# Upgrade Guide for Cisco Catalyst 9000 Series Switches and Smart Switches

**Cisco C9350 Series Smart Switches**
Updated August 1, 2026

© 2026 Cisco Systems, Inc. All rights reserved.

# Full Cisco Trademarks with Software License

Full Cisco Trademarks with Software License

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

# Topics included


**1 Overview....5**

Deployment scenarios....6
Upgrade modes....6
Upgrade summary....7

**2 Pre-upgrade Readiness and Validation....9**

Pre-upgrade workflow....10
Limitations and known issues for device upgrade and recovery....11
Pre-upgrade checklist....12
Perform pre-upgrade validation and remediation....13
Command reference to validate system readiness....15
Additional References....16

**3 Upgrading Devices....17**

Upgrade workflow....18
Device upgrade process....19
Upgrade the device in install mode....19
Upgrade the device in bundle mode....23
Upgrade the device in ISSU mode....26
Upgrade the device in ISSU mode: one-step procedure....26
Upgrade the device in ISSU mode: three-step procedure....29
Upgrade the device using xFSU....31

**4 Post-Upgrade Validation....35**

Post-upgrade workflow....36
Recommendations for device health check....37
Command set for system and application health checks....39
Verify install state, ISSU state, and redundancy....41
Verify CPU, memory, and control-processor health....44
Verify hardware and traffic stability....46
Verify physical StackWise health....49
Verify StackWise Virtual system health....50
Verify network services and application recovery....51
Verify application-hosting health....53
Additional references....54

**5 Troubleshooting Device Upgrades....57**

Operational boundaries and diagnostic reference....58
Guidelines for troubleshooting....58
Common upgrade failure symptoms and resolution reference....59
First-response commands for troubleshooting upgrade issues....60
Determine the actual boot and install state....60
Troubleshoot image-copy and file-validation issues....62
Troubleshoot normal install-mode upgrade issues....66
Troubleshoot legacy bundle-mode issues....71
Troubleshoot ISSU issues....73
Troubleshoot xFSU issues....77
Troubleshoot SMU issues....79
Troubleshoot stack, supervisor, and synchronization issues....81
Troubleshoot a device that boots the previous release....83
Recover from ROMMON or bootloader failures....86
Troubleshoot post-upgrade system or service issues....87
Troubleshoot Catalyst Center SWIM upgrades....90
Technical support criteria....91
Collect diagnostics data....91
Additional references....93

**6 Rolling Back Device Upgrades....95**
Rollback conditions and recovery actions....96
Perform device rollback and recovery....96

# 1 Overview

**Topics:**


- Deployment scenarios
- Upgrade modes
- Upgrade summary

## Deployment scenarios

Details the supported deployment scenarios for Cisco Catalyst 9000 series switches, including standalone operation, stacking, StackWise Virtual, and dual supervisor or high availability options.

This topic details the supported deployment scenarios for Cisco Catalyst 9000 series switches, enabling you to select the appropriate configuration—standalone, stacking, StackWise Virtual, or dual supervisor or high availability—for your network requirements.

#### Standalone

A single Catalyst 9000 switch deployed as an independent access, distribution, or core switch when chassis- or stack-level redundancy is not required.

#### Stack

A group of Catalyst 9200 or 9300 switches connected with StackWise and managed as one logical switch to scale access ports and simplify operations.

#### StackWise Virtual

Two supported Catalyst 9400, 9500, or 9600 switches clustered as one logical switch for simplified management, MEC, and high availability. StackWise Virtual deployment also supports quad supervisors.

#### Dual supervisor or High Availability

A modular Catalyst 9400 or 9600 chassis with redundant supervisors using NSF or SSO to maintain forwarding during supervisor switchover.

## Upgrade modes

Details the supported upgrade modes for Cisco Catalyst 9000 series switches, including bundle and install modes, in-service software upgrade, and extended fast software upgrade.

This topic explains the different upgrade modes available for Cisco Catalyst 9000 series switches to help you choose and implement the appropriate upgrade method for your deployment scenario.

#### Bundle mode

Bundle mode boots the switch directly from the monolithic Cisco IOS XE .bin image, with packages extracted to RAM at boot. The boot variable points directly to the .bin file. For stack, dual-supervisor, or StackWise Virtual scenarios, copy the image to every required flash location before reloading.

#### Install mode

Install mode expands the .bin image into packages and uses packages.conf to identify the package set. The boot variable must point to packages.conf, and the install workflow can add, activate, commit, abort, and roll back versions.

#### In-service software upgrade

In-service software upgrade (ISSU) upgrades the software while the system continues forwarding traffic. It is supported only in install mode on eligible HA designs such as StackWise Virtual or dual-supervisor systems.

**Extended fast software upgrade**

Extended fast software upgrade (xFSU) for Catalyst 9300 stackable access switches, designed to reduce upgrade downtime by completing as much of the upgrade as possible before the data plane is taken offline.

## Upgrade summary

Outlines the supported upgrade modes for each platform, and provides comparative details and platform-specific notes to guide upgrade selection and workflow optimization.

This topic summarizes the supported upgrade modes for each platform in Cisco Catalyst 9000 series switches, and provides comparative details and platform-specific notes to guide upgrade selection and workflow optimization.

Cisco Catalyst 9000 series switches support Cisco IOS XE in either bundle mode or install mode. Install mode is recommended because it expands the .bin image into individual packages and boots through `packages.conf`, enabling faster boot times, improved memory utilization, rollback support, and modern install workflows.

**Table 1: Supported upgrade modes and key platform notes**

| Platform | Supported upgrade modes | Key platform notes |
|---|---|---|
| Catalyst 9200 | • Bundle mode<br>• Install mode | • ISSU is not supported.<br>• Bundle mode and install mode can move directly from 16.x to 17.x or within 17.x, subject to target release notes. |
| Catalyst 9300 | • Bundle mode<br>• Install mode<br>• xFSU | • ISSU is not supported.<br>• xFSU can be used in eligible install-mode scenarios to reduce downtime. |
| Catalyst 9400 | • Bundle mode<br>• Install mode<br>  • ISSU | ISSU is supported for eligible High Availability, dual-supervisor, quad-supervisor, and StackWise Virtual deployments. |
| Catalyst 9500 | • Bundle mode<br>• Install mode<br>  • ISSU | ISSU is supported for eligible StackWise Virtual deployments, subject to release compatibility. |
| Catalyst 9600 | • Bundle mode<br>• Install mode<br>  • ISSU | ISSU is supported for eligible dual-supervisor, quad-supervisor, and StackWise Virtual deployments. |

# 2 Pre-upgrade Readiness and Validation

**Topics:**

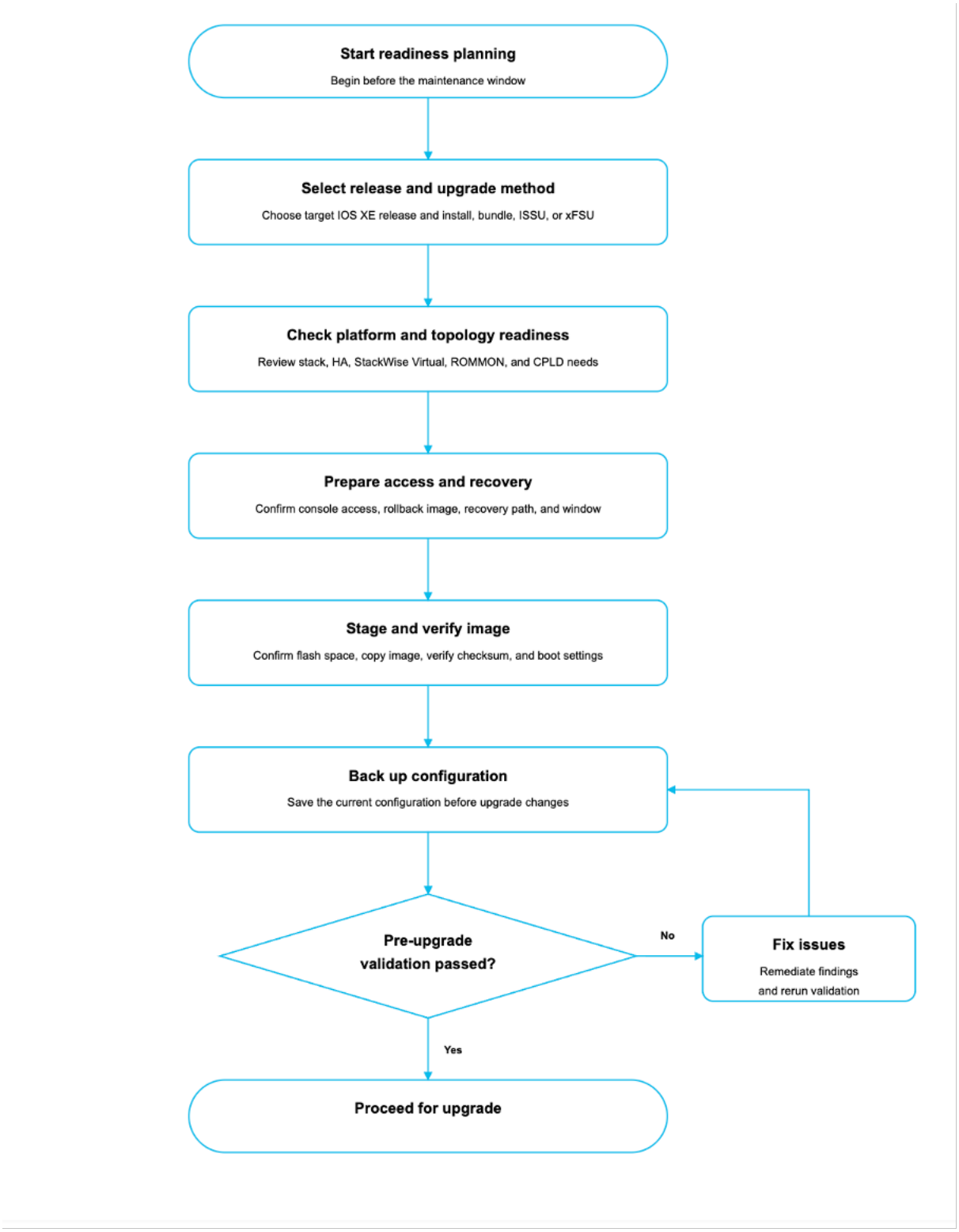

- Pre-upgrade workflow
- Limitations and known issues for device upgrade and recovery
- Pre-upgrade checklist
- Perform pre-upgrade validation and remediation
- Command reference to validate system readiness
- Additional References

## Pre-upgrade workflow

Details the pre-upgrade workflow, illustrating each step required before initiating an upgrade.

This topic provides the sequence of steps in the pre-upgrade workflow to help you properly prepare your systems before starting an upgrade.

**Figure 1: Pre-upgrade workflow**




## Limitations and known issues for device upgrade and recovery

Provides restrictions and operational requirements for software upgrades and system recovery processes. These guidelines ensure system stability and prevent service interruptions during maintenance activities.

Follow these guidelines when performing system upgrades:

- Legacy **request platform software install** commands are supported only for Catalyst 9300 and 9500 workflows and are deprecated as of Cisco IOS XE 16.10.1.
- Auto-upgrade is disabled by default, is not available in bundle mode, and does not recover a supervisor from ROMMON.
- Booting from USB or TFTP in install mode is not supported; ROMMON or bundle-mode recovery can use USB or TFTP boot paths.
- On Catalyst 9400 switches, manually initiate the supervisor CPLD upgrade if mandated in the target release notes; CPLD auto-upgrade is not supported. For other Catalyst 9000 series switches, follow the platform-specific firmware upgrade instructions. Do not perform the Catalyst 9400 switches CPLD procedure in other series of switches.
- Do not use xFSU if the upgrade requires a microcode or FPGA update, or if the topology does not meet documented uplink requirements.

## Pre-upgrade checklist

Details the essential checklist items and actions that must be completed before initiating an upgrade, ensuring that all prerequisites for device readiness are systematically addressed.

This topic outlines the pre-upgrade checklist for Catalyst 9000 series switches, including image and upgrade method selection, platform and topology validation, maintenance and recovery planning, image and boot configuration, and ISSU-specific preparation steps.

For a quick reference to the recommended Cisco IOS XE releases, upgrade checks, upgrade steps generation, and ROMMON upgrade guidance, refer the Cisco Catalyst 9K Upgrade Compatibility Matrix tool.

### Release and upgrade method selection

- Select the target image using Cisco recommended release guidance and product-specific release notes. Refer these documents:
  - Recommended Releases for Catalyst 9200/9300/9400/9500/9600 Platforms
  - Recommended Releases for Cisco C9350/C9610 Series Smart Switches
- Confirm the platform-specific upgrade method: install mode, bundle mode, ISSU, or xFSU. Prefer install mode for modern Catalyst 9000 upgrade workflows.

### ISSU readiness

Confirm ISSU compatibility by verifying that the platform, topology, and source-to-destination release pair are supported in the ISSU Compatibility Matrix. Also, refer In-Service Software Upgrade (ISSU) Release Matrix.

### Platform and topology readiness

- For Catalyst 9200 and 9300 stacks, validate image placement, stack member state, and version consistency across all members.
- For Catalyst 9400, 9500, and 9600 High Availability or StackWise Virtual deployments, verify redundancy state and plan supervisor or chassis reload behavior.
- For Catalyst 9400, review ROMMON and CPLD requirements in the target release notes before the upgrade. For instance, CPLD upgrades are manual and might be traffic-impacting. Refer ROMMON and CPLD versions.

**Maintenance, access, and recovery readiness**

- Reserve a 2–3-hour maintenance window and include rollback time.
- Take a backup of the latest configuration from the device.
- Ensure console access, management reachability, and privileged EXEC or configuration access are available during the maintenance window.
- Keep a validated rollback image and a reachable recovery path, such as TFTP or a FAT32-formatted USB drive with enough capacity for the IOS XE image and recovery files.
- Arrange physical access when USB recovery, stack re-cabling, or an individual member power cycle may be required.
- Use cleanup commands carefully because inactive cleanup can remove critical .bin files and rollback points.

**Image, flash, and boot readiness**

- Download the target image from Cisco Software Download Center for the exact switch model and verify the checksum using the **verify /md5** command after the transfer.
- Keep both the current and target IOS XE images available during the maintenance window.
- Verify that flash has enough free space for the new image and package expansion. Remove inactive packages only after confirming that required rollback files are no longer needed.
- Before proceeding with an upgrade, confirm that the boot variable is set correctly and the boot manual state is disabled.
- In install mode, confirm that the boot variable points to `packages.conf`, using the platform-specific path shown in **show boot** or **show bootvar** outputs. Do not manually edit `packages.conf` or change the boot statement to point to an inactive package during an install operation.
- In bundle mode, confirm that the switch boots from the `monolithic .bin` image and that the image is present on each required member or supervisor.

## Perform pre-upgrade validation and remediation

Provides instructions for performing pre-upgrade validation and remediation, including saving the current configuration, verifying and freeing up Flash space, checking boot and ROMMON variables, and acquiring and validating the software image required for upgrade.

Use this procedure to perform validation steps prior to the upgrade process, emphasizing key checks to confirm compatibility and operational status before proceeding.

**Procedure**

1. Save the current configuration before changing boot variables or initiating install operations.

   **Example**

   ```
   Switch# write memory
   ```

2. Check and free up Flash space on the device.

**Example**

```
Switch# show file systems
Switch# dir flash: | include free
Switch# install remove inactive
```

Use show file systems and platform-specific **dir** commands to verify available space. Clean inactive files when needed, noting that cleanup can remove .bin files.

3. Check if the system boot variables are rightly set for each upgrade mode. Also confirm if auto-boot is enabled.

**Example**

```
Switch# show boot system
Current Boot Variables:
BOOT variable = flash:packages.conf;

Boot Variables on next reload:
BOOT variable = flash:packages.conf;

Manual Boot = no
```

Here, both the current boot variable and the boot variable for the next reload point to `flash:packages.conf` which is set for install mode upgrade. To set the boot variables, refer the upgrade procedure for respective upgrade mode.

The `Manual Boot = no` output indicates that auto-boot is enabled.

4. Check ROMMON variables.

**Example**

```
Switch# show romvar
-----------------------------------

ROMMON variables for Active Switch

-----------------------------------

ABNORMAL_RESET_COUNT=0
AUTO_SWITCH_CONSOLE_DISABLE=0
BAUD=9600
BOARDID=20608
BOOT=flash:packages.conf;
BOOTLDR=
BSI=0
CALL_HOME_DEBUG=0000000000000
CONFIG_FILE=
CONSOLE_MIRRORING_DISABLE=0
CSDL_ENTROPY_REQUIREMENT_DISABLE=
CSDL_MODE_DISABLE=
DC_COPY=yes
DISABLE_QUACK_AUTH=1
D_STACK_DAD=
D_STACK_DISTR_STACK_LINK1=
D_STACK_DISTR_STACK_LINK2=
D_STACK_DOMAIN_NUM=
```

```
D_STACK_MODE=
ENABLE_BREAK=yes
FIPS_MODE=
LICENSE_BOOT_LEVEL=network-advantage+dna-advantage,all:C9500_12Q;
MAC_ADDR=00:BF:77:B8:14:80
MANUAL_BOOT=no
MCP_STARTUP_TRACEFLAGS=00000000:00000000
MODEL_NUM=C9500-12Q
MODEL_REVISION_NUM=B0
MOTHERBOARD_ASSEMBLY_NUM=73-18139-02
MOTHERBOARD_REVISION_NUM=A0
MOTHERBOARD_SERIAL_NUM=FOC22017AFE
RANDOM_NUM=1056193287
RECOVERY_RELOAD_DISABLE=
RET_2_RCALTS=
RET_2_RTS=
ROMMON_AUTOBOOT_ATTEMPT=3
ReloadReason=Image Install RET_2_RCALTS=1787576730
SWITCH_IGNORE_STARTUP_CFG=0
SWITCH_NUMBER=1
SWITCH_PRIORITY=15
SYSTEM_SERIAL_NUM=FCW2202A5V8
TAG_ID=E2003412015F1D000066DD8E
TEMPLATE=distribution
USB_DISABLE=0
VERSION_ID=V01
```

ROMMON-related validation is important especially when release notes mandate a ROMMON update or ISSU requires matching firmware.

5. Acquire and validate the image.

   **Example**

   ```
   Switch# copy tftp://Location/directory/<file_name>.bin flash:
   Switch# copy usbflash0:<file_name>.bin flash:
   Switch# verify /md5 flash:<file_name>.bin
   ```

   Specify the exact image location.

