



Third-Party Applications

This chapter contains the following sections:

- [About Third-Party Applications](#), on page 1
- [Installing Signed Third-Party RPMs by Importing Keys Automatically](#), on page 1
- [Installing Signed RPM](#), on page 3
- [Persistent Third-Party RPMs](#), on page 8
- [Installing RPM from VSH](#), on page 9
- [Third-Party Applications](#), on page 13

About Third-Party Applications

The RPMs for the Third-Party Applications are available in the repository at . These applications are installed in the native host by using the **dnf** command in the Bash shell or through the NX-OS CLI.

When you enter the **dnf install rpm** command, a Cisco **DNF** plug-in gets executed. This plug-in copies the RPM to a hidden location. On switch reload, the system reinstalls the RPM.

For configurations in `/etc`, a Linux process, **incron**, monitors artifacts that are created in the directory and copies them to a hidden location, which gets copied back to `/etc`.

Installing Signed Third-Party RPMs by Importing Keys Automatically

Setup the yum repo to point to the keys and RPM.

```
root@switch# cat /etc/yum/repos.d/puppet.repo
[puppet]
name=Puppet RPM
baseurl=file:///bootflash/puppet
enabled=1
ggpcheck=1
ggpkey=http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
```

```

metadata_expire=0

cost=500

bash-4.2# yum install puppet-enterprise

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 ...
puppet                     | 951 B     00:00 ...
thirdparty                 | 951 B     00:00 ...

Setting up Install Process

Resolving Dependencies

--> Running transaction check

---> Package puppet-enterprise.x86_64 0:3.7.1.rc2.6.g6cdc186-1.pe.nxos will be installed

--> Finished Dependency Resolution

Dependencies Resolved

=====

Package                Arch      Version                               Repository      Size
=====

Installing:

puppet-enterprise      x86_64    3.7.1.rc2.6.g6cdc186-1.pe.nxos      puppet          14 M

Transaction Summary

=====

Install      1 Package

Total download size: 14 M

Installed size: 46 M

Is this ok [y/N]: y

Retrieving key from file:///bootflash/RPM-GPG-KEY-puppetlabs

Importing GPG key 0x4BD6EC30:

  Userid: "Puppet Labs Release Key (Puppet Labs Release Key) <info@puppetlabs.com>"
  From   : /bootflash/RPM-GPG-KEY-puppetlabs

```

```
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
Warning! Standby is not ready. This can cause RPM database inconsistency.
If you are certain that standby is not booting up right now, you may proceed.
Do you wish to continue?
Is this ok [y/N]: y
Warning: RPMDB altered outside of yum.
Installing : puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64          1/1
/sbin/ldconfig: /usr/lib/libboost_regex.so.1.49.0 is not a symbolic link
Installed:
puppet-enterprise.x86_64 0:3.7.1.rc2.6.g6cdc186-1.pe.nxos

Complete!
```

Installing Signed RPM

Checking a Signed RPM

Run the following command to check if a given RPM is signed or not.

```
Run, rpm -K rpm_file_name
```

Not a signed RPM

```
bash-4.2# rpm -K bgp-1.0.0-r0.lib32_n3600.rpm
bgp-1.0.0-r0.lib32_n3600.rpm: (sha1) dsa sha1 md5 OK
```

Signed RPM

```
bash-4.2#
rpm -K puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64.rpm
puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64.rpm: RSA sha1 MD5 NOT_OK
bash-4.2#
```

Signed third-party rpm requires public GPG key to be imported first before the package can be installed otherwise **yum** will throw the following error:

```
bash-4.2#
yum install puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64.rpm -q

Setting up Install Process

warning: rpmts_HdrFromFdno: Header V4 RSA/SHA1 signature: NOKEY, key ID 4bd6ec30

Cannot open: puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64.rpm. Skipping.

Error: Nothing to do
```

Installing Signed RPMs by Manually Importing Key

- Copy the GPG keys to `/etc rootfs` so that they are persisted across reboots.

```
bash-4.2# mkdir -p /etc/pki/rpm-gpg
bash-4.2# cp -f RPM-GPG-KEY-puppetlabs /etc/pki/rpm-gpg/
```

