Engineering
Simplicity

How to Configure the NFX350

Published
2025-12-12

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

YEAR 2000 NOTICE

END USER LICENSE AGREEMENT

Table of Contents

About This Guide

Use this guide to perform initial provisioning, configure Junos OS features, chain multiple virtualized network functions, monitor, and manage the NFX350 Series devices.

1

CHAPTER

Overview

IN THIS CHAPTER

NFX350 Overview

IN THIS SECTION

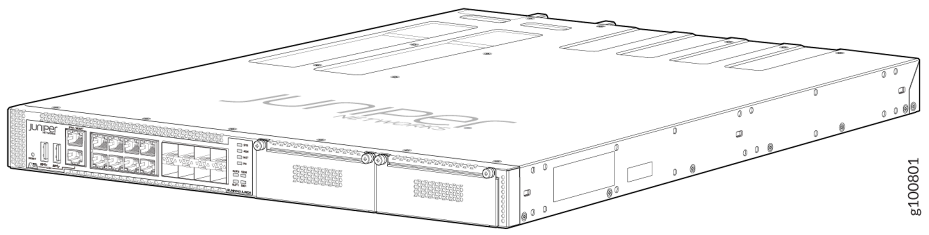
The Juniper Networks NFX350 Network Services Platform is a secure, automated, software-driven customer premises equipment (CPE) platform that delivers virtualized network and security services on demand. The NFX350 is part of the Juniper Cloud CPE solution, which leverages Network Functions Virtualization (NFV).

The NFX350 platform completes the uCPE portfolio to provide end-to-end platforms for medium, large, and extra-large deployments. In addition to IPsec and SD-WAN functionality, the NFX350 provides features such as LAN or WAN isolation, software and hardware resiliency, redundant power supply, Baseboard Management Controller, and serial over LAN.

The NFX350 has the Intel Skylake-D processor which provides increased throughput and cache. Integrated QAT helps accelerate applications that perform cryptographic operations such as IPsec.

Figure 1 on page 2 shows the NFX350 device.

Figure 1: NFX350 Device



Some typical deployment scenarios where you can use the NFX350 are:

- MSP/SP large/extra-large deployments requiring platform resiliency
- IOT gateway
- Resource-intensive deployments

Software Architecture

The architecture is designed to provide a unified control plane that functions as a single management point. Key components in the software include the JCP, JDM, Layer 2 data plane, Layer 3 data plane, and VNFs.

Figure 2 on page 3 and Figure 3 on page 4 illustrate the software architecture of the NFX350 in throughput, hybrid, and compute modes.

Figure 2: NFX350 NextGen Software Architecture (Throughput Mode)

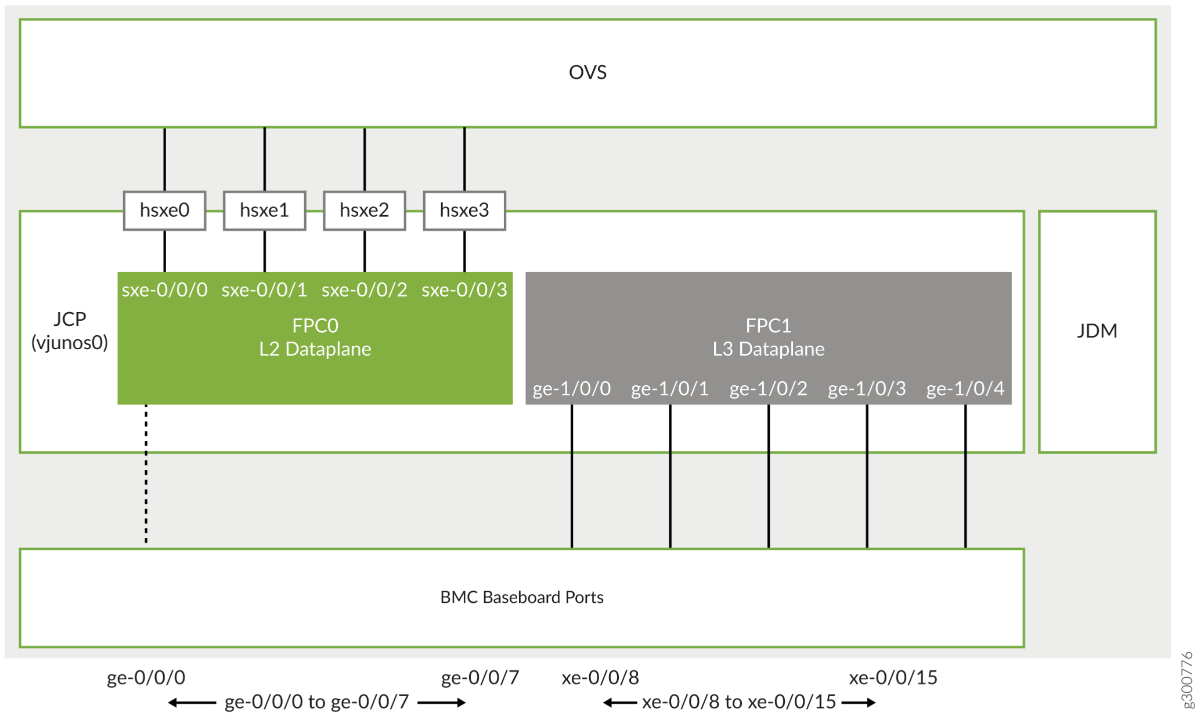
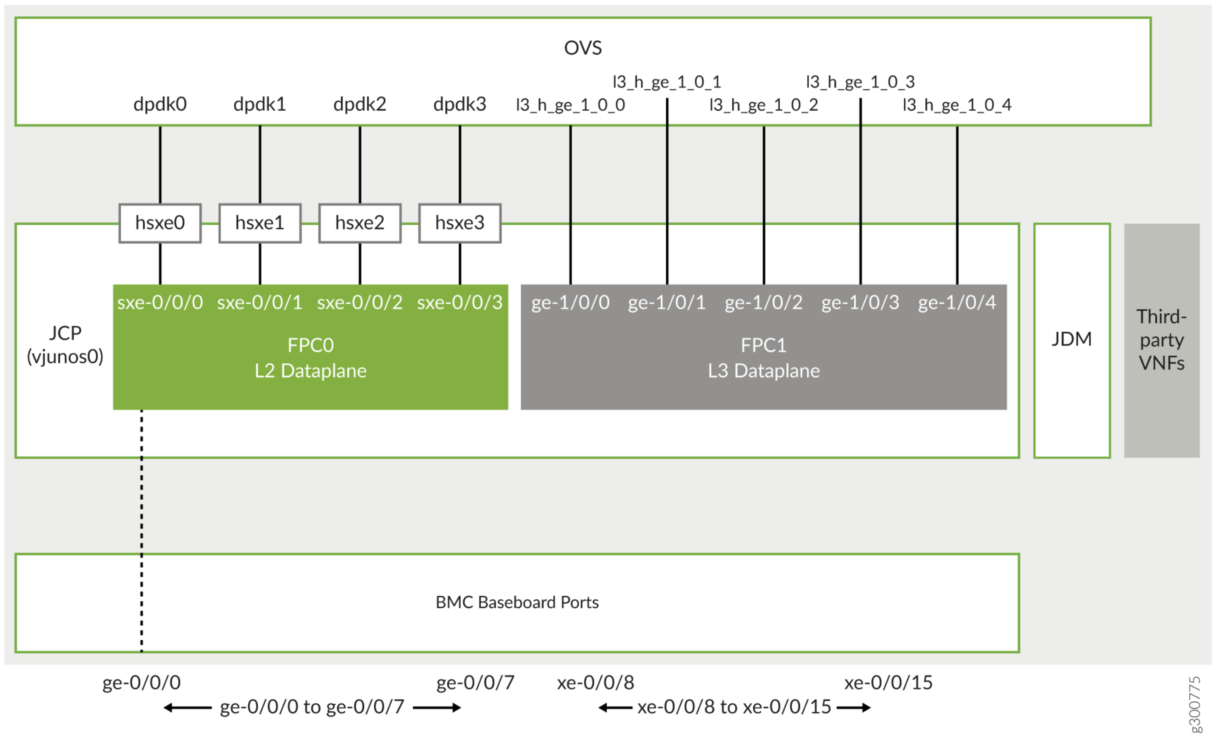


