

Third-Party Applications

- About Third-Party Applications, on page 1
- Guidelines and Limitations, on page 1
- Installing Python2 and Dependent Packages, on page 2
- Installing Third-Party Native RPMs/Packages, on page 2
- Persistent Third-Party RPMs, on page 4
- Installing RPM from VSH, on page 4
- Third-Party Applications, on page 9

About Third-Party Applications

Third-Party 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, **incrond**, monitors artifacts that are created in the directory and copies them to a hidden location, which gets copied back to `/etc`.

Guidelines and Limitations

RPMs for the third-party applications have the following guidelines and limitations:

- The NX-OS 10.1(1) release has a new operating system and rootfs, based on NX-Linux(Cisco's proprietary Linux distribution), so third-party RPMs that were built using WRL5/WRL8 might not be compatible with NX-Linux, so the third-party software might not work. In this case, remove old versions of your apps used with previous releases and replace them with new software that is compatible with NX-Linux..
- Guidelines and instructions for installing signed RPMs are provided in the *Cisco Nexus 9000 Series NX-OS Software Upgrade and Downgrade Guide, Release 9.2(x)*, including DNF and VSH CLI options for managing RPMs, signed and nonsigned RPM installations, the clean-up of repositories, and so on.
- The third-party applications are started during switch startup. It is possible that a third-party application could be started before its communication interface is up, or before the routing between the switch and any communication peer or server is established. Therefore, all third-party applications should be written to be robust in case of communication failure, and the application should retry establishing the connection.

If an application is not resilient in the presence of a communication failure, a “wrapper” application might be required to establish that any communication peer is reachable before starting the desired application, or restart the desired application if necessary.

- Beginning with Cisco NX-OS Release 10.2(3)F, Python2 and dependent RPMs are removed from NX-OS. However, you can install Python2 and dependent RPMs from devhub site as package group packagegroup-nxos-64-python-2-deprecated-rpms.

Installing Python2 and Dependent Packages

The following is the complete workflow of package installation:

```
switch# cat /etc/dnf/repos.d/open-nxos.repo
[open-nxos]
name=open-nxos
baseurl=https://devhub.cisco.com/artifactory/open-nxos/10.2.3/
enabled=1
gpgcheck=0
sslverify=0

dnf info packagegroup-nxos-64-python-2-deprecated-rpms
dnf install packagegroup-nxos-64-python-2-deprecated-rpms
The output of these cmds will be available post KR3F CCO.
```

Installing Third-Party Native RPMs/Packages

The complete workflow of package installation is as follows:

Procedure

Configure the repository on the switch to point to the Cisco repository where agents are stored.

```
bash-4.2# cat /etc/dnf/repos.d/open-nxos.repo
[open-nxos]
name=open-nxos



enabled=1
gpgcheck=0
sslverify=0
```

Instructions for using the CLIs to import the digital signature are available in the section "Using Install CLIs for Digital Signature Support" in the *Cisco Nexus 9000 Series NX-OS Software Upgrade and Downgrade Guide, Release 9.2(x)*.

An example of installation of an RPM using *dnf*, with full install log.

Example:

```
bash-4.2# dnf install splunkforwarder
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching, protect-packages
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package splunkforwarder.x86_64 0:6.2.3-264376 will be installed
```

```
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================
 Package              Arch                 Version           Repository              Size
================================================================================================
Installing:
splunkforwarder       x86_64               6.2.3-264376      open-nxos               13 M

Transaction Summary
================================================================================================
Install       1 Package

Total size: 13 M
Installed size: 34 M
Is this ok [y/N]: y
Downloading Packages:
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : splunkforwarder-6.2.3-264376.x86_64
                                                                                1/1
complete

Installed:
  splunkforwarder.x86_64 0:6.2.3-264376


Complete!
bash-4.2#
```

An example of querying the switch for successful installation of the package, and verifying that its processes or services are up and running.

Example:

```
bash-4.2# dnf info splunkforwarder
Loaded plugins: downloadonly, importpubkey, localrpmDB, patchaction, patching, protect-packages
Fretta              |  951 B     00:00 ...
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        : splunkforwarder
Arch        : x86_64
Version     : 6.2.3
Release     : 264376
Size        : 34 M
Repo        : installed
From repo   : open-nxos
Summary     : SplunkForwarder
License     : Commercial
Description : The platform for machine data.
```

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.