- Import the keys using the below command

```
bash-4.2# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
```

```
bash-4.2#
```

```
bash-4.2# rpm -q gpg-pubkey
```

```
gpg-pubkey-4bd6ec30-4c37bb40
```

```
bash-4.2# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
```

```
bash-4.2#
```

```
bash-4.2# rpm -q gpg-pubkey
```

```
gpg-pubkey-4bd6ec30-4c37bb40
```

- Install the signed RPM with *yum* command

```
bash-4.2#
yum install puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64.rpm

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 ...

Setting up Install Process
```

```
Examining puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64.rpm:
puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64

Marking puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64.rpm to be installed

Resolving Dependencies

--> Running transaction check

---> Package puppet-enterprise.x86_64 0:3.7.1.rc2.6.g6cdc186-1.pe.nxos will be installed

--> Finished Dependency ResolutionDependencies Resolved
```

```
=====
Package           Arch      Version                               Repository
Size
```

```
Installing:
```

```
puppet-enterprise x86_64 3.7.1.rc2.6.g6cdc186-1.pe.nxos /puppet-enterprise-
46 M
3.7.1.rc2.6.g6cdc186-1.
pe.nxos.x86_64
```

```
Transaction Summary
```

```
=====
Install          1 Package
```

```
Total size: 46 M
```

```
Installed size: 46 M
```

```
Is this ok [y/N]: y
```

```
Downloading Packages:
```

```
Running Transaction Check
```

```
Running Transaction Test
```

```
Transaction Test Succeeded
```

```
Running Transaction
```

```
Installing : puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64
```

```
1/1
```

```
Installed:
```

```
puppet-enterprise.x86_64 0:3.7.1.rc2.6.g6cdc186-1.pe.nxos
```

```
Complete!
bash-4.2#
```

Installing Signed Third-Party RPMs by Importing Keys Automatically

Setup the yum repo to point to the keys and RPM.

```
root@switch# cat /etc/yum/repos.d/puppet.repo
```

```
[puppet]
```

```
name=Puppet RPM
```

```
baseurl=file:///bootflash/puppet
```

```
enabled=1
```

```
gpgcheck=1
```

```
gpgkey=http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
```

```
metadata_expire=0
```

```
cost=500
```

```
bash-4.2# yum install puppet-enterprise
```

```
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 ...
puppet                      |  951 B    00:00 ...
thirdparty                  |  951 B    00:00 ...
```

```
Setting up Install Process
```

```
Resolving Dependencies
```

```
--> Running transaction check
```

```
---> Package puppet-enterprise.x86_64 0:3.7.1.rc2.6.g6cdc186-1.pe.nxos will be installed
```

```
--> Finished Dependency Resolution
```

```
Dependencies Resolved
```

```
=====
```

Package	Arch	Version	Repository	Size
=====				

```
Installing:
```

```
puppet-enterprise    x86_64    3.7.1.rc2.6.g6cdc186-1.pe.nxos    puppet    14 M
```

```
Transaction Summary
```

```
=====
```

```
Install          1 Package
```

```
Total download size: 14 M
```

```
Installed size: 46 M
```

```
Is this ok [y/N]: y
```

```
Retrieving key from file:///bootflash/RPM-GPG-KEY-puppetlabs
```

```
Importing GPG key 0x4BD6EC30:
```

```
  Userid: "Puppet Labs Release Key (Puppet Labs Release Key) <info@puppetlabs.com>"
```

```
  From   : /bootflash/RPM-GPG-KEY-puppetlabs
```

```
Is this ok [y/N]: y
```

```
Downloading Packages:
```

```
Running Transaction Check
```

```
Running Transaction Test
```

```
Transaction Test Succeeded
```

```
Running Transaction
```

```
Warning! Standby is not ready. This can cause RPM database inconsistency.
```

```
If you are certain that standby is not booting up right now, you may proceed.
```

```
Do you wish to continue?
```

```
Is this ok [y/N]: y
```

```
Warning: RPMDB altered outside of yum.
```

```
Installing : puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64
```

```
1/1
```

```
/sbin/ldconfig: /usr/lib/libboost_regex.so.1.49.0 is not a symbolic link
```

```
Installed:
```

```
puppet-enterprise.x86_64 0:3.7.1.rc2.6.g6cdc186-1.pe.nxos
```

```
Complete!
```