6. Run the checks that apply to the platform, topology, and selected upgrade method before activating the image.

   Refer Command reference to validate system readiness on page 15 for the list of commands.

## Command reference to validate system readiness

Lists the commands used to validate system readiness prior to image installation or activation, including verification of platform, topology, release version, boot mode, file systems, image integrity, redundancy, ISSU state, and module health.

This topic lists the commands to validate system readiness prior to image installation or activation, including verification of platform, topology, release version, boot mode, file systems, image integrity, redundancy, ISSU state, and module health.

**Table 2: List of commands to validate system readiness**

| Verify | Command |
|---|---|
| Current release | **show version \| include IOS XE** |
| Boot mode | **show version \| include INSTALL** |
| Boot variable | **show boot system or show boot** |
| File systems/free space | **show file systems** |
| Image presence | **dir flash: or dir bootflash:** |
| Image integrity | **verify /md5 flash:<***file_name***>.bin****verify /md5 flash:***file_name***.bin** |
| Redundancy state | **show redundancy** |
| Stack setup | **show switch** |
| ISSU state | **show issu state detail** |
| Install state | **show install summary** |
| Module health | **show module** |
| Package pointer | **more flash:packages.conf \| include pkg** |

## Additional References

Guides you to recommended Cisco IOS XE releases, upgrade checks, step generation, and ROMMON upgrade instructions using the Cisco Catalyst 9K Upgrade Compatibility Matrix tool.

For a quick reference to the recommended Cisco IOS XE releases, upgrade checks, upgrade steps generation, and ROMMON upgrade guidance, refer the Cisco Catalyst 9K Upgrade Compatibility Matrix tool.

# 3 Upgrading Devices

**Topics:**


- Upgrade workflow
- Device upgrade process
- Upgrade the device in install mode
- Upgrade the device in bundle mode
- Upgrade the device in ISSU mode
- Upgrade the device using xFSU

## Upgrade workflow

Outlines the available upgrade methods and illustrates the upgrade workflow.

This topic guides you through each step of the upgrade workflow.

**Figure 2: Upgrade workflow**

# Device upgrade process

Outlines the upgrade process, highlighting key processes such as image extraction, boot partition updates, firmware installation, and manual firmware component upgrades to build foundational understanding, including guidance for handling placeholders and understanding sample outputs during system upgrades.

This topic outlines the upgrade process, highlighting key processes such as image extraction, boot partition updates, firmware installation, and manual firmware component upgrades to build foundational understanding. It also provides guidance for handling placeholders and understanding sample outputs during system upgrades.

### Upgrade sequence for components

- Software image is copied and validated first.
- Install mode expands the packages and updates the `packages.conf` file during the activation phase. Do not edit it manually.
- Primary ROMMON or bootloader is automatically upgraded on first boot when a newer ROMMON is included and the hardware requires it. It is not downgraded when reverted.
- Golden ROMMON or backup bootloader must be manually upgraded with the **upgrade rom-monitor capsule golden switch** command in privileged EXEC mode when required. In case of a switch stack, perform the upgrade on the active switch and all members of the stack.
- For Catalyst 9400, CPLD upgrade is manual after the software image upgrade and causes a supervisor power cycle and traffic disruption. For details, refer ROMMON and CPLD versions.

### Firmware or image installation

The device upgrade procedures listed in this chapter cover these upgrade modes:

- Install mode
- Bundle mode
- ISSU
  - One-step ISSU
  - Three-step ISSU with rollback option
- xFSU

The sample outputs given in the upgrade procedure sections are truncated and for illustrative purpose only. Replace all placeholders with values for the exact platform and release. Lines beginning with '>>>' are meant to describe the output and are not generated by the device.

# Upgrade the device in install mode

Guides you through upgrading supported Catalyst 9000 series switches or stacks in install mode.

Use this procedure to upgrade a Catalyst 9000 switch or switch stack in install mode.

When executing the commands and verifying the outputs, replace *CURRENT_RELEASE*, *TARGET_RELEASE*, *IMAGE.bin*, and *SHA-512-VALUE* values in the sample outputs with the actual values of your deployment.

### Before you begin

For switch stacks, verify that all members report the expected software state before and after the upgrade.

**Procedure**

1. Verify the current release, installation state, boot variables, and stack readiness before the upgrade.
   a) Verify that the switch is running from the INSTALL-mode packages file and identify the current Cisco IOS XE release.

      **Example**

      ```
      switch# show version | include IOS XE Software|System image file
      Cisco IOS XE Software, Version <CURRENT_RELEASE>
      System image file is "flash:packages.conf"
      ```

      The `System image file` value must be `flash:packages.conf`. The `packages.conf` file confirms that the switch is running in INSTALL mode.

   b) Verify that all stack members are in INSTALL mode and running the same software release version.

      **Example**

      ```
      switch# show version | begin Switch Ports
      Switch Ports Model              SW Version       SW Image       Mode
      ------ ----- -----              ----------       --------       ----
      *    1 48    C9300-48P          <CURRENT>        CAT9K_IOSXE    INSTALL
           2 48    C9300-48P          <CURRENT>        CAT9K_IOSXE    INSTALL
      ```

      Every stack member must show `INSTALL` in the `Mode` column and must show the same release in the `SW Version` column.

   c) Confirm that the current image is activated and committed.

      **Example**

      ```
      switch# show install summary
      State (St): I - Inactive, U - Activated & Uncommitted,
                  C - Activated & Committed
      --------------------------------------------------------------------------
      Type  St   Filename/Version
      --------------------------------------------------------------------------
      IMG   C    <CURRENT_RELEASE>

      Auto abort timer: inactive
      ```

      A state value of `C` means that the current image is activated and committed.

      The `Auto abort timer: inactive` output confirms that no previous uncommitted installation or rollback timer is active.

   d) Confirm that the current boot variables and boot variables on next reload point to the INSTALL-mode package file. If not, set it accordingly.

      **Example**

      ```
      switch# show boot
      Current Boot Variables:
      ```

```
BOOT variable = flash:packages.conf;

Boot Variables on next reload:
BOOT variable = flash:packages.conf;

Manual Boot = no
```

Here, both the current boot variable and the boot variable for the next reload point to `flash:packages.conf`. The `Manual Boot = no` output indicates that auto-boot is enabled.

**Example**

If the boot variables were not rightly set, follow these steps to set it and to disable manual boot:

```
Switch(config)# no boot system
Switch(config)# boot system flash:packages.conf
Switch(config)# no boot manual
```

2. Validate the target image by verifying its SHA-512 checksum.

   **Example**

   ```
   switch# verify /sha512 flash:<IMAGE.bin>
   verify /sha512 (flash:<IMAGE.bin>) = <SHA-512-VALUE>
   ```

   The returned checksum value must exactly match the SHA-512 value that is posted on Cisco Software Download site.

3. Save the running configuration before you start the installation.

   **Example**

   ```
   switch# write memory
   Building configuration...
   [OK]
   ```

   The `[OK]` message confirms that the configuration was saved successfully.

4. Add, activate, and commit the target image as the persistent software version.

   **Example**

   ```
   switch# install add file flash:<IMAGE.bin> activate commit
   install_add_activate_commit: START

   --- Starting initial file syncing ---
   Copying flash:<IMAGE.bin> to Switch 1 2
   Info: Finished copying to the selected Switch
   >>> Indicates that the target image was copied to the required stack members

   --- Starting Add ---
   Performing Add on all members
    [1] Finished Add package(s) on Switch 1
    [2] Finished Add package(s) on Switch 2
   ```

```
Add: Passed on [1 2]
>>> Indicates that the image was validated and expanded successfully on every
 member

--- Starting Activate ---
Performing Activate on all members
Activate: Passed on [1 2]
>>> Indicates that the target packages were activated successfully

--- Starting Commit ---
Performing Commit on all members
Commit: Passed on [1 2]
>>> Indicates that the target image was made persistent for future reloads

SUCCESS: install_add_activate_commit
>>> Indicates that the complete add, activate, and commit operations succeeded

Initializing Hardware......

Preparing to autoboot. [Press Ctrl-C to interrupt]  0
boot: attempting to boot from [flash:packages.conf]
>>> Indicates that the switch is booting in INSTALL mode
...
...
...
>>> [Routine boot messages are omitted ]
```

The installation process copies the target image to the required stack members, validates and expands the packages, activates the target packages, and commits the image for future reloads.

5. Verify the post-upgrade state on the device.
   a) Verify that the switch is running the target Cisco IOS XE release in INSTALL mode after the reload.

      **Example**

      ```
      switch# show version | include IOS XE Software|System image file
      Cisco IOS XE Software, Version <TARGET_RELEASE>

      System image file is "flash:packages.conf"
      ```

      The output must show Cisco IOS XE Software Version *<TARGET_RELEASE>*.

      The `System image file` value must display `flash:packages.conf` which indicates that the switch remains in install mode.

   b) Confirm that the target image is activated and committed after the upgrade.

      **Example**

      ```
      switch# show install summary
      Type  St   Filename/Version
      ---------------------------------------------------------------------
      IMG   C    <TARGET_RELEASE>
      >>> Indicates that the target image is activated and committed

      Auto abort timer: inactive
      ```

      The `IMG` entry must show state `C` for the *<TARGET_RELEASE>*.

The `Auto abort timer: inactive` output confirms that no incomplete installation or rollback timer remains.

c) Confirm that every switch stack member has rejoined and is ready.

**Example**

```
switch# show switch
Switch#  Role      Priority  State
-----------------------------------
*1       Active    15        Ready
 2       Standby   14        Ready
```

Every stack member must show `Ready` in the `State` column. Verify that the expected active and standby roles are present.

---

The Catalyst 9000 switch or switch stack runs the target Cisco IOS XE release in install mode. The target image is activated and committed, no rollback timer remains active, and all stack members are ready.

## Upgrade the device in bundle mode

Guides you through upgrading Catalyst 9000 switches or stacks in bundle mode.

Use this procedure to upgrade a Catalyst 9000 switch or switch stack in bundle mode.

When executing the commands and verifying the outputs, replace *<CURRENT_RELEASE>*, *<TARGET_RELEASE>*, *<IMAGE.bin>*, and *SHA-512-VALUE* values in the sample outputs with the actual values of your deployment.

In Bundle mode, the switch boots directly from a `.bin` image instead of from the INSTALL-mode `packages.conf` file.

**Procedure**

---

1. Identify the current Cisco IOS XE release and verify that the switch is booting directly from a bundle image.

   **Example**

   ```
   switch# show version | include IOS XE Software|System image file
   Cisco IOS XE Software, Version <CURRENT_RELEASE>
   System image file is "flash:<CURRENT_IMAGE.bin>"
   ```

   The `System image file` value must point to a file such as `flash:<CURRENT_IMAGE.bin>`. Booting directly from a `.bin` file indicates bundle mode.

2. Confirm that the switch or stack members are running in bundle mode.

   **Example**

   ```
   switch# show version | begin Switch Ports
   Switch Ports Model              SW Version       SW Image       Mode
   ```

```
------ ----- -----              ----------       --------       ----
*    1 48    C9300-48P          <CURRENT>        CAT9K_IOSXE    BUNDLE
```

The `Mode` column must show `BUNDLE`.

For switch stacks, verify that every member shows bundle mode and the expected software release.

3. Validate the target bundle image.
   a) Confirm that the target bundle image exists in flash.

      **Example**

      ```
      switch# dir flash:<IMAGE.bin>
      Directory of flash:/

      1234567  -rw-  <FILE-SIZE>  <DATE>  <IMAGE.bin>
      ```

      Confirm that the target image exists locally. In a switch stack, the image must exist on every stack member.

   b) Verify the SHA-512 checksum of the target bundle image.

      **Example**

      ```
      switch# verify /sha512 flash:<IMAGE.bin>
      verify /sha512 (flash:<IMAGE.bin>) = <SHA-512-VALUE>
      ```

      The returned SHA-512 checksum must exactly match the value that is posted on official Cisco Software Download Center.

4. Configure the switch to boot from the target bundle image and enable autoboot.

   **Example**

   ```
   switch# configure terminal
   switch(config)# no boot system >>> This clears existing boot statements.
   switch(config)# boot system flash:<IMAGE.bin> >>> This sets the target bundle
    image as the boot image.
   switch(config)# no boot manual >>> This enables autoboot.
   switch(config)# end
   ```

5. Save the new boot configuration.

   **Example**

   ```
   switch# write memory
   Building configuration...
   [OK]
   ```

   The `[OK]` message confirms that the new boot configuration was saved successfully.

6. Verify that the current and next boot variables point to the target bundle image.

**Example**

```
switch# show boot
Current Boot Variables:
BOOT variable = flash:<IMAGE.bin>;

Boot Variables on next reload:
BOOT variable = flash:<IMAGE.bin>;  >>>The next reload will use the target
bundle image


Manual Boot = no >>>Autoboot is enabled
```

Both the current boot variable and the boot variable for the next reload must point to `flash:<IMAGE.bin>`.

The `Manual Boot = no` output confirms that autoboot is enabled.

7. Reload the switch or switch stack to boot from the target bundle image.

**Example**

```
switch# reload
Proceed with reload? [confirm]

Initializing Hardware......

Preparing to autoboot. [Press Ctrl-C to interrupt]  0
boot: attempting to boot from [flash:<IMAGE.bin>] >>> The boot loader selected
 the target .bin image

>>> [Routine boot messages are omitted]
```

Confirm the reload when prompted.

8. Verify that the switch is running the target Cisco IOS XE release after the reload.

**Example**

```
switch# show version | include IOS XE Software|System image file
Cisco IOS XE Software, Version <TARGET_RELEASE>
>>> Confirm that the intended target release is running

System image file is "flash:<IMAGE.bin>" >>> The switch is still booting
directly from the bundle image
```

The output must show Cisco IOS XE Software Version *<TARGET_RELEASE>*. The `System image file` value must continue to point directly to `flash:<IMAGE.bin>`, confirming that the switch is still booting from the bundle image.

9. Confirm that the switch remains in Bundle mode after the upgrade.

**Example**

```
switch# show version | begin Switch Ports
```

```
Switch Ports Model              SW Version       SW Image       Mode
------ ----- -----              ----------       --------       ----
*    1 48    C9300-48P          <TARGET>         CAT9K_IOSXE    BUNDLE
```

The `Mode` column must show `BUNDLE`. This confirms that the switch was upgraded in Bundle mode and was not converted to INSTALL mode.

---

The Catalyst 9000 switch or switch stack runs the target Cisco IOS XE release in bundle mode. The boot variables point to the target bundle image, autoboot is enabled, and the switch was not converted to INSTALL mode.

# Upgrade the device in ISSU mode

## Upgrade the device in ISSU mode: one-step procedure

Guides you through upgrading Catalyst 9000 switch using one-step ISSU workflow, and confirming the post-upgrade redundancy and software state on the device.

Use this procedure to upgrade a Cisco Catalyst 9000 switch using one-step In-Service Software Upgrade (ISSU) workflow–add, activate, and commit in a single step.

When executing the commands and verifying the outputs, replace *<CURRENT_RELEASE>*, *<TARGET_RELEASE>*, *<IMAGE.bin>*, and *SHA-512-VALUE* values in the sample outputs with the actual values of your deployment.

To perform device upgrade in ISSU mode using three-step workflow, refer Upgrade the device in ISSU mode: three-step procedure on page 29

### Before you begin

> **Note**
>
> ISSU is not supported on Cisco Catalyst 9300 series switches.

Proceed only when the ISSU Compatibility Matrix explicitly supports the exact platform, topology, source release, and target release.

#### Procedure

---

1. Verify that SSO redundancy is operating and that the standby control plane is ready.

   **Example**

```
switch# show redundancy
Configured Redundancy Mode = sso
Operating Redundancy Mode = sso      >>> Stateful Switchover is operating

Communications = Up                 >>> The active and standby control planes
 can communicate

Current Software state = ACTIVE
Current Software state = STANDBY HOT >>> The standby is synchronized and ready
```

```
 to assume the active role

BOOT = flash:packages.conf;          >>> ISSU requires INSTALL mode
```

- The `Configured Redundancy Mode` and `Operating Redundancy Mode` values must show `sso`.
- The `Communications = Up` output confirms that the active and standby control planes can communicate.
- The `Current Software state = ACTIVE` and `Current Software state = STANDBY HOT` values confirm that the standby is synchronized and ready to assume the active role.
- The `BOOT = flash:packages.conf;` value confirms that the switch is running in INSTALL mode, which is required for ISSU.

2. Confirm that no ISSU operation is already in progress.

   **Example**

```
switch# show issu state detail
No ISSU operation is in progress
```

   The output confirms that no previous ISSU operation needs to be completed or cleared before the new upgrade starts.

3. Confirm that the current source image is activated and committed.

   **Example**

```
switch# show install summary
Type  St   Filename/Version
-----------------------------------------
IMG   C    <CURRENT_RELEASE>              >>> The source image is activated
 and committed

Auto abort timer: inactive                  >>> No earlier uncommitted
installation is active
```

   The `IMG` entry must show state `C` for <CURRENT_RELEASE>, which means that the source image is activated and committed. The `Auto abort timer: inactive` output confirms that no earlier uncommitted installation is active.

4. Start the one-step ISSU workflow and commit the target image automatically.

   **Example**

```
switch# install add file flash:<MATRIX-APPROVED-IMAGE.bin> activate issu
commit
install_add_activate_commit: START

%INSTALL-5-INSTALL_START_INFO:
 install_engine: Started install one-shot ISSU   >>> The one-step ISSU workflow
 has started and will commit automatically

--- Starting initial file syncing ---
[1]: Copying flash:<IMAGE.bin> from switch 1 to switch 2
[2]: Finished copying to switch 2               >>> The target image is
```

```
available on both ISSU participants

--- Starting Add ---
Add: Passed on [1 2]                              >>> The target image was
validated and expanded successfully

STAGE 0: Initial System Level Sanity Check
========================================================
--- Verifying install_issu supported ---
--- Verifying standby is in Standby Hot state ---
--- Verifying booted from valid media ---
--- Verifying AutoBoot mode is enabled ---
Finished Initial System Level Sanity Check       >>> The initial ISSU
prerequisites passed

STAGE 1: Installing software on Standby
========================================================
install_remote: Passed on [2]                    >>> The target software was
 installed on the standby

STAGE 2: Restarting Standby
========================================================
--- Starting standby reload ---

>>> [Standby reload messages omitted]

%HA_CONFIG_SYNC-6-BULK_CFGSYNC_SUCCEED: Bulk Sync succeeded >>> Configuration
 synchronization with the restarted standby succeeded

%RF-5-RF_TERMINAL_STATE: Terminal state reached for (SSO)   >>> The standby
returned to a stable SSO state

STAGE 3: Installing software on Active
========================================================
install_active: Passed on [1]                     >>> The target
software was installed on the active system

STAGE 4: Restarting Active
========================================================
--- Starting active reload ---                   >>> A switchover
 occurs. The management session can temporarily disconnect

%IOSXEBOOT-4-ISSU_ONE_SHOT: ISSU finished successfully     >>> IOS XE reports
 successful completion of the ISSU sequence

%INSTALL-5-INSTALL_COMPLETED_INFO:
 install_engine: Completed install commit ISSU             >>> The target
image was committed automatically

SUCCESS: install_add_activate_commit                       >>> The complete
 one-step ISSU operation succeeded
```

5. Verify that redundancy returned to a healthy SSO state after the ISSU operation.

**Example**

```
switch# show redundancy | include Operating|Communications|STANDBY
Operating Redundancy Mode = sso
Communications = Up
```

```
Current Software state = STANDBY HOT    >>> Redundancy returned to its normal
 healthy state
```

6. Confirm that the ISSU operation has fully completed.

   **Example**

   ```
   switch# show issu state detail
   No ISSU operation is in progress
   ```

   The output must show `No ISSU operation is in progress`, confirming that the ISSU operation has fully completed.

