

Candidate Configuration

- Overview, on page 1
- Guidelines and Limitations, on page 3
- Perform Integrity Check for Candidate Config, on page 8
- Configuration Candidate session, on page 9

Overview

Candidate configuration is a subset of the running-config which checks whether the Candidate config exists in the running-config without any additions or modifications or deletions.

To check the integrity of the candidate config, use the following commands:

- `show diff running-config`
- `show diff startup-config`

For more information on the CLIs, refer to Perform Integrity Check for Candidate Config, on page 8.

Examples

No Difference Between Running and Candidate Config

```
switch# show diff running-config bootflash:base_running.cfg
switch#
```

Difference Between Running and Candidate

```
switch# show diff running-config bootflash:modified-running.cfg unified
--- running-config
+++ User-config
@@ -32,11 +32,11 @@

interface Ethernet1/1
   mtu 9100
   link debounce time 0
   beacon
-  ip address 2.2.2.2/24
+  ip address 1.1.1.1/24
   no shutdown

interface Ethernet1/2
```

```
interface Ethernet1/3
switch#
```

Difference Between Running and Partial Candidate

```
switch# show file bootflash:intf_vlan.cfg
interface Vlan101
  no shutdown
  no ip redirects
  ip address 1.1.2.1/24 secondary
  ip address 1.1.1.1/24
switch#
switch# show diff running-config bootflash:intf_vlan.cfg partial unified
--- running-config
+++ User-config
@@ -3897,10 +3883,14 @@
   mtu 9100
   ip access-group IPV4_EDGE in
   ip address 2.2.2.12/26 tag 54321

 interface Vlan101
+ no shutdown
+ no ip redirects
+ ip address 1.1.2.1/24 secondary
+ ip address 1.1.1.1/24

 interface Vlan102
   description Vlan102
   no shutdown
   mtu 9100
switch#
```

Partial Configuration Diff Merged

```
switch# show file po.cfg
interface port-channel500
description po-123
switch#
switch# sh run int po500

!Command: show running-config interface port-channel500
!Running configuration last done at: Fri Sep 29 12:27:28 2023
!Time: Fri Sep 29 12:30:24 2023

version 10.4(2) Bios:version 07.69

interface port-channel500
  ip address 192.0.2.0/24
  ipv6 address 2001:DB8:0:ABCD::1/48

switch#

switch# show diff running-config po.cfg partial merged unified
--- running-config
+++ User-config
@@ -124,10 +110,11 @@
interface port-channel100
 interface port-channel500
   ip address 192.0.2.0/24
   ipv6 address 2001:DB8:0:ABCD::1/48
+ description po-123
 interface port-channel4096
 interface Ethernet1/1
switch#
```

Guidelines and Limitations

- The line numbers that are displayed in the generated running config do not match with the candidate config as they are internally generated one.
- If there is any difference between the configuration of running and candidate, then it is displayed inline as output.
- If the whole block of configuration in the candidate file is a new addition, it will be appended at the end of the generated running config.
- When the candidate config has an SNMP or an AAA user CLI with clear-text password, the SNMP user is seen as a diff even when the user is already configured.
- EIGRP address family IPv4 configs are recommended to configure under the EIGRP address family hierarchy and not under the router mode hierarchy in the candidate file, before running a partial diff.
- If the target/candidate file has a default command (for example, `- log-neighbor-warnings;`) configured directly under the **router eigrp** mode and not one of its submodes, that is, **address-family ipv4 unicast** or **address-family ipv6 unicast** , then partial-diff shows + displayed in the output of the default command (for example, `+ log-neighbor-warnings`) in the diff.
- For noncase sensitive commands, if there is a letter case distinction between the commands in the running config and candidate-config files, then the output of **partial diff** displays both the commands due to the difference in letter case.
- Cleartext passwords are allowed in case of partial diff candidate CONFIG_FILE as the user database gets synced between SNMP and AAA (Security).
- Configuration profile, maintenance profile (mmode) and scheduler mode configurations are not supported.

Integrity Check Guidelines

1. To perform an integrity check on a full running configuration file, the user must supply a running configuration generated by an NX-OS system and should not use the **partial** keyword. The **partial** keyword treats the input candidate as a subset of the running configuration.