Note The rules in iptables are not persistent across reboots when they are modified in a bash-shell.

To make the modified iptables persistent, see Making an Iptable Persistent Across Reloads.

Installing RPM from VSH

Package Addition

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

SUMMARY STEPS

1. **show install package**
2. **install add ?**
3. **install add** *rpm-packagename*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**show install package**	Displays the packages and versions that already exist.
Step 2	**install add ?**	Determine supported URIs.

	Command or Action	Purpose
Step 3	**install add** *rpm-packagename*	The **install add** 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 package
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.el5.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, on page 5.

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.el5.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

Procedure

	Command or Action	Purpose
Step 1	**show install inactive**	Displays the list of packages that were added and not activated.
Step 2	**install activate** *rpm-packagename*	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.el5 thirdparty
eigrp.lib32_n9000 1.0.0-r0                                            groups-rep
o
sysinfo.x86_64    1.0.0-7.0.3                                         patching
switch# install activate chef-12.0-1.el5.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

Procedure

	Command or Action	Purpose
Step 1	**install deactivate** *package-name*	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

Procedure

	Command or Action	Purpose
Step 1	**install remove** *package-name*	Removes the RPM package.

Example

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

```
switch# install remove chef-12.0-1.el5.x86_64.rpm
```

Displaying Installed Packages

SUMMARY STEPS

1. **show install packages**

DETAILED STEPS

Procedure

	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

Procedure

	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

Procedure

	Command or Action	Purpose
Step 1	install add *package-name* activate upgrade	Upgrade a package.