Figure 3: NFX350 NextGen Software Architecture (Hybrid or Compute Mode)



Key components of the system software include:

- Linux—The host OS, which functions as the hypervisor.
- VNF—A VNF is a virtualized implementation of a network device and its functions. Linux functions as the hypervisor, and it creates and runs the VNFs. The VNFs include functions such as firewalls, routers, and WAN accelerators.

  You can connect VNFs together as blocks in a chain to provide networking services. The NFX350 supports up to eight VNFs thereby enabling increased network functions and port density.
- JCP—Junos virtual machine (VM) running on the host OS, Linux. The JCP functions as the single point of management for all the components.

  The JCP supports:

  - Layer 2 to Layer 3 routing services
  - Layer 3 to Layer 4 security services
  - Layer 4 to Layer 7 advanced security services

  In addition, the JCP enables VNF lifecycle management.

- JDM—An application container that manages VNFs and provides infrastructure services. The JDM functions in the background. Users cannot access the JDM directly.
- L2 data plane—Manages Layer 2 traffic. The Layer 2 dataplane forwards the LAN traffic to the Open vSwitch (OVS) bridge, which acts as the NFV backplane. The Layer 2 dataplane is mapped to the virtual FPC0 on the JCP.
- L3 data plane—Provides data path functions for the Layer 3 to Layer 7 services. The Layer 3 data plane is mapped to the virtual FPC1 on the JCP.
- Open vSwitch (OVS) bridge—The OVS bridge is a VLAN-aware system bridge that acts as the NFV backplane to which the VNFs, FPC1, and FPC0 connect. Additionally, you can create custom OVS bridges to isolate connectivity between different VNFs.

  On NFX350, you can configure up to 72 OVS interfaces, which includes the VNF and FPC1 interfaces.

For the list of supported features, see Feature Explorer.

NFX350 Models

Table 1 on page 5 lists the NFX350 device models and its specifications. For more information, see the *NFX350 Hardware Guide*.

Table 1: NFX350 Series Device Models and Specifications

	NFX350-S1	NFX350-S2	NFX350-S3
CPU	8-core Intel Skylake D-2146NT	12-core Intel Skylake D-2166NT	16-core Intel Skylake D-2187NT
RAM	32 GB	64 GB	128 GB
Storage	100 GB SSD	100 GB SSD	100 GB SSD
Form Factor	Rack	Rack	Rack
Ports	Eight 1-Gigabit Ethernet RJ-45 ports	Eight 1-Gigabit Ethernet RJ-45 ports	Eight 1-Gigabit Ethernet RJ-45 ports

Table 1: NFX350 Series Device Models and Specifications *(Continued)*

	NFX350-S1	NFX350-S2	NFX350-S3
	Eight 10-Gigabit Ethernet SFP+ ports	Eight 10-Gigabit Ethernet SFP+ ports	Eight 10-Gigabit Ethernet SFP+ ports
	One management/ Intelligent Platform Management Interface (IPMI) port	One management/ Intelligent Platform Management Interface (IPMI) port	One management/ Intelligent Platform Management Interface (IPMI) port
	One console port (RJ-45 and mini-USB)	One console port (RJ-45 and mini-USB)	One console port (RJ-45 and mini-USB)
	Two USB 3.0 port	Two USB 3.0 port	Two USB 3.0 port
LTE support	Yes	Yes	Yes
Expansion module support	Two expansion module slots (one dual slot width NFX-LTE-AA/AE expansion module slot width expansion module)	Two expansion module slots (one dual slot width NFX-LTE-AA/AE expansion module slot width expansion module)	Two expansion module slots (one dual slot width NFX-LTE-AA/AE expansion module slot width expansion module)
Supported expansion modules	• NFX-LTE-AE—Expansion module with an LTE modem supporting the frequency bands in Europe and North America. • NFX-LTE-AA—Expansion module with an LTE modem supporting the frequency bands in Asia, Australia, and New Zealand.	• NFX-LTE-AE—Expansion module with an LTE modem supporting the frequency bands in Europe and North America. • NFX-LTE-AA—Expansion module with an LTE modem supporting the frequency bands in Asia, Australia, and New Zealand.	• NFX-LTE-AE—Expansion module with an LTE modem supporting the frequency bands in Europe and North America. • NFX-LTE-AA—Expansion module with an LTE modem supporting the frequency bands in Asia, Australia, and New Zealand.

Interfaces

The NFX350 device includes the following network interfaces:

- Eight 1-Gigabit Ethernet RJ-45 ports. The ports follow the naming convention, ge-0/0/*n*, where *n* ranges from 0 to 7. These ports are used for LAN connectivity.
- Eight 10-Gigabit uplink ports that support small form-factor pluggable plus (SFP+) transceivers. The ports follow the naming convention xe-0/0/*n*, where the value of *n* is ranges from 8 to 15. These ports are used as WAN uplink ports.
- A dedicated management port labeled **MGMT** (fxp0) functions as the out-of-band management interface. The fxp0 interface is assigned the IP address 192.168.1.1/24.
- Four static interfaces, sxe-0/0/0, sxe-0/0/1, sxe-0/0/2, and sxe-0/0/3, which connect the Layer 2 data plane (FPC0) to the OVS backplane.

NOTE: By default, all the network ports connect to the Layer 2 data plane.

For the list of supported transceivers for your device, see https://apps.juniper.net/hct/product/#prd=NFX350.

Performance Modes

IN THIS SECTION

NFX350 devices offer various operational modes. You can either select the operational mode of the device from a pre-defined list of modes or specify a custom mode.

- Throughput mode—Provides maximum resources (CPU and memory) for Junos software.