Release Specific Guidelines

1. Beginning with Cisco NX-OS Release 10.2(3)F, Candidate config integrity check option is introduced on all Cisco Nexus switches.
2. Beginning from Cisco NX-OS Release 10.4(3)F, you can also use polymorphic commands in candidate configuration to perform partial diff.

Partial Diff of Default Commands for Multicast Components Guidelines

The content of this section is applicable from Cisco NX-OS Release 10.4(3)F.

If the default commands of multicast components are present in the candidate CONFIG_FILE, they are seen in show diff as follows:

Multicast Component	Default Commands in show diff
PIM	ip access-list copp-system-p-acl-pim 10 permit pim any 224.0.0.0/24 20 permit udp any any eq pim-auto-rp ip access-list copp-system-p-acl-pim-mdt-join ip access-list copp-system-p-acl-pim-reg 10 permit pim any any
PIM6	ipv6 access-list copp-system-p-acl-pim6 10 permit pim any ff02::d/128 20 permit udp any any eq pim-auto-rp ipv6 access-list copp-system-p-acl-pim6-reg 10 permit pim any any
IGMP	ip access-list copp-system-p-acl-igmp 10 permit igmp any 224.0.0.0/3 class-map copp-system-p-class-normal-igmp
MLD	ipv6 access-list copp-system-p-acl-mld 10 permit icmp any any mld-query 20 permit icmp any any mld-report 30 permit icmp any any mld-reduction 40 permit icmp any any mldv2

show diff running-config *file_url* [unified] [partial] [merged] Command Guidelines

- When using the **unified** , **partial** , and **merged** option to review the differences for the following PBR commands, the diff outputs are as mentioned below:
  - **set ip next-hop**
  - **set ip default next-hop**
  - **set ip default vrf next-hop**
  - **set ipv6 next-hop**
  - **set ipv6 default next-hop**
  - **set ipv6 default vrf next-hop**

1. If the candidate next-hops are a subset of running next-hops (in the same order and sequence), and candidate additive flags are a subset of running flags, then the diff output is empty as shown in the following table:

Candidate Config	Running Config	Partial Unified Merged Diff Output
route-map rmap1 permit 10 set ip next-hop 1.1.1.1 2.2.2.2 load-share	route-map rmap1 permit 10 set ip next-hop 1.1.1.1 2.2.2.2 3.3.3.3 load-share force-order	<no-diff>

2. If the candidate next-hops are a subset of running next-hops (in the same order and sequence), and the candidate has some extra additive flags which are not present in running config, then the diff output appends any additional flags present in the candidate config to the running config, similar to command line behavior as shown in the following table:

Candidate Config	Running Config	Partial Unified Merged Diff Output
route-map rmap1 permit 10 set ip next-hop 1.1.1.1 2.2.2.2 load-share force-order	route-map rmap1 permit 10 set ip next-hop 1.1.1.1 2.2.2.2 3.3.3.3 load-share drop-on-fail	route-map rmap1 permit 10 - set ip next-hop 1.1.1.1 2.2.2.2 3.3.3.3 load-share drop-on-fail + set ip next-hop 1.1.1.1 2.2.2.2 3.3.3.3 load-share force-order drop-on-fail

3. If candidate next-hops are not a subset of running next-hops (in the same order and sequence), and the candidate and running record can have any additive flag, then the diff output indicates this with a '–' for the running config record and a '+' for the candidate config record.

   This distinction is important, particularly when using with PBR commands, where the sequence of next-hops is critical. Even if the next-hops IP addresses are identical, their order affects functionality.

   For example, '1.1.1.1 2.2.2.2' is different from '2.2.2.2 1.1.1.1'.



Important If there is an additive flag in the running config that you wish to retain after merging with the candidate config, you must explicitly include that flag in the candidate config. This ensures that the needed flags are preserved in the final, merged configuration.

Candidate Config	Running Config	Partial Unified Merged Diff Output
route-map rmap1 permit 10 set ip next-hop 1.1.1.1 2.2.2.2 load-share drop-on-fail	route-map rmap1 permit 10 set ip next-hop 2.2.2.2 1.1.1.1 load-share force-order	route-map rmap1 permit 10 - set ip next-hop 2.2.2.2 1.1.1.1 load-share force-order + set ip next-hop 1.1.1.1 2.2.2.2 load-share drop-on-fail

- When **Partial Unified** or **Partial Unified Merged** option is used, all the PBR commands are mutually exclusive and cannot coexist within the same parent route-map. Therefore, if a candidate configuration specifies multiple mutually exclusive PBR commands under a single route-map, only the last command variant will be shown in the partial diff output.