7. Confirm that the target image is activated and committed after ISSU.

   **Example**

   ```
   switch# show install summary
   Type  St   Filename/Version
   ----------------------------------------
   IMG   C    <TARGET_RELEASE>                    >>> The target image is
   activated and committed

   Auto abort timer: inactive                   >>> No rollback timer remains
    active
   ```

   The `IMG` entry must show state `C` for <TARGET_RELEASE>. The `Auto abort timer: inactive` output confirms that no rollback timer remains active.

---

The Catalyst 9000 switch runs the target Cisco IOS XE release after a successful ISSU operation. Redundancy is operating in SSO mode, no ISSU operation remains in progress, the target image is activated and committed, and no rollback timer remains active.

## Upgrade the device in ISSU mode: three-step procedure

Guides you through upgrading Catalyst 9000 switches or stacks using three-step ISSU workflow–add, activte, and commit performed in three separate steps–providing an option to abort the operation and return to the previously committed image if commit validation fails.

Use this procedure to upgrade a Cisco Catalyst 9000 switch using three-step In-Service Software Upgrade (ISSU) workflow–add, activte, and commit performed in three separate steps. This three-step procedure gives an opportunity to abort the ISSU operation and to return to the previously committed image if commit validation fails after activating the image.

Replace *<CURRENT_RELEASE>*, *<TARGET_RELEASE>*, *<IMAGE.bin>*, and *SHA-512-VALUE* values in the sample outputs with the actual values for your deployment.

To perform device upgrade in ISSU mode using one-step workflow, refer Upgrade the device in ISSU mode: one-step procedure on page 26

**Before you begin**

> **Note**
>
> ISSU is not supported on Cisco Catalyst 9300 series switches.

Proceed only when the ISSU Compatibility Matrix explicitly supports the exact platform, topology, source release, and target release.

**Procedure**

1. Add the image on the device.

   **Example**

   ```
   switch# install add file flash:<IMAGE.bin>
   SUCCESS: install_add
   ```

   The `SUCCESS:` field value `install_add` confirms that the image was added but not yet activated.

2. Activate the image.

   **Example**

   ```
   switch# install activate issu
   SUCCESS: install_activate
   ```

   The `SUCCESS:` field value `install_activate` confirms that the image was activated but not yet comitted.

3. Verify the ISSU state and check if you can rollback to the previously committed image.

   **Example**

   ```
   switch# show issu state detail
   Current state: Activated state
   Auto abort timer: remaining time before rollback: 01:43:55
   ```

   The `Auto abort timer:` output in this example indicates that the rollback opportunity is still available.

4. (Optional) Abort the ISSU operation if validation fails before commit operation.

   **Example**

   ```
   switch# install abort issu
   ```

   The system attempts to return to the previously committed image.

5. Commit the install operation.

**Example**

```
switch# install commit
SUCCESS: install_commit
```

The commit operation makes the target image permanent and ends the normal rollback opportunity.

6. Confirm that the ISSU operation has fully completed.

**Example**

```
switch# show issu state detail
No ISSU operation is in progress
```

The output must show `No ISSU operation is in progress`, confirming that the ISSU operation has fully completed.

7. Confirm that the current source image is activated and committed.

**Example**

```
switch# show install summary
Type  St   Filename/Version
----------------------------------------
IMG   C    <CURRENT_RELEASE>               >>> The source image is activated
 and committed

Auto abort timer: inactive                  >>> No earlier uncommitted
installation is active
```

The `IMG` entry must show state `C` for <CURRENT_RELEASE>, which means that the source image is activated and committed. The `Auto abort timer: inactive` output confirms that no earlier uncommitted installation is active.

---

The Catalyst 9000 switch runs the target Cisco IOS XE release after a successful three-step ISSU operation. Redundancy is operating in SSO mode, no ISSU operation remains in progress, the target image is activated and committed, and no rollback timer remains active.

## Upgrade the device using xFSU

Guides you through verifying xFSU eligibility, performing the upgrade workflow, and confirming successful reload-fast and post-upgrade software state on supported Catalyst 9300 switches running Cisco IOS XE.

Use this procedure to upgrade supported Catalyst 9300 switches that is running Cisco IOS XE by using Extended Fast Software Upgrade (xFSU).

When executing the commands and verifying the outputs, replace *<TARGET_RELEASE>*, *<IMAGE.bin>*, and any site-specific values in this example with the actual values of your deployment.

xFSU is a reduced-downtime upgrade workflow that applies only to supported Catalyst 9300 deployments and supported source-to-target release paths.

**Before you begin**

Follow these guidelines for xFSU upgrade:

- Use xFSU only after confirming the supported platform and release path and when the switch passes the eligibility check.
- xFSU pre-commit abort or rollback workflow is not supported.
- xFSU downgrade is not supported.
- After a successful committed xFSU operation, returning to an older image is a separate normal install or downgrade operation. Perform this only when the path is supported with normal maintenance impact.
- The **clear xfsu state** command clears the recorded last-reload state; it does not restore the previous image. Do not use it while an operation is active or before collecting diagnostics data.

**Procedure**

1. Verify that the switch is eligible for xFSU.

   **Example**

```
switch# show xfsu eligibility
Eligibility Check          Status
=======================    =========
Autoboot Enabled           Yes         >>> Automatic boot behavior satisfies
the xFSU requirement

Install Mode               Yes         >>> The switch is running in INSTALL
mode. xFSU cannot run from BUNDLE mode.

Supported FPGA            Yes        >>> The installed FPGA version supports
 xFSU

MacSec                     Eligible    >>> The current MACsec configuration
does not block xFSU

Spanning Tree             Eligible   >>> The spanning-tree role and topology
 satisfy the device checks

XFSU Eligibility Summary
============================
XFSU Eligibility: ELIGIBLE           >>> The device-side checks passed.
```

   The output must show `XFSU Eligibility: ELIGIBLE`. This confirms that the device-side xFSU checks passed, but the supported source-to-target path must still be verified in the release-specific guide.

2. Start the xFSU add, activate, and commit workflow for the target image.

   **Example**

```
switch# install add file flash:<IMAGE.bin> activate xfsu commit
install_add_activate_commit: START

STACK_GR: Inside xFSU check if switch stack
Checking STP eligibility: Eligible

[1]: Performing xFSU-pre-check
```

```
SUCCESS: xFSU-pre-check finished
SUCCESS: xFSU requirement pre-check  >>> The runtime configuration and topology
 prechecks passed

--- Starting initial file syncing ---
Info: Finished copying flash:<IMAGE.bin> to the selected switch(es) >>> The
image is available on all required members

--- Starting Add ---
Add: Passed on [1]    >>> The target image was validated and expanded
successfully

--- Verifying platform-specific xFSU admission criteria ---
SUCCESS: xFSU image pre-check    >>> The platform and target image passed the
 final xFSU admission check

This operation requires a fast reload of the system.
Do you want to proceed? [y/n] y

--- Starting Activate ---
Activate: Passed on [1]          >>> The target packages were activated

--- Starting Commit ---
Commit: Passed on [1]            >>> The target image was made persistent

SUCCESS to verify packages
SUCCESS to verify before xFSU    >>> The final package and xFSU verification
 passed

[1]: Performing Upgrade_Service
SUCCESS: Upgrade_Service finished

Initializing Hardware......
Reload Fast Detected             >>> The switch entered the xFSU
reduced-downtime reload workflow

boot: attempting to boot from [flash:packages.conf]
>>> The switch is booting the target packages in INSTALL mode

>>> [Routine boot messages omitted]
```

The xFSU workflow performs runtime and topology prechecks, synchronizes the image to the required members, validates and expands the packages, activates the target packages, and commits the target image.

3. Verify that the intended target release is running and that reload-fast was used.

   **Example**

```
switch# show version | include IOS XE Software|Last reload reason
Cisco IOS XE Software, Version <TARGET_RELEASE>

Last reload reason: Image Install with Reloadfast
```

   The output must show Cisco IOS XE Software Version <TARGET_RELEASE>. The `Last reload reason: Image Install with Reloadfast` output confirms that the reload was performed through the xFSU reload-fast workflow.

4. Confirm that the forwarding-engine fast-reload operation completed.

**Example**

```
switch# show logging | include FAST_RELOAD_COMPLETE
%FED_IPC_MSG-5-FAST_RELOAD_COMPLETE:
 Fast reload operation complete
```

The `%FED_IPC_MSG-5-FAST_RELOAD_COMPLETE` log message confirms that the forwarding-engine fast-reload operation completed.

5. Verify the xFSU platform status after the upgrade.

**Example**

```
switch# show xfsu status
XFSU PLATFORM Status:
 Data plane update done, Upgrade done in 1590 ms.
```

The `Data plane update done` and `Upgrade done` output confirms that the data-plane update completed. The reported upgrade time varies by device and deployment.

6. Confirm that the target image is activated and committed after xFSU.

**Example**

```
switch# show install summary
Type  St   Filename/Version
-----------------------------------------
IMG   C    <TARGET_RELEASE>

Auto abort timer: inactive
```

The `IMG` entry must show state `C` for <TARGET_RELEASE>. The `Auto abort timer: inactive` output confirms that no incomplete installation or rollback timer remains.

---

The supported Catalyst 9300 switch runs the target Cisco IOS XE release after a successful xFSU operation. Reload-fast completed, the data-plane update is done, the target image is activated and committed, and no rollback timer remains active.

# 4 Post-Upgrade Validation

**Topics:**

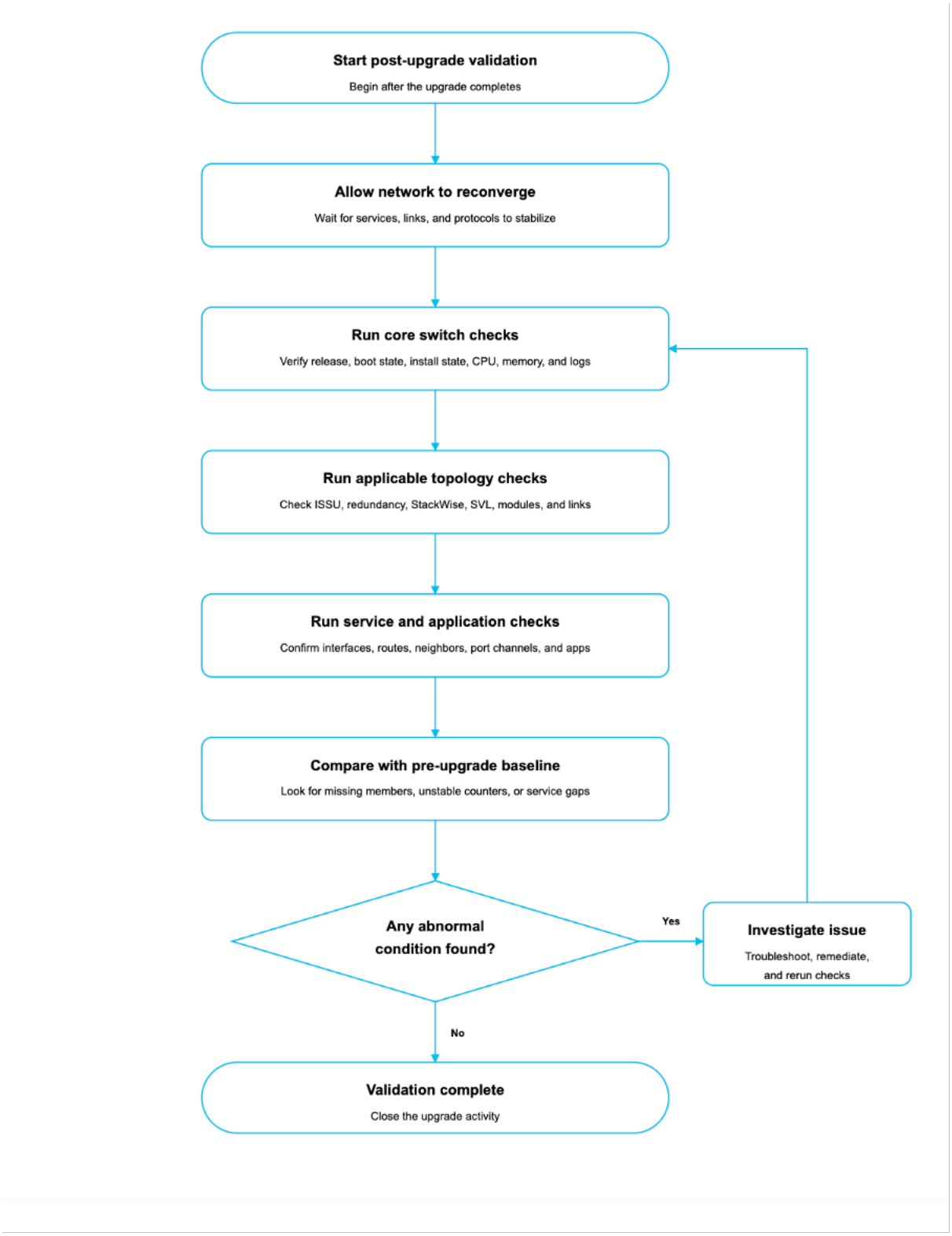

- Post-upgrade workflow
- Recommendations for device health check
- Command set for system and application health checks
- Verify install state, ISSU state, and redundancy
- Verify CPU, memory, and control-processor health
- Verify hardware and traffic stability
- Verify physical StackWise health
- Verify StackWise Virtual system health
- Verify network services and application recovery
- Verify application-hosting health
- Additional references

## Post-upgrade workflow

Outlines the post-upgrade workflow, including required steps and considerations to ensure successful completion after a system upgrade

This topic enables you to effectively follow the post-upgrade workflow, ensuring all necessary steps and checks are completed for a stable and fully operational system after an upgrade.

**Figure 3: Post-upgrade workflow**




## Recommendations for device health check

Outlines recommendations for determining device health, including pass criteria and indicators for further analysis based on install state, redundancy, stack and module status, resource utilization, counters, network recovery, logs, applications, and service confirmation.

### Pass criteria

A device passes the health check when the intended release is running and properly activated, all redundancy and stack members are healthy, resource use remains within normal limits, counters are stable, expected interfaces and applications have recovered, logs contain no critical errors, and normal service is confirmed.

The pass criteria includes these:

- Pass criteria - install and redundancy state
  - The intended release is running and `packages.conf` file is used in install mode.
  - The image is activated and committed, with no active rollback timer.
  - No ISSU transaction remains in progress.
  - SSO is operating, communications are up, and the peer is STANDBY HOT where applicable.
  - Configuration-synchronization failure lists are empty.
- Pass criteria - platform and stack health
  - Every expected stack member, supervisor, line card, SVL, and DAD link is healthy.
  - CPU, memory, load, fans, power, and temperature remain healthy.
  - Error, discard, stack-link, and redundancy counters are stable between samples.
- Pass criteria - network, logs, and application recovery
  - Expected interfaces, port channels, routes, neighbors, FHRP groups, and hosted applications have recovered.
  - Logs contain no unexplained crash, traceback, critical platform, install, or ISSU messages.
  - Application owners and monitoring systems confirm normal service.

### When to investigate further

Initiate further investigation if the device reports warning or critical statistics, experiences persistent high CPU or declining memory, shows discrepancies in image or peer status, has incomplete install or ISSU transactions, reveals missing or unhealthy stack and links, registers increasing errors, fails to restore network baselines, or presents unexpected log incidents.

- Install, redundancy, and stack indicators
  - The active and standby processors run different images or the peer is not STANDBY HOT.
  - An ISSU or install transaction remains pending, incomplete, failed, or rolled back.
  - A stack member is missing, not Ready, or the physical ring reports `Half`.
  - SVL or DAD links are down, suspended, pending, or not ready.
- Interface, network, and application indicators
  - Interface errors, discards, stack CRC errors, or redundancy errors continue increasing.
  - Routing neighbors, route counts, FHRP groups, or hosted applications do not return to their baseline state.
  - An unexpected reload, crash, core file, traceback, or critical log message is present.

## Command set for system and application health checks

Lists the command set for performing system and application health checks on switches, covering core, redundancy, StackWise, StackWise Virtual, network services, and application hosting checks, as well as additional commands for troubleshooting abnormal conditions across various hardware and feature configurations.

This topic describes how to select and use appropriate system and application health check commands for different switch platforms and configurations, enabling consistent pre- and post-upgrade assessments and effective troubleshooting of core, redundancy, stacking, network service, and application hosting operations.

Use this command set as a reference for selecting the checks that apply to a specific switch, topology, and configured feature set.

Run the same checks before and after the upgrade. Allow the network to reconverge, then compare the results with the pre-upgrade baseline.

The commands are read-only. Some commands apply only to a stack, modular chassis, StackWise Virtual system, ISSU-capable system, or a switch using application hosting.

The reference sessions are illustrative. Exact fields and values vary by model, topology, configuration, and IOS XE release. Comment lines in the examples explain the command output and are not generated by the device.

**Core checks for every switch**

```
terminal length 0
show clock
show version
show boot
show install summary

show platform resources
show platform software status control-processor brief
show processes cpu sorted
show processes cpu history
show processes memory sorted

show environment all
show interfaces status
show interfaces status err-disabled
show interfaces counters errors
show etherchannel summary
show spanning-tree summary totals
show spanning-tree inconsistentports
show ip interface brief
show ip route summary

show logging | last 100
show license summary
```

**ISSU and redundancy checks**

Run these only on an ISSU-capable dual-supervisor, StackWise, or StackWise Virtual system.

