



Optionality in Cisco NX-OS Software

This chapter describes optionality in Cisco NX-OS software.

- [Optionality in Cisco NX-OS Software, on page 1](#)
- [Using Modular Packages, on page 2](#)
- [Booting the NX-OS Image in Base or Full Mode, on page 3](#)
- [Information About RPMs, on page 4](#)
- [Information About DNF Commands, on page 18](#)
- [Configuring an FTP server and Setting up a Local FTP YUM Repository, on page 32](#)
- [Creating User Roles for Install Operation, on page 36](#)
- [Compacting Cisco NX-OS Software Images, on page 36](#)

Optionality in Cisco NX-OS Software

Beginning with Cisco NXOS Release 9.2(1), Cisco NX-OS software image supports modular package management. Cisco NX-OS software now provides flexibility to add, remove, and upgrade the features selectively without changing the base NX-OS software.

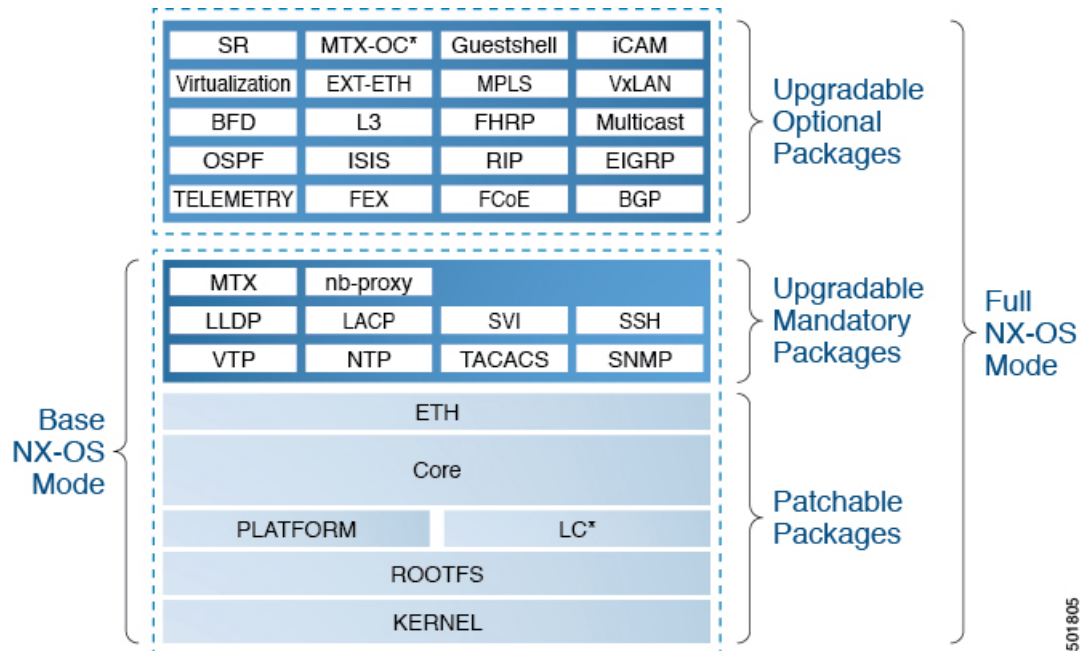
The advantages for using modular Cisco NX-OS software are:

- Lean NX-OS software
- Asynchronous delivery of the features and the fixes: Quick fixes are provided that are independent of the releases, including new features.
- Reduced footprint of binaries and libraries at run time

Cisco NX-OS software is provisioned to boot the NX-OS software in two modes as described in the following illustration:

- Base NX-OS mode
- Full NX-OS mode

Figure 1: Optionality in Cisco NX-OS Software



- Base NX-OS mode contains:
 - Upgradable mandatory packages
 - Patchable packages
- Full NX-OS mode contains:
 - Upgradable optional packages
 - Upgradable mandatory packages
 - Patchable packages



Note The default mode is full NX-OS mode.

In base NX-OS mode, basic Layer 2 and Layer 3 features are available. All dynamic routing features (for example, BGP, OSPF, EIGRP, RIP, and ISIS) and other optional feature RPMs are not available by default. You have to install the optional feature RPMs on top of the base image.

In full NX-OS mode, all feature RPMs are installed during boot time when Ethernet plugin is activated by the plugin manager. There is no change in the user behavior as compared to the previous releases.

Using Modular Packages

The Cisco NX-OS software image is traditionally constructed with the packaging that forms a Cisco Linux distribution. It makes upgrading certain packages difficult as each package is large in size.

This section describes a new package management for the Cisco NX-OS software image. Beginning with Cisco NX-OS Release 9.2(1), some NXOS features are considered as optional, for example, BGP, OSPF, VXLAN, MPLS, Segment Routing.

Each modular package has the following important characteristics:

- Upgrade functionality: The modular packages can be independently upgraded. The modular packages should be used from the same release as performing upgrades on these packages across multiple releases is not supported.
- Optionality: The modular packages are optional, for example, these packages can be removed or uninstalled at run time. The removal of the modular packages does not affect bringing-up the system and it does not affect any other functionality of the switches.



Note All APIs exported by the modular package should be used only after the installation of the feature.

RPM and DNF

RPM (Red Hat Package Manager) is the package management system used for packaging in the Linux Standard Base (LSB). The RPM command options are grouped into three subgroups for:

- Querying and verifying packages
- Installing, upgrading, and removing packages
- Performing miscellaneous functions

rpm is the command name for the main command that is used with RPM, whereas **.rpm** is the extension that is used for the RPM files.

Dandified YUM (Yellowdog Updater, Modified) or DNF is an open source command-line tool for RPM based Linux systems. It allows users and system administrators to easily install, update, remove, or search software packages on the systems. DNF adds the automatic updates and the package management, including dependency management, to the RPM systems. In addition to understanding the installed packages on a system, DNF works with the repositories that are collections of the packages and they are typically accessible over a network connection.

Booting the NX-OS Image in Base or Full Mode

You can now boot the NX-OS image in base or full mode. The full boot mode installs the complete NX-OS software which is similar to the software of the previous releases. This is the default boot mode. The base boot mode has no optional RPMs installed.

To use the command line option, see the following steps:

- Use the **install reset nxos base** option to install the NX-OS image in the base boot mode using the VSH prompt. After reload, the switch is in the base mode with no optional packages installed.
- Use the **install reset nxos full** option to install the NX-OS image in the full boot mode using the VSH prompt. After reload, the switch is in the full mode with the optional packages automatically installed.

For more information, see Using Install CLIs for Feature RPM Operation section.

Information About RPMs

RPMs can be upgraded or downgraded to a new software version using NXOS install commands or by using DNF commands. An upgradable RPM can be optional or mandatory.

See the following sections for more information about optional and mandatory RPMs.

Format of the RPM

The general format of a RPM is <name>-<version>-<release>.<arch>.rpm. The same format is followed for NXOS feature RPMs.

- Name: package name, for example, BGP
- Version in <x.y.x.b> format: <major.minor.patch.build_number>, for example, 2.0.1.0
- Release: The branch from which the RPM is created, for example, 9.2.1
- Arch: The architecture type of the RPM, for example, lib32_n9000

See the following table for more information on the naming convention, for example, fex-2.0.0.0-9.2.1.lib32_n9000.rpm:

Table 1: RPM Naming Convention

RPM Naming Convention Example: fex-2.0.0.0-9.2.1.lib32_n9000.rpm	Description
fex	Indicates the name of the component.
2	Indicates that the RPM is not backward compatible. Configuration loss takes place during an upgrade.
0	Indicates the incremental API changes/CLI changes/Schema changes with backward compatibility. It is applicable to the new features on top of the existing capabilities. No configuration is lost during an upgrade.
0	Indicates a bug fix without any functionality change. No configuration is lost during an upgrade.
0	This number tracks how many times the component has changed during the development cycle of a release. This value will be 0 for all the release images.
9.2.1	Indicates the release number or the distribution version for the RPM. It aligns to the NVR format. Since the feature RPM is only applicable to a NXOS release, this field has NXOS release version number present.
lib32_n9000	Indicates the architecture type of the RPM.