  Example-1: In this example, the candidate configuration includes multiple PBR commands under a single route-map **rmap1** :

```
route-map rmap1 permit 10
  set ip next-hop 1.1.1.1 2.2.2.2
  set ipv6 next-hop 3::3
  set ip next-hop verify-availability 4.4.4.4
  set ip next-hop verify-availability 5.5.5.5
  set ip vrf green next-hop 6.6.6.6
  set ip vrf blue next-hop 7.7.7.7 8.8.8.8
```

  Before the generation of the partial-diff output, the above candidate configuration is automatically converted to the following:

```
route-map rmap1 permit 10
  set ip vrf green next-hop 6.6.6.6
  set ip vrf blue next-hop 7.7.7.7 8.8.8.8
```

Example-2: In this example, the candidate configuration includes multiple 'set ip next-hop verify-availability' commands with different track IDs specified for the route-map **rmap2** . Since track IDs cannot be modified for the same next-hop, these commands are mutually exclusive:

```
route-map rmap2 permit 10
  set ip next-hop verify-availability 1.1.1.1 track 1
  set ip next-hop verify-availability 2.2.2.2 track 20
  set ip next-hop verify-availability 2.2.2.2 track 30
  set ip next-hop verify-availability 2.2.2.2 track 40
  set ip next-hop verify-availability 3.3.3.3 track 3
```

Before generating the partial-diff output, the system will automatically consolidate these commands by retaining only the last **set ip next-hop verify-availability** command for each next-hop IP address as shown below:

```
                                route-map rmap2 permit 10
  set ip next-hop verify-availability 1.1.1.1 track 1
                                set ip next-hop verify-availability 2.2.2.2 track 40
  set ip next-hop verify-availability 3.3.3.3 track 3
```

- When the **Partial Unified Merged** option is used, to review the differences for the **verify-availability** command variants, the track ID for a given next-hop is not modifiable.

  Therefore, if the candidate and running configurations contain the same next-hop but have different track IDs under the same parent route-map, the candidate record cannot simply be merged with the running record, as per command line behavior. Therefore, to apply the candidate record with different track ID for the same next-hop, the corresponding running config record must be removed first ('–' for the running configuration record in the diff) and then when the candidate record is merged, it will be appended at the end of the last record under the same parent route-map ('+' for the candidate config record).

  The following table shows the sample candidate and running configuration with the 'Partial Unified Merged' output for different use cases as mentioned below:

  1. If the track ID is different for the same next-hop under candidate and running config, then the diff output is as shown in the following table:

Candidate Config	Running Config	Partial Unified Merged Diff Output

route-map rmap1 permit 10
  set ip next-hop
verify-availability 1.1.1.1
 track 1
  set ip next-hop
verify-availability 2.2.2.2
 track 20
  set ip next-hop
verify-availability 3.3.3.3
 track 3 load-share
	route-map rmap1 permit 10
	  set ip next-hop
	verify-availability 1.1.1.1
	 track 1
	  set ip next-hop
	verify-availability 2.2.2.2
	 track 2
	  set ip next-hop
	verify-availability 3.3.3.3
	 track 3 load-share
		route-map test permit 10
		  set ip next-hop
		verify-availability 1.1.1.1
		track 1
		 - set ip next-hop
		verify-availability 2.2.2.2
		track 2
		  set ip next-hop
		verify-availability 3.3.3.3
		track 3
		 + set ip next-hop
		verify-availability 2.2.2.2
		track 20 load-share

  2. If track ID is not present under candidate config but present in running config for the same next-hop, then the diff output is empty as shown in the following table:

Candidate Config	Running Config	Partial Unified Merged Diff Output

route-map rmap1 permit 10
  set ip next-hop
verify-availability 1.1.1.1
 track 1
  set ip next-hop
verify-availability 2.2.2.2
 	
  set ip next-hop
verify-availability 3.3.3.3
 track 3
	route-map rmap1 permit 10
	  set ip next-hop
	verify-availability 1.1.1.1
	 track 1
	  set ip next-hop
	verify-availability 2.2.2.2
	 track 2
	  set ip next-hop
	verify-availability 3.3.3.3
	 track 3
		no-diff

3. If track ID is not present under running config but present in candidate config for the same next-hop, then the diff output is as shown in the following table:

Candidate Config	Running Config	Partial Unified Merged Diff Output

route-map rmap1 permit 10
  set ip next-hop
verify-availability 1.1.1.1
 track 1
  set ip next-hop
verify-availability 2.2.2.2
 track 20
  set ip next-hop
verify-availability 3.3.3.3
 track 3
	route-map rmap1 permit 10
	  set ip next-hop
	verify-availability 1.1.1.1
	 track 1
	  set ip next-hop
	verify-availability 2.2.2.2
	
	  set ip next-hop
	verify-availability 3.3.3.3
	 track 3
		route-map rmap1 permit 10
		  set ip next-hop
		verify-availability 1.1.1.1
		track 1
		 - set ip next-hop
		verify-availability 2.2.2.2
		  set ip next-hop
		verify-availability 3.3.3.3
		track 3
		 + set ip next-hop
		verify-availability 2.2.2.2
		track 20

Partial Diff of RPM Commands Guidelines

The content of this section is applicable from Cisco NX-OS Release 10.4(3)F.

When using the unified, partial, and merged option to review the differences for the following RPM commands, the diff outputs are as follows:

- In the candidate configuration, the RPM commands will undergo syntactic validation as reflected in the diff output. However, semantic validation will not be performed in the diff output. It is the user's responsibility to ensure that the commands in the candidate configuration are semantically accurate.

  If the command in the Candidate-config is semantically incorrect, the diff may incorrectly indicate that the command is executable, but in actual it may not.

- Ensure that you provide the sequence number mandatorily for the following commands in the Candidate-config file:
  - **ip prefix-list list-name seq** *seq* { **deny** | **permit** } *prefix*
  - **ipv6 prefix-list list-name seq** *seq* { **deny** | **permit** } *prefix*
  - **mac-list list-name seq** *seq* { **deny** | **permit** } *prefix*
  - **ip community-list** { **standard** | **expanded** } *list-name* seq *seq* { **deny** | **permit** } *expression*
  - **ip extcommunity-list** { **standard** | **expanded** } *list-name* seq *seq* { **deny** | **permit** } *expression*
  - **ip large-community-list** { **standard** | **expanded** } *list-name* seq *seq* { **deny** | **permit** } *expression*

- **ip-as-path access-list** *list-name* seq *seq* { **deny** | **permit** } *expression*

- When the following commands include an expression string that has spaces enclosed in quotes within the Candidate-config, there will be no differences displayed in the diff output:
  - **ip community-list expanded** *list-name* seq *seq* { **deny** | **permit** } *expression*
  - **ip extcommunity-list expanded** *list-name* seq *seq* { **deny** | **permit** } *expression*
  - **ip large-community-list expanded** *list-name* seq *seq* { **deny** | **permit** } *expression*
  - **ip-as-path access-list** *list-name* seq *seq* { **deny** | **permit** } *expression*

Candidate Config	Running Config	Partial Unified [Merged] Diff Output
`ip community-list expanded list_abc seq 10 permit "1:1 "`	`ip community-list expanded list_abc seq 10 permit "1:1"`	`no-diff`
`ip extcommunity-list expanded list_abc seq 10 permit "1:1  "`	`ip extcommunity-list expanded list_abc seq 10 permit "1:1"`	`no-diff`
`ip large-community-list expanded list_abc seq 10 permit "1:1:1   "`	`ip large-community-list expanded list_abc seq 10 permit "1:1:1"`	`no-diff`
`ip as-path access-list list_abc seq 10 permit "1 "`	`ip as-path access-list list_abc seq 10 permit "1"`	`no-diff`

Perform Integrity Check for Candidate Config

To perform the integrity check, use the following commands:

Before you begin

Note Before performing the integrity check, ensure that the running config and the candidate config belong to the same image version.