The **show install issu history** is available in Cisco IOS XE 16.10.1, and later. An ISSU path must still be approved by the current Cisco compatibility matrix.

```

show issu state detail
show install issu history
show redundancy
```

```
show redundancy states

show redundancy config-sync failures mcl
show redundancy config-sync failures prc
show redundancy config-sync failures bem
show redundancy counters
show redundancy switchover history
```

**Physical StackWise checks**

```
show switch detail
show switch stack-ring speed
show switch stack-ports summary
show switch stack-ports detail
```

**StackWise Virtual checks**

```
show stackwise-virtual link
show stackwise-virtual bandwidth
show stackwise-virtual neighbors
show stackwise-virtual dual-active-detection
```

**Configured network-service checks**

Run only the protocols used by the switch.

For a non-default VRF, use the corresponding VRF-aware command.

```
show ip ospf neighbor summary
show ip eigrp neighbors
show ip bgp summary
show isis neighbors
show bfd neighbors details
show standby brief
show vrrp brief
```

**Optional application-hosting checks**

```
show iox-service
show app-hosting list
show app-hosting detail appid <APP_ID>
show app-hosting utilization appid <APP_ID>
```

**Additional checks for further analysis**

Use these additional set of commands to troubleshoot if any of the checks done earlier fails or gives an error.

The `switch active`, `R0`, `F0`, `module`, and `location` syntax vary by platform. Use command help (`?`) to select the one supported by the switch.

```
show install log
show processes cpu platform sorted 5min location switch active R0
```

```
show processes memory platform sorted location switch active R0
show platform software process list switch active R0 summary
show platform software object-manager switch active F0 statistics
show diagnostic result module all detail
show logging onboard switch active uptime detail
dir crashinfo:
```

## Verify install state, ISSU state, and redundancy

Guides you through verifying install and ISSU readiness, checking ISSU operation history, assessing redundancy health and synchronization, and confirming configuration synchronization to ensure system stability before and after an upgrade.

Use these checks before an upgrade on systems where install state, ISSU state, or redundancy must be validated.

**Procedure**

1. Verify install and ISSU readiness
   a) Verify install-mode boot variables.

      **Example**

      ```
      switch# show boot
      Current Boot Variables:
      BOOT variable = flash:packages.conf;

      Boot Variables on next reload:
      BOOT variable = flash:packages.conf;
      Manual Boot = no
      ```

      The `packages.conf` confirms install-mode boot.

      Manual Boot = no confirms that automatic boot is enabled.

   b) Verify committed install package state.

      **Example**

      ```
      switch# show install summary
      [ Switch 1 2 ] Installed Package(s) Information:
      State (St): I - Inactive, U - Activated & Uncommitted,
                  C - Activated & Committed,
                  D - Deactivated & Uncommitted
      --------------------------------------------------------------
      Type  St   Filename/Version
      --------------------------------------------------------------
      IMG   C    <CURRENT_RELEASE>
      --------------------------------------------------------------
      Auto abort timer: inactive
      --------------------------------------------------------------
      ```

      `C` means that the running image is activated and committed.

      An inactive auto-abort timer means that no uncommitted installation is waiting to roll back.

      Investigate an unexpected U, D, active timer, or different image state across members.

   c) Confirm that no ISSU operation is in progress.

**Example**

```
switch# show issu state detail
--- Starting local lock acquisition on switch 1 ---
Finished local lock acquisition on switch 1

No ISSU operation is in progress
```

This is the expected state before starting ISSU and after ISSU has completed.

Do not start another upgrade if the output reports an active, pending, added, activated, failed, or stalled ISSU operation.

d) Review ISSU operation history.

**Example**

```
switch# show install issu history
>>> Most recent ISSU operation is displayed
```

Verify that the most recent entry belongs to the intended change window and completed successfully.

Failed, aborted, rolled-back, or incomplete entries require review. Output fields vary by IOS XE release.

**2.** Verify redundancy health.

a) Verify SSO redundancy health

**Example**

```
switch# show redundancy
Redundant System Information:
------------------------------
Switchovers system experienced = 1
Standby failures = 0
Last switchover reason = ISSU

Hardware Mode = Duplex
Configured Redundancy Mode = sso
Operating Redundancy Mode = sso
Communications = Up

Current Processor Information:
------------------------------
Current Software state = ACTIVE
BOOT = flash:packages.conf;

Peer Processor Information:
---------------------------
Current Software state = STANDBY HOT
BOOT = flash:packages.conf;
```

A healthy redundant system has SSO configured and operating, `Communications` as `Up`, one ACTIVE processor, and one STANDBY HOT processor.

Both processors must run the intended release and use `packages.conf`.

A standalone switch has no standby peer; STANDBY HOT is therefore not expected.

b) Verify redundancy state synchronization.

**Example**

```
switch# show redundancy states
       my state = 13 -ACTIVE
     peer state = 8  -STANDBY HOT
           Mode = Duplex

Redundancy Mode (Operational) = sso
Redundancy Mode (Configured)  = sso
Redundancy State              = sso
Communications                = Up
```

The peer has completed synchronization and reached STANDBY HOT.

Do not begin ISSU while the peer is `COLD`, `INIT`, `RECOVERY`, `DISABLED`, absent, or changing state.

**3.** Verify configuration synchronization

a) Confirm that there are no MCL configuration-synchronization failures.

**Example**

```
switch# show redundancy config-sync failures mcl
Mismatched Command List
-----------------------
The list is Empty
```

No configured command is unsupported or mismatched on the standby.

b) Confirm that there are no PRC configuration-synchronization failures.

**Example**

```
switch# show redundancy config-sync failures prc
PRC Failed Command List
-----------------------
The list is Empty
```

Active and standby parser return codes match.

c) Confirm that there are no BEM configuration-synchronization failures.

**Example**

```
switch# show redundancy config-sync failures bem
BEM Failed Command List
-----------------------
The list is Empty
```

No best-effort configuration-synchronization failure is recorded.

All three lists should be empty. Troubleshoot a mismatched command before ISSU or use a normal install-mode upgrade.

## Verify CPU, memory, and control-processor health

Guides you through verifying switch CPU, memory, and control-processor health, including checking platform resources, IOSd CPU and memory usage, and performing additional platform process diagnostics as needed.

Use these checks to determine whether the switch CPU, memory, and control processor are healthy.

**Procedure**

1. Verify platform and control-processor health.
   a) Verify platform resource health.

      **Example**

```
switch# show platform resources
State Acronym: H - Healthy, W - Warning, C - Critical

Resource                 Usage          Max       Warning  Critical  State
-------------------------------------------------------------------------------
Control Processor        7.20%          100%      90%      95%       H
 DRAM                    2701MB (69%)   3883MB    90%      95%       H
```

      - `H` means that the resource is within the platform-defined healthy range.
      - `W` means that it requires investigation.
      - `C` means that the platform-defined critical level has been reached.

      Use the thresholds displayed by the switch; do not apply one fixed CPU or memory percentage to every platform.

   b) Verify control-processor health.

      **Example**

```
switch# show platform software status control-processor brief
Load Average
 Slot    Status    1-Min  5-Min  15-Min
1-RP0    Healthy    0.21   0.18    0.16
2-RP0    Healthy    0.18   0.15    0.14

Memory (kB)
 Slot    Status     Total       Used (Pct)       Free (Pct)
1-RP0    Healthy    3976852     2766284 (70%)    1210568 (30%)
2-RP0    Healthy    3976852     2680416 (67%)    1296436 (33%)

CPU Utilization
 Slot  CPU  User  System  Nice  Idle   IRQ  SIRQ  IOwait
1-RP0    0  4.40    1.70  0.00  93.80  0.00  0.10    0.00
2-RP0    0  3.80    1.20  0.00  94.90  0.00  0.10    0.00
```

      `Healthy` indicates that load and memory are within the switch-calculated limits.

      In an HA system, both active and standby processors should report healthy.

      `Warning`, `Critical`, offline, unknown, or repeatedly stale statistics requires further analysis.

2. Review IOSd CPU and memory health.

a) Review IOSd CPU utilization.

**Example**

```
switch# show processes cpu sorted
CPU utilization for five seconds: 8%/1%; one minute: 7%; five minutes: 6%
 PID  Runtime(ms)  Invoked   uSecs   5Sec  1Min  5Min  TTY  Process
-----------------------------------------------------------------------
 123     35303496  679878082    51   1.10% 1.02% 0.98%   0  IOSXE-RP Punt
 Se
 247     47096761   92250875   510   0.70% 0.62% 0.60%   0  Spanning Tree
 229     27381066  963726156    28   0.40% 0.35% 0.33%   0  IP Input
```

The first line of the sample output displays the total IOSd CPU utilization for five seconds. The value after the `/` is interrupt-driven CPU utilization.

The `one-minute` and `five-minute` values help distinguish a short spike from sustained load.

The sorted output places the highest-consuming IOSd process first.

b) Review CPU utilization history.

**Example**

```
switch# show processes cpu history
```

The output displays 60-second, 60-minute, and 72-hour CPU graphs.

Isolated peaks can occur during an upgrade. Repeated high averages after the switch has settled are more important.

c) Review IOSd processor-pool memory.

**Example**

```
switch# show processes memory sorted
Processor Pool Total: 25954228  Used: 8368640  Free: 17585588

 PID  TTY   Allocated     Freed    Holding  Getbufs  Retbufs  Process
-----------------------------------------------------------------------
   0    0     8629528    689900    6751716        0        0  *Init*
 123    0      420000    180000     252000        0        0  IOSXE-RP Punt
 Se
 247    0      190000     72000     130000        0        0  Spanning Tree
```

This output reports IOSd processor-pool memory, not all Linux system memory.

Compare multiple samples. Continuously falling `freed` memory combined with continuously growing `Holding` memory require review.

**3.** Run additional platform CPU and memory checks when needed.

a) **`Review platform process CPU utilization.`**

**Example**

```
switch# show processes cpu platform sorted 5min location switch active R0
CPU utilization for five seconds: 3%; one minute: 4%; five minutes: 4%

   Pid    PPid   5Sec  1Min  5Min  Status    Size  Name
-----------------------------------------------------------------
 18662       1     2%    2%    2%  S       694272 linux_iosd-imag
 17546       1     1%    1%    1%  S       249856 fed main event
 17835       1     0%    0%    0%  S        75776 sif_mgr
```

This includes IOS XE Linux processes that are not individually shown by the **show processes cpu sorted** command.

Identify a process that remains unexpectedly high in the one-minute and five-minute columns.

b) Review platform process memory utilization.

**Example**

```
switch# show processes memory platform sorted location switch active R0
System memory: 3976852K total, 2762884K used, 1213968K free
Lowest: 1189400K

   Pid      Text      Data   Stack  Dynamic      RSS  Name
-----------------------------------------------------------------
 18662    149848    684864     136       80   684864  linux_iosd-imag
  9655      3787    264964     136    18004   264964  wcm
 17546       324    248588     132   103908   248588  fed main event
```

`RSS` is the physical RAM currently used by each process.

A steadily increasing RSS for one process, combined with falling free memory, requires further analysis.

Use these checks only if the quick checks show an abnormal condition. The syntax varies by platform.

## Verify hardware and traffic stability

Guides you through verifying hardware status, interface and traffic stability, port-channel and spanning-tree health, and reviewing reload reasons, logs, and crash files after an upgrade.

Use these checks to validate hardware status, interface stability, reload reason, logs, and crash artifacts.

**Procedure**

1. Verify hardware health
   a) Verify fan, power, and temperature health.

**Example**

```
switch# show environment all
Switch 1 FAN 1 is OK
Switch 1 FAN 2 is OK
```

```
Switch 1 FAN 3 is OK
FAN PS-1 is OK
FAN PS-2 is OK

Switch 1: SYSTEM TEMPERATURE is OK
Inlet Temperature Value: 25 Degree Celsius
Temperature State: GREEN
Hotspot Temperature Value: 35 Degree Celsius
Temperature State: GREEN
```

`OK` and `GREEN` indicate normal fan and temperature operation.

`NOT PRESENT` can be normal for an intentionally empty power-supply bay; compare it with the intended redundancy design.

`FAULTY`, failed, yellow, red, airflow mismatch, or an unexpectedly absent power supply requires attention.

**2.** Verify interface and traffic stability.

a) Check for error-disabled interfaces.

**Example**

```
switch# show interfaces status err-disabled
Port      Name      Status       Reason
---------------------------------------
```

The sample output here did not list any interface. This means that no interface is currently error-disabled.

b) Review interface error counters

**Example**

```
switch# show interfaces counters errors
Port        Align-Err  FCS-Err  Xmit-Err  Rcv-Err  UnderSize  OutDiscards
-------------------------------------------------------------------------
Gi1/0/1             0        0         0        0          0            0
Gi1/0/47            0        0         0        0          0            0
Gi1/0/48            0        0         0        0          0            0
```

Focus on counters that continue to increase, not only on historical non-zero totals.

Increasing FCS-Err or receive errors (`Rcv-Err`) commonly point to a cable, optic, NIC, interface, or duplex problem.

Increasing output discards can indicate congestion or traffic bursts.

c) Verify port-channel bundle health.

**Example**

```
switch# show etherchannel summary
Flags: D - down   P - bundled   R - Layer3   S - Layer2   U - in use

Group  Port-channel  Protocol  Ports
------+-------------+---------+-------------------------------
1      Po1(SU)       LACP      Gi1/0/47(P) Gi1/0/48(P)
```

The flags `S` and `U` mean that the Layer 2 port channel is in use.

`P` means that each physical interface is successfully bundled.

`D`, `I`, suspended, or stand-alone member states require further analysis.

d) Check for spanning-tree inconsistent ports.

**Example**

```
switch# show spanning-tree inconsistentports
Name                 Interface              Inconsistency
-------------------- ---------------------- ------------------

Number of inconsistent ports (segments) in the system : 0
```

Zero inconsistent ports is the expected result.

Normal STP blocking ports are not failures; inconsistent ports require further analysis.

**3.** Review reload, logging, and crash indicators.

a) Verify reload reason and uptime

**Example**

```
switch# show version | include uptime|returned|Last reload reason
switch uptime is 1 hour, 18 minutes
System returned to ROM by Image Install
Last reload reason: Image Install
```

`Image Install` value for the `Last reload reason` field is expected after a planned software upgrade.

An unexpected crash, watchdog, power loss, software failure, or unknown reason requires further analysis.

b) Check logs for critical post-upgrade events.

**Example**

```
switch# show logging | include
CRASH|TRACEBACK|OOM|PLATFORM-[0-3]|INSTALL-[0-3]|ISSU-[0-3]
switch#
```

No output means that the retained local buffer has no matching messages.

The buffer can wrap or be cleared during a reload. Also check the external syslog server.

c) Check for new crash files.

**Example**

```
switch# dir crashinfo:
Directory of crashinfo:/

No files in directory
```

`No files in directory` is the expected result which indicates that there is no new crash file.

Preserve any crash or system-report file whose timestamp matches the upgrade or reported problem.

## Verify physical StackWise health

Guides you through verifying the physical health of StackWise systems, including checks for stack membership, ring speed and redundancy, and stack-port synchronization and stability.

Use these checks on a physical StackWise system to confirm stack membership, ring health, and stack-port stability.

**Procedure**

Verify stack membership and ring health.

a) Verify stack member and stack-port health.

**Example**

```
switch# show switch detail
Switch/Stack Mac Address : <STACK_MAC>

Switch#  Role      Priority  Version  State
-------------------------------------------
*1       Active    15        V03      Ready
 2       Standby   14        V03      Ready
 3       Member     1        V03      Ready

Stack Port Status             Neighbors
Switch#  Port 1  Port 2       Port 1  Port 2
-------------------------------------------
1        OK      OK           2       3
2        OK      OK           3       1
3        OK      OK           1       2
```

Every expected member is present and Ready.

Both stack ports are OK, and the neighbor relationships form a complete ring.

b) Verify StackWise ring speed and redundancy.

**Example**

```
switch# show switch stack-ring speed
Stack Ring Speed         : 480G
Stack Ring Configuration : Full
Stack Ring Protocol      : StackWise
```

`Full` means that the stack has a complete, redundant ring.

`Half` indicates a broken ring, reduced bandwidth, and loss of ring redundancy.

The expected ring speed depends on the switch models and stack configuration.

c) Verify stack-port synchronization and stability.

**Example**

```
switch# show switch stack-ports summary
Sw#/Port#  Status  Neighbor  Link OK  Link Active  Sync OK  #Changes
-------------------------------------------------------------------
1/1        OK      2         Yes      Yes          Yes      1
1/2        OK      3         Yes      Yes          Yes      1
2/1        OK      3         Yes      Yes          Yes      1
2/2        OK      1         Yes      Yes          Yes      1
3/1        OK      1         Yes      Yes          Yes      1
3/2        OK      2         Yes      Yes          Yes      1
```

The output shows that all stack ports are synchronized and active.

Capture this output twice. An increasing `#Changes` value indicates stack-link instability.

## Verify StackWise Virtual system health

Guides you through verifying StackWise Virtual system health, including StackWise Virtual Link status, dual-active detection, and module and supervisor redundancy on StackWise Virtual systems and modular chassis.

### Before you begin

Use these checks on a StackWise Virtual system or modular chassis to confirm StackWise Virtual Link (SVL), dual-active detection (DA, and module health.

**Procedure**

Verify StackWise Virtual system health.

a) Verify StackWise Virtual Link status.

**Example**

```
switch# show stackwise-virtual link
Stackwise Virtual Link (SVL) Information:
-----------------------------------------
Switch  SVL  Ports                          Link-Status  Protocol-Status
------  ---  -----------------------------  -----------  ---------------
1       1    FortyGigabitEthernet1/1/1      U            R
2       1    FortyGigabitEthernet2/1/1      U            R
```

The `Link-Status U` means that the physical SVL interface is Up.

The `Protocol-Status R` means that the SVL protocol is Ready.

Some platforms display `P` for a correctly bundled protocol state; use the legend printed by the switch.

b) Verify dual-active-detection status.

**Example**

```
switch# show stackwise-virtual dual-active-detection
Recovery Reload for switch 1: Enabled
```

```
Recovery Reload for switch 2: Enabled

Dual-Active-Detection Configuration:
------------------------------------
Switch  Dad port                        Status
------  ------------------------------  ------
1       FortyGigabitEthernet1/0/3       up
2       FortyGigabitEthernet2/0/3       up
```

The output shows that dual-active-detection links are configured and up on both switches.

A down or missing DAD path reduces protection against a dual-active condition.

c) Verify module and supervisor redundancy status.

