual nodes within the GI or RAC environments share a single file system and utilize a cluster file system such as Oracle Cluster File System 2 (OCFS2), in addition to sharing the same Home. Although this configuration is more disk space-efficient, the process of patching becomes a bit more complicated as the different nodes are utilizing the same resources/disk space.

**Note**: GI shared home installations can be patched only in non-rolling mode.

In a non-shared GI/RAC home, sometimes referred to as Private GI/RAC home, each node in the cluster maintains a complete copy of the Oracle software tree on local storage. This is the most common way in which Oracle Grid Infrastructure and Real Application Clusters are installed.

# Index

## R



## S



## T