  NOTE: Starting in Junos OS Release 21.1R1, mapping OVS to Layer 3 data plane interface is not supported in throughput mode on NFX350 devices. If the OVS mapping is present in releases prior to Junos OS Release 21.1R1, you must change the mapping before upgrading the device to Junos OS Release 21.1R1 to prevent a configuration commit failure.

- Hybrid mode—Provides a balanced distribution of resources between the Junos software and third-party VNFs.
- Compute mode—Provides minimal resources for Junos software and maximum resources for third-party VNFs.
- Custom mode—Provides an option to allocate resources to Layer 3 data plane and NFV backplane.

NOTE: Compute, hybrid, and throughput modes are supported in Junos OS Release 19.4R1 or later. Custom mode is supported in Junos OS Release 21.1R1 or later.The default mode is throughput in Junos OS Releases prior to 21.4R1. Starting in Junos OS Release 21.4R1, the default mode is compute.

In throughput mode, you must map SR-IOV VF to Layer 3 data plane interfaces on an NFX350 device. Three SR-IOV (VFs) are reserved from each NIC (SXE or HSXE) to support a maximum of 12 Layer 3 Dataplane interfaces. For example:

```
user@host# set vmhost virtualization-options interfaces ge-1/0/1 mapping interface hsxe0
```

NOTE: You cannot create VNFs on Throughput mode.

In hybrid mode and compute mode, you can map Layer 3 data plane interfaces to either SR-IOV or OVS on an NFX350 device. For example:

Map Layer 3 data plane interfaces to either SR-IOV:

```
user@host# set vmhost virtualization-options interfaces ge-1/0/1 mapping interface hsxe0
```

Map Layer 3 Dataplane interfaces to either OVS:

```
user@host# set vmhost virtualization-options interfaces ge-1/0/1
```

In hybrid or compute mode, you can create VNFs using the available CPUs on each mode. You can check the CPU availability by using the `show vmhost mode` command. Each VNF can have maximum of eight user interfaces apart from the two management interfaces. You can attach the VNF interfaces to either OVS or SR-IOV interfaces.

NOTE: You cannot attach single VNF interface to both SR-IOV and OVS. However, you can attach different interfaces from the same VNF to SR-IOV and OVS.

Seven SR-IOV (VFs) are reserved from each NIC (SXE or HSXE) to create VNF interfaces, and supports up to a maximum of 28 SR-IOV VNF interfaces per device. You can view the available free VFs by using the `show system visibility network`.

NOTE: When the mapping to a particular Layer 3 data plane interface changes between SR-IOV NICs (eg, hsxe0 to hsxe1) or from hsxe*x* to OVS or vice versa, then FPC1 restarts automatically.

To change the current mode, run the `request vmhost mode` *`mode-name`* command. The `request vmhost mode ?` command lists only the pre-defined modes such as hybrid, compute, and throughput modes.

Before switching to a mode, issue the `show system visibility cpu` and `show vmhost mode` commands to check the availability of CPUs. When switching between operational modes, ensure that resource and configuration conflicts do not occur.

For example, if you move from compute mode that supports VNFs to throughput mode that does not support VNFs, conflicts occur:

```
user@host# run request vmhost mode throughput
error: Mode cannot be changed; Reason: No CPUs are available for VNFs in the desired mode, but
there is atleast one VNF currently configured
```

If the Layer 3 dataplane is not mapped to SR-IOV, then switching from hybrid or compute mode to throughput mode results in an error.

You can define a custom mode template in Junos configuration by using the following commands:

1. `user@host#` **set vmhost mode custom *custom-mode-name* layer-3-infrastructure cpu count *count***
2. `user@host#` **set vmhost mode custom *custom-mode-name* layer-3-infrastructure memory size *mem-size***
3. `user@host#` **set vmhost mode custom *custom-mode-name* nfv-back-plane cpu count *count***
4. `user@host#` **set vmhost mode custom *custom-mode-name* nfv-back-plane memory size *mem-size***

Starting in Junos OS Release 22.1R1, you can opt to configure the CPU quota for the Layer 3 data plane by using the `set vmhost mode custom` *`custom-mode-name`* `layer-3-infrastructure cpu colocation quota` *`quota-value`* command, where *quota-value* can range from 1 through 99. If you configure `cpu colocation quota`, then the sum total of the CPU quotas of the cpu colocation components must be less than or equal to 100. You must configure `cpu count` using numeric values and not keywords like MIN as MIN can have different values for different components.

The number of CPUs and the specific CPUs (by CPU ID) available for VNF usage in a custom mode is automatically determined based on the `cpu count` and `cpu colocation quota` in the custom mode configuration and the internally fixed CPU allocation for other Juniper system components.

The amount of memory, in terms of 1G units, available for VNF usage in a custom mode is automatically determined based on the custom mode specific memory size configuration and the per-SKU internally fixed memory allocation for other Juniper system components. Note that this number is only an approximate value and the actual maximum memory allocation for VNFs might be less than that.

If you do not configure the memory size for a VNF, then the memory is considered as 1G (default value).

CPU count for both NFV backplane and Layer 3 data plane must be configured in integral numbers.

Memory for Layer 3 data plane and NFV backplane must be specified in Gigabytes in a custom mode. The memory specified through a custom mode is created and backed by 1G huge pages for NFV backplane usage and 2M huge pages for Layer 3 data plane usage. It is recommended to configure NFV backplane memory size in integral numbers, whereas Layer 3 data plane memory can be configured in decimals.

You must configure the CPU count and memory for both Layer 3 data plane and NFV backplane. The CPU and memory resources for the remaining Junos software infrastructure is internally determined by the device.

Custom mode template supports a keyword MIN, which is a device-specific pre-defined value for allocating minimal resources.

flex and *perf* are the custom mode templates that are present in the default Junos configuration.

- *flex* mode—Uses MIN keyword for allocating resources to system components such as Layer 3 data plane and NFV backplane. In this mode, device provides maximum memory and CPUs to third-party VNFs.

  To allocate resources in *flex* mode:

  1. `user@host#` **set vmhost mode custom *custom-mode-name* layer-3-infrastructure cpu count MIN**

  2. `user@host#` **set vmhost mode custom *custom-mode-name* layer-3-infrastructure memory size MIN**

  3. `user@host#` **set vmhost mode custom *custom-mode-name* nfv-back-plane cpu count MIN**

  4. `user@host#` **set vmhost mode custom *custom-mode-name* nfv-back-plane memory size MIN**

  In flex mode, you can configure a maximum of:

  - 8 IPSec VPN tunnels

  - 16 IFL
  - 4 IFD

- *perf* mode—Another example custom mode template that is available in the default Junos configuration.

NOTE: Currently, Layer 3 data plane supports only MIN in a custom mode for both CPU count and memory size.

When the device is in custom mode with MIN keyword, only basic firewall features are supported and you can use Layer 3 data plane only for IPsec termination.

When you allocate CPUs to NFV backplane and Layer 3 data plane, the device allocates full cores. When a full core is allocated to NFV backplane, both the logical CPUs on that hyper-threaded core are allocated to it. However, to get the optimal performance, the device disables one of the logical CPUs and is still counted as 2 CPUs allocated. When full cores are not available, the device allocates individual CPUs from different cores.