Optional RPMs and Their Associated Features

The optional RPMs are the RPMs that can be installed to enable the features without affecting the native NXOS behavior or they can be removed using the **install deactivate** command from the switch.

Optional RPMs, for example, EIGRP are not a part of the base software. They can be added, upgraded, and removed as required using either **dnf** or **install** CLI commands from the switch.

See the following list of the optional RPMs and their associated features:

Table 2: List of Optional RPMs and Their Associated Features

Package Name	Associated Features
BGP	feature bgp
BFD	feature bfd
Container-tracker	feature container-tracker
EIGRP	feature eigrp
Ext-Eth	<ul style="list-style-type: none"> • feature openflow • feature evb • feature imp • feature netflow • feature sla_sender • feature sla_responder • feature sla twamp-server • feature sflow
EXT_ETH_LOWMEM	<ul style="list-style-type: none"> • feature evb • feature netflow
FCoE	<ul style="list-style-type: none"> • feature-set fcoe • feature-set fcoe-npv
FEX	feature-set fex
FHRP	<ul style="list-style-type: none"> • feature hsrp • feature vrrpv3
HW TELEMETRY	feature hw telemetry
iCAM	feature icam
ISIS	feature isis

Package Name	Associated Features
MPLS	<ul style="list-style-type: none"> • feature mpls segment-routing • feature mpls evpn
Multicast	<ul style="list-style-type: none"> • feature pim • feature pim6 • feature msdp • feature ngmvpn
NIA	NA
NXSDK	NA
OSPF	<ul style="list-style-type: none"> • feature ospf • feature ospfv3
RIP	feature rip
SDAA	NA
Services	feature catena
SR	feature mpls segment-routing traffic-engineering
TELEMETRY	feature telemetry
Virtualization	NA
VM Tracker	feature vmtracker
VXLAN	<ul style="list-style-type: none"> • feature nv overlay • feature fabric forwarding

Guidelines for NX-OS Feature RPM Installation

See the following NX-OS system RPM repositories that are present in the Cisco NX-OS Series switches for the RPM management.



Note Avoid manually copying the RPMs to system repositories. Instead use the install or DNF commands.

Table 3: RPM Repositories That Are Present in the Switches

Repository Name	Repository Path	Description
groups-repo	/rpms	Part of the bundled NX-OS image. It is used to keep all the RPMs that are bundled as part of the NX-OS image. All RPMs based in this repository are known as base RPMs.
localdb	/bootflash/.rpmstore/patching/localrepo	Used for RPM persistency. When a user adds a NX-OS feature RPM as part of install add command, the RPM is copied to this location and it is persisted during the reloads. User has the responsibility to clean the repository. To add a RPM to this repository, use install add command. To remove a RPM from this repository, use install remove command. DNF commands can be used to populate the repository too. The maximum space for the repository is 200Mb along with the patching repository for Cisco Nexus 9000 Series switches except Cisco Nexus 3000 Series switches. For Cisco Nexus 3000 Series switches, the maximum space for the repository is 20 Mb only.
patching	/bootflash/.rpmstore/patching/patchrepo	Used for RPM persistency. When a user adds a NX-OS patch RPM to the switch, the patch RPM is copied to this repository.
thirdparty	/bootflash/.rpmstore/thirdparty	Used for RPM persistency when a user adds a third party RPM.

The **groups-repo** and **localdb** repositories hold the NX-OS feature RPMs that should be installed during the system boot or during activation. DNF commands or **install** command can be used for the installation or the removal of these RPMs.

The following rules are applied to the feature RPM installation procedure during boot or install time:

- Only RPMs with the same NX-OS release number should be selected for the installation.
- Base RPMs cannot be added to the **localdb** repository.

Guidelines for Third-Party RPM Installation

Currently, any third-party package that is not provided by Cisco is allowed to be installed on the device, even when it is not signed by Cisco.

Starting with release 10.1(x) any third-party package that is not signed by Cisco is not allowed to be installed on the device. However, if you wish to bypass this and install the software, you can configure the device to enable the third-party software installation. The configuration will persist as a normal configuration and can be verified by using the **running-config** command. Following this configuration, you can install any third-party software with the known risks.

Using Install CLIs for Feature or Third-Party RPM Operation

See the following reference table for using install CLIs for the feature RPM operations:

Table 4: Reference for Install CLIs for the Feature RPM Operations

CLI	Description
install reset	<p>This operation removes all the patches, persisted configurations, upgraded packages, third-party installed packages, unsaved configurations, and reloads the switch's previous mode (Full/Base) with the default packages.</p> <p>The install reset command also performs write erase operation. The following message is displayed at the prompt:</p> <pre>switch(config)# install reset</pre> <hr/> <p>WARNING!!This operation will remove all patches, upgraded packages, persisted etc configs, third party packages installed, startup configuration(write erase) and reload the switch with default packages.</p> <hr/> <p>Do you want to proceed with reset operation? (y/n)? [n]</p>
install reset nxos base	<p>This operation installs NXOS in base mode by removing all patches, upgraded packages, persisted etc configurations, third-party packages installed, startup configuration (write erase), and reloads the switch with the default packages.</p>
install reset nxos full	<p>This operation installs NXOS with full mode by removing all patches, upgraded packages, persisted etc configs, third-party packages installed, startup configuration (write erase), and reloads the switch with the default packages (with mandatory and optional RPMs).</p>

CLI	Description
install add <>	Adds an RPM file to the respective repository and updates the repository (patch/feature/third-party).
install activate < <i>rpm name</i> >	Installs an RPM that is present in the repository.
install commit < <i>rpm name</i> >	Used for the patch RPMs. Makes the patch persist during the reload.
install deactivate < <i>rpm name</i> >	Un-installs an RPM. Beginning with Cisco NX-OS Release 10.1(1), when you use this command to deactivate RPMs, the options to either downgrade to the base version of RPM or to uninstall RPM appear. You can select the option that you desire and the operation will proceed.
install remove < <i>rpm name</i> >	Removes an RPM file from the repository and updates the repository.
sh install active	Displays the list of the installed RPMs in the system apart from base rootfs RPMs. (features/patch/third-party).
sh install inactive	Displays the list of the RPMs that are present in the repository but they are not installed.
sh install packages	Lists all the RPMs that are installed including rootfs RPMs.

CLI	Description
<p>[no] system software allow third-party</p>	<p>Beginning with Cisco NX-OS Release 10.1(1) the third-party RPM installations are not allowed to be installed on the device by default. This command bypasses this restriction and configures the device to enable the third-party software installation.</p> <p>The following command shows the error message when you activate third-party RPM without applying the third-party configuration:</p> <pre>switch(config)# install activate pbwMonitor-1.0-1.5.0.x86_64.rpm</pre> <p>Install operation 193 failed because package is not signed by Cisco.Enable TPS installation using 'system software allow third-party' CLI at Tue Nov 17 04:23:10 2020</p> <p>The following command shows activating third-party RPM installations after applying the configuration:</p> <pre>switch(config)# system software allow third-party switch(config)# 2020 Nov 17 04:25:41 switch %% VDC-1 %% %USER-2-SYSTEM_MSG: <<%PATCH-INSTALLER-2-TPS_FEATURE_ENABLED>> User has enabled TPS installation - patch_installer</pre> <pre>switch(config)# install activate pbwMonitor-1.0-1.5.0.x86_64.rpm [#####] 100% Install operation 194 completed successfully at Tue Nov 17 04:25:58 2020</pre> <p>The following command shows disabling the third-party configuration:</p> <pre>switch(config)# no system software allow third-party switch(config)# 2020 Nov 17 04:27:17 switch %% VDC-1 %% %USER-2-SYSTEM_MSG: <<%PATCH-INSTALLER-2-TPS_FEATURE_DISABLED>> User has disabled TPS installation - patch_installer</pre>



Note If you are using ISSU or upgrading to Cisco NX-OS Release 10.1.1 release from an earlier version, you must manually apply the third-party configuration within the first 30 minutes after the upgrade to ensure the third-party RPMs get installed.