**Example**

```
switch# show module
Mod  Card Type            Model              Status
---  -------------------  -----------------  --------
1    Line Card            <MODEL>            ok
5    Supervisor Module    <MODEL>            ok
6    Supervisor Module    <MODEL>            ok

Mod  Redundancy Role  Operating Mode  Redundancy Status
---  ---------------  --------------  -----------------
5    Active           sso             Active
6    Standby          sso             Standby Hot
```

Every installed module is recognized and reports ok in this output.

A redundant modular chassis has one Active and one Standby Hot supervisor.

## Verify network services and application recovery

Guides you through verifying recovery of routed interfaces, route summaries, OSPF neighbors, BGP sessions, HSRP group states, and application or monitoring reachability after an upgrade, comparing results with the pre-upgrade baseline.

Run only the protocol and application checks that apply to the switch configuration, then compare the results with the pre-upgrade baseline.

**Procedure**

1. Verify interface and route recovery.
   a) Verify routed interface and switched virtual interface (SVI) status.

   **Example**

```
switch# show ip interface brief
Interface              IP-Address      OK? Method Status
Protocol
Vlan10                  192.0.2.2       YES NVRAM  up
up
```

```
Vlan20                  198.51.100.2    YES NVRAM  up
up
Port-channel10          203.0.113.1     YES NVRAM  up
up
Loopback0               192.0.2.254     YES NVRAM  up
up
```

Expected routed interfaces and SVIs must return to `up/up`.

An administratively down interface can be normal if it was intentionally shut.

b) Compare route summary with the baseline.

**Example**

```
switch# show ip route summary
Route Source    Networks  Subnets
connected       0         8
static          1         2
ospf 10         0         126
Total           7         136
```

Compare route totals by source with the pre-upgrade baseline.

Small temporary differences can occur during reconvergence; sustained unexplained losses require further analysis.

**2.** Verify routing protocol recovery.

a) Verify OSPF neighbor recovery.

**Example**

```
switch# show ip ospf neighbor summary
DOWN         0
INIT         0
EXSTART      0
EXCHANGE     0
LOADING      0
FULL         2
Total count  2
```

Compare the FULL and total neighbor counts with the pre-upgrade baseline.

Persistent `INIT`, `EXSTART`, `EXCHANGE`, or `LOADING` states require further analysis.

b) Verify BGP session and prefix recovery.

**Example**

```
switch# show ip bgp summary
Neighbor        V    AS  InQ OutQ Up/Down  State/PfxRcd
203.0.113.2     4 65002    0    0 1d02h              84
203.0.113.6     4 65003    0    0 1d02h              91
```

A number in `State/PfxRcd` means that the BGP session is established.

`Idle`, `Active`, or `Connect` means that the session is not established.

Compare received-prefix counts with the pre-upgrade baseline.

3. Verify gateway and application reachability.
   a) Verify HSRP group state.

**Example**

```
switch# show standby brief
Interface  Grp  Pri  State    Active          Standby         Virtual IP
Vl10        10  110  Active   local           192.0.2.3       192.0.2.1
Vl20        20   90  Standby  198.51.100.3    local           198.51.100.1
```

Expected HSRP groups should settle in Active or Standby.

Verify the active peer, standby peer, priority, and virtual IP against the baseline.

   b) Test agreed application or monitoring reachability.

**Example**

```
switch# ping 192.0.2.10 repeat 20
Type escape sequence to abort.
Sending 20, 100-byte ICMP Echos to 192.0.2.10:
!!!!!!!!!!!!!!!!!!!!
Success rate is 100 percent (20/20), round-trip min/avg/max = 1/2/4 ms
```

Test agreed application destinations, management systems, gateways, and monitoring collectors.

ICMP success alone does not validate the complete application. Obtain confirmation from the application owner.

## Verify application-hosting health

Outlines procedures for verifying IOx service health, checking hosted application state, reviewing application details, and assessing resource utilization against pre-upgrade baselines to ensure application-hosting integrity after an upgrade.

Run these checks only when IOx or application hosting was in use before the upgrade.

**Procedure**

Verify IOx and hosted application health.

a) Verify IOx service health.

**Example**

```
switch# show iox-service
IOx Infrastructure Summary:
---------------------------
IOx service (CAF)          : Running
IOx service (HA)           : Running
IOx service (IOxman)       : Running
Libvirtd                   : Running
Dockerd                    : Running
Application DB Sync Info   : Available
```

The output shows that required IOx services are running. Exact services vary by platform and application configuration.

b) Verify hosted application state

**Example**

```
switch# show app-hosting list
App id                                   State
---------------------------------------------------
MONITORING_APP                           RUNNING
```

Every application that was running before the upgrade should return to `RUNNING` state.

c) Review hosted application details

**Example**

```
switch# show app-hosting detail appid MONITORING_APP

>>>Application details are displayed
```

Verify the application state, virtual interfaces, storage, and configuration against the pre-upgrade output.

d) Review hosted application resource utilization.

**Example**

```
switch# show app-hosting utilization appid MONITORING_APP

>>> Application CPU, memory, and storage utilization are displayed
```

Compare application utilization with the pre-upgrade baseline.

Do further analysis on sustained resource exhaustion or a large unexplained change.

## Additional references

Provides links to Cisco references for Catalyst 9000 series switches, including health checks, ISSU, control-plane troubleshooting, latency and packet drops, StackWise, StackWise Virtual, and application hosting to support further analysis and device health verification.

These links support follow-up analysis and related device health check work.

- Perform Catalyst Health and Configuration Check
- Perform ISSU on Catalyst 3850 and Catalyst 9000 Series Switches
- Troubleshoot Control Plane Operations on Catalyst 9000 Switches
- Troubleshooting Network Latency and Packet Drops on Catalyst 9000 Switches
- Verify and Troubleshoot Stackwise on Catalyst 9200 and 9300
- Cisco Catalyst 9000 Platform StackWise Virtual White Paper

- Understand%20App%20Hosting%20on%20Catalyst%209000%20Switches

# 5 Troubleshooting Device Upgrades

**Topics:**


- Operational boundaries and diagnostic reference
- Guidelines for troubleshooting
- Common upgrade failure symptoms and resolution reference
- First-response commands for troubleshooting upgrade issues
- Determine the actual boot and install state
- Troubleshoot image-copy and file-validation issues
- Troubleshoot normal install-mode upgrade issues
- Troubleshoot legacy bundle-mode issues
- Troubleshoot ISSU issues
- Troubleshoot xFSU issues
- Troubleshoot SMU issues
- Troubleshoot stack, supervisor, and synchronization issues
- Troubleshoot a device that boots the previous release
- Recover from ROMMON or bootloader failures
- Troubleshoot post-upgrade system or service issues
- Troubleshoot Catalyst Center SWIM upgrades
- Technical support criteria
- Collect diagnostics data
- Additional references

## Operational boundaries and diagnostic reference

Outlines the operating boundaries, key terminology and provides diagnostic references for troubleshooting upgrade issues on Cisco Catalyst 9000 series switch without changing the switch state.

Use this reference to identify operating boundaries, understand image and upgrade mode restrictions, familiarize with key terminology, and utilize diagnostic references for troubleshooting Cisco Catalyst 9000 series switch upgrades without disrupting switch operations.

#### Scope and important boundaries

- Always use the image, release notes, compatibility matrix, and procedures for the exact product ID and release.
- Keep continuous console access and an out-of-band recovery path throughout the change.
- Common Catalyst 9000 image names include `cat9k_iosxe` and `cat9k_lite_iosxe`. You must not rename an image or use one platform family's image on another family.
- Install mode boots `packages.conf` file. Legacy BUNDLE mode boots the monolithic `.bin` file.
- ISSU and xFSU are eligibility-based methods, not generic alternatives to a normal upgrade. Use them only when the current exact Cisco matrix or platform guide approves the full source-to-target path and the device passes its readiness checks.
- xFSU examples in this guide only apply to supported Catalyst 9300 series switches.

#### Key terminology

These are some of the common terms used in this guide:

- Upgrade modes:
  - `Install mode`: where the switch boots expanded IOS XE packages through the `packages.conf` file.
  - `Bundle mode`: where the switch boots the monolithic IOS XE `.bin` image directly.
  - `ISSU`: In-Service Software Upgrade on an eligible redundant platform and exact approved release path.
  - `xFSU`: Extended Fast Software Upgrade, a reduced-downtime method on eligible Catalyst 9300 series switches.
- `SMU`: Software Maintenance Upgrade, a release- and platform-specific patch package.
- `SSO`: Stateful Switchover between an ACTIVE and a synchronized STANDBY HOT processor.
- `SVL`: StackWise Virtual.
- `SWIM`: Software Image Management in Cisco Catalyst Center.
- `ROMMON`: the low-level bootloader or recovery environment whose prompt is `switch:`.
- `R0 and F0`: platform locations for the route or control processor, and forwarding processor. Exact location and member syntax is platform-dependent.

## Guidelines for troubleshooting

Outlines essential troubleshooting guidelines to prevent service interruptions, data loss, or upgrade complications during software updates, install procedures, and system recoveries before identifying the switch state and failure cause.

Follow these troubleshooting guidelines to prevent service interruptions, data loss, or upgrade complications during software updates, install procedures, and system recoveries before identifying the switch state and failure cause.

- Do not power-cycle a device while it is updating ROMMON, bootloader, CPLD, or FPGA components.
- Do not interrupt an **install** command with Ctrl-C.
- Do not manually edit `packages.conf` or manually delete active `.pkg` files.
- Do not run a second upgrade while another INSTALL, ISSU, xFSU, SMU, or Catalyst Center SWIM task is active.
- Do not run **install remove inactive** command until the deletion list has been reviewed and the rollback decision is complete.
- Do not force ISSU or xFSU after an eligibility failure.
- Do not use **clear install state**, **format flash:**, or **emergency-install** as customer self-recovery. These actions can remove rollback and SMU state, reload the system, or erase flash. Use them only when directed by Cisco Support for the exact platform.

## Common upgrade failure symptoms and resolution reference

Outlines common upgrade failure symptoms—including copy, hash, install state, ISSU, xFSU, SMU, boot, ROMMON, and post-upgrade service issues—and guides diagnostic checks and recommended recovery

This topic outlines common upgrade failure symptoms—including copy, hash, install state, ISSU, xFSU, SMU, boot, ROMMON, and post-upgrade service issues—and guides diagnostic checks and recommended recovery

**Table 3: Common upgrade failure symptoms and resolutions**

| Failure symptom | Diagnostic checks and resolution steps |
|---|---|
| Copy fails or stops | Verify reachability, VRF, source interface, credentials, server path, ACL or firewall, DNS, filesystem name, and free space. |
| Hash does not match Cisco Software Download | Do not install the file; remove only that bad staged image, copy it again, and verify it again. |
| Not enough space | Identify active files and rollback requirements, then use the supported inactive-package cleanup workflow. |
| Log displays, `Super package already added` | Verify the **show install summary** command output; the target might be already in inactive state. |
| Image is active but uncommitted | Validate the new release, then either commit it or abort before the rollback timer expires. |
| Stack member did not upgrade | Inspect the member state, filesystem, free space, and synchronized image. |
| ISSU is rejected | Check the exact release path, INSTALL mode, STANDBY HOT/SSO state, configuration synchronization, ROMMON, and storage. |
| xFSU is not eligible | Use the reason shown by **show xfsu eligibility**; correct it only if xFSU is supported, otherwise use a normal INSTALL-mode upgrade. |
| SMU is rejected | Verify exact platform, base release, package state, and the SMU readme. |
| Device boots the previous release | Inspect the boot variable, `packages.conf` references, install state, and install log. |

| Failure symptom | Diagnostic checks and resolution steps |
|---|---|
| Device reaches the ROMMON or bootloader prompt, `(switch:)` | Preserve console output and inspect valid local boot files before attempting recovery. |
| Services are unhealthy after the upgrade | Compare the pre-upgrade baseline and decide whether to commit, abort, or execute the approved rollback. |

## First-response commands for troubleshooting upgrade issues

Outlines the essential commands for collecting first-response evidence during upgrade troubleshooting, including checks for version, boot, install, redundancy, ISSU, xFSU, filesystem, platform resources, processes, environment, and platform-supported logging.

Use the commands listed in this topic to efficiently gather critical diagnostic information using platform-supported commands for troubleshooting upgrade issues, ensuring comprehensive assessment of version, installation, redundancy, process, and environment status.

Run the commands supported by the respective platform. The switch active, R0, F0, module, member, and filesystem syntax might vary for each product.

```
switch# show clock
switch# show version
switch# show boot
switch# show install summary
switch# show install active
switch# show install committed
switch# show install inactive
switch# show install uncommitted
switch# show install log
switch# show switch
switch# show redundancy
switch# show redundancy states
switch# show issu state detail
switch# show xfsu eligibility
switch# show xfsu status
switch# show graceful-reload
switch# show file systems
switch# dir flash:
switch# dir crashinfo:
switch# show platform resources
switch# show platform software status control-processor brief
switch# show processes cpu sorted
switch# show processes memory sorted
switch# show environment all
switch# show logging | last 200
```

## Determine the actual boot and install state

Guides you through verifying the switch software mode, boot variables, install package state, and rollback timer to assess the current boot and install conditions before retrying an upgrade or confirming the device is in the expected install mode.

Use this procedure to verify the switch software mode, boot variables, install package state, and rollback timer to assess the current boot and install conditions before retrying an upgrade or confirming the device is in the expected install mode.

Stop before retrying if different stack members or supervisors report different unexpected states. Preserve the show install log and the complete original session.

**Procedure**

1. Identify the current switch software mode.

   **Example**

   ```
   switch# show version | begin Switch Ports
   Switch Ports Model        SW Version  SW Image       Mode
   ------ ----- ------------ ----------- -------------- -------
   *    1 48    C9300-48P    17.xx.x     CAT9K_IOSXE   INSTALL
   ```

   `INSTALL` value for the `Mode` means that the device boots expanded packages through `packages.conf`. `BUNDLE` means it boots a `.bin` file.

2. Verify boot variables and automatic boot state.

   **Example**

   ```
   switch# show boot
   Current Boot Variables:
   BOOT variable = flash:packages.conf;

   Boot Variables on next reload:
   BOOT variable = flash:packages.conf;
   Manual Boot = no
   ```

   A normal INSTALL-mode system points both current and next-reload boot variables to `packages.conf` and has automatic boot enabled.

3. Verify install package state and rollback timer.

   **Example**

   ```
   switch# show install summary
   State (St): I - Inactive, U - Activated & Uncommitted,
               C - Activated & Committed, D - Deactivated & Uncommitted
   --------------------------------------------------------------
   Type  St   Filename/Version
   --------------------------------------------------------------
   IMG   C    <CURRENT_RELEASE>
   --------------------------------------------------------------
   Auto abort timer: inactive
   --------------------------------------------------------------
   ```

   - `C` indicates that the image is active and persistent.
   - `I` indicates that the image is added but inactive.
   - `U` indicates that the installation process is running but not committed.
   - `D` indicates that the image is deactivated but the change is not committed.

An inactive timer is expected when no staged install or rollback decision is pending.

## Troubleshoot image-copy and file-validation issues

Guides you through troubleshooting image-copy and file-validation issues by confirming connectivity and storage, resolving image transfer problems, verifying filenames and SHA-512 hashes, addressing image compatibility, and safely managing free space for upgrade, ISSU, xFSU, SMU, or recovery operations.

**Procedure**

1. Confirm connectivity and storage.
   a) Verify writable filesystems and available storage.

      **Example**

      ```
      switch# show file systems | include flash|usb
      *  11250098176  4123000832  disk  rw  bootflash: flash:
         8000000000  2950000000  disk  rw  flash-2:
                  -           -  disk  rw  usbflash0:
      ```

      Confirm the exact writable filesystem, available space on every required member, and whether USB is detected.

   b) Test transfer-server reachability.

      **Example**

      ```
      switch# ping <SERVER_IP>
      Type escape sequence to abort.
      Sending 5, 100-byte ICMP Echos to <SERVER_IP>, timeout is 2 seconds:
      !!!!!
      Success rate is 100 percent (5/5)
      ```

      Specify the IP address of the server.

      Reachability succeeds in the default routing table. Use the appropriate VRF-aware ping when management traffic uses another VRF.

   c) Verify the route to the transfer server.

      **Example**

      ```
      switch# show ip route <SERVER_IP>
      Routing entry for <SERVER_PREFIX>
        Known via <ROUTING_SOURCE>
        * <NEXT_HOP>, via <INTERFACE>
      ```

      The switch has a route to the transfer server. Confirm that the return path, ACLs, and firewall also permit the chosen protocol.

2. Troubleshoot TFTP transfer behavior.
   a) Verify if the TFTP image copy action was successful.

**Example**

```
switch# copy tftp://192.0.2.10/images/<IMAGE.bin> flash:
Destination filename [<IMAGE.bin>]?
Accessing tftp://192.0.2.10/images/<IMAGE.bin>...
Loading <IMAGE.bin> from 192.0.2.10: !!!!!!!!!!!!!!!!!
[OK - <BYTES> bytes]
<BYTES> bytes copied in <SECONDS> secs
```

The `OK` message confirms transfer completion, not image integrity. Verify the SHA-512 hash next.

b) Investigate a TFTP timeout.

**Example**

```
switch# copy tftp://192.0.2.10/images/<IMAGE.bin> flash:
%Error opening tftp://192.0.2.10/images/<IMAGE.bin> (Timed out)
```

Check the server IP and path, routing/VRF, source interface, UDP reachability, server service, ACL, and firewall. Do not repeatedly retry without correcting the failed dependency.

**3.** Troubleshoot SCP transfer behavior.

a) Verify an interactive SCP image copy.

**Example**

```
switch# copy scp: flash:
Address or name of remote host []? 192.0.2.20
Source username []? upgradeuser
Source filename []? /images/<IMAGE.bin>
Destination filename [<IMAGE.bin>]?
Password:
!!!!!!!!!!!!!!!!!
<BYTES> bytes copied in <SECONDS> secs
```

SCP requires reachable SSH service, valid AAA credentials, and permission to read the exact server path.

b) Investigate SCP permission or path failure

**Example**

```
switch# copy scp://upgradeuser@192.0.2.20//images/<IMAGE.bin> flash:
%Error opening scp://... (Permission denied)
```

The log indicates that the network reachability exists, but authentication, authorization, file permission, or path validation failed. Correct the server-side account or path rather than changing the switch image.

**4.** Troubleshoot USB transfer behavior.

a) Verify USB media detection.

**Example**

```
switch# show file systems | include usb
               -           -  disk  rw  usbflash0:
```

`usbflash0:` confirms that supported media is detected.