While allocating CPUs for VNF usage, the device allocates full cores. Both the logical CPUs on that core are enabled. When full cores are not available, the device allocates individual CPUs from different cores.

NOTE: The requested CPU count and memory should not exceed the total CPU count and memory available on the system.

When the device is operating in custom mode, you can make changes to the custom mode configuration. Reboot the device for the changes to take effect.

Commit checks are performed for basic validation when a custom mode is defined in the configuration and when you change the device mode to a custom mode.

You cannot delete a custom mode configuration when the device is operating in the same mode.

To delete a custom mode configuration when the device is operating in custom mode:

1. Change the device mode from custom mode to another mode.

2. Delete the custom mode configuration.

When the device in a custom mode is downgraded to an image that does not support custom mode, then the default throughput mode is applied on the device.

NOTE: Before performing such an image downgrade process, you must remove all VNF configurations from the device.

When multiple custom modes are configured in the device and when the device is in a custom mode other than the *flex* or *perf* custom mode, which are defined in the factory-default Junos configuration, you cannot reset the device configuration to factory-default configuration. Before you reset such a device to factory-default Junos configuration, you must change the device mode to one of the pre-defined modes such as compute, hybrid, throughput, or to the *flex* or *perf* custom mode that are already defined in the factory-default configuration.

Core to CPU Mapping on NFX350

The following tables list the CPU to core mappings for the NFX350 models:

NFX350-S1								
Core	0	1	2	3	4	5	6	7
CPU	0, 8	1, 9	2, 10	3, 11	4, 12	5, 13	6, 14	7, 15

NFX350-S2												
Core	0	1	2	3	4	5	6	7	8	9	10	11
CPU	0, 12	1, 13	2, 14	3, 15	4, 16	5, 17	6, 18	7, 19	8, 20	9, 21	10, 22	11, 23

NFX350-S3																
Core	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CPU	0, 16	1, 17	2, 18	3, 19	4, 20	5, 21	6, 22	7, 23	8, 24	9, 25	10, 26	11, 27	12, 28	13, 29	14, 30	15, 31

Benefits and Uses

The NFX350 provides the following benefits:

- Highly scalable architecture that supports multiple Juniper VNFs and third-party VNFs on a single device. The modular software architecture provides high performance and scalability for routing, switching, and security enhanced by carrier-class reliability.
- Integrated security, routing, and switching functionality in a single control plane simplifies management and deployment.
- A variety of flexible deployments. A distributed services deployment model ensures high availability, performance, and compliance. The device provides an open framework that supports industry standards, protocols, and seamless API integration.
- Wireless WAN support through the LTE module provides more flexibility in deployments.
- Secure boot feature safeguards device credentials, automatically authenticates system integrity, verifies system configuration, and enhances overall platform security.
- Automated configuration eliminates complex device setup and delivers a plug-and-play experience.
- Increased storage capacity through two external hard disks.

Junos OS Releases Supported on NFX Series Hardware

The Table 2 on page 13 provides details of Junos OS software releases supported on the NFX Series devices.

Table 2: Supported Junos OS Releases on NFX Series Devices

NFX Series Platform	Supported Junos OS Release	Software Package	Software Downloads Page
NFX150	18.1R1 or later	nfx-3 jinstall-host-nfx-3-x86-64-<*release-number*>- secure-signed.tgz install-media-host-usb-nfx-3-x86-64-<*release-number*>- secure.img	NFX150 Software Download Page

Table 2: Supported Junos OS Releases on NFX Series Devices *(Continued)*

NFX Series Platform	Supported Junos OS Release	Software Package	Software Downloads Page
NFX250	17.2R1 through 19.1R1	nfx-2 jinstall-host-nfx-2-flex-x86-64-<*release-number* >-secure-signed.tgz install-media-host-usb-nfx-2-flex-x86-64-<*release-number*>- secure.img	NFX250 Software Download Page
	19.1 R1 or later	nfx-3 jinstall-host-nfx-3-x86-64-<*release-number*>-secure-signed.tgz install-media-host-usb-nfx-3-x86-64-<*release-number*>-secure.img	NFX250 Software Download Page
NFX350	19.4 R1 or later	nfx-3 jinstall-host-nfx-3-x86-64-<*release-number*>-secure-signed.tgz install-media-host-usb-nfx-3-x86-64-<*release-number*>-secure.img	NFX350 Software Download Page

SEE ALSO

NFX250 Overview

Baseboard Management Controller Overview

IN THIS SECTION

A Baseboard Management Controller (BMC) is a specialized micro controller, used for remotely managing and recovering NFX350 device.

NOTE: You cannot access BMC through the Management port or console.

Managing BMC

IN THIS SECTION

Using Junos CLI, you can upgrade BMC firmware, check the BMC firmware version, and perform device power cycle.

Perform Power Cycle of the NFX350 Device

You can perform power cycle of the mainboard CPU or device using Junos CLI.

NOTE: The `power-cycle` command performs only the power cycle of mainboard CPU. You cannot perform the power cycle of BMC and CPLD by using this command.

To perform a power cycle:

```
user@host> request vmhost power-cycle
Power cycle the vmhost ? [yes,no] (yes)
```

BMC Firmware Upgrade

You can upgrade BMC using Junos CLI.

NOTE: For BMC upgrade, the image version must be greater than the currently running BMC firmware version.

- To upgrade BMC after copying the firmware to the device file system:

  NOTE: The NFX350 device remains operational during the BMC upgrade process.

  ```
  user@host> request system firmware upgrade jfirmware bmc file BMC-firmware-path
  ```

  For example,

  ```
  user@host> request system firmware upgrade jfirmware bmc file /var/public/nfx-3-
  jfirmware-19.4R1.5.tgz
  Validated nfx-3-jfirmware-19.4R1.5.tgz
  BMC firmware upgrade initiated
  Check progress using "request system firmware upgrade jfirmware
  ..<type>.. progress“
  ```

  After the BMC firmware upgrade is completed, BMC reboots automatically.

- To check the status of BMC firmware upgrade:

  ```
  user@host> request system firmware upgrade jfirmware bmc progress
  BMC upgrade in progress
  ```

After the firmware is successfully loaded, `BMC upgrade successful` message is displayed.

View BMC Firmware Version

To view the BMC firmware version:

```
user@host> show system inventory firmware | match BMC
BMC Version: 00.06_00.02
```

NFX Product Compatibility

IN THIS SECTION

Hardware Compatibility

To obtain information about the components that are supported on your devices, and special compatibility guidelines with the release, see the Hardware Guide and the Interface Module Reference for the product.

To determine the features supported on NFX Series devices in this release, use the Juniper Networks Feature Explorer, a Web-based application that helps you to explore and compare Junos OS feature information to find the right software release and hardware platform for your network. Find Feature Explorer at: https://pathfinder.juniper.net/feature-explorer/.

Hardware Compatibility Tool

For a hardware compatibility matrix for optical interfaces and transceivers supported across all platforms, see the Hardware Compatibility Tool.