Using Install CLIs for Digital Signature Support

Use the following CLI commands to install CLIs for digital signature support:

SUMMARY STEPS

1. switch#install add bootflash:<keyfile> gpg-key
2. switch#install verify package <package-name>
3. OR switch#install verify bootflash:<RPM file>

DETAILED STEPS

	Command or Action	Purpose
Step 1	switch#install add bootflash:<keyfile> gpg-key Example: <pre>install add bootflash:RPM-GPG-KEY-puppetlabs gpg-key [#####] 100% Install operation 304 completed successfully at Thu Jun 19 16:40:28 2018</pre>	Cisco release RPMs are signed with Cisco GPG (GNU Privacy Guard) key. The public GPG key is present at /etc/pki/rpm-gpg/arm-Nexus9k-rel.gpg . To add other public keys from different sources, use the steps in this section.
Step 2	switch#install verify package <package-name>	Verifies the package.
Step 3	OR switch#install verify bootflash:<RPM file> Example: <pre>switch# install verify bootflash:vxlan-2.0.0.0-9.2.1.lib32_n9000.rpm RSA signed switch#</pre>	Use step 2 or 3 to verify whether the RPM file is a signed or non-signed file.

Querying All Installed RPMs

Complete the following step to query all the installed RPMs:

SUMMARY STEPS

1. show install packages

DETAILED STEPS

	Command or Action	Purpose
Step 1	show install packages Example: <pre>switch# show install packages Boot Image: NXOS Image: bootflash:/nxos.9.2.1.bin ----- Installed Packages attr.x86_64 2.4.47-r0.0 installed Unsigned</pre>	Queries all the installed RPMs.

Command or Action	Purpose
<pre>aufs-util.x86_64 3.14+git0+b59a2167a1-r0.0 installed Unsigned base-files.n9000 3.0.14-r89.0 installed Unsigned base-passwd.lib32_x86 3.5.29-r0.1.0 installed Unsigned bash.lib32_x86 4.3.30-r0.0 installed Unsigned bfd.lib32_n9000 2.0.0.0-9.2.1 installed Signed bgp.lib32_n9000 2.0.0.0-9.2.1 installed Signed binutils.x86_64 2.25.1-r0.0 installed Unsigned bridge-utils.x86_64 1.5-r0.0 installed Unsigned busybox.x86_64 1.23.2-r0.0 installed Unsigned busybox-udhcpc.x86_64 1.23.2-r0.0 installed Unsigned bzip2.x86_64 1.0.6-r5.0 installed Unsigned ca-certificates.all 20150426-r0.0 installed Unsigned cgroup-lite.x86_64 1.1-r0.0 installed Unsigned chkconfig.x86_64 1.3.58-r7.0 installed Unsigned container-tracker.lib32_n9000 2.0.0.0-9.2.1 installed Signed containerd-docker.x86_64 0.2.3+gitaa8187dbd3b7ad67d8e5e3a15115d3eef43a7ed1-r0.0 installed Unsigned core.lib32_n9000 2.0.0.0-9.2.1 installed Signed coreutils.lib32_x86 8.24-r0.0 installed Unsigned cpio.x86_64 2.12-r0.0 installed Unsigned cracklib.lib32_x86 2.9.5-r0.0 installed Unsigned cracklib.x86_64 2.9.5-r0.0 installed Unsigned createrepo.x86_64 0.4.11-r9.0 installed Unsigned cronie.x86_64 1.5.0-r0.0 installed Unsigned curl.lib32_x86 7.60.0-r0.0 installed Unsigned db.x86_64 6.0.30-r0.0 installed Unsigned dbus-1.lib32_x86 1.8.20-r0.0 installed Unsigned dhcp-client.x86_64 4.3.2-r0.0 installed Unsigned dhcp-server.x86_64 4.3.2-r0.0 installed Unsigned switch#</pre>	

Installing the RPMs Using One Step Procedure

The CLIs for both install and upgrade RPMs are the same. See the following step to install the RPMs using one step procedure:

Procedure

Command or Action	Purpose
<p>Step 1</p> <pre>install add <rpm> activate</pre> <p>Example:</p> <pre>switch# install add bootflash:chef.rpm activate Adding the patch (/chef.rpm) [#####] 100% Install operation 868 completed successfully at Tue May 8 11:20:10 2018 Activating the patch (/chef.rpm)</pre>	Installs and activates the RPM.

	Command or Action	Purpose
	<pre>[#####] 100% Install operation 869 completed successfully at Tue May 8 11:20:20 2018</pre>	

Example

```
switch# show install active
Boot Image:
    NXOS Image: bootflash:/nxos.9.2.1.bin

Active Packages:
bgp-2.0.1.0-9.2.1.lib32_n9000
chef-12.0.0alpha.2+20150319234423.git.1608.b6eb10f-1.el5.x86_64

Active Base Packages:
lACP-2.0.0.0-9.2.1.lib32_n9000
lldp-2.0.0.0-9.2.1.lib32_n9000
mtx-device-2.0.0.0-9.2.1.lib32_n9000
mtx-grpc-agent-2.0.0.0-9.2.1.lib32_n9000
mtx-infra-2.0.0.0-9.2.1.lib32_n9000
mtx-netconf-agent-2.0.0.0-9.2.1.lib32_n9000
mtx-restconf-agent-2.0.0.0-9.2.1.lib32_n9000
mtx-telemetry-2.0.0.0-9.2.1.lib32_n9000
ntp-2.0.0.0-9.2.1.lib32_n9000
nxos-ssh-2.0.0.0-9.2.1.lib32_n9000
snmp-2.0.0.0-9.2.1.lib32_n9000
svi-2.0.0.0-9.2.1.lib32_n9000
tacacs-2.0.0.0-9.2.1.lib32_n9000
vtp-2.0.0.0-9.2.1.lib32_n9000
switch(config)#
```

Installing the RPMs Using Two Steps Procedure

The CLIs for both install and upgrade RPMs are the same. See the following steps to install the RPMs using two steps procedure:

SUMMARY STEPS

1. **install add** <rpm>
2. **install activate** <rpm>

DETAILED STEPS

	Command or Action	Purpose
Step 1	<pre>install add <rpm> Example: switch# install add bootflash:vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm</pre>	Installs the RPM.

	Command or Action	Purpose
	<pre>[#####] 100% Install operation 892 completed successfully at Thu Jun 7 13:56:38 2018 switch(config)# sh install inactive grep vxlan vxlan-2.0.1.0-9.2.1.lib32_n9000</pre>	
Step 2	install activate <rpm> Example:	Activates the RPM.

Example

```
switch#install activate vxlan

[#####] 100%
Install operation 891 completed successfully at Thu Jun  7 13:53:07 2018

switch# show install active | grep vxlan

vxlan-2.0.0.0-9.2.1.lib32_n9000

switch# sh install inactive | grep vxlan

switch#
```

Upgrading the RPMs Using One Step

The CLIs for both install and upgrade RPMs are the same. See the following steps to upgrade the RPMs:

Procedure

	Command or Action	Purpose
Step 1	install add <rpm>activate upgrade Example: <pre>switch(config)# install add bootflash:bgp-2.0.2.0-9.2.1.lib32_n9000.rpm activate upgrade Adding the patch (/bgp-2.0.2.0-9.2.1.lib32_n9000.rpm) [#####] 100% Install operation 870 completed successfully at Tue May 8 11:22:30 2018</pre>	Installs the RPM.