Adding Signed RPM into Repo

Step 1 Copy signed RPM to repo directory

Step 2 Import the corresponding key for the create repo to succeed

```
bash-4.2# ls
puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64.rpm  RPM-GPG-KEY-puppetlabs
bash-4.2#
bash-4.2# rpm --import RPM-GPG-KEY-puppetlabs
bash-4.2# createrepo .
1/1 - puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64.rpm
Saving Primary metadata
Saving file lists metadata
Saving other metadata
bash-4.2#
```

Without importing keys

```
bash-4.2# ls
puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64.rpm  RPM-GPG-KEY-puppetlabs
bash-4.2#
bash-4.2# createrepo .
warning: rpmts_HdrFromFdno: Header V4 RSA/SHA1 signature: NOKEY, key ID 4bd6ec30

Error opening package - puppet-enterprise-3.7.1.rc2.6.g6cdc186-1.pe.nxos.x86_64.rpm

Saving Primary metadata
Saving file lists metadata
Saving other metadata
```

Step 3 Create repo config file under /etc/yum/repos.d pointing to this repo

```
bash-4.2# cat /etc/yum/repos.d/puppet.repo
[puppet]
name=Puppet RPM
baseurl=file:///bootflash/puppet
enabled=1
gpgcheck=1
gpgkey=file:///bootflash/puppet/RPM-GPG-KEY-puppetlabs
#gpgkey=http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
metadata_expire=0
cost=500

bash-4.2# yum list available puppet-enterprise -q
Available Packages
puppet-enterprise.x86_64          3.7.1.rc2.6.g6cdc186-1.pe.nxos
                                puppet
bash-4.2#
```

Persistent Third-Party RPMs

The following is the logic behind persistent third-party RPMs:

- A local **dnf** repository is dedicated to persistent third-party RPMs. The `/etc/yum/repos.d/thirdparty.repo` points to `/bootflash/.rpmstore/thirdparty`.
- Whenever you enter the **dnf install third-party.rpm** command, a copy of the RPM is saved in `/bootflash/.rpmstore/thirdparty`.
- During a reboot, all the RPMs in the third-party repository are reinstalled on the switch.

- Any change in the `/etc` configuration files persists under `/bootflash/.rpmstore/config/etc` and they are replayed during boot on `/etc`.
- Any script that is created in the `/etc` directory persists across reloads. For example, a third-party service script that is created under `/etc/init.d/` brings up the apps during a reload.

Installing RPM from VSH

Package Addition

NX-OS feature RPMs can also be installed by using the VSH CLIs.

SUMMARY STEPS

1. `show install packages`
2. `install add ?`
3. `install add rpm-packagename`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>show install packages</code>	Displays the packages and versions that already exist.
Step 2	<code>install add ?</code>	Determine supported URIs.
Step 3	<code>install add rpm-packagename</code>	The <code>install add</code> command copies the package file to a local storage device or network server.

Example

The following example shows how to activate the Chef RPM:

```
switch# show install packages
switch# install add ?
WORD          Package name
bootflash:   Enter package uri
ftp:         Enter package uri
http:        Enter package uri
modflash:    Enter package uri
scp:         Enter package uri
sftp:        Enter package uri
tftp:        Enter package uri
usb1:        Enter package uri
usb2:        Enter package uri
volatile:    Enter package uri
switch# install add
bootflash:chef-12.0.0alpha.2+20150319234423.git.1608.b6eb10f-1.e15.x86_64.rpm
[#####] 100%
Install operation 314 completed successfully at Thu Aug 6 12:58:22 2015
```

What to do next

When you are ready to activate the package, go to [Package Activation](#).



Note Adding and activating an RPM package can be accomplished in a single command:

```
switch#
install add bootflash:chef-12.0.0alpha.2+20150319234423.git.1608.b6eb10f-1.e15.x86_64.rpm
activate
```

Package Activation

Before you begin

The RPM has to have been previously added.