Software Version Compatibility

This section lists the vSRX Virtual Firewall and Cloud CPE Solution software releases that are compatible with the Junos OS releases on the NFX Series devices.

NOTE:

- Starting in Junos OS Release 18.1R1, NFX Series devices support the same version of platform software and vSRX Virtual Firewall. For example, see Table 3 on page 18.

NFX250 Software Version Compatibility

This section lists the vSRX Virtual Firewall and CloudCPE Solution software releases that are compatible with the Junos OS releases on the NFX250 devices:

Table 3: Software Compatibility Details with vSRX Virtual Firewall and Cloud CPE Solution

NFX250 Junos OS Release	vSRX Virtual Firewall	Cloud CPE Solution
15.1X53-D40.3	15.1X49-D40.6	Cloud CPE Solution 2.0
15.1X53-D41.6	15.1X49-D40.6	Cloud CPE Solution 2.1
15.1X53-D102.2	15.1X49-D61	Cloud CPE Solution 3.0
15.1X53-D47.4	15.1X49-D100.6	Cloud CPE Solution 3.0.1
15.1X53-D490	15.1X49-D143	Cloud CPE Solution 4.0
15.1X53-D495	15.1X49-D160	Cloud CPE Solution 4.1
15.1X53-D496	15.1X49-D170	Cloud CPE Solution 4.1
15.1X53-D45.3	15.1X49-D61	Not applicable
17.2R1	15.1X49-D78.3	Not applicable

Table 3: Software Compatibility Details with vSRX Virtual Firewall and Cloud CPE Solution *(Continued)*

NFX250 Junos OS Release	vSRX Virtual Firewall	Cloud CPE Solution
17.3R1	15.1X49-D78.3	Not applicable
17.4R1	15.1X49-D78.3	Not applicable
15.1X53-D471	15.1X49-D143	Not applicable
18.1R1	18.1R1	Not applicable
18.1R2	18.1R2	Not applicable
18.1R3	18.1R3	Not applicable
18.2R1	18.2R1	Not applicable
18.3R1	18.3R1	Not applicable
18.4R1	18.4R1	Not applicable

2

CHAPTER

Initial Configuration

IN THIS CHAPTER

Initial Configuration on NFX350 Devices

IN THIS SECTION

Factory Default Settings

The NFX350 is shipped with the following factory default settings:

Table 4: Security Policies

Source Zone	Destination Zone	Policy Action
trust	trust	permit
trust	untrust	permit

Table 5: Interfaces

Port Label	Interface	Security Zone	DHCP State	IP Address
0/0 to 0/7	ge-0/0/0 to ge-0/0/7	trust	server	192.168.2.1/24
0/8 to 0/15	xe-0/0/8 to xe-0/0/15	untrust	client	ISP assigned
MGMT	fxp0	N/A	N/A	192.168.1.1/24

The device is shipped with the following services enabled in the default security policy: DHCP, HTTP, HTTPS, and SSH.

To provide secure traffic, a basic set of screens are configured on the untrust zone.

Enabling Basic Connectivity

1. Ensure that the device is powered on.
2. Connect to the console port:
    a. Plug one end of the Ethernet cable into the console port on your device.

    b. Connect the other end of the Ethernet cable to the RJ-45 to DB-9 serial port adapter shipped with your device.

    c. Connect the RJ-45 to DB-9 serial port adapter to the serial port on the management device. Use the following values to configure the serial port:

    Bits per second—9600; Parity—None; Data bits—8; Stop bits—1; Flow control—None.

    NOTE: Alternately, you can use the USB cable to connect to the mini-USB console port on the device. To use the mini-USB console port, you must download the USB driver from the following page and install the driver on the management device: https://www.juniper.net/support/downloads/junos.html
3. Use any terminal emulation program such as HyperTerminal to connect to the device console. The CLI displays a login prompt.
4. Log in as **root**. If the software completes booting before you connect to the console, you might need to press the Enter key for the prompt to appear.

```
login: root
```

5. Start the CLI.

```
root@:~ # cli
root@>
```

6. Enter configuration mode.

```
root@> configure
[edit]
root@#
```

7. Change the password for the root administration user account.

```
[edit]
root@# set system root-authentication plain-text-password
New password: password
Retype new password: password
```

8. Enable SSH service for the root user.

```
[edit]
root@# set system services ssh root-login allow
```

9. (Optional) Enable Internet connection for the devices connected on LAN by setting the DNS IP.

```
[edit]
root@# set access address-assignment pool junosDHCPPool family inet dhcp-attributes name-
server dns-server-ip
```

10. Commit the configuration.

```
[edit]
root@# commit
```

Establishing the Connection

1. Connect the device to the ISP by connecting one of the WAN ports (0/8 through 0/15) to the ISP. The device is assigned an IP address by the ISP through DHCP.

   NOTE: For information about interfaces, see Table 5 on page 21.

2. Connect the laptop to one of the front panel LAN ports (0/0 to 0/7). The laptop is assigned an IP address by the DHCP server running on the device.
3. Open a browser window on your laptop, navigate to https://www.juniper.net, and verify your connectivity.

Zero Touch Provisioning on NFX Series Devices

IN THIS SECTION

Understanding Zero Touch Provisioning

Zero Touch Provisioning (ZTP) allows you to provision and configure an NFX Series device in your network automatically, with minimal manual intervention. ZTP allows you to make configuration changes or software upgrades without logging into the device. NFX Series devices support ZTP with Sky Enterprise, which is a cloud-based network management application. For more information on Sky Enterprise, see Sky Enterprise Documentation.

The initial provisioning process involves the following components:

- NFX Series device—Sends requests to Juniper's Redirect Server.
- Redirect server—Provides authentication and authorization for the devices in a network to access their assigned central servers for the boot images and initial configuration files. The redirect server resides at Juniper Networks.

  Connectivity to the redirect server can be through IPv4 or IPv6 network. Depending on the source address, the redirect server redirects the ZTP to the corresponding Central Server with IPv4 or IPv6 address.

  The NFX Series device is shipped with a factory default configuration. The factory default configuration includes the URL of the redirect server, that is used to connect to the central servers by using a secure encrypted connection.
- Central server—Manages the network and the NFX Series devices located remotely. The central server is located at a central geographical location. Alternately, you can use Contrail Service Orchestration (CSO) along with Sky Enterprise. CSO deploys the network services and Sky Enterprise manages the devices in the network.