	Command or Action	Purpose
	<pre> Activating the patch (/bgp-2.0.2.0-9.2.1.lib32_n9000.rpm) [#####] 100% Install operation 871 completed successfully at Tue May 8 11:22:40 2018 </pre>	

Example

```

switch(config)# show install active

Boot Image:
  NXOS Image: bootflash:/nxos.9.2.1.bin

Active Packages:
bgp-2.0.2.0-9.2.1.lib32_n9000
chef-12.0.0alpha.2+20150319234423.git.1608.b6eb10f-1.el5.x86_64

Active Base Packages:
  lacp-2.0.0.0-9.2.1.lib32_n9000
  lldp-2.0.0.0-9.2.1.lib32_n9000
  mtx-device-2.0.0.0-9.2.1.lib32_n9000
  mtx-grpc-agent-2.0.0.0-9.2.1.lib32_n9000
  mtx-infra-2.0.0.0-9.2.1.lib32_n9000
  mtx-netconf-agent-2.0.0.0-9.2.1.lib32_n9000
  mtx-restconf-agent-2.0.0.0-9.2.1.lib32_n9000
  mtx-telemetry-2.0.0.0-9.2.1.lib32_n9000
  ntp-2.0.0.0-9.2.1.lib32_n9000
  nxos-ssh-2.0.0.0-9.2.1.lib32_n9000
  snmp-2.0.0.0-9.2.1.lib32_n9000
  svi-2.0.0.0-9.2.1.lib32_n9000
  tacacs-2.0.0.0-9.2.1.lib32_n9000
  vtp-2.0.0.0-9.2.1.lib32_n9000
    
```

Downgrading the RPMs

The downgrade procedure needs a special CLI attribute. See the following step to downgrade the RPMs using the one step procedure:

Procedure

	Command or Action	Purpose
<p>Step 1</p>	<pre> install add <rpm>activate downgrade Example: switch(config)# install add bootflash:bgp-2.0.1.0-9.2.1.lib32_n9000.rpm activate downgrade Adding the patch </pre>	<p>Downgrades the RPM.</p>

	Command or Action	Purpose
	<pre>(/bgp-2.0.1.0-9.2.1.lib32_n9000.rpm) [#####] 100% Install operation 872 completed successfully at Tue May 8 11:24:43 2018 Activating the patch (/bgp-2.0.1.0-9.2.1.lib32_n9000.rpm) [#####] 100% Install operation 873 completed successfully at Tue May 8 11:24:52 2018</pre>	

Example

```
switch(config)# show install active
Boot Image:
  NXOS Image: bootflash:/nxos.9.2.1.bin

Active Packages:
  bgp-2.0.1.0-9.2.1.lib32_n9000
  chef-12.0.0alpha.2+20150319234423.git.1608.b6eb10f-1.e15.x86_64

Active Base Packages:
  lACP-2.0.0.0-9.2.1.lib32_n9000
  lldp-2.0.0.0-9.2.1.lib32_n9000
  mtx-device-2.0.0.0-9.2.1.lib32_n9000
  mtx-grpc-agent-2.0.0.0-9.2.1.lib32_n9000
  mtx-infra-2.0.0.0-9.2.1.lib32_n9000
  mtx-netconf-agent-2.0.0.0-9.2.1.lib32_n9000
  mtx-restconf-agent-2.0.0.0-9.2.1.lib32_n9000
  mtx-telemetry-2.0.0.0-9.2.1.lib32_n9000
  ntp-2.0.0.0-9.2.1.lib32_n9000
  nxos-ssh-2.0.0.0-9.2.1.lib32_n9000
  snmp-2.0.0.0-9.2.1.lib32_n9000
  svi-2.0.0.0-9.2.1.lib32_n9000
  tacacs-2.0.0.0-9.2.1.lib32_n9000
  vtp-2.0.0.0-9.2.1.lib32_n9000
switch(config)#
```

Uninstalling the RPMs

See the following steps to uninstall the RPMs:

SUMMARY STEPS

1. **install deactivate** *<rpm>*

DETAILED STEPS

	Command or Action	Purpose
Step 1	<p>install deactivate <rpm></p> <p>Example:</p> <pre>switch(config)# install deactivate bgp Base RPM found. Do you want to downgrade to base version(y/n) [n] y Downgrading to the base version [#####] 100% Install operation 190 completed successfully at Tue Nov 17 04:10:40 2020</pre> <p>Example:</p> <pre>switch(config)# install deactivate bgp Base RPM found. Do you want to downgrade to base version(y/n) [n] n</pre> <hr/> <pre>WARNING!! This operation will remove 'bgp-3.0.0.0-9.4.1.lib32_n9000' related configuration from running-configuration on successful completion. Update startup-configuration accordingly.</pre> <hr/> <pre>[#####] 100% Install operation 9 completed successfully at Tue Nov 17 05:05:59 2020</pre>	<p>Downgrades to the base version of RPM, if one exists in the groups-repo (/rpms), or uninstalls the RPM completely from the switch.</p> <p>To downgrade to the base version, enter y. To completely uninstall the RPM, enter n in the command prompt.</p>

Removing the RPMs

See the following steps to remove the RPMs:

SUMMARY STEPS

1. **install remove** <rpm>

DETAILED STEPS

	Command or Action	Purpose
Step 1	<p>install remove <rpm></p> <p>Example:</p> <pre>switch(config)# show install inactive grep vxlan vxlan-2.0.0.0-9.2.1.lib32_n9000 switch(config)# install remove vxlan Proceed with removing vxlan? (y/n)? [n] y [#####] 100%</pre>	Removes the RPM from the repository.

Command or Action	Purpose
Install operation 890 Removal of base rpm package is not permitted at Thu Jun 7 13:52:15 2018	

Information About DNF Commands

See the following sections for more information about DNF commands.



Note DNF commands do not support ctrl+c. Install commands do support ctrl+c. If DNF commands are aborted using ctrl+c, manual cleanup must be performed using `"/isan/bin/patching_utils.py --unlock"`.

Performing Package Operations Using the DNF Commands

See the following sections for performing package operations using the DNF commands:



Note DNF commands are accessed only from the BASH shell on the box and they are not allowed from the NXOS VSH terminal.



Note Make sure that as a sudo user, you have access to the super user privileges.

Finding the Base Version RPM of the Image

Use the `ls /rpms` command to find the base version RPM of the image. The base RPM version is the pre-installed RPM that is archived in the system image.

```
#ls /rpms
```

```
bfd-2.0.0.0-9.2.1.lib32_n9000.rpm
ins_tor_sdk_t2-1.0.0.0-9.2.0.77.lib32_n9000.rpm
mtx-netconf-agent-2.0.0.0-9.2.1.lib32_n9000.rpm      snmp-2.0.0.0-9.2.1.lib32_n9000.rpm
bgp-2.0.0.0-9.2.1.lib32_n9000.rpm
ins_tor_sdk_t3-1.0.0.0-9.2.0.77.lib32_n9000.rpm
mtx-restconf-agent-2.0.0.0-9.2.1.lib32_n9000.rpm    sr-2.0.0.0-9.2.1.lib32_n9000.rpm
container-tracker-2.0.0.0-9.2.1.lib32_n9000.rpm     isis-2.0.0.0-9.2.1.lib32_n9000.rpm
      mtx-telemetry-2.0.0.0-9.2.1.lib32_n9000.rpm     svi-2.0.0.0-9.2.1.lib32_n9000.rpm
eigrp-2.0.0.0-9.2.1.lib32_n9000.rpm                  lacp-2.0.0.0-9.2.1.lib32_n9000.rpm
      nbproxy-2.0.0.0-9.2.1.lib32_n9000.rpm
tacacs-2.0.0.0-9.2.1.lib32_n9000.rpm
ext-eth-2.0.0.0-9.2.1.lib32_n9000.rpm                 lldp-2.0.0.0-9.2.1.lib32_n9000.rpm
      ntp-2.0.0.0-9.2.1.lib32_n9000.rpm
telemetry-2.3.4.0-9.2.1.lib32_n9000.rpm
fcoe-2.0.0.0-9.2.1.lib32_n9000.rpm                    mcast-2.0.0.0-9.2.1.lib32_n9000.rpm
      nxos-ssh-2.0.0.0-9.2.1.lib32_n9000.rpm
virtualization-2.0.0.0-9.2.1.lib32_n9000.rpm
```

```

fex-2.0.0.0-9.2.1.lib32_n9000.rpm
ospf-2.0.0.0-9.2.1.lib32_n9000.rpm
fhrp-2.0.0.0-9.2.1.lib32_n9000.rpm
  repodata
vxlan-2.0.0.0-9.2.1.lib32_n9000.rpm
guestshell-2.0.0.0-9.2.1.lib32_n9000.rpm
  rip-2.0.0.0-9.2.1.lib32_n9000.rpm
icam-2.0.0.0-9.2.1.lib32_n9000.rpm
  services-2.0.0.0-9.2.1.lib32_n9000.rpm
mpls-2.0.0.0-9.2.1.lib32_n9000.rpm
vtp-2.0.0.0-9.2.1.lib32_n9000.rpm
mtx-device-2.0.0.0-9.2.1.lib32_n9000.rpm
mtx-grpc-agent-2.0.0.0-9.2.1.lib32_n9000.rpm
mtx-infra-2.0.0.0-9.2.1.lib32_n9000.rpm