SUMMARY STEPS

1. show install inactive
2. install activate *rpm-packagename*

DETAILED STEPS

	Command or Action	Purpose
Step 1	show install inactive	Displays the list of packages that were added and not activated.
Step 2	install activate <i>rpm-packagename</i>	Activates the package.

Example

The following example shows how to activate a package:

```
switch# show install inactive
Boot image:
    NXOS Image: bootflash:///yumcli6.bin

Inactive Packages:
    sysinfo-1.0.0-7.0.3.x86_64
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching,
                : protect-packages
Available Packages
chef.x86_64          12.0.0alpha.2+20150319234423.git.1608.b6eb10f-1.e15  thirdparty
eigrp.lib32_n3600  1.0.0-r0                                           groups-rep
o
sysinfo.x86_64     1.0.0-7.0.3                                       patching
switch# install activate chef-12.0-1.e15.x86_64.rpm
[#####] 100%
Install operation completed successfully at Thu Aug 6 12:46:53 2015
```

Deactivating Packages

SUMMARY STEPS

1. `install deactivate package-name`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>install deactivate <i>package-name</i></code>	Deactivates the RPM package.

Example

The following example shows how to deactivate the Chef RPM package:

```
switch# install deactivate chef
```

Removing Packages

Before you begin

Deactivate the package before removing it. Only deactivated RPM packages can be removed.

SUMMARY STEPS

1. `install remove package-name`

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>install remove <i>package-name</i></code>	Removes the RPM package.

Example

The following example shows how to remove the Chef RPM package:

```
switch# install remove chef-12.0-1.e15.x86_64.rpm
```

Displaying Installed Packages

SUMMARY STEPS

1. `show install packages`

DETAILED STEPS

	Command or Action	Purpose
Step 1	show install packages	Displays a list of the installed packages.

Example

The following example shows how to display a list of the installed packages:

```
switch# show install packages
```

Displaying Detail Logs

SUMMARY STEPS

1. show tech-support install

DETAILED STEPS

	Command or Action	Purpose
Step 1	show tech-support install	Displays the detail logs.

Example

The following example shows how to display the detail logs:

```
switch# show tech-support install
```

Upgrading a Package

SUMMARY STEPS

1. install add *package-name* activate upgrade

DETAILED STEPS

	Command or Action	Purpose
Step 1	install add <i>package-name</i> activate upgrade	Upgrade a package.

Example

The following example show how to upgrade a package:

```
switch# install add bootflash:bgp-1.0.1-r0.1ib32_n3600.rpm activate ?
downgrade Downgrade package
forced      Non-interactive
```

```

upgrade      Upgrade package
switch# install add bootflash:bgp-1.0.1-r0.lib32_n3600.rpm activate upgrade
[#####] 100%
Install operation completed successfully at Thu Aug  6 12:46:53 2015

```

Downgrading a Package

SUMMARY STEPS

1. install add *package-name* activate downgrade

DETAILED STEPS

	Command or Action	Purpose
Step 1	install add <i>package-name</i> activate downgrade	Downgrade a package.

Example

The following example shows how to downgrade a package:

```

switch# install add bootflash:bgp-1.0.1-r0.lib32_n3600.rpm activate ?
downgrade  Downgrade package
forced      Non-interactive
upgrade     Upgrade package
switch# install add bootflash:bgp-1.0.1-r0.lib32_n3600.rpm activate downgrade
[#####] 100%
Install operation completed successfully at Thu Aug  6 12:46:53 2015

```

Third-Party Applications

NX-OS

For more information about NX-API REST API object model specifications, see <https://developer.cisco.com/media/dme/index.html>

collectd

collectd is a daemon that periodically collects system performance statistics and provides multiple means to store the values, such as RRD files. Those statistics can then be used to find current performance bottlenecks (i.e. performance analysis) and predict future system load (that is, capacity planning).

For additional information, see <https://collectd.org>.