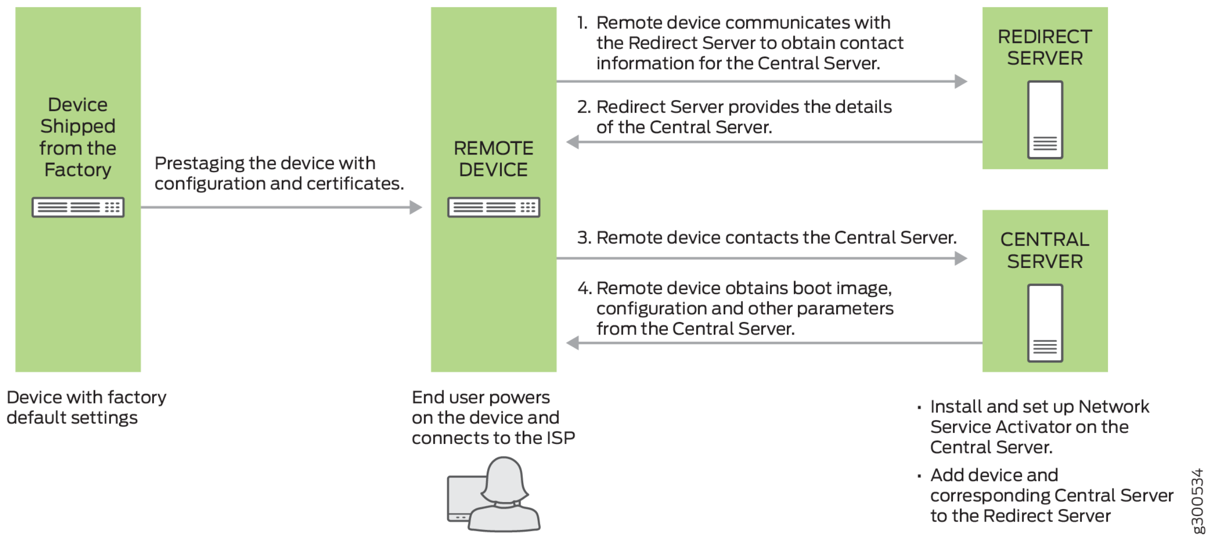
Pre-staging an NFX Series Device

Prestaging is an optional step for the device to by-pass Juniper's Redirect Server and to connect to a customer specific Redirect Server or a Regional Server for authentication and authorization in the network. Prestaging involves copying and applying certificates and customer specific configuration from a specific directory in the device before the device is shipped to the customer site for installation.

The customer specific resources are stored internally. When the device boots up with the factory default configuration, the prestage resources are copied and the configuration is applied on the device.

Figure 4 on page 25 illustrates the workflow of prestaging the NFX Series devices.

Figure 4: Workflow for Prestaging an NFX Series Device


Device Shipped from the Factory
Prestaging the device with configuration and certificates.
REMOTE DEVICE
1. Remote device communicates with the Redirect Server to obtain contact information for the Central Server.
2. Redirect Server provides the details of the Central Server.
REDIRECT SERVER
3. Remote device contacts the Central Server.
4. Remote device obtains boot image, configuration and other parameters from the Central Server.
CENTRAL SERVER
Device with factory default settings
End user powers on the device and connects to the ISP
• Install and set up Network Service Activator on the Central Server.
• Add device and corresponding Central Server to the Redirect Server
g300534

The prestage workflow proceeds as follows:

1. The device is shipped from the factory with the factory default configuration.

2. To prestage the device, the customer specific resources such as certificates and configuration are copied to the device by a user or ISP.

   To add the prestage configuration and certificates, run:

   ```
   user@host>request system phone-home pre-stage add configuration file
   user@host>request system phone-home pre-stage add certificates file/files
   ```

3. After the device is prestaged, the device is shipped to the end user.

4. The end user powers on the remote device and connects the device to the ISP by connecting one of the WAN ports (0/12 and 0/13) to the ISP. For more information, see *Initial Configuration on NFX250 NextGen Devices* .

5. The device applies the prestage configuration and uses the certificates to authenticate the customer specific Redirect Server or Regional Server.

6. The Redirect Server or Regional Server sends the corresponding Central Server information to the device.

7. The device sends a provisioning request to the Central Server. The Central Server responds with the boot image and the configuration that is provisioned on the Central Server for that particular device.

8. The device fetches the boot image and configuration file from the Central Server.

9. The device upgrades to the boot image and applies the configuration to start the services and become operational.

To delete the prestage configuration and certificates, run:

```
user@host>request system phone-home pre-stage delete configuration file
user@host>request system phone-home pre-stage delete certificate all | file
user@host>request system phone-home pre-stage delete all
```

To verify the prestage configuration and certificates, run:

```
user@host>show system phone-home pre-stage configuration
user@host>show system phone-home pre-stage certificate
user@host>show system phone-home pre-stage
```

The prestage resources are not deleted when you upgrade the image by using the `request system software add image` command or when you zeroize the device by using the `request system zeroize` command.

The default configuration for phone-home is:

```
user@jdm# set system phone-home server https://redirect.juniper.net
user@jdm# set system phone-home upgrade-image-before-configuration
```

To enable trace operation:

```
user@jdm# set system phone-home traceoptions file file-name size file-size
user@jdm# set system phone-home traceoptions flag [all | config | function | misc | socket |
state-machine]
```

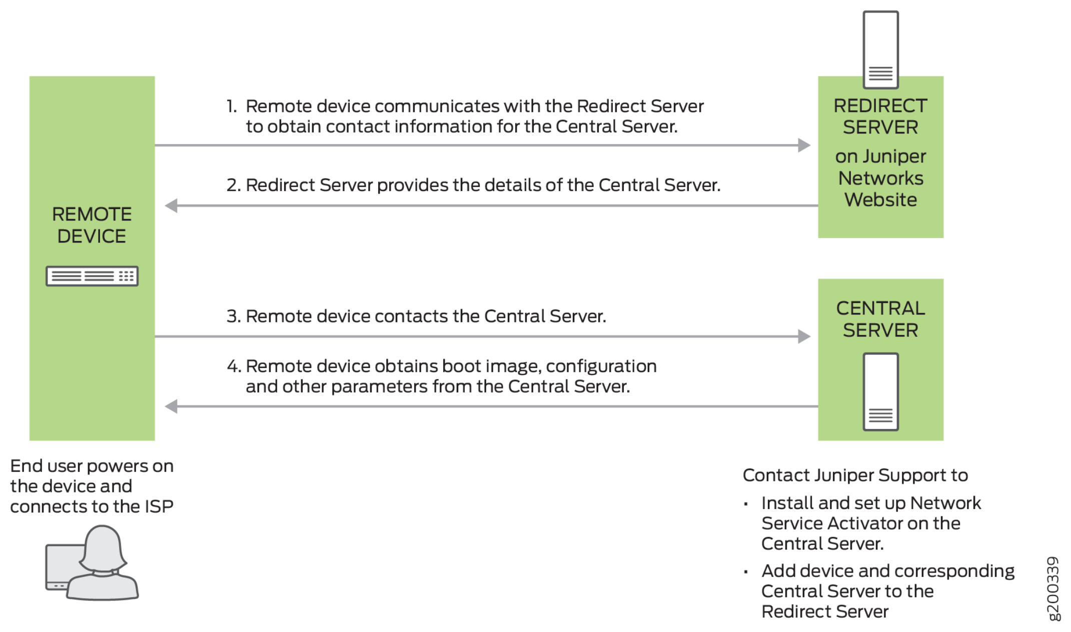
To disable trace operation:

```
user@jdm# set system phone-home traceoptions no-remote-trace
```

Provisioning an NFX Series Device

Figure 5 on page 27 illustrates the workflow of the initial provisioning of NFX Series devices.