```

Checking the List of the Installed RPMs

Use the **dnf list installed** command to query the feature and third party RPMs and grep a specific RPM. See the following example for feature RPMs:

```

bash-4.2# dnf list installed | grep lib32_n9000

bfd.lib32_n9000                2.0.0.0-9.2.1          @groups-repo
core.lib32_n9000              2.0.0.0-9.2.1          installed
eth.lib32_n9000               2.0.0.0-9.2.1          installed
guestshell.lib32_n9000        2.0.0.0-9.2.1          @groups-repo
lACP.lib32_n9000              2.0.0.0-9.2.1          installed
linecard2.lib32_n9000         2.0.0.0-9.2.1          installed
lldp.lib32_n9000              2.0.0.0-9.2.1          installed
mcast.lib32_n9000             2.0.0.0-9.2.1          @groups-repo
mtx-device.lib32_n9000        2.0.0.0-9.2.1          installed
mtx-grpc-agent.lib32_n9000    2.0.0.0-9.2.1          installed
mtx-infra.lib32_n9000         2.0.0.0-9.2.1          installed
mtx-netconf-agent.lib32_n9000 2.0.0.0-9.2.1          installed
mtx-restconf-agent.lib32_n9000 2.0.0.0-9.2.1          installed
mtx-telemetry.lib32_n9000     2.0.0.0-9.2.1          installed
nbproxy.lib32_n9000           2.0.0.0-9.2.1          installed
ntp.lib32_n9000               2.0.0.0-9.2.1          installed
nxos-ssh.lib32_n9000          2.0.0.0-9.2.1          installed
ospf.lib32_n9000              2.0.0.0-9.2.1          @groups-repo
platform.lib32_n9000          2.0.0.0-9.2.1          installed
snmp.lib32_n9000              2.0.0.0-9.2.1          installed
svi.lib32_n9000               2.0.0.0-9.2.1          installed
tacacs.lib32_n9000            2.0.0.0-9.2.1          installed
tor.lib32_n9000               2.0.0.0-9.2.0.77       installed
virtualization.lib32_n9000    2.0.1.0-9.2.1          @localdb
vtp.lib32_n9000               2.0.0.0-9.2.1          installed
vxlan.lib32_n9000             2.0.0.0-9.2.1          @groups-repo
...

```

Getting Details of the Installed RPMs

The **dnf info <rpmname>** command lists out the detailed info of the installed RPM.

```
dnf info vxlan
```

```

Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
protect-packages
groups-repo

```

```

| 1.1 kB    00:00 ...
localdb
| 951 B     00:00 ...

```

```

patching
| 951 B    00:00 ...
thirdparty
| 951 B    00:00 ...

Installed Packages
Name       : vxlan
Arch      : lib32_n9000
Version   : 2.0.0.0
Release   : 9.2.1
Size      : 6.4 M
Repo      : installed
From repo : groups-repo
Summary   : Cisco NXOS VxLAN
URL       : http://cisco.com/
License   : Proprietary
Description : Provides VxLAN support

```

Installing the RPMs

Installing the RPMs downloads the RPMs and copies the respective program to the switches. See the following example for installing the RPMs from a remote server (that is reachable in the network):

```

bash-4.3# dnf install
http://10.0.0.2/modularity/rpms/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm

Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
protect-packages
groups-repo
| 1.1 kB    00:00 ...
localdb
| 951 B     00:00 ...
localdb/primary
| 886 B     00:00 ...
localdb
| 1/1
patching
| 951 B     00:00 ...
thirdparty
| 951 B     00:00 ...

Setting up Install Process
vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm
| 1.6 MB    00:00
Examining /var/tmp/yum-root-RaANgb/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm:
vxlan-2.0.1.0-9.2.1.lib32_n9000
Marking /var/tmp/yum-root-RaANgb/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package vxlan.lib32_n9000 0:2.0.1.0-9.2.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

```

Package	Repository	Arch	Version	Size
Installing:				
vxlan		lib32_n9000	2.0.1.0-9.2.1	6.4 M
	/vxlan-2.0.1.0-9.2.1.lib32_n9000			

Transaction Summary

```

=====
Install          1 Package

Total size: 6.4 M
Installed size: 6.4 M
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : vxlan-2.0.1.0-9.2.1.lib32_n9000                1/1

starting pre-install package version mgmt for vxlan
pre-install for vxlan complete
starting post-install package version mgmt for vxlan
post-install for vxlan complete

Installed:
  vxlan.lib32_n9000 0:2.0.1.0-9.2.1

```

Complete!

See the following example for installing the RPMs from local bootflash:

```
sudo dnf install /bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm
```

```

Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
protect-packages
groups-repo

```

```

localdb                | 1.1 kB    00:00 ...
patching                | 951 B     00:00 ...
thirdparty              | 951 B     00:00 ...
                       | 951 B     00:00 ...

```

Setting up Install Process

```

Examining /bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm: vxlan-2.0.1.0-9.2.1.lib32_n9000
Marking /bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm as an update to
vxlan-2.0.0.0-9.2.1.lib32_n9000

```

Resolving Dependencies

```

--> Running transaction check
---> Package vxlan.lib32_n9000 0:2.0.0.0-9.2.1 will be updated
---> Package vxlan.lib32_n9000 0:2.0.1.0-9.2.1 will be an update
--> Finished Dependency Resolution

```

Dependencies Resolved

```

=====
Package Arch Repository
Version Size
-----
Updating:
  vxlan lib32_n9000
2.0.1.0-9.2.1 /vxlan-2.0.1.0-9.2.1.lib32_n9000

```

6.4 M

Transaction Summary

Upgrade 1 Package

Total size: 6.4 M

Is this ok [y/N]: y

Downloading Packages:

Running Transaction Check

Running Transaction Test

Transaction Test Succeeded

Running Transaction

Updating : vxlan-2.0.1.0-9.2.1.lib32_n9000

1/2

starting pre-install package version mgmt for vxlan

pre-install for vxlan complete

starting post-install package version mgmt for vxlan

post-install for vxlan complete

Cleanup : vxlan-2.0.0.0-9.2.1.lib32_n9000

2/2

Updated:

vxlan.lib32_n9000 0:2.0.1.0-9.2.1

Complete!