Example

The following example shows how to upgrade a package:

```
switch# install add bootflash:bgp-1.0.1-r0.lib32_n9000.rpm activate ?
downgrade  Downgrade package
forced     Non-interactive
upgrade    Upgrade package
switch# install add bootflash:bgp-1.0.1-r0.lib32_n9000.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

Procedure

	Command or Action	Purpose
Step 1	install add *package-name* 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_n9000.rpm activate ?
downgrade  Downgrade package
forced     Non-interactive
upgrade    Upgrade package
switch# install add bootflash:bgp-1.0.1-r0.lib32_n9000.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/docs/nx-api-dme-model-9-3-1-reference/

DevOps Configuration Management Tools

For DevOps configuration management tools, refer to the following links:

- Ansible 2.0 Release(Nexus Support), Ansible Release Index
- Ansible NX-OS Sample Modules, Ansible NX-OS Sample Modules
- Puppet, Puppet Forge Cisco Puppet
- Cisco Puppet Module(Git), Cisco Network Puppet Module
- Chef, Chef Supermarket Cisco Cookbook
- Cisco Chef Cookbook(Git), Cisco Network Chef Cookbook

V9K

To download a virtual Nexus 9000 switch, for an ESX5.1/5.5, VirtualBox, Fusion, and KVM, go to https://software.cisco.com/portal/pub/download/portal/select.html?&mdfid=286312239&flowid=81422&softwareid=282088129.

Automation Tool Educational Content

For a free book on Open NX-OS architecture and automation, see http://www.cisco.com/c/dam/en/us/td/docs/switches/datacenter/nexus9000/sw/open_nxos/programmability/guide/Programmability_Open_NX-OS.pdf

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 (for example, 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 that is 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 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 that is designed to supplant proprietary link layer protocols such as EDP or CDP. The goal of LLDP is to provide an intervendor 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 the following through the Nagios remote plug-in executor (NRPE) and through SSH or SSL tunnels:

- Network services through ICMP, SNMP, SSH, FTP, HTTP, and so on
- Host resources, such as CPU load, disk usage, system logs, and so on
- Alert services for servers, switches, applications
- Services

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 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 search, visualization, and prepackaged 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 a minimal impact on performance.

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

tcollector

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

tcollector 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.
- Deduplicates repeated values.
- 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 a description of the contents of packets on a network interface that match a Boolean expression. The description is preceded by a timestamp, printed, by default, as hours, minutes, seconds, and fractions of a second since midnight.

tcpdump can be run with the following flags:

- -w, which causes it to save the packet data to a file for later analysis.
- -r, which causes it to read from a saved packet file rather than to read packets from a network interface.
- -V, which causes it to read a list of saved packet files.

In all cases, tcpdump processes only the packets that match the expression.

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

TShark

TShark is a network protocol analyzer on the CLI. Tshar lets you capture packet data from a live network, or read packets from a previously saved capture file. You can print either a decoded form of those packets to the standard output or write the packets to a file. TShark's native capture file format is pcap, the format that is used by **tcpdump** and various other tools also. TShark can be used within the Guest Shell 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.

Third-Party Application (TPA) Container Hosting

Beginning with Cisco NX-OS Release 10.6(2), Cisco Nexus 9000 switches support hosting third-party applications (TPA) in Linux containers directly on the switch. This feature enables administrators and automation systems to deploy, manage, and monitor containerized applications natively on the switch, without the need for external servers.

Benefits of Third-Party Application Hosting

This capability provides several benefits:

- Flexible deployment of custom and vendor applications: Allows you to run Linux-based applications—such as DHCP/TFTP servers, syslog collectors, or monitoring tools—inside containers directly on the switch, supporting a wide range of use cases at the network edge.
- Integrated management and automation: Provides NX-OS CLI commands and programmable APIs (such as gNMI and gNOI) for configuring, monitoring, and managing TPA containers, enabling integration with automated workflows and backend systems.
- Operational visibility and troubleshooting: Offers real-time status, resource utilization, and log retrieval for running containers, simplifying troubleshooting and performance analysis.
- Persistent and reliable application operation: Ensures that critical applications can be configured to automatically restart following device upgrades or reboots, supporting continuous and automated service delivery.
- Support for secure and compliant deployments: Aligns with Cisco's security and compliance best practices, including VRF-aware deployment, secure communication, and persistent storage for configuration and credentials.
- Designed for edge and cloud integration: Supports on-box application hosting and cloud-native management in distributed and hybrid environments.

How Container-Based Third-Party Application (TPA) Hosting Works

This capability leverages integration between the NX-OS operating system, the integrated Docker engine, and AppManager to provide on-switch container lifecycle management for TPA hosting.

Summary

The following components play key roles in container-based TPA hosting:

- NX-OS: Manages feature enablement, configuration, and monitoring for containerized applications.
- Docker Engine: Provides the container runtime environment and manages the lifecycle of Linux application containers.
- AppManager: Coordinates between NX-OS and Docker for container control, lifecycle actions, and status reporting.

This process enables secure, persistent, and automated hosting of Linux container-based third-party applications directly on the switch.

Workflow

The process for hosting and managing container-based third-party applications (TPA) involves the following stages:

1. Feature Enablement: Enabling the TPA feature starts the Docker engine as a managed service inside NX-OS.
2. Internal Communication: A loopback interface is configured for all communication between NX-OS and the Docker engine, ensuring traffic isolation for containers.

3. Container Storage and Startup: Application container images are stored on the switch and started by the Docker engine according to TPA configuration.
4. Persistence: Persistent storage (host-mounted volumes) ensures container data and credentials survive reboots and upgrades.
5. Lifecycle Management: AppManager and related NX-OS processes control container start, stop, restart, and monitor status.
6. Monitoring and Logging: NX-OS provides commands to retrieve container status, resource usage, and logs from Docker and the containers.
7. Automation and APIs: TPA hosting integrates with NX-OS CLI and programmable APIs (such as gNMI/gNOI) for automation and external management.
8. Security and Isolation: VRF-aware communication and Unix domain sockets are used for secure, isolated operations within the switch.

Guidelines and Limitations

Observe the following feature compatibility and operational restrictions when using TPA Container Hosting:

- Only loopback interfaces are supported for use as the transport interface between NX-OS and Docker containers. This interface must be explicitly configured for TPA communication.
- ND-ISSU (Non-Disruptive In-Service Software Upgrade) is not supported when the Docker Box (TPA Container Hosting) feature is enabled.
- Manual cleanup required: If **dockerbox** or native Docker was previously enabled, performing a **write erase** followed by a reload will not automatically remove the **/bootflash/dockerpart** filesystem. You must manually delete **/bootflash/dockerpart** to reclaim disk space or before re-enabling Docker features.
- Change in **write erase** behavior: In future releases, **write erase** may automatically remove **/bootflash/dockerpart** if native Docker was previously deployed. This can result in Docker containers being brought up again. Be aware of this change if you rely on persistent container data after a **write erase**.
- The Docker Box (TPA Container Hosting) feature cannot co-exist with the **nxsecure** or **appmgr** features.
- SSH connectivity through kstack: SSH connections can be accepted over the Linux kernel network stack (kstack) in addition to the existing netstack path. The SSH daemon (sshd) will start on the configured loopback interface and default SSH port.

Consider the following resource and operational recommendations for successful deployment and maintenance:

- For optimal performance and reliability, it is recommended to use a switch platform with at least 6 CPU cores and 64 GB of memory when running application containers.
- Store configuration files, credentials, and persistent application data in dedicated volumes or host-mount directories to ensure data is preserved across device reboots and upgrades.
- Critical application containers should be configured to automatically restart after a system upgrade or reboot, ensuring continuous operation of essential services.
- Advanced monitoring and performance optimization for containers is not included in the current implementation. Users can utilize native Docker commands for basic resource and performance tracking.

- The total storage available for Docker images is limited to 5 GB, which is allocated from the **/bootflash/** space on the switch.
- The supported Docker version is **20.10.25-ce** and the supported containerd version is **1.6.19**. To check the currently installed versions, use **run bash sudo docker version** for Docker and **docker-containerd -v** for containerd from the switch CLI.
- Do not upgrade Docker or containerd independently. Upgrading to Free and Open Source Software (FOSS) versions not validated by Cisco can lead to incompatibility and loss of support. Always use the Docker and containerd versions provided with your Cisco NX-OS release.
- If you use scripts to deploy Docker containers, ensure that the scripts perform all necessary checks and validations required for successful and reliable container deployment.

Configure Third-Party Application (TPA) Container Hosting

Use these steps to enable and configure container-based third-party application (TPA) hosting on Cisco Nexus 9000 switches.

Before you begin

Identify the loopback interface you will use for Docker service communication.

Procedure

Step 1 Enable the TPA feature.

Example:

```
feature dockerbox
```

This command enables Docker-based third-party application hosting on the switch.

Step 2 Enter TPA configuration mode.

Example:

```
dockerbox
```

This command places the switch in TPA (Docker Box) configuration mode.

Step 3 Configure the transport interface for Docker service communication.

Example:

```
transport-interface <loopback-interface>
```

Only loopback interfaces are supported for Docker service communication. Replace **loopback-interface** with your chosen loopback interface.

Container-based TPA hosting is now enabled and configured. You can now deploy and manage application containers on the switch.

What to do next

Verify your configuration using the **show dockerbox status** and related commands. Refer to the verification section for example commands and expected output.

Verify Third-Party Application (TPA) Container Hosting

Use these steps to verify the status and operation of third-party application (TPA) container hosting on Cisco Nexus 9000 switches.

These verification steps allow you to check the health and operation of TPA hosting and its containers using NX-OS CLI commands.

Before you begin

Ensure that TPA hosting is configured and enabled on your switch.

Procedure

Step 1 Display the status of TPA hosting.

Example:

```
show dockerbox status
```

This command displays the current status of the TPA (Docker) service.

```
<example output placeholder>
```

Step 2 View internal state and debug information.

Example:

```
show system internal dockerbox
```

This command provides detailed internal state information for TPA hosting.

```
<example output placeholder>
```

Step 3 Display event history.

Example:

```
show system internal dockerbox event-history [error|debug|msgs]
```

Use this command to view error logs, debug messages, or message history related to TPA hosting.

```
<example output placeholder>
```

Step 4 Show container information.

Example:

```
show system internal dockerbox containers [brief|stats|logs]
```

This command displays information about running containers, including summaries, resource stats, or logs.

```
<example output placeholder>
```

Step 5 View TPA hosting logs.

Example:

```
show system internal dockerbox log [errors]
```

This command displays Docker service logs for TPA hosting. Use the **errors** option to show only error messages.

```
<example output placeholder>
```