For ROMMON recovery, Cisco Catalyst 9000 recovery guidance specifies supported FAT32-formatted USB media.

> **Note**
>
> If usbflash0: is absent, reset the media, confirm its filesystem and platform support, and use the exact platform recovery guide. Do not format the switch flash to resolve a USB detection issue.

b) Verify that the image exists on USB.

**Example**

```
switch# dir usbflash0:<IMAGE.bin>
Directory of usbflash0:/

<FILE-ID>  -rw-  <BYTES>  <DATE>  <IMAGE.bin>
```

c) Copy the image from USB to flash.

**Example**

```
switch# copy usbflash0:<IMAGE.bin> flash:
Destination filename [<IMAGE.bin>]?
Copy in progress...!!!!!!!!!!!!!!!!!!
[OK - <BYTES> bytes]
```

5. Verify the downloaded filename and hash.

a) Verify that the image exists in flash.

**Example**

```
switch# dir flash:<IMAGE.bin>
Directory of flash:/

<FILE-ID>  -rw-  <BYTES>  <DATE>  <IMAGE.bin>
```

b) Verify the SHA-512 hash.

**Example**

```
switch# verify /sha512 flash:<IMAGE.bin>
.....................................................................
```

```
Done!
verify /sha512 (flash:<IMAGE.bin>) = <128-CHAR-SHA512-VALUE>
```

Compare the entire value with the hash for the exact filename on the official Cisco Software Download website. Similar filenames are not interchangeable.

c) Identify a hash mismatch.

**Example**

```
switch# verify /sha512 flash:<IMAGE.bin>
verify /sha512 (flash:<IMAGE.bin>) = <DEVICE-SHA512>
```

You must not install the image if `DEVICE-SHA512` differs from the Cisco published value, or if the file is wrong, incomplete, corrupted, or altered.

Guidance

Prefer SHA-512 when Cisco Software Download lists it.

Hash mismatch example:

Remove only the confirmed bad staged image, copy it again from a trusted source, and repeat the verification. Do not delete `packages.conf`, active .pkg files, crash evidence, or rollback files.

**6.** Resolve image or package incompatibility.

a) Identify an unsupported image during install.

**Example**

```
switch# install add file flash:<IMAGE.bin> activate commit
FAILED: Image is not supported on this platform
```

This error occurs when the filename, platform family, product ID, release path, or cryptographic variant does not match the switch variant.

Do not rename the file or try to bypass the check.

b) Verify the exact product ID.

**Example**

```
switch# show inventory
NAME: "Switch 1", DESCR: "<PLATFORM_DESCRIPTION>"
PID: <EXACT_PRODUCT_ID> , VID: <VERSION> , SN: <SERIAL>
```

c) Verify the current release and image source

**Example**

```
switch# show version | include IOS XE Software|System image file
Cisco IOS XE Software, Version <CURRENT_RELEASE>
System image file is "flash:packages.conf"
```

d) Download the correct software image from the Cisco Software Download website by selecting the exact product ID, current release, target release, and required image family.

Confirm whether the deployment requires the standard, NPE, or other platform-specific image variant and whether an intermediate release or ROMMON update is required. Images of Cisco Catalyst 9000 and C9000 Smart switch families are not interchangeable.

**7.** Safely handle insufficient space.

a) Check free space on flash and retained files.

**Example**

```
switch# dir flash:
Directory of flash:/

<FILE-ID>  -rw-  <BYTES>  <DATE>  packages.conf
<FILE-ID>  -rw-  <BYTES>  <DATE>  <IMAGE.bin>
...
11250098176 bytes total (4123000832 bytes free)
```

Compare free space with the exact target release notes. Check the data for every stack-member or supervisor filesystem involved in the operation.

b) Remove inactive packages after reviewing the deletion list.

**Example**

```
switch# install remove inactive
install_remove: Removing IMG
Cleaning up unnecessary package files
Preparing packages list to delete ... done.

The following files will be deleted:
[1]:
/flash/<OLD-INACTIVE-FILE-1>
/flash/<OLD-INACTIVE-FILE-2>

Do you want to remove the above files? [y/n] y
SUCCESS: Files deleted.
Post_Remove_Cleanup: Passed
```

Read the complete deletion list before answering `Yes (y)`. This command can also select a newly copied `.bin` file.

Use the supported cleanup workflow before copying the target image whenever possible.

Do not run cleanup while an uncommitted upgrade or a possible rollback is still being evaluated. Archive the required crash files externally before removing them.

## Troubleshoot normal install-mode upgrade issues

Guides you through troubleshooting normal INSTALL-mode upgrade issues, including resolving super package conditions, handling add or activate failures, addressing uncommitted or inconsistent install states, and performing rollbacks after commit.

**Procedure**

1. Resolve a scenario which states that super package has been already added.
   a) Identify the target image that has been added already.

      **Example**

      ```
      switch# install add file flash:<IMAGE.bin> activate commit
      FAILED: install_add_activate_commit : Super package already added.
      Add operation not allowed. 'install remove inactive' can be used
      to discard added packages
      ```

   b) Confirm the inactive target image.

      **Example**

      ```
      switch# show install summary
      Type  St   Filename/Version
      --------------------------------
      IMG   C    <CURRENT_RELEASE>
      IMG   I    <TARGET_RELEASE>
      --------------------------------
      Auto abort timer: inactive
      ```

      `I` state means that the target image has already been added and expanded, but it is not active.

   c) Verify the inactive package.

      **Example**

      ```
      switch# show install inactive
      IMG   I    <TARGET_RELEASE>
      ```

      Confirm that the inactive target is the intended, hash-verified release.

   d) Activate the staged package

      **Example**

      ```
      switch# install activate
      ...
      Activate: Passed on [1]
      SUCCESS: install_activate
      ```

   e) Confirm the uncommitted target state.

      **Example**

      ```
      switch# show install summary
      IMG   U    <TARGET_RELEASE>
      Auto abort timer: active on install_activate,
      time before rollback - 01:58:40
      ```

Validate the target while it is uncommitted. Commit it only if the post-upgrade checks pass; otherwise use the supported abort method.

f) Commit the validated target image.

**Example**

```
switch# install commit
...
Commit: Passed on [1]
SUCCESS: install_commit
```

Commit makes the validated target persistent.

Follow these guidelines:

- Preserve **show install** log. If the staged target is the exact verified image that you still intend to run, continue the supported three-step upgrade workflow instead of adding it again.
- If you are abandoning or re-staging the inactive target and no required rollback package must be retained, run the **install remove inactive** command, review the proposed deletions, and retry the approved one-step workflow. Do not rerun the **add** command or do not run **clear install state** command.

**2.** Analyse image add or activate operation failure.

a) Identify the failed install phase.

**Example**

```
switch# install add file flash:<IMAGE.bin> activate commit
install_add_activate_commit: START
...
Add: Passed on [1]
Activate: Failed on [1]
FAILED: install_add_activate_commit <FAILURE_REASON>
```

In this example, the image add operation succeeded, but activation failed. The target can remain staged or the install state can be incomplete.

b) Review the current install state.

**Example**

```
switch# show install summary

>>> <CURRENT AND TARGET STATES>
```

The output displays the current and target states.

c) Preserve and review the install log.

**Example**

```
switch# show install log
```

```
[<OPERATION_ID>|install_add]: END SUCCESS flash:<IMAGE.bin>
[<OPERATION_ID>|install_activate]: END FAILURE <FAILURE_REASON>
```

The install log identifies the failed phase. Preserve the log before attempting any cleanup or retry activity.

Check the exact error, image-platform compatibility, member state, space, boot mode, and any earlier uncommitted operation. If the supported abort operation or inactive image cleanup operation does not restore a clear state, contact Cisco Support.

3. Resolve an activation that succeeded but was not committed.
   a) Confirm the uncommitted target state.

      **Example**

      ```
      switch# show install summary
      Type  St   Filename/Version
      -------------------------------
      IMG   U    <TARGET_RELEASE>
      -------------------------------
      Auto abort timer: active on install_activate,
      time before rollback - 01:52:08
      ```

      The `U` state means that the target is running but not persistent.

      The active timer protects the switch by returning it to the prior committed image if validation or commit does not finish.

   b) Commit the running target image if all post-upgrade checks pass.

      **Example**

      ```
      switch# install commit
      install_commit: START
      Commit: Passed on [1]
      SUCCESS: install_commit
      ```

      Commit operation makes the running image persistent and stops the automatic rollback timer.

   c) Abort the uncommitted target if the new release is unhealthy and it has not been committed.

      **Example**

      ```
      switch# install abort
      This operation may require a reload of the system.
      Do you want to proceed? [y/n] y
      ...
      SUCCESS: install_abort
      ```

      Abort operation returns the installation to its prior state and allows you to reload the switch. Keep console access throughout the operation.

   Do not routinely stop the auto-abort timer; it is an important recovery safeguard.

   Perform commit action only after the new release and services are validated.

4. Resolve an active or inconsistent install state.

a) Review the unexpected install state.

**Example**

```
switch# show install summary

>>> <UNEXPECTED I, U, OR D STATE>
```

Identify the displayed install state—`UNEXPECTED`, `I`, `U`, or `D`.

b) Preserve and review the failed transaction log.

**Example**

```
switch# show install log

>>> <LAST OPERATION DID NOT REACH A SUCCESSFUL END STATE>
```

The log indicates that the last operation did not reach a successful end state. The previous transaction is incomplete. Starting another upgrade in this case can make device recovery harder.

Use only the abort, commit, rollback, or cleanup action supported for the displayed state and exact release. If those supported actions fail, then collect the required diagnostic data and contact Cisco Support.

> **Note**
>
> The **clear install state** is not a normal cleanup command; it can remove provisioned SMUs and rollback points and can reload the switch or stack.

5. Perform a roll back action after commit action.

a) List available rollback points.

**Example**

```
switch# show install rollback
ID    Label          Description
----------------------------------------
2     No Label       No Description
1     <KNOWN_GOOD>   <PREVIOUS_RELEASE>
```

This step confirms that the required saved rollback point exists.

b) Inspect the intended rollback ID.

**Example**

```
switch# show install rollback id 1
Rollback id - 1
Type  St   Filename/Version
```

```
---------------------------------
IMG   C    <PREVIOUS_RELEASE>
```

Confirm the exact rollback ID, release, and retained files before proceeding.

c) Roll back to the selected rollback ID.

**Example**

```
switch# install rollback to id 1
install_rollback: START
...
SUCCESS: install_rollback
```

A rollback action is disruptive and can reload the system. Validate the exact path and expected impact in the platform release notes.

Do not remove inactive files until the rollback decision is complete. If the requested rollback point is absent, use an approved normal install or downgrade procedure only when that path is supported.

## Troubleshoot legacy bundle-mode issues

Guides you through troubleshooting legacy BUNDLE-mode upgrade issues, including recovering stack members that are unable to boot the bundle image and performing supported conversions from BUNDLE mode to INSTALL mode.

Use this procedure to troubleshoot legacy bundle-mode upgrade issues, including recovering stack members that are unable to boot the bundle image and to perform supported conversions from bundle mode to install mode.

Use the exact procedure for source release because older releases can require a different syntax. Do not mix deprecated **request platform software package** commands with the current install workflow unless the exact source release requires them.

Cisco recommends install mode. Use bundle mode only when the exact Cisco Catalyst 9000 platform and release support it.

**Procedure**

1. Recover a stack member that cannot boot the bundle image.
   a) Identify the mismatched stack member.

   **Example**

   ```
   switch# show switch
   Switch#  Role      State
   -------------------------
   *1       Active    Ready
    2       Member    V-Mismatch
   ```

   The `V-Mismatch` state means the member is running incompatible software or did not boot the intended image.

   b) Verify that the bundle image exists on the active member.

**Example**

```
switch# dir flash:<IMAGE.bin>

 <IMAGE.bin>
```

c) Verify whether the image is missing from the affected member.

**Example**

```
switch# dir flash-2:<IMAGE.bin>
%Error opening flash-2:<IMAGE.bin> (No such file or directory)
```

In bundle mode the complete `.bin` must be available to every stack member through its configured boot path.

d) Copy the bundle image to the affected member.

**Example**

```
switch# copy flash:<IMAGE.bin> flash-2:
Destination filename [<IMAGE.bin>]?
Copy in progress...!!!!!!!!!!!!!!!!!!
[OK - <BYTES> bytes]
```

e) Verify the copied image hash on the affected member.

**Example**

```
switch# verify /sha512 flash-2:<IMAGE.bin>
verify /sha512 (flash-2:<IMAGE.bin>) = <MATCHING_SHA512>
```

The affected member now has the complete hash-verified bundle image. Confirm that the saved boot statement uses the exact image on all members before the approved reload.

Guidance

Copy and verify the exact same image on the affected member before the next approved reload. Also confirm the boot variable on every member. Do not format member flash as the first recovery action.

**2.** Convert from bundle to install mode. If the platform supports conversion and the target image is verified, use the supported one-step install workflow.

a) Confirm current BUNDLE mode.

**Example**

```
switch# show version | begin Switch Ports
Switch Ports Model        SW Version  SW Image       Mode
------ ----- ------------ ----------- -------------- -------
*    1 48    C9300-48P    <RELEASE>   CAT9K_IOSXE   BUNDLE
```

b) Configure the next boot target for INSTALL mode.

**Example**

```
switch# configure terminal
switch(config)# no boot system
switch(config)# boot system flash:packages.conf
```

```
switch(config)# no boot manual
switch(config)# end
```

c) Save the boot configuration.

**Example**

```
switch# write memory
Building configuration...
[OK]
```

d) Verify the next boot target.

**Example**

```
switch# show boot
Boot Variables on next reload:
BOOT variable = flash:packages.conf;
Manual Boot = no
```

Confirm the persistent next-reload target before running the installation. A stack can require the platform-specific switch-all boot syntax.

e) Run the supported conversion workflow for install operation.

**Example**

```
switch# install add file flash:<IMAGE.bin> activate commit
...
Add: Passed on [1]
Activate: Passed on [1]
Commit: Passed on [1]
SUCCESS: install_add_activate_commit
```

The conversion is disruptive. Do not reload manually if the **install** command fails before `packages.conf` file is created.

After the automatic reload, verify the install mode, the `packages.conf` file , and committed state.

## Troubleshoot ISSU issues

Guides you through troubleshooting ISSU issues by verifying readiness, resolving rejection due to configuration synchronization or software mismatch, and investigating stalled ISSU operations, including log collection, commit, and abort procedures.

Use this procedure to troubleshoot ISSU issues by verifying readiness, resolving rejection due to configuration synchronization or software mismatch, and analyse stalled ISSU operations, including log collection, commit, and abort procedures.

The three-step ISSU workflow has an automatic rollback timer. The **install commit** operation makes the new software permanent and closes the rollback window. The one-step ISSU workflow commits automatically and therefore has no ISSU pre-commit rollback window.

If **install abort issu** does not clear the state, preserve the diagnostics data and contact Cisco Support. Do not self-run service internal or clear install state. Do not run **install remove inactive** while rollback is still required.

### Before you begin

Verify ISSU compatibility. Refer the ISSU Compatibility Matrix and In-Service Software Upgrade (ISSU) Release Matrix.

**Procedure**

1. Confirm ISSU readiness.
   a) Confirm that no ISSU operation is active.

      **Example**

      ```
      switch# show issu state detail
      Current ISSU Status: Enabled
      Previous ISSU Operation: N/A
      ...
      No ISSU operation is in progress
      ```

      This is the expected starting state. The ISSU status indicating `Enabled` alone is not sufficient; confirm that no ISSU operation is active.

   b) Verify STANDBY HOT and SSO state.

      **Example**

      ```
      switch# show redundancy states
      my state = 13 -ACTIVE
      peer state = 8 -STANDBY HOT
      Redundancy Mode (Operational) = sso
      Redundancy Mode (Configured)  = sso
      Communications = Up
      ```

      An ISSU-capable redundant system requires a synchronized STANDBY HOT peer, operational SSO, and communications Up.

   c) Verify configuration synchronization.

      **Example**

      ```
      switch# show redundancy config-sync failures mcl
      Mismatched Command List
      -----------------------
      The list is Empty
      ```

   d) Verify parser return-code synchronization.

      **Example**

      ```
      switch# show redundancy config-sync failures prc
      The list is Empty
      ```

   e) Verify best-effort synchronization.

**Example**

```
switch# show redundancy config-sync failures bem
The list is Empty
```

Empty lists mean no reported active-to-standby synchronization failures.

f) Verify ISSU compatibility inputs.

**Example**

```
switch# show version
switch# show boot
switch# show module
switch# show install summary
switch# show issu comp-matrix
```

2. Resolve ISSU rejection by configuration synchronization.
   a) Identify MCL synchronization failures and remove or modify the incompatible commands.

**Example**

If the device displays this ERROR log, run the **show redundancy config-sync failures historic mcl** command.

```
ERROR: MCL-Terminating the current ISSU Operation.
ISSU Abort operation is initiated

ERROR: Once ISSU abort is done please check full list of
mismatched commands via:
show redundancy config-sync failures historic mcl
```

```
switch# show redundancy config-sync failures historic mcl
...
<INCOMPATIBLE_COMMANDS>
```

The output indicates that at least one configured command cannot synchronize to the target release.

Review each command and its service impact. Remove or modify it only when it is operationally safe. If it cannot be removed, use a normal INSTALL-mode upgrade; do not force the ISSU operation.

   b) Identify software or ROMMON mismatch.

**Example**

```
%REDUNDANCY-2-IPC: IOS versions do not match.
```

The error log indicates that there is a mismatch of software and ROMMON versions.

Compare the software and ROMMON on both supervisors or members. ISSU must not continue until the exact platform requirements are met.

3. Analyse an ISSU operation that appears stalled.
   a) Check the current ISSU stage.

**Example**

```
switch# show issu state detail
Current ISSU Status: In Progress
Previous ISSU Operation: N/A
```

`In Progress` status can be normal during a transition. Compare the current stage and timestamps with the expected change-window timing.

b) Recognize a new ISSU operation that is blocked.

**Example**

```
switch# install add file flash:<IMAGE.bin> activate issu commit
FAILED: ISSU in progress; please use install abort issu to abort
the ISSU operation OR use install commit to commit the ISSU operation.
```

This error log indicates that an earlier three-step ISSU transaction is still active. Do not start another add or activation step in between.

c) Collect ISSU, install, redundancy, and log state.

**Example**

```
switch# show clock
switch# show issu state detail
switch# show issu sessions
switch# show issu entities
switch# show issu negotiated
switch# show install summary
switch# show install log
switch# show redundancy
switch# show redundancy states
switch# show logging | include ISSU|INSTALL|REDUNDANCY
```

d) Commit an intentionally uncommitted ISSU target if post-upgrade validation has passed and the active three-step ISSU is awaiting commit.