Ganglia

Ganglia is a scalable distributed monitoring system for high-performance computing systems such as clusters and grids. It is based on a hierarchical design targeted at federations of clusters. It leverages widely used technologies such as XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualization. It uses engineered data structures and algorithms to achieve very low per-node overheads and high concurrency. The implementation is robust, has been ported to an extensive set of operating systems and processor architectures, and is currently in use on thousands of clusters around the world. It has been used to link clusters across university campuses and around the world and can scale to handle clusters with 2000 nodes.

For additional information, see <http://ganglia.info>.

Iperf

Iperf was developed by NLANR/DAST to measure maximum TCP and UDP bandwidth performance. Iperf allows the tuning of various parameters and UDP characteristics. Iperf reports bandwidth, delay jitter, datagram loss.

For additional information, see <http://sourceforge.net/projects/iperf/> or <http://iperf.sourceforge.net>.

LLDP

The link layer discover protocol (LLDP) is an industry standard protocol designed to supplant proprietary link layer protocols such as EDP or CDP. The goal of LLDP is to provide an inter-vendor compatible mechanism to deliver link layer notifications to adjacent network devices.

For more information, see <https://vincentbernat.github.io/lldpd/index.html>.

Nagios

Nagios is open source software that monitors network services (through ICMP, SNMP, SSH, FTP, HTTP etc), host resources (CPU load, disk usage, system logs, etc.), and alert services for servers, switches, applications, and services through the Nagios remote plugin executor (NRPE) and through SSH or SSL tunnels.

For more information, see <https://www.nagios.org/>.

OpenSSH

OpenSSH is an open-source version of the SSH connectivity tools that encrypts all traffic (including passwords) to effectively eliminate eavesdropping, connection hijacking, and other attacks. OpenSSH provides secure tunneling capabilities and several authentication methods, and supports all SSH protocol versions.

For more information, see <http://www.openssh.com>.

Quagga

Quagga is a network routing software suite that implements various routing protocols. Quagga daemons are configured through a network accessible CLI called a "vty".



Note Only Quagga BGP has been validated.

For more information, see <http://www.nongnu.org/quagga/>.

Splunk

Splunk is a web based data collection, analysis, and monitoring tool that has a search, visualization and pre-packaged content for use-cases. The raw data is sent to the Splunk server using the Splunk Universal Forwarder. Universal Forwarders provide reliable, secure data collection from remote sources and forward that data into the Splunk Enterprise for indexing and consolidation. They can scale to tens of thousands of remote systems, collecting terabytes of data with minimal impact on performance.

For additional information, see http://www.splunk.com/en_us/download/universal-forwarder.html.

tccollector

tccollector is a client-side process that gathers data from local collectors and pushes the data to Open Time Series Database (OpenTSDB).

tccollector has the following features:

- Runs data collectors and collates the data,
- Manages connections to the time series database (TSD),
- Eliminates the need to embed TSD code in collectors,
- De-duplicates repeated values, and
- Handles wire protocol work.

For additional information, see http://opentsdb.net/docs/build/html/user_guide/utilities/tcollector.html.

tcpdump

Tcpdump is a CLI application that prints out a description of the contents of packets on a network interface that match the boolean expression; the description is preceded by a time stamp, printed, by default, as hours, minutes, seconds, and fractions of a second since midnight. It can also be run with the `-w` flag, which causes it to save the packet data to a file for later analysis, and/or with the `-r` flag, which causes it to read from a saved packet file rather than to read packets from a network interface. It can also be run with the `-V` flag, which causes it to read a list of saved packet files. In all cases, only packets that match expression will be processed by tcpdump.

For more information, see <http://www.tcpdump.org/manpages/tcpdump.1.html>.

Tshark

TShark is a network protocol analyzer on the CLI. It lets you capture packet data from a live network, or read packets from a previously saved capture file. You can either print a decoded form of those packets to the standard output or write the packets to a file. TShark's native capture file format is the pcap format, which is

also the format used by **tcpdump** and various other tools. Tshark can be used within the Guest Shell 2.1 after removing the `cap_net_admin` file capability.

```
setcap  
cap_net_raw=ep /sbin/dumpcap
```



Note This command must be run within the Guest Shell.

For more information, see <https://www.wireshark.org/docs/man-pages/tshark.html>.