See the following example for installing the RPM if it is available in a repository:

dnf install eigrp

Upgrading the RPMs

See the following example for upgrading the RPMs from a remote server (that is reachable in the network):

```

bash-4.3# dnf upgrade
http://10.0.0.2/modularity/rpms/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm

Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
protect-packages
groups-repo
localdb | 1.1 kB 00:00 ...
patching | 951 B 00:00 ...
thirdparty | 951 B 00:00 ...
Setting up Upgrade Process
vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm | 1.6 MB 00:00
Examining /var/tmp/yum-root-RaANgb/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm:
vxlan-2.0.1.0-9.2.1.lib32_n9000
Marking /var/tmp/yum-root-RaANgb/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm as an update to
vxlan-2.0.0.0-9.2.1.lib32_n9000
Resolving Dependencies
--> Running transaction check

```

```

---> Package vxlan.lib32_n9000 0:2.0.0.0-9.2.1 will be updated
---> Package vxlan.lib32_n9000 0:2.0.1.0-9.2.1 will be an update
--> Finished Dependency Resolution

```

Dependencies Resolved

Package	Repository	Arch	Version	Size
Updating:				
vxlan		lib32_n9000	2.0.1.0-9.2.1	
	/vxlan-2.0.1.0-9.2.1.lib32_n9000			6.4 M
Transaction Summary				

Upgrade 1 Package

```

Total size: 6.4 M
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
** Found 1 pre-existing rpmdb problem(s), 'yum check' output follows:
busybox-1.23.2-r0.0.x86_64 has missing requires of busybox-syslog
  Updating   : vxlan-2.0.1.0-9.2.1.lib32_n9000                1/2

starting pre-install package version mgmt for vxlan
pre-install for vxlan complete
starting post-install package version mgmt for vxlan
post-install for vxlan complete
  Cleanup   : vxlan-2.0.0.0-9.2.1.lib32_n9000                2/2

Updated:
  vxlan.lib32_n9000 0:2.0.1.0-9.2.1

```

Complete!

See the following example for upgrading the RPMs from local bootflash:

```
sudo dnf upgrade /bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm
```

```

Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
protect-packages
groups-repo

```

```

localdb           | 1.1 kB    00:00 ...
patching          | 951 B     00:00 ...
thirdparty        | 951 B     00:00 ...

```

Setting up Upgrade Process

```
Examining /bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm: vxlan-2.0.1.0-9.2.1.lib32_n9000
```

```

Marking /bootflash/vxlan-2.0.1.0-9.2.1.lib32_n9000.rpm as an update to
vxlan-2.0.0.0-9.2.1.lib32_n9000
Resolving Dependencies
--> Running transaction check
---> Package vxlan.lib32_n9000 0:2.0.0.0-9.2.1 will be updated
---> Package vxlan.lib32_n9000 0:2.0.1.0-9.2.1 will be an update
--> Finished Dependency Resolution

```

Dependencies Resolved

Package Version	Arch Size	Repository
Updating: vxlan 2.0.1.0-9.2.1	lib32_n9000 6.4 M	/vxlan-2.0.1.0-9.2.1.lib32_n9000

Transaction Summary

Upgrade 1 Package

```

Total size: 6.4 M
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Updating      : vxlan-2.0.1.0-9.2.1.lib32_n9000

```

1/2

```

starting pre-install package version mgmt for vxlan
pre-install for vxlan complete
starting post-install package version mgmt for vxlan
post-install for vxlan complete
Cleanup      : vxlan-2.0.0.0-9.2.1.lib32_n9000

```

2/2

```

Updated:
  vxlan.lib32_n9000 0:2.0.1.0-9.2.1

```

Complete!

See the following example for upgrading the RPMs if it is available in any repository:

```
dnf upgrade eigrp
```

Downgrading the RPMs

See the following example for downgrading the RPMs from a remote server (that is reachable in the network):

```
sudo dnf  
downgrade vxlan-2.0.0.0-9.2.1.lib32_n9000
```



```
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
protect-packages
Setting up Downgrade Process
groups-repo
```

```
localdb          | 1.1 kB    00:00 ...
localdb/primary  | 951 B    00:00 ...
localdb          | 1.3 kB    00:00 ...
                2/2
patching
thirdparty       | 951 B    00:00 ...
                | 951 B    00:00 ...
```

```
Resolving Dependencies
--> Running transaction check
---> Package vxlan.lib32_n9000 0:2.0.0.0-9.2.1 will be a downgrade
---> Package vxlan.lib32_n9000 0:2.0.1.0-9.2.1 will be erased
--> Finished Dependency Resolution
```

Dependencies Resolved

Package	Version	Size	Arch	Repository
---------	---------	------	------	------------

Downgrading:				
vxlan	2.0.0.0-9.2.1	1.6 M	lib32_n9000	groups-repo

Transaction Summary

Downgrade 1 Package

Total download size: 1.6 M

Is this ok [y/N]: y

Downloading Packages:

Running Transaction Check

Running Transaction Test

Transaction Test Succeeded

Running Transaction

Installing : vxlan-2.0.0.0-9.2.1.lib32_n9000

1/2

starting pre-install package version mgmt for vxlan

pre-install for vxlan complete

starting post-install package version mgmt for vxlan

post-install for vxlan complete

Cleanup : vxlan-2.0.1.0-9.2.1.lib32_n9000

2/2

Removed:

vxlan.lib32_n9000 0:2.0.1.0-9.2.1

```
Installed:
  vxlan.lib32_n9000 0:2.0.0.0-9.2.1
```

```
Complete!
```

See the following example for downgrading the RPMs from local bootflash:

```
dnf downgrade /bootflash/eigrp-2.0.0-9.2.1.lib32_n9000.rpm
```

See the following example for downgrading the RPMs if it is available in any repository:

```
dnf downgrade eigrp
```

Deleting the RPMs

Deleting the RPMs de-installs the RPMs and removes any configuration CLI of the feature. Use the **dnf erase** *<rpm>* command to delete the RPMs.

```
bash-4.2# sudo dnf erase vxlan
```

```
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
protect-packages
Setting up Remove Process
Resolving Dependencies
--> Running transaction check
---> Package vxlan.lib32_n9000 0:2.0.1.0-9.2.1 will be erased
--> Finished Dependency Resolution
```

```
Dependencies Resolved
```

Package	Arch	Repository	Version	Size
Removing:				
vxlan	lib32_n9000	@/vxlan-2.0.1.0-9.2.1.lib32_n9000	2.0.1.0-9.2.1	6.4 M

```
Transaction Summary
```

```
Remove      1 Package
```

```
Installed size: 6.4 M
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Erasing      : vxlan-2.0.1.0-9.2.1.lib32_n9000
```

```
1/1
```

```
starting pre-remove package version mgmt for vxlan
```

```

pre-remove for vxlan complete

Removed:
  vxlan.lib32_n9000 0:2.0.1.0-9.2.1

Complete!

```

Support for DNF Groups

The support for DNF groups is part of the package management. It simplifies the management of the packages for the administrators and it provides greater flexibility.

The administrators can group a list of packages (RPMs) into a logical group and they can perform various operations. DNF supports the following group commands:

- `groupinstall`
- `groupinfo`
- `groupinstall`
- `groupremove`
- `groupupdate`

DNF groups can be broadly classified as L2, L3, routing, and management.

Using the `groupinstall` Command

In Linux, number of packages are bundled to particular group. Instead of installing individual packages with `dnf`, you can install particular group that will install all the related packages that belongs to the group. For example to list all the available groups, use the `dnf groupinstall` command:

```

bash-4.4# dnf groupinstall
Last metadata expiration check: 0:00:00 ago on Fri 08 Mar 2024 12:26:33 PM UTC.
] --- B/s | 0 B    --:-- ETA
Available Groups:
  management
  routing
  L2
  L3
bash-4.4#

```

Using the `groupmembers` Command

Use `dnf groupinfo` command to display the description and the contents of a package group. The command lists out the feature members of the group.

```

bash-4.4# dnf groupinfo l2
Last metadata expiration check: 0:00:00 ago on Fri 08 Mar 2024 12:27:44 PM UTC.
] --- B/s | 0 B    --:-- ETA

Group: L2
Mandatory Packages:
  lacp
  lldp
  svi

```

```

vtp
bash-4.4#

```

Using the groupinstall Command

This command is for both install and upgrade of the members RPM. If the member is not installed, it will install the highest version available. If the member is already installed and higher RPM is available, it will upgrade that member.

```

bash-4.4# dnf groupinstall l3
Last metadata expiration check: 0:00:00 ago on Fri 08 Mar 2024 12:38:05 PM UTC.
] --- B/s | 0 B    --:-- ETA
Not a redundant system. Nothing todo
Dependencies resolved.