Procedure

Step 1 Display the differences between the running and user given candidate config using the command **show diff running-config** *file_url* [**unified**] [**merged**]

Example:

```
switch# show diff running-config bootflash:candidate.cfg partial unified
```

- *file_url* : File path to compare with.

- **unified** : Displays the difference between running and user configuration in unified format.
- **merged** : Enter **merged** only if sub-commands need to be merged instead of replace.

Step 2 Display the differences between the startup and user given candidate config using the command **show diff startup-config** *file_url* [**unified**]

Example:

```
switch# show diff startup-config bootflash:candidate.cfg unified
```

- *file_url* : File path to compare with.
- **unified** : Displays the difference between startup and user configuration in unified format.

Configuration Candidate session

A Configuration Candidate session is a configuration management feature that allows you to stage, review, and validate configuration changes before committing them to the running configuration on a Cisco Nexus switch.

- Staged configuration changes are isolated from the active running configuration until you explicitly commit them.
- You can compare staged changes to the current running configuration to review differences and validate intended updates.
- Candidate session supports committing all changes at once, discarding staged changes, and rolling back to previous commit points.
- Candidate session can be initiated for both manual CLI-based changes and automated workflows using supported programming interfaces.
- Sessions are not persistent across switch reloads; any uncommitted changes are lost if the switch is restarted.
- Only a limited number of Candidate sessions can exist at one time, and only one user can access a session at a time.

Candidate session Use Cases and Administrator Value

- Prepare and review configuration changes in advance, minimizing the risk of unintended disruptions.
- Validate changes by viewing differences before applying them, which helps identify configuration errors or conflicts.
- Apply all intended changes at once, or abort the session to discard them without affecting the active configuration.
- Integrate with automation and programmatic interfaces for batch configuration workflows.

Actors in the Process

- Network administrator: Initiates and manages the configuration session.
- Switch CLI infrastructure: Provides the commands and session environment for staging, reviewing, and committing changes.
- Data Management Engine (DME): Maintains the candidate configuration database for staging changes.

In summary, Candidate session allows you to isolate changes from the active configuration, review differences, validate updates, and then commit or discard them as a single unit.

Guidelines and Limitations for Configuration Candidate Session

Beginning with NX-OS Release 10.6(3)F, candidate session (Configuration Staging) is supported on Cisco Nexus 9000 switches. Candidate sessions allow you to stage and review configuration changes before committing them to the running configuration.

- Creation of a new configuration candidate session is blocked if the current running configuration contains unsupported commands. The error message displayed when a command is not supported is as follows:

  ```
  switch# configure candidate
  Error: Current running-config has commands that are not supported by the candidate
  session.
  Please check 'show candidate-config last-unsupported' and remove the unsupported
  configurations from the running-config.
  switch#
  ```

  To resolve this issue, run the following command to identify unsupported configurations:

  ```
  switch# show candidate-config last-unsupported
  ```

  Then remove the unsupported commands from the running configuration before creating a new candidate session.

- Candidate session is removed if the device reloads or the supervisor switches.
- Only one candidate session can be active.
- Only one user can access and modify each candidate session at a time.
- The system maintains a history of up to 50 commit entries; older entries are overwritten as new commits are made.
- A candidate session supports a maximum configuration size of 35,000 lines.
- Interface breakout and the associated configuration of the newly created interfaces cannot be performed in the same candidate session. Use two separate candidate sessions: first to perform the interface breakout, and second to apply the required interface configurations
- A candidate session name can contain up to 64 alphanumeric characters, hyphens (-), and underscores (_). Spaces and other special characters are not supported. Names cannot start with special characters

Configuring a Candidate Session

This task describes how to start a configuration session, stage configuration changes, review the changes, and commit or discard them.

Use candidate session to safely stage multiple configuration changes before committing them to the device.

- You can review your staged changes and compare them with the running configuration before applying them.
- Candidate session helps prevent accidental changes by allowing you to review and validate before commit.

Before you begin

- Device running NX-OS Release 10.6(3)F or later.
- RBAC permissions for candidate session creation and commit.
- Only one candidate session can be active on the device.

Procedure

Step 1 Start a configuration session and copy configuration from file to the candidate session.

Use the following command to start a candidate session:

```
switch#configure candidate [name <session-name>]
```

You can provide a session name, but it is optional. If you do not specify one, the system automatically generates a name in the format session-x.