Figure 5: Workflow for Initial Provisioning of an NFX Series Device



REMOTE DEVICE
REDIRECT SERVER on Juniper Networks Website
CENTRAL SERVER
1. Remote device communicates with the Redirect Server to obtain contact information for the Central Server.
2. Redirect Server provides the details of the Central Server.
3. Remote device contacts the Central Server.
4. Remote device obtains boot image, configuration and other parameters from the Central Server.
End user powers on the device and connects to the ISP
Contact Juniper Support to
• Install and set up Network Service Activator on the Central Server.
• Add device and corresponding Central Server to the Redirect Server
g200339

NOTE: Contact Juniper Support to add the device and the corresponding central server to the redirect server.

The provisioning workflow proceeds as follows:

1. The end user powers on the remote device, and connects the remote device to the ISP through the WAN ports.

2. The remote device transmits its X.509 certificate and fully qualified domain name (FQDN) as a provisioning request to the redirect server.

3. The redirect server searches its data store for the central server that an administrator has specified for the remote device, and confirms that the remote device's request corresponds to the X.509 certificate specified for the server.

4. The redirect server sends contact information for the central server to the remote device.

5. The remote device sends a request to the central server for the URL of the boot image and the location of the initial configuration file. The central server responds with the requested information.

6. The remote device fetches the boot image and configuration file from the central server.

7. The remote device upgrades to the boot image (if the boot image is different from the image running on the NFX Series device), and applies the configuration to start the services and become operational.

Provisioning an NFX Series Device Using Sky Enterprise

Figure 5 on page 27 illustrates the workflow of the initial provisioning of NFX Series devices using Sky Enterprise.

The provisioning workflow proceeds as follows:

1. The end user powers on the remote device, and connects the remote device to the ISP through the WAN ports.

2. The NFX Series device transmits its X.509 certificate and fully qualified domain name (FQDN) as a provisioning request to the Redirect Server.

3. The Redirect Server connects the device to Sky Enterprise.

4. Click the link in the authorization e-mail that you receive from Sky Enterprise. Alternately, you can use the Sky Enterprise application to authorize the device.

5. The NFX Series device registers with Sky Enterprise.

6. The initial configuration of the device begins. The initial configuration process takes about 60 seconds.

3

CHAPTER

Generating YANG Files

IN THIS CHAPTER

YANG files on NFX350 Devices

IN THIS SECTION

Understanding YANG on NFX350 Devices

YANG is a standards-based, extensible data modeling language that is used to model the configuration and operational state data, remote procedure calls (RPCs), and server event notifications of network devices. The NETMOD working group in the IETF originally designed YANG to model network management data and to provide a standard for the content layer of the Network Configuration Protocol (NETCONF) model. However, YANG is protocol independent, and YANG data models can be used independent of the transport or RPC protocol and can be converted into any encoding format supported by the network configuration protocol.

Juniper Networks provides YANG modules that define the Junos OS configuration hierarchy and operational commands and Junos OS YANG extensions. You can generate the modules on the device running Junos OS.

YANG uses a C-like syntax, a hierarchical organization of data, and provides a set of built-in types as well as the capability to define derived types. YANG stresses readability, and it provides modularity and flexibility through the use of modules and submodules and reusable types and node groups.

A YANG module defines a single data model and determines the encoding for that data. A YANG module defines a data model through its data, and the hierarchical organization of and constraints on that data. A module can be a complete, standalone entity, or it can reference definitions in other modules and submodules as well as augment other data models with additional nodes.

A YANG module defines not only the syntax but also the semantics of the data. It explicitly defines relationships between and constraints on the data. This enables you to create syntactically correct configuration data that meets constraint requirements and enables you to validate the data against the model before uploading it and committing it on a device.

YANG uses modules to define configuration and state data, notifications, and RPCs for network operations in a manner similar to how the Structure of Management Information (SMI) uses MIBs to model data for SNMP operations. However, YANG has the benefit of being able to distinguish between operational and configuration data. YANG maintains compatibility with SNMP's SMI version 2 (SMIv2), and you can use libsmi to translate SMIv2 MIB modules into YANG modules and vice versa. Additionally, when you cannot use a YANG parser, you can translate YANG modules into YANG Independent Notation (YIN), which is an equivalent XML syntax that can be read by XML parsers and XSLT scripts.

For information about YANG, see RFC 6020, *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*, and related RFCs.

For more information, see YANG Modules Overview, Using Juniper Networks YANG Modules, and show system schema.

Generating YANG Files

You can generate YANG files for JCP on NFX350 devices.

To generate YANG files for JCP:

1. Log in to the NFX device using SSH or console:

```
login: root
```

2. Start the CLI:

```
root@:~# cli
{master:0}
root>
```

3. Create a temporary directory to store the generated YANG files:

```
{master:0}
root> file make-directory /var/public/yang_files
{master:0}
root> file list /var/public/yang_files
/var/public/yang_files:
{master:0}
root>
```

4. Generate YANG files for JCP:

```
{master:0}
root> show system schema module all format yang output-directory /var/public/yang_files
```

5. Verify whether YANG files are generated in the specified target directory:

```
{master:0}
root> file list /var/public/yang_files
/var/public/yang_files:

junos-common-types@2019-01-01.yang
junos-nfx-conf-access-profile@2019-01-01.yang
junos-nfx-conf-access@2019-01-01.yang
junos-nfx-conf-accounting-options@2019-01-01.yang
junos-nfx-conf-applications@2019-01-01.yang
...Output truncated...
```

6. Copy the generated JCP YANG files from the NFX device to the YANG based tools or orchestrators by using the `scp` or `file copy` command.

4

CHAPTER

Configuring Interfaces

IN THIS CHAPTER

Configuring the In-Band Management Interface on NFX350

In in-band management, you configure a network interface as a management interface and connect it to the management device. You can configure any of the ge-1/0/x ports, where x ranges from 0 to 4, as in-band management interfaces.

To configure in-band management:

1. Log in to the JCP CLI and enter configuration mode:

```
root@host% cli
root@host> configure
```

2. Configure VLAN tagging:

```
root@host# set interfaces ge-1/0/x vlan-tagging
root@host# set interfaces ge-1/0/x unit n vlan-id mgmt-vlan-id
root@host# set interfaces ge-1/0/x unit n family inet address address/prefix-length
```

To configure a LAN port for in-band management:

1. Configure the management VLAN:

```
root@host# set vlans mgmt-vlan vlan-id vlan-id
```

2. Add the physical network interface and the service interface as members of the VLAN:

```
root@host# set interfaces ge-0/0/x unit 0 family ethernet-switching vlan members mgmt-vlan
```

   Where **x** ranges from 0 to 7.

```
root@host# set interfaces sxe-0/0/x unit 0 family ethernet-switching vlan members mgmt-vlan
```

   Where **x** ranges from 0 to 3.