**Example**

```
switch# install commit
...
SUCCESS: install_commit
```

The commit operation makes the ISSU target persistent and closes its rollback window.

e) If the active three-step ISSU is uncommitted and must be abandoned, then abort that active uncommitted ISSU operation.

**Example**

```
switch# install abort issu
...rollback messages...
SUCCESS: ISSU abort completed
```

This operation returns the system to the previously installed software and can reload supervisors or members. Use it only after confirming that ISSU is active and not committed.

## Troubleshoot xFSU issues

Guides you through troubleshooting xFSU issues on Catalyst 9300 series switches, including resolving eligibility failures, analyzing incomplete status, verifying reload-fast events, investigating failed or stalled operations, and understanding the rollback boundary and commit behavior for supported upgrade workflows.

Use this procedure to troubleshoot xFSU issues on Catalyst 9300 series switches, including resolving eligibility failures, analyzing incomplete status, verifying reload-fast events, investigating failed or stalled operations, and understanding the rollback boundary and commit behavior for supported upgrade workflows.

### Before you begin

- Refer the xFSU guidelines mentioned in the *Upgrade* chapter.
- Understand the xFSU rollback boundary: The xFSU command includes commit operation. After a successful committed xFSU, returning to an older image is a separate normal install or downgrade operation. Perform the downgrade operation only when the path is supported with normal maintenance impact. The **clear xfsu state** command clears the recorded last-reload state; it does not restore the previous image. Do not use it while an operation is active or before collecting the diagnostics data.

### Procedure

1. Resolve xFSU eligibility failures.
   a) Review xFSU eligibility failure details.

      **Example**

      ```
      switch# show xfsu eligibility
      Eligibility Check       Status
      ==================      ======
      Autoboot Enabled        Yes
      Install Mode            Yes
      Spanning Tree           Eligible
      MacSec                  Eligible

      FAILED: FPGA on switch 1 does not support xFSU.
      Please upgrade the FPGA via a normal install.

      XFSU Eligibility Summary
      ============================
      XFSU Eligibility: NOT ELIGIBLE
      ============================
      ```

      Do not start xFSU in this case. This device requires a normal INSTALL-mode upgrade to update the unsupported FPGA.

   b) Verify graceful-reload client readiness.

**Example**

```
switch# show graceful-reload
Graceful Reload Infra Status: Not running
Client OSPF          Status: Up
Client GR_CLIENT_BGP Status: Up
Client LACP_xFSU     Status: Up
Client GR_CLIENT_RIB Status: Up
Client GR_CLIENT_FIB Status: Up
```

Configured graceful-reload clients relevant to the deployment must be `Up` before xFSU.

On a supported stack, also confirm that full-ring stack, required license, and SSO checks report status as eligible. If SSO is not ready, wait for the standby and stack to become healthy; do not force the operation.

Typical blockers for xFSU eligibility include unsupported FPGA, disabled autoboot, BUNDLE mode, partial stack ring, unsupported spanning-tree condition, MACsec or feature restrictions, or an unsupported topology-protocol combination. Use the exact reason in the output and refer the xFSU guide.

2. Analyse incomplete xFSU status.
   a) Check an intermediate xFSU status.

**Example**

```
switch# show xfsu status
XFSU PLATFORM Status: Stack converged, waiting for standby reconnect
```

This can be a normal intermediate state. If it remains unchanged beyond the approved maintenance plan, collect diagnostics and do not start another xFSU.

A completed xFSU platform status looks like this:

```
switch# show xfsu status
XFSU PLATFORM Status: Data plane update done, Upgrade done in 1590 ms.
```

   b) Verify the device reload reason.

**Example**

```
switch# show version | include reason
Last reload reason: Image Install with Reloadfast
```

```
switch# show logging | include FAST
%FED_IPC_MSG-5-FAST_RELOAD_COMPLETE: Fast reload operation completed
```

The reload reason and completion message confirm the reduced-downtime reload path.

   c) Collect diagnostics data for failed or stalled xFSU.

**Example**

```
switch# show xfsu eligibility
switch# show xfsu status
switch# show graceful-reload
switch# show switch
switch# show install summary
switch# show install log
switch# show version running
```

```
switch# show logging
switch# show platform hardware fpga switch 1 | include Version
switch# show inventory
switch# show etherchannel summary
switch# show spanning-tree
switch# dir crashinfo-1:
switch# show tech-support xfsu
```

Older releases can use `reloadfast` variants of the status and technical-support commands.

## Troubleshoot SMU issues

Guides you through confirming SMU compatibility, troubleshooting SMU add or activation failures, rolling back uncommitted SMUs, and properly removing committed SMUs using the INSTALL workflow.

Use this procedure to confirm SMU compatibility, troubleshoot SMU add or activation failures, roll back uncommitted SMUs, and properly remove committed SMUs using the INSTALL workflow.

You must not manually delete a committed SMU. Follow the exact SMU readme: procedure.

**Procedure**

1. Confirm exact SMU compatibility.

   The IOS XE SMU installation uses INSTALL mode. Cisco publishes SMUs for eligible Extended Maintenance Releases (EMRs), and the package must exactly match the platform and base release. Follow the SMU `readme` because activation and removal impact differs by package.

   a) Verify the base release and INSTALL mode.

      **Example**

      ```
      switch# show version | include IOS XE Software|System image file
      Cisco IOS XE Software, Version <BASE_RELEASE>
      System image file is "flash:packages.conf"
      ```

      The `packages.conf` file confirms INSTALL mode. The SMU must match this exact base release and platform.

   b) Verify committed base image state.

      **Example**

      ```
      switch# show install summary
      Type  St   Filename/Version
      --------------------------------
      IMG   C    <BASE_RELEASE>
      --------------------------------
      Auto abort timer: inactive
      ```

      The output shows that base image is committed and no earlier install transaction is waiting.

   c) Verify the hash value of SMU file.

**Example**

```
switch# verify /sha512 flash:<SMU_FILE>
verify /sha512 (flash:<SMU_FILE>) = <SHA512_VALUE>
```

Compare this value with the one present in Cisco Software Download page.

d) Inspect SMU package metadata.

**Example**

```
switch# show install package flash:<SMU_FILE>
Name: <SMU_FILE>
Version: <MATCHING_BASE_RELEASE>
Platform: <MATCHING_PLATFORM>
Package Type: SMU
Defect ID: <BUG_ID>
...
```

Confirm the exact platform, release, package type, and whether activation causes a process restart, switchover, or reload.

**2.** Investigate SMU add or activation failure.

a) Identify the rejected SMU package.

**Example**

```
switch# install add file flash:<SMU_FILE> activate commit
install_add_activate_commit: START
...
SMU_ADD: Failed on [1]
FAILED: <COMPATIBILITY_OR_PACKAGE_ERROR>
```

The output shows that package was rejected. Do not rename it or force activation.

b) Preserve and review the install log.

**Example**

```
switch# show install log
[<OPERATION_ID>|SMU_ADD]: END FAILURE <FAILURE_REASON>
```

Preserve the exact failure logs and confirm the platform, base release, package hash, and package readme.

**3.** Roll back an uncommitted SMU.

For the currently pending uncommitted transaction, the exact release can also support **install abort**. Use the command only after confirming the displayed state.

a) Verify that the SMU is active but uncommitted.

**Example**

```
switch# show install uncommitted
SMU   U    flash:<SMU_FILE>
```

The output shows that SMU is active but not committed.

b) Roll back to the last committed install state.

**Example**

```
switch# install rollback to committed
install_rollback: START
...
SMU_ROLLBACK: Passed on [1]
SUCCESS: install_rollback
```

The switch returns to the last committed install state.

The SMU rollback requires a reload.

4. Remove a committed SMU.
   a) Deactivate the committed SMU.

**Example**

```
switch# install deactivate file flash:<SMU_FILE>
...
SMU_DEACTIVATE: Passed on [1]
SUCCESS: install_deactivate flash:<SMU_FILE>
```

   b) Verify the deactivated SMU state.

**Example**

```
switch# show install summary
SMU   D    flash:<SMU_FILE>
IMG   C    <BASE_RELEASE>
```

The status `D` means the SMU is deactivated but the change is not yet committed.

   c) Remove the inactive SMU package.

**Example**

```
switch# install remove file flash:<SMU_FILE>
...
SMU_REMOVE: Passed on [1]
SUCCESS: install_remove flash:<SMU_FILE>
```

Deactivation can restart a process or reload the device, as stated in the SMU metadata or readme. The remove step deletes the already inactive package.

## Troubleshoot stack, supervisor, and synchronization issues

Guides you through troubleshooting stack-member version mismatches, resolving filesystem and synchronization failures, enabling automatic member upgrade recovery, and addressing redundant-peer states that are not STANDBY HOT.

Use this procedure to troubleshoot stack-member version mismatches, resolvie filesystem and synchronization failures, enable automatic member upgrade recovery, and address redundant-peer states that are not STANDBY HOT.

**Procedure**

1. Resolve a stack member version mismatch state.

   Follow these guidelines:

   - Confirm that the member is physically stable and recognized, then check its filesystem and storage. In INSTALL mode, use the supported automatic upgrade or recovery workflow for the exact release. Do not format the member flash as a first response.
   - For an INSTALL-mode stack whose mismatched member is still participating in the stack, the exact release can support automatic member upgrade.
   - Automatic member upgrade can start a few minutes after the mismatch is detected. It cannot recover a member that is already at the `switch: bootloader` prompt or is not participating in the stack. Use console access and the exact platform recovery guide in that condition.

   a) Identify the mismatched stack member.

      **Example**

      ```
      switch# show switch
      Switch#  Role      Mac Address       Priority  Version  State
      -------------------------------------------------------------
      *1       Active    <MAC>             15        V01      Ready
       2       Member    <MAC>             14        V02      V-Mismatch
      ```

      The `V-Mismatch` state means member 2 is running software that is incompatible with the active member.

   b) Check the affected member filesystem and free space.

      **Example**

      ```
      switch# dir flash-2:
      Directory of flash-2:/
      ...
      <FREE_BYTES> bytes free
      ```

      Check the expected image or package files and free space of the affected member.

   c) Review install synchronization failure details.

      **Example**

      ```
      switch# show install log
      [2]: Failed to copy file flash:<IMAGE.bin> from active switch 1
      operation aborted
      ```

      The output shows that image synchronization to member 2 failed.

   d) Enable automatic member upgrade when supported.

      **Example**

      ```
      switch# configure terminal
      switch(config)# software auto-upgrade enable
      switch(config)# end
      ```

   e) Save the auto-upgrade configuration.

**Example**

```
switch# write memory
Building configuration...
[OK]
```

Auto-upgrade is disabled by default and is available only in INSTALL mode. The active switch checks a mismatched joining member and automatically supplies compatible software.

f) Verify that the recovered member is ready.

**Example**

```
switch# show switch
Switch#  Role      State
------------------------
*1       Active    Ready
 2       Member    Ready
```

The `Ready` state confirms that the member completed compatibility recovery and joined the stack. Capture the logs and validate its running release and install state.

2. Resolve a redundant peer that is not STANDBY HOT.

Review software and ROMMON version alignment, configuration synchronization, peer console logs, modules, and platform health. A normal disruptive upgrade might be better than trying to force a reduced-downtime method.

a) Verify the peer redundancy state.

**Example**

```
switch# show redundancy states
my state = 13 -ACTIVE
peer state = 4 -STANDBY COLD
Redundancy Mode (Operational) = rpr
Communications = Up
```

The output shows that peer is not synchronized for ISSU. Do not start ISSU until the exact cause is corrected and STANDBY HOT/SSO is restored.

## Troubleshoot a device that boots the previous release

Guides you through diagnosing a switch that boots the previous release by inspecting INSTALL and BUNDLE mode boot targets, validating packages.conf and related package references, and correcting the persistent boot variable following Cisco recommended procedures.

Use this procedure to diagnose a switch that boots the previous release by inspecting INSTALL and BUNDLE mode boot targets, validate `packages.conf` and related package references, and to correct the persistent boot variable following Cisco recommended procedures.

**Procedure**

1. Diagnose INSTALL-mode boot to the old release.

Do not manually edit `packages.conf` or any `.pkg` file. If another validated `.conf` exists, verify all its referenced files and follow the exact Cisco recovery procedure before changing the boot target.

a) Verify the running release and boot source.

**Example**

```
switch# show version | include IOS XE Software|System image file
Cisco IOS XE Software, Version <UNEXPECTED_OLD_RELEASE>
System image file is "flash:packages.conf"
```

Check if the output shows the old release version.

b) Verify the current boot variables.

**Example**

```
switch# show boot
Current Boot Variables:
BOOT variable = flash:packages.conf;

Boot Variables on next reload:
BOOT variable = flash:packages.conf;
Manual Boot = no
```

The INSTALL-mode boot setting is present in this output. Check the packages which the provisioning file references.

c) Inspect `packages.conf` references.

**Example**

```
switch# more flash:packages.conf | include .pkg
boot rp 0 0 rp_boot cat9k-rpboot.<OLD_RELEASE>.SPA.pkg
iso  rp 0 0 rp_base cat9k-rpbase.<OLD_RELEASE>.SPA.pkg
iso  rp 0 0 rp_daemons cat9k-rpbase.<OLD_RELEASE>.SPA.pkg
```

The `packages.conf` still references the old package set. This indicates that activation or commit did not complete as expected.

d) Verify the committed install state.

**Example**

```
switch# show install summary
IMG   C    <OLD_RELEASE>
```

Check if the output shows the old release version.

e) Preserve and review the install log.

**Example**

```
switch# show install log
[<OPERATION_ID>|install_add]: END SUCCESS flash:<TARGET_IMAGE.bin>
[<OPERATION_ID>|install_activate]: END FAILURE <FAILURE_REASON>
```

Preserve the log; it identifies the failed stage.

**2.** Diagnose a persistent BUNDLE boot variable that still points to an old BUNDLE image.

a) Verify that the switch booted the old bundle image.

**Example**

```
switch# show version | include IOS XE Software|System image file
Cisco IOS XE Software, Version <UNEXPECTED_OLD_RELEASE>
System image file is "flash:<OLD_IMAGE.bin>"
```

b) Verify the next-reload boot variable.

**Example**

```
switch# show boot
Boot Variables on next reload:
BOOT variable = flash:<OLD_IMAGE.bin>;
Manual Boot = no
```

As per the output, the next reload still selects the old monolithic image.

c) Verify that the `packages.conf` file exists.

**Example**

```
switch# dir flash:packages.conf
Directory of flash:/
<FILE_ID> -rw- <SIZE> <DATE> packages.conf
```

d) Inspect target package references.

**Example**

```
switch# more flash:packages.conf | include .pkg
<TARGET_RELEASE_PACKAGE_REFERENCES>
```

Before changing the boot variable, verify that the `packages.conf` file belongs to the intended committed release and that every referenced `.pkg` file is present.

3. Correct the persistent boot target after validating the provisioning file and all referenced packages.

a) Configure `packages.conf` file as the boot target.

**Example**

```
switch# configure terminal
switch(config)# no boot system
switch(config)# boot system flash:packages.conf
switch(config)# no boot manual
switch(config)# end
```

b) Save the corrected boot configuration.

**Example**

```
switch# write memory
Building configuration...
[OK]
```

c) Verify the corrected boot target.

**Example**

```
switch# show boot
Boot Variables on next reload:
BOOT variable = flash:packages.conf;
Manual Boot = no
```

The output shows that corrected persistent boot target is `packages.conf`. A stack can require the platform-specific **switch-all** syntax.

Do not reload merely because the boot variable was corrected. First confirm the install state, image or package integrity, console access, and the approved maintenance plan.

## Recover from ROMMON or bootloader failures

Guides you through diagnosing ROMMON or bootloader failures, validating INSTALL-mode provisioning files, and temporarily booting a known-good image from local or USB media when no valid provisioning file exists.

Use this procedure to diagnose ROMMON or bootloader failures, validating install-mode provisioning files, and temporarily booting a known-good image from local or USB media when no valid provisioning file exists.

**Procedure**

1. Confirm ROMMON or bootloader failure.

   **Example**

   ```
   boot: attempting to boot from [flash:packages.conf]
   ERROR: failed to boot from flash:packages.conf (Not Found)
   ```

   The typical symptom for ROMMON or bootloader failure includes these error messages on the device console.

2. Confirm that the `packages.conf` file is present and that every referenced `.pkg` file is present.

   **Example**

   ```
   switch: dir flash:
   ...
   packages.conf
   cat9k-rpboot.<RELEASE>.SPA.pkg
   cat9k-rpbase.<RELEASE>.SPA.pkg
   ```

3. Boot the validated install-mode provisioning file if a valid `packages.conf` file exists.

   **Example**

   ```
   switch: boot flash:packages.conf
   ```

4. Temporarily boot a known-good image when no valid provisioning file exists.
   - Boot a known-good local image from flash.

     ```
     switch: boot flash:<KNOWN_GOOD_IMAGE.bin>
     ```

     This can temporarily boot the monolithic image. After the IOS XE operating system starts, verify the image and use the supported procedure to restore INSTALL mode.
   - Boot a known-good USB image from a supported USB media.

     ```
     switch: dir usbflash0:
     switch: boot usbflash0:<KNOWN_GOOD_IMAGE.bin>
     ```

     Use the exact platform recovery guide. Direct boot does not automatically create a correct persistent INSTALL-mode state.

   If no validated boot file exists, stop and use the exact Cisco platform recovery procedure or contact Cisco Support. Do not format the flash or use emergency-install process. It can erase software, files, and configuration and platform restrictions also apply.

## Troubleshoot post-upgrade system or service issues

Guides you through post-upgrade troubleshooting by verifying the committed state of the target image, comparing system and service health with baseline data, and collecting diagnostics data for unexpected reloads or crash conditions.

Use this procedure to troubleshoot the committed state of the target image, compare system and service health with baseline data, and to collect diagnostics data for unexpected reloads or crash conditions.

Compare the post-upgrade results with the pre-upgrade baseline:

If the new image is uncommitted and service health is unacceptable, use the supported abort or rollback. If it is already committed, use the approved rollback point or a supported normal install/downgrade plan. Collect evidence before the action because logs can be overwritten.

**Procedure**

1. Verify whether the target image is committed.
   a) Check for an uncommitted target image.

      **Example**

      ```
      switch# show install summary
      IMG   U    <TARGET_RELEASE>
      Auto abort timer: active on install_activate,
      time before rollback - 01:40:20
      ```

      The output shows that the target is uncommitted. If service validation fails, preserve the diagnostics data and use the supported abort or rollback procedure before the timer expires.