Use the following command to copy configuration from file to candidate session:

```
switch(config-c-<session-name>)# import configuration <uri0>
```

Step 2 Stage configuration changes, review and compare them in the session.

Use the following command to stage configuration changes in the session

```
switch(config-c-<session-name>)# interface Ethernet1/1
```

The session accepts supported configuration commands. Unsupported commands are rejected with an error.

Use the following command to review staged changes and compare them with the running configuration:

```
switch(config-c-<session-name>)# show candidate-config diff
```

Use the diff command to see what changes will be made when the session is committed.

Step 3 Use any of the following commands to commit the staged changes.

- `switch(config-c-<session-name>)# commit [timeout <seconds>]`
- `switch# configure candidate name <session-name> commit`

Committed changes are applied to the running configuration.

Step 4 To add a description to the commit, use this command.

```
switch(config-c-<session-name>)# commit description this is a test
```

The description can contain up to 128 characters. You can use alphanumeric characters, spaces, underscores (_), hyphens (-), and periods (.). Enter a multiword description without quotation marks. Quotation marks cause an invalid command error.

Step 5 Use any of the following commands to discard the candidate session without applying the changes and delete it.

- `switch(config-c-<session-name>)# abort`
- `switch# configure candidate name <session-name> abort`

Step 6 Confirm or abort the commit before a timeout.

Use the following command to confirm or abort the commit before a timeout:

```
switch# configure candidate name <session-name> commit [confirm] | abort
```

Note
If you choose to abort, all staged changes are discarded and the session is deleted.

Step 7 Reset the session configuration to the default configuration.

Use the following command to reset the session configuration to the default configuration:

```
switch(config-c-<session-name>)# factory-reset candidate
```

Note
If you are importing the complete running configuration into the session, factory-reset the existing candidate session before initiating the import procedure

Step 8 Clean up the old Commit IDs from the system.

Use the following command to clean up the old Commit IDs from the system:

```
switch# clear configuration commits {diskspace <size> | oldest <num>}
```

Step 9 Roll back to a previous commit using its ID.

Use the following command to roll back to a previous commit using its ID:

```
switch#  rollback configuration commit <id>
```

This step is optional and can be used if you need to revert changes.

Step 10 Delete the candidate session

Use the following command to delete the candidate session:

```
switch#  no configure candidate name  <session-name>
```

Configuration changes are staged, reviewed, and either applied to or discarded from the device.

After committing or aborting the session, verify the device configuration as needed.

Verifying a Candidate Session

Verification ensures that staged changes are correct before you commit them and that committed changes are present in the running configuration.

Confirm that the candidate session contains the intended staged changes and that committed changes have taken effect.

Perform verification after staging configuration changes and again after committing them to the device.

Procedure

Step 1 Display the staged configuration for the current session using the `switch(config-c-<session-name>)#` **`show candidate-config`** command.

This command shows all commands that are currently staged in the configuration session

Step 2 Compare staged changes with the running configuration using the `switch(config-c-<session-name>)#` **`show candidate-config diff`** command.

This command displays the differences between the staged configuration and the running configuration

Step 3 Verify committed changes in the running configuration by using the `switch#` **`show running-config`** command.

Use this step after you commit the session to ensure the intended changes are present.

Step 4 View commit history for candidate session using the `switch#` **`show configuration commit list`** command.

This shows a list of commit IDs that you can use for audit or rollback.

Step 5 View commit information for a session name by using the `switch#` **`show configuration commit`** `{` **`session`** `<name>` `|` **`changes session`** `<name>}` command.

Step 6 View execution logs of last rollback by using the `switch#` **`show configuration rollback log`** command.

Step 7 View the commit ID info by using the `switch#` **`show configuration commit`** `{[`**`changes`**`]` `<commit-id>|` `list}` command.

Step 8 View the pending and completed candidate session by using the `switch#` **`show candidate summary`** `[`**`pending`** `|` **`committed`**`]` command.

Step 9 View the configuration of given candidate name by using the `switch#` **`show candidate-config name`** `<session-name>` command.

The candidate session is verified. You can confirm that staged changes match the intended configuration and that committed changes are present in the running configuration.