```

Group	Packages
-------	----------

Marking packages as installed by the group:

@L3	bfd
-----	-----

Is this ok [y/N]: y

Complete!

Install operation 10 completed successfully at Fri Mar 8 12:38:08 2024.

[#####] 100%

Using the groupupdate Command

Use the **dnf groupupdate** command to update any existing installed group packages.

```

bash-4.4# dnf groupupdate l3

```

```

Last metadata expiration check: 0:00:00 ago on Wed 13 Mar 2024 12:30:11 PM UTC.

```

```

] --- B/s | 0 B    --:-- ETA

```

Dependencies resolved.

Group	Packages
-------	----------

Marking packages as installed by the group:

@L3	bfd
-----	-----

Package	Arch Repository	Size	Version
---------	--------------------	------	---------

Installing group packages:

bfd	lib32_64_n9000 groups-repo	562 k	2.0.0.0-10.4.3
-----	-------------------------------	-------	----------------

Transaction Summary

Install 1 Package

Total size: 562 k

Installed size: 2.3 M

Is this ok [y/N]: y

Downloading Packages:

Running transaction check

Transaction check succeeded.

```

Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :                               1/1

  Running scriptlet: bfd-2.0.0.0-10.4.3.lib32_64_n9000
                                                    1/1
starting pre-install package version mgmt for bfd
pre-install for bfd complete
  Installing      : bfd-2.0.0.0-10.4.3.lib32_64_n9000
                                                    1/1

  Running scriptlet: bfd-2.0.0.0-10.4.3.lib32_64_n9000
                                                    1/1
starting post-install package version mgmt for bfd
post-install for bfd complete
  Verifying      : bfd-2.0.0.0-10.4.3.lib32_64_n9000
                                                    1/1

Installed:
  bfd.lib32_64_n9000 2.0.0.0-10.4.3

Complete!
Install operation 14 completed successfully at Wed Mar 13 12:30:23 2024.

[#####] 100%
bash-4.4#

```

Using the grouperase Command

Use the **dnf grouperase** command to delete the groups or all the RPM members of the group.

```

bash-4.4# dnf grouperase l3
Dependencies resolved.

```

```

=====
Group                                     Packages
=====
Marking packages as removed by the group:
@L3                                       bfd
=====

Package      Repository      Arch      Size      Version
=====
Removing:
bfd          @System        lib32_64_n9000 2.3 M     2.0.0.0-10.4.3

Transaction Summary
=====
Remove 1 Package

Freed space: 2.3 M
Is this ok [y/N]: y
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :                               1/1

  Running scriptlet: bfd-2.0.0.0-10.4.3.lib32_64_n9000

```

```

starting pre-remove package version mgmt for bfd
pre-remove for bfd complete
  Erasing          : bfd-2.0.0.0-10.4.3.lib32_64_n9000
                    1/1
  Running scriptlet: bfd-2.0.0.0-10.4.3.lib32_64_n9000
                    1/1
starting post-remove package version mgmt for bfd
post-remove for bfd complete

  Verifying        : bfd-2.0.0.0-10.4.3.lib32_64_n9000
                    1/1

Removed:
  bfd.lib32_64_n9000 2.0.0.0-10.4.3
                                          Complete!
Install operation 11 completed successfully at Fri Mar  8 12:38:41 2024.

[#####] 100%
bash-4.4#

```

Finding Repositories

This command lists the repositories that the switch has along with the number of RPMs it has to those repositories.

```

bash-4.3# dnf repolist all

Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
protect-packages
groups-repo

localdb          | 1.1 kB    00:00 ...
patching         | 951 B     00:00 ...
thirdparty      | 951 B     00:00 ...
repo id         | 951 B     00:00 ...
repo name
status
groups-repo
  Groups-RPM Database
enabled: 37
localdb
  Local RPM Database
enabled: 6
patching
  Patch-RPM Database
enabled: 0
thirdparty
  Thirdparty RPM Database
enabled: 0
open-nxos
  open-nxos
disabled
repolist: 43

```

Finding the Installed DNF Version

See the following example for listing the installed DNF version:

```
dnf --version
```

```
3.4.3
  Installed: rpm-5.4.14-r0.0.x86_64 at 2018-06-02 13:04
  Built      : Wind River <info@windriver.com> at 2018-04-27 08:36
  Committed: Wind River <info@windriver.com> at 2018-04-27

  Installed: yum-3.4.3-r9.0.x86_64 at 2018-06-02 13:05
  Built      : Wind River <info@windriver.com> at 2018-04-27 08:36
  Committed: Wind River <info@windriver.com> at 2018-04-27
```

Mapping the NX-OS CLI to the DNF Commands

See the following table for mapping the NX-OS CLI to the DNF commands:

Table 5: Patching Command Reference

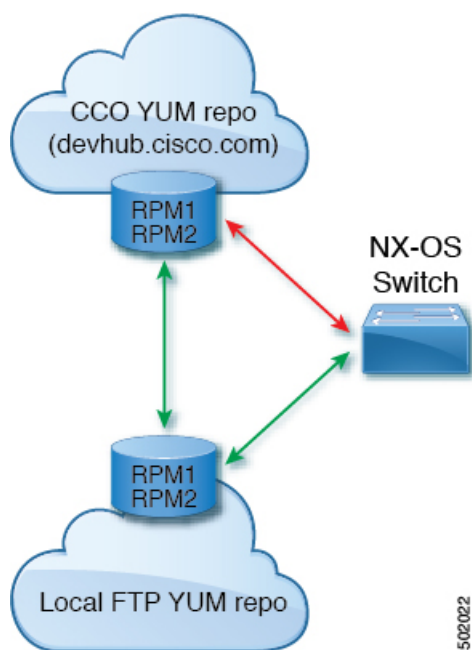
NX-OS CLI Commands	DNF Commands
show install inactive	dnf list --patch-only available
show install active	dnf list --patch-only installed
show install committed	dnf list --patch-only committed
show install packages	dnf list --patch-only
show install pkg-info	dnf info --patch-only
show install log	dnf history --show-patch-log where log_cmd: <ul style="list-style-type: none"> • opid= - Log that is specific to an operation ID. • last - Shows the latest operation log. • reverse – Shows the log in reverse order. • detail – Show detailed log. • from= - Shows logging from a specific operation ID.
clear install log	dnf history --clear-patch-log= where clear_log_cmd: <ul style="list-style-type: none"> • all - Clears the complete log. • - Clears the logs above this operation ID.
install add	dnf install --add bootflash:/

NX-OS CLI Commands	DNF Commands
install remove	dnf install --remove
install remove inactive	dnf install --remove all
install activate	dnf install --no-persist --nocommit Note By default, all packages are activated and committed.
install deactivate	dnf erase --nocommit Note By default, all packages are de-activated and committed.
install commit	dnf install --commit
Install commit	dnf install --commit all

Configuring an FTP server and Setting up a Local FTP YUM Repository

For setting up a local FTP YUM repository, you have to first create an FTP server, create a local FTP YUM repository, and configure the Cisco NX-OS switch to reach the FTP server as outlined in the following illustration.

Figure 2: Configuring an FTP server and Setting up a Local FTP YUM Repository



502022



Note For Cisco NX-OS Release 10.1(1), visit <https://devhub.cisco.com/artifactory/open-nxos/10.1.1/> for Cisco `open-nxos` repository.