   Guidance

If the new image is uncommitted and service health is unacceptable, use the supported abort or rollback. If it is already committed, use the approved rollback point or a supported normal install/downgrade plan. Collect evidence before the action because logs can be overwritten.

2. Compare system and service health with the pre-upgrade baseline.
   a) Verify platform resource health

      **Example**

      ```
      switch# show platform resources
      Control Processor   <VALUE>   ... State H
      DRAM                <VALUE>   ... State H
      ```

      The status `H` means healthy. Warning or critical state, sustained high CPU, or abnormal memory use requires investigation.

   b) Verify process-health counters.

      **Example**

      ```
      switch# show platform software process list switch active R0 summary
      Zombies          : 0
      Stopped          : 0
      Paging           : 0
      ```

      Zero is expected for these process-health counters.

   c) Review sustained platform CPU.

      **Example**

      ```
      switch# show processes cpu platform sorted 5min location switch active R0
      ...
      ```

      Compare sustained process CPU with the pre-upgrade baseline; a brief re-convergence spike is normal.

   d) Verify forwarding object-manager stability.

      **Example**

      ```
      switch# show platform software object-manager switch active F0 statistics
      Object update: Pending-issue: 0, Pending-acknowledgement: 0
      Stale-objects: 0
      Error-objects: 0
      Paused-types: 0
      ```

      Persistent nonzero pending, stale, error, or paused objects can indicate forwarding-programming instability.

   e) Review interface error counters.

      **Example**

      ```
      switch# show interfaces counters errors
      <INTERFACE COUNTERS>
      ```

      Compare new errors, discards, or link changes with the pre-upgrade values.

   f) Verify port channel state.

**Example**

```
switch# show etherchannel summary
<PORT-CHANNEL STATE>
```

g) Verify spanning-tree inconsistent ports.

**Example**

```
switch# show spanning-tree inconsistentports
Name                 Interface              Inconsistency
-------------------- ---------------------- ------------------
Number of inconsistent ports (segments) in the system : 0
```

Port channels, spanning tree, routing neighbors, first-hop redundancy, and application hosting must return to their expected state.

If the new image is uncommitted and service health is unacceptable, use the supported abort or rollback procedure. If it is already committed, use the approved rollback point or a supported normal install or downgrade plan. Collect diagnostics data before the action because logs can be overwritten.

3. Investigate unexpected system reload or crash.
   a) Verify reload reason and up time.

**Example**

```
switch# show version | include uptime|returned|Last reload reason
switch uptime is 18 minutes
System returned to ROM by LocalSoft
Last reload reason: LocalSoft
```

A `LocalSoft` or `CPU catastrophic error` is not a normal image-install reload. Collect crash diagnostics data immediately.

   b) Review onboard reset history.

**Example**

```
switch# show logging onboard switch active uptime detail
Number of resets        : 4
Current reset reason    : LocalSoft
Current reset timestamp : <EVENT_TIME>
Current uptime          : 0 years 0 weeks 0 days 0 hours 18 minutes
```

Record the reason and timestamp and correlate them with the console session.

   c) Locate crash, core, and system-report files.

**Example**

```
switch# dir /recursive all-filesystems | include Directory
of|crashinfo_|core.gz|fullcore|system-report|kernel.rp|threshold_lowmem_info
Directory of crashinfo:/
  <FILE_ID> -rw- <SIZE> <EVENT_TIME> switch-system-report_<TIMESTAMP>.tar.gz
Directory of flash:/core/
  <FILE_ID> -rw- <SIZE> <EVENT_TIME> switch_<PROCESS>_<TIMESTAMP>.core.gz
```

Preserve files whose timestamps match the event. Do not delete them to make space before copying them externally.

## Troubleshoot Catalyst Center SWIM upgrades

Guides you through troubleshooting Cisco Catalyst Center SWIM upgrades, including reviewing task and device state, collecting device-side data, and verifying inventory, platform health, and compliance before retrying failed image distribution, readiness, activation, reload, or post-check operations.

Use this procedure to troubleshoot the upgrade issues if the upgrade was started from Cisco Catalyst Center.

### Before you begin

Follow these guidelines and steps if the upgrade was started from Cisco Catalyst Center:

- Record the parent task ID, child task ID, full error, timestamp or timezone, device, product ID, current release, target release, boot mode, and topology.
- Check image-repository metadata and the intended golden-image assignment.
- Confirm inventory reachability, compliance, device credentials, platform health, and available storage.
- If the inventory version is stale, correct reachability and resynchronize the device before deciding to retry.
- Review the complete task history to identify whether the failure occurred during distribution, readiness, activation, reload, or post-check.
- Do not start a parallel CLI upgrade while a SWIM task is active.
- Retry only after both the Catalyst Center task and device install state are clear.

#### Procedure

Collect device-side SWIM evidence using these commands.

Replace switch or member and location values with the syntax supported on the exact platform.

#### Example

```
switch# show tech-support install
switch# show platform software install-manager switch 1 R0 operation history
detail
switch# show platform software install-manager switch 1 R0 operation current
detail
switch# show install summary
switch# show install log
switch# show logging
switch# show switch
switch# show redundancy
switch# show platform resources
```

Save the Catalyst Center task export together with the switch console log.

## Technical support criteria

Outlines the criteria for stopping self-recovery and opening a Cisco Support case for repeated ROMMON recovery, boot failures, synchronization issues, install inconsistencies, failed updates, unexpected reloads, or persistent recovery and rollback problems.

Stop self-recovery and open a Cisco Support case when any of these conditions applies:

- The device repeatedly returns to ROMMON or no validated boot image or provisioning file is available.
- A stack member or supervisor remains missing, version mismatch, or unable to synchronize after storage and image checks.
- ISSU or xFSU remains active or stalled even after performing the abort procedure.
- Install state remains inconsistent after the supported commit, abort, rollback, or inactive cleanup proedure.
- A bootloader, ROMMON, CPLD, or FPGA update fails or is interrupted.
- An unexpected reload, process crash, watchdog, low-memory event, or forwarding failure occurs.
- The suggested recovery process requires clearing install state, flash formatting, emergency-install, service internal, or manual changes to package file.
- The correct rollback point or required package files are absent.
- The same validated operation fails again after the underlying cause was corrected.

## Collect diagnostics data

Guides you through recording case context, gathering detailed failure and device information, running supported evidence commands, creating and verifying support files after confirming free space, and adding feature-specific technical-support output as required for diagnostics and Cisco Support.

Use this procedure to collect diagnostics data before contacting Cisco Support for device upgrade issues.

> **Note**
>
> Commands that require service internal or standby-specific trace access must be run only when directed by Cisco Support.

### Procedure

1. Record case context and failure details.
   a) Record exact event time and timezone.
   b) Record the product IDs, serial-number mapping if required for the case, topology, stack or SVL or HA roles, and affected member or supervisor.
   c) Record the source release, target release, boot mode, upgrade method, exact image or SMU filename, and published hash.
   d) Record the complete console session from the first command through the failure and recovery attempt.
   e) Record every prompt response, failure message, service impact, and whether the issue is reproducible.

f) Record whether the failure occurred during copy, add, activate, reload or switchover, commit, rollback, or post-validation process.
g) Record Catalyst Center task IDs and exports when SWIM initiated the upgrade.

2. Run the supported diagnostics commands.

**Example**

```
terminal length 0
show clock
show version
show boot
show install summary
show install active
show install committed
show install inactive
show install uncommitted
show install log

show switch
show switch detail
show redundancy
show redundancy states
show redundancy config-sync failures historic mcl
show issu state detail
show install issu history
show xfsu eligibility
show xfsu status
show graceful-reload

show module
show platform
show inventory
show environment all
show platform resources
show processes cpu sorted
show processes cpu platform sorted 5min
show processes memory platform sorted

show logging
show logging onboard switch active uptime detail
show reload history
show file systems
dir flash:
dir crashinfo:
dir /recursive all-filesystems | include Directory
of|crashinfo_|core.gz|fullcore|system-report|kernel.rp|threshold_lowmem_info

show tech-support install
show platform software install-manager switch 1 R0 operation current detail
show platform software install-manager switch 1 R0 operation history detail
```

The **show reload history** is available only in newer IOS XE releases; use **onboard uptime history** instead on older releases where **show reload history** is not available.

The install-manager syntax varies by stack member, supervisor, and platform location. Use command help to select the supported form.

3. Create support files after confirming free space on the device.
a) Collect the output of **show tech-support** command and save to a file on the device.

**Example**

```
switch# show tech-support | redirect flash:ShowTech-after-upgrade.txt
```

b) Create a platform trace archive.

**Example**

```
switch# request platform software trace archive
Creating archive file [flash:switch_1_RP_0_trace_archive-<TIMESTAMP>.tar.gz]
Done with creation of the archive file
```

The trace archive preserves IOS XE process traces that can later be overwritten.

c) Verify support files are present in the flash.

**Example**

```
switch# dir flash: | include ShowTech|trace_archive|.tar.gz
<FILE_ID> -rw- <SIZE> <DATE> ShowTech-after-upgrade.txt
<FILE_ID> -rw- <SIZE> <DATE> switch_1_RP_0_trace_archive-<TIMESTAMP>.tar.gz
```

4. Copy these files and all event-matching crash or system-report files off the switch and attach them to the case. Add feature-specific technical-support output when applicable.
5. Collect stack, StackWise Virtual, diagnostic, platform, or xFSU support output.

**Example**

```
switch# show tech-support stack
switch# show tech-support stackwise-virtual switch all
switch# show tech-support diagnostic
switch# show tech-support platform
switch# show tech-support xfsu
```

## Additional references

Provides links to Cisco resources for Catalyst 9000 upgrade and recovery, bootloader and ROMMON procedures, mode conversion, compatibility and release planning, ISSU, xFSU, SMU, SWIM troubleshooting, reloads, and operational diagnostics data collection.

Use these guides and technical references to support upgrade troubleshooting, recovery, compatibility validation, software-maintenance upgrade handling, Catalyst Center SWIM troubleshooting, and diagnostics data collection.

**Compatibility and release planning**

- Cisco IOS XE Software Forensic Data Collection Procedures
- Catalyst 9000 Series Comp Matrix Tool
- In-Service Software Upgrade (ISSU)
- Recommended Releases for Catalyst 9200/9300/9400/9500/9600 Platforms

**Upgrade and recovery guides**

- Upgrade Guide for Catalyst 9000 Switches
- Recover Catalyst 9000 Switches from Upgrade Failures
- Troubleshoot Rommon on Catalyst 9000 Series Switches
- Change Between Bundle/Install Mode on Catalyst 9000 Switches

**ISSU, xFSU, and SMU references**

- Troubleshoot ISSU Upgrade Failures on Catalyst 9000 Series Switches
- Cisco IOS XE 17 - High Availability
- In-Service Software Upgrade (ISSU)
- System Management Configuration Guide, Cisco IOS XE 26.x.x (Catalyst 9300 Switches) - Performing Extended Fast Software Upgrade
- Understand Extended Fast Software Upgrade on Catalyst 9300 Series Switches
- System Management Configuration Guide, Cisco IOS XE 26.x.x (Catalyst 9300 Switches) - Software Maintenance Upgrade

**Operations and diagnostics data collection**

- Troubleshoot Catalyst Center SWIM Feature
- Troubleshoot Unexpected Reloads on Catalyst 9000 Switches
- Troubleshoot Output from Catalyst 9000 Switches

# 6 Rolling Back Device Upgrades

**Topics:**


- Rollback conditions and recovery actions
- Perform device rollback and recovery

## Rollback conditions and recovery actions

Details the conditions under which rollback operations must be performed, including device and image states, and outlines the corresponding rollback and recovery actions based on various scenarios.

This topic enables users to identify rollback conditions and apply appropriate recovery actions based on device and image states to ensure system stability and successful rollback processes.

**Table 4: Rollback conditions and recovery actions**

| Current device or image state | Description | Recovery action |
|---|---|---|
| New image state is `I` | Image is added, but not active | Remove it only if the change is canceled |
| New image state is `U` | Image is active, but not committed | Abort before the rollback timer expires |
| New image state is `C` | Image is active and committed | Use a saved rollback point or reinstall an approved previous image |
| SMU is `U` or `D` | SMU change is not committed | Return to the last committed package state |
| System image is a `.bin` file | Switch is using direct image boot | Restore the verified previous `.bin` boot entry and reload the device |
| Device prompt is in `switch:` mode | IOS XE did not boot | Boot a valid `packages.conf` or known-good image |

## Perform device rollback and recovery

Provides detailed instructions for device rollback and recovery.

Use this procedure to perform rollback and recovery on a device when the new software does not pass validation or the device does not boot normally.

Replace the sample values given inside the angular brackets (<...>), and use the filesystem name, such as flash: or bootflash:, as shown by the device.

**Before you begin**

**Note**

- A rollback can reload the device and interrupt the traffic. Keep console or out-of-band access available prior to initiating the rollback procedure.
- Do not use commands such as, **emergency-install** or **format flash:**, or perform actions such as deleting active `.pkg` files or clearing the install state as standard recovery steps. These actions can erase the configuration, software, and rollback information.
- Software rollback does not automatically restore an older configuration. It may also not downgrade ROMMON, the bootloader, FPGA, or CPLD firmware.

- Save the configuration and the complete console or session logs.
- Confirm that a verified known-good image is available.
- Do not remove inactive files until the rollback decision is complete.
- Do not interrupt an install operation or power-cycle the switch.

**Procedure**

1. Check the current install state of the device.

   **Example**

   ```
   switch# show install summary
   switch# show install uncommitted
   switch# show install rollback
   switch# show boot
   switch# show version
   ```

2. Identify the recovery action based on the install state of the device. Refer, Rollback conditions and recovery actions on page 96.
3. Perform the recovery step based on the device or image condition.
   - Condition: Image is added, but not active.

     ```
     switch# show install summary
     Type  St   Filename/Version
     ---------------------------------
     IMG   C    <current-version>
     IMG   I    <staged-version>
     ```

     This output shows that the current image version is still active.

     To cancel the staged change, perform this step. This command can remove the images and packages needed for a later rollback. Hence, make sure to review the proposed list of files before giving confirmation to remove the files.

     ```
     switch# install remove inactive
     ```

```
<review the proposed deletion list>
Do you want to remove the above files? [y/n] y
...
SUCCESS: install_remove
```

- Condition: New version is active, but not committed.

  This is the preferred rollback window.

```
switch# show install summary
Type  St   Filename/Version
-------------------------------
IMG   U    <new-version>

Auto abort timer: active
Time before rollback: <remaining-time>
```

If validation fails, do not commit. Instead, abort the install operation. This can cause a device reload and management access might drop.

```
switch# install abort
This operation may require a reload of the system.
Do you want to proceed? [y/n] y
...
SUCCESS: install_abort
```

If the command is rejected, use the **install abort ?** command and follow the syntax shown on the device.

If the timer expires before the commit operation, the switch returns to the last committed state. Do not stop the timer when rollback protection is required. Stopping the timer does not commit the image.

Check the install state after rollback:

```
switch# show install summary
Type  St   Filename/Version
-------------------------------
IMG   C    <previous-version>

Auto abort timer: inactive
```

```
switch# show install uncommitted
No Uncommitted Packages
```

The rejected image might still be in inactive state. Keep it until validation and log collection processes are complete.

- Condition: New image version is already committed.

  After commit operation, the install abort option is no longer available.

  Check for a saved rollback point:

```
switch# show install rollback
ID    Label       Description
-----------------------------------
<id>  <label>     <previous-version>
```

```
switch# show install rollback id <id>
Rollback id - <id>
IMG   C    <previous-version>
```

Confirm the version, required files, and reload impact. You must not select a rollback ID by the number alone.

Perform the rollback operation to the respective rollback point or ID.

```
switch# install rollback to id <id>
install_rollback: START
...
SUCCESS: install_rollback
```

If no usable point exists, return to the verified previous image as a new software change:

```
switch# verify /sha512 flash:<previous-image.bin>
switch# install add file flash:<previous-image.bin>
switch# install activate auto-abort-timer <minutes>
<validate after the reload>
switch# install commit
```

Confirm that the return path is supported. An intermediate release or configuration change might be required.

- Condition: SMU change is uncommitted.

  For an uncommitted SMU change, perform the rollback operation to committed state.

```
switch# install rollback to committed
...
SUCCESS: install_rollback
```

To remove a committed SMU, follow its Cisco README file.

These are the common commands:

```
switch# install deactivate file flash:<smu-file>
switch# show install summary
SMU   D    flash:<smu-file>

switch# install remove file flash:<smu-file>
```

SMU deactivation might restart processes or reload the switch.

- Condition: Switch boots a `.bin` image.

  Use this only when the **show version** command output confirms that the switch is booting a `.bin` image and the product supports this method.

```
switch# verify /sha512 flash:<previous-image.bin>
switch# configure terminal
switch(config)# no boot system
switch(config)# boot system flash:<previous-image.bin>
switch(config)# end
switch# copy running-config startup-config
switch# show boot
switch# reload
```

Confirm that the image is present everywhere required before reloading the device.

- Condition: The device is in `switch:` prompt.

```
switch: dir flash:
switch: cat flash:packages.conf
switch: boot flash:packages.conf
```

Use the `packages.conf` file only when all the referenced `.pkg` files are present. Do not edit it manually.

If the package set is missing or invalid, boot a verified image:

```
switch: boot flash:<known-good-image.bin>
```

OR

```
switch: dir usbflash0:
switch: boot usbflash0:<known-good-image.bin>
```

For TFTP recovery:

```
switch: set IP_ADDRESS <switch-ip>
switch: set IP_SUBNET_MASK <subnet-mask>
switch: set DEFAULT_GATEWAY <gateway-ip>
switch: ping <tftp-server-ip>
switch: boot tftp://<tftp-server-ip>/<known-good-image.bin>
```

Direct `.bin` boot uses the bundle boot method. After IOS XE loads, copy the correct image to flash and restore the supported installation state.

**4.** Validate the rollback operation using these **show** commands.

**Example**

```
switch# show version
switch# show boot
switch# show install summary
switch# show install uncommitted
switch# show switch
switch# show redundancy states
switch# show interfaces status
switch# show etherchannel summary
switch# show spanning-tree summary totals
switch# show platform resources
switch# show environment all
switch# show logging
```

Confirm that

- the intended release is active and committed
- the rollback timer is inactive and no install operation is incomplete
- expected members, interfaces, protocols, and services are healthy

- CPU, memory, power, temperature, and fans are healthy, and
- no new crash, traceback, or repeated error is present on the device.

---