Creating an FTP Server on Red Hat Enterprise Linux 7 (RHEL7) Virtual Machine

Complete the following steps to create an FTP server on Red Hat Enterprise Linux 7 (RHEL7) Virtual Machine (VM):

SUMMARY STEPS

1. `dnf install vsftpd`
2. `systemctl start vsftpd`
3. `systemctl status vsftpd`
4. `firewall-cmd --zone=public --permanent --add-port=21/tcp`
5. `firewall-cmd --zone=public --permanent --add-service=ftp`
6. `firewall-cmd --reload`
7. `wget ftp:// <ip of FTP server> /test.txt`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>dnf install vsftpd</code>	Installs vsftpd, an FTP server.
Step 2	<code>systemctl start vsftpd</code>	Starts the FTP Server.
Step 3	<code>systemctl status vsftpd</code>	Checks the status of the FTP Server.
Step 4	<code>firewall-cmd --zone=public --permanent --add-port=21/tcp</code>	Allows access to the FTP services from the external systems and opens port 21.
Step 5	<code>firewall-cmd --zone=public --permanent --add-service=ftp</code>	Adds the FTP service.
Step 6	<code>firewall-cmd --reload</code>	Reloads the server.
Step 7	<code>wget ftp:// <ip of FTP server> /test.txt</code>	Hosts a file in the FTP server (for example, test.txt) and attempts Wget of that file. Note Note that <code>/var/ftp/</code> is the default home directory of the FTP server.

Creating a Local FTP YUM Repository

Complete the following steps to synchronize the external repository RPMs to the FTP server and create a local FTP YUM repository:

SUMMARY STEPS

1. **touch /etc/yum.repos.d/local.repo**
2. **vim /etc/yum.repos.d/local.repo**
3. **cat /etc/yum.repos.d/local.repo**
4. **dnf repolist**
5. **nohup reposync -r <repo-name mentioned in the local.repo> -p <directory path to sync>&**
6. **tail -f nouhup.out**

DETAILED STEPS

	Command or Action	Purpose
Step 1	touch /etc/yum.repos.d/local.repo Example: <pre>bash-4.3#touch /etc/yum.repos.d/local.repo</pre>	Creates a repository file under /etc/yum.repos.d/ , for example, creates local.repo repository and adds the base URL.
Step 2	vim /etc/yum.repos.d/local.repo Example: <pre>bash-4.3#vim /etc/yum.repos.d/local.repo [localrepo] name=localrepo baseurl= https://devhub.cisco.com/artifactory/open-nxos/7.0-3-I2-1/x86_64/ enabled=1 gpgcheck=0 sslverify=0</pre>	Edit the repository file and copy the localrepo details. Note Modify the base URL to the required repository URL.
Step 3	cat /etc/yum.repos.d/local.repo Example: <pre>bash-4.3#cat /etc/yum.repos.d/local.repo [localrepo] name=localrepo baseurl= https://devhub.cisco.com/artifactory/open-nxos/7.0-3-I2-1/x86_64/ enabled=1 gpgcheck=0 sslverify=0</pre>	Verify the local repository data to proceed further.
Step 4	dnf repolist Example: <pre>bash-4.3# dnf repolist Loaded plugins: fastestmirror, langpacks Loading mirror speeds from cached hostfile * base: mirror.dhakacom.com * extras: mirror.dhakacom.com * updates: mirror.dhakacom.com repo id repo name status base/7/x86_64 CentOS-7 - Base 9,911 extras/7/x86_64 CentOS-7 - Extras 313 localrepo localrepo 687</pre>	Checks the reachability of the repository.

	Command or Action	Purpose
	updates/7/x86_64 CentOS-7 - Updates 711 repolist: 11,622	
Step 5	<p>nohup reposync -r <i><repo-name mentioned in the local.repo></i> -p <i><directory path to sync></i> &</p> <p>Example:</p> <pre>nohup reposync -r localrepo -p /var/ftp/ &</pre> <p>This command creates a directory with the name local.repo inside /var/ftp/ and downloads all the packages from devhub.cisco.com to the directory.</p>	Synchronizes all the packages from the external repository to the FTP server home directory.
Step 6	tail -f nouhup.out	Checks the status of the synchronization.

Configuring a Switch to Reach an FTP Server

Complete the following steps to configure a switch to reach an FTP server:

SUMMARY STEPS

1. **run bash sudo su**
2. **ip netns exec management ping** *<ip_address>*
3. **touch /etc/yum/repos.d/ftp.repo**
4. **vim /etc/yum/repos.d/ftp.repo**
5. **cat /etc/yum/repos.d/ftp.repo**
6. **ip netns exec management bash**
7. **dnf repolist**
8. **dnf list available**

DETAILED STEPS

	Command or Action	Purpose
Step 1	run bash sudo su	Logs in as a sudo user.
Step 2	ip netns exec management ping <i><ip_address></i>	Checks the reachability of the FTP server address from the switch using the ping command.
Step 3	<p>touch /etc/yum/repos.d/ftp.repo</p> <p>Example:</p> <pre>bash-4.3#touch /etc/yum/repos.d/ftp.repo</pre>	Creates a repository file under /etc/yum/repos.d/ , for example, creates ftp.repo repository.
Step 4	<p>vim /etc/yum/repos.d/ftp.repo</p> <p>Example:</p> <pre>bash-4.3#vim /etc/yum/repos.d/ftp.repo</pre> <pre>[ftp] name=ftp baseurl=</pre>	<p>Edit the repository file and copy the ftp repo details.</p> <p>Note Modify the base URL to the required ftp server IP.</p>

	Command or Action	Purpose
	ftp://198.51.100.1/localrepo/ enabled=1 gpgcheck=0 sslverify=0	
Step 5	cat /etc/yum/repos.d/ftp.repo Example: bash-4.3# cat /etc/yum/repos.d/ftp.repo [ftp] name=ftp baseurl=ftp://198.51.100.1/localrepo/ enabled=1 gpgcheck=0 sslverify=0	Creates a repository file on the switch with the FTP server address as the URL.
Step 6	ip netns exec management bash	Uses the Bash shell prompt.
Step 7	dnf repolist Example: bash-4.3# dnf repolist Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching, : protect-packages groups-repo 1.1 kB 00:00 ... localdb 951 B 00:00 ... patching 951 B 00:00 ... thirdparty 951 B 00:00 ... thirdparty/primary 758 B 00:00 ... thirdparty 1/1 repo id repo name status groups-repo Groups-RPM Database 37 localdb Local RPM Database 0 patching Patch-RPM Database 0 thirdparty Thirdparty RPM Database 1 ftp ftp 686 repolist: 724	Checks the reachability of newly created repository.
Step 8	dnf list available	Lists the available packages in the new repository.

Creating User Roles for Install Operation

The **install** command is only available to the users of admin role. The **install** command can be available to a user by RBAC. See RBAC configuration guidelines for the same.

Compacting Cisco NX-OS Software Images

Cisco NX-OS software image compaction reduces the size of the image file before completing a copy request. Use SCP, HTTP, or HTTPS as the source and bootflash or USB as the destination. The following example uses SCP and bootflash:

```
switch# copy scp://user@scpserver.cisco.com//download/nxos64.10.1.1.bin  
bootflash:nxos64.10.1.1.bin compact vrf management use-kstack
```

```
user1@10.65.42.196's password:  
nxos64.10.1.1.bin 100% 1501MB 8.4MB/s 02:58  
Copy complete, now saving to disk (please wait)...  
Copy complete.
```

The **compact** keyword compacts the NX-OS image before copying the file to the supervisor module.



Note Software image compaction is only supported on SCP, HTTP, or HTTPS. If you attempt compaction with any other protocol, the system returns the following error:

```
Compact option is allowed only with source as scp/http/https and destination  
as bootflash or usb
```



Note Compacted images are not supported with LXC boot mode.



Note Software image compaction is only supported on Cisco Nexus 9300-series platform switches.