NOTE: If the device is in throughput mode, you must map ge-1/0/*x* to sxe-0/0/*x* by using the `set vmhost virtualization-options interfaces ge-1/0/x mapping interface hsxex` command. If the device is in hybrid or compute mode, you can map ge-1/0/*x* to OVS by using the `set vmhost virtualization-options interfaces ge-1/0/x` command or map to SR-IOV by using the `set vmhost virtualization-options interfaces ge-1/0/x mapping interface hsxex` command. After you change the the mapping, FPC1 restarts automatically.

ADSL2 and ADSL2+ Interfaces on NFX350 Devices

IN THIS SECTION

ADSL Interface Overview

IN THIS SECTION

Asymmetric digital subscriber line (ADSL) technology is part of the *x*DSL family of modem technologies that use existing twisted-pair telephone lines to transport high-bandwidth data. ADSL lines connect service provider networks and customer sites over the "last mile" of the network—the loop between the service provider and the customer site.

ADSL transmission is asymmetric because the downstream bandwidth is typically greater than the upstream bandwidth. The typical bandwidths of ADSL2 and ADSL2+ circuits are defined in Table 6 on page 36.

Table 6: Standard Bandwidths of DSL Operating Modes

Operating Modes	Upstream	Downstream
ADSL2	1—1.5 Mbps	12—14 Mbps
ADSL2+	1—1.5 Mbps	24—25 Mbps

ADSL2 and ADSL2+ support the following standards:

- LLCSNAP bridged 802.1q
- VC MUX bridged

Supported security devices with xDSL SFP can use PPP over Ethernet(PPPoE) to connect through ADSL lines only.

ADSL2 and ADSL2+

The ADSL2 and ADSL2+ standards were adopted by the ITU in July 2002. ADSL2 improves the data rate and reach performance, diagnostics, standby mode, and interoperability of ADSL modems.

ADSL2+ doubles the possible downstream data bandwidth, enabling rates of 20 Mbps on telephone lines shorter than 5000 feet (1.5 km).

ADSL2 uses seamless rate adaptation (SRA) to change the data rate of a connection during operation with no interruptions or bit errors. The ADSL2 transceiver detects changes in channel conditions—for example, the failure of another transceiver in a multicarrier link—and sends a message to the transmitter to initiate a data rate change. The message includes data transmission parameters such as the number of bits modulated and the power on each channel. When the transmitter receives the information, it transitions to the new transmission rate.

Example: Configuring ADSL SFP Interface on NFX350 Devices

IN THIS SECTION

Requirements

This example uses the following hardware and software components:

- NFX350 device running the Junos OS Release 19.4R1 version, which supports the reoptimized architecture.

Overview

In this example, you are configuring ADSL SFP interface on an NFX350 device with the following configurations:

- Physical interface **- ge-0/0/11**
- ADSL SFP options **- vpi3, vci34, and encap llcsnap-bridged-802dot1q**

NOTE: Ensure that connectivity to the host is not lost during the configuration process.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To configure ADSL SFP interfaces on NFX350 devices:

1. Connect to the host.

```
user@host> configure
[edit]
user@host#
```

2. Allocate hugepages:

```
user@host# run show system visibility memory
user@host# set system memory hugepages size 1024 count 5
```

Reboot the device.

3. Configure virtual interfaces:

```
user@host# set vmhost virtualization-options interfaces ge-1/0/3
user@host# set vmhost virtualization-options interfaces ge-1/0/4
user@host# commit
```

4. Create VLANs using VLAN IDs:

```
user@host# set vlans vlan100 vlan-id 100
user@host# set vlans vlan101 vlan-id 101
user@host# set vlans vlan200 vlan-id 200
user@host# set vlans vlan50 vlan-id 50
```

5. Configure interfaces:

```
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan50
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan100
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan101
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan200
user@host# set interfaces ge-0/0/11 native-vlan-id 50
user@host# set interfaces ge-0/0/11 dsl-sfp-options adsl-options vpi 3
user@host# set interfaces ge-0/0/11 dsl-sfp-options adsl-options vci 32
user@host# set interfaces ge-0/0/11 dsl-sfp-options adsl-options encap llcsnap-
bridged-802dot1q
user@host# set interfaces ge-0/0/11 unit 0 family ethernet-switching interface-mode trunk
```

```
user@host# set interfaces ge-0/0/11 unit 0 family ethernet-switching vlan members vlan50
user@host# set interfaces ge-0/0/11 unit 0 family ethernet-switching vlan members vlan101
user@host# set interfaces ge-1/0/3 vlan-tagging
user@host# set interfaces ge-1/0/3 unit 0 vlan-id 50
user@host# set interfaces ge-1/0/3 unit 0 family inet address 130.1.1.11/24
user@host# set interfaces ge-1/0/3 unit 0 family inet6 address 2001::1/64
```

6. Commit the configuration.

```
user@host# commit and-quit
user@host> exit
```

Results

VDSL2 Interfaces on NFX350 Devices

IN THIS SECTION

VDSL Interface Overview

IN THIS SECTION

Very-high-bit-rate digital subscriber line (VDSL) technology is part of the *x*DSL family of modem technologies that provide faster data transmission over a single flat untwisted or twisted pair of copper wires. The VDSL lines connect service provider networks and customer sites to provide high bandwidth applications (triple-play services) such as high-speed Internet access, telephone services like VoIP, high-definition TV (HDTV), and interactive gaming services over a single connection.

VDSL2 is an enhancement to G.993.1 (VDSL) and permits the transmission of asymmetric (half-duplex) and symmetric (full-duplex) aggregate data rates up to 100 Mbps on short copper loops using a bandwidth up to 17 MHz. The VDSL2 technology is based on the ITU-T G.993.2 (VDSL2) standard, which is the International Telecommunication Union standard describing a data transmission method for VDSL2 transceivers.

The VDSL2 uses discrete multitone (DMT) modulation. DMT is a method of separating a digital subscriber line signal so that the usable frequency range is separated into 256 frequency bands (or channels) of 4.3125 KHz each. The DMT uses the Fast Fourier Transform (FFT) algorithm for demodulation or modulation for increased speed.

VDSL2 interface supports Packet Transfer Mode (PTM). The PTM mode transports packets (IP, PPP, Ethernet, MPLS, and so on) over DSL links as an alternative to using Asynchronous Transfer Mode (ATM). PTM is based on the Ethernet in the First Mile (EFM) IEEE802.3ah standard.

VDSL2 provides backward compatibility with ADSL2 and ADSL2+ because this technology is based on both the VDSL1-DMT and ADSL2/ADSL2+ recommendations.

VDSL2 Vectoring Overview

Vectoring is a transmission method that employs the coordination of line signals that reduce crosstalk levels and improve performance. It is based on the concept of noise cancellation, like noise-cancelling headphones. The ITU-T G.993.5 standard, "Self-FEXT Cancellation (Vectoring) for Use with VDSL2 Transceivers," also known as G.vector, describes vectoring for VDSL2.

The scope of Recommendation ITU-T G.993.5 is specifically limited to the self-FEXT (far-end crosstalk) cancellation in the downstream and upstream directions. The FEXT generated by a group of near-end transceivers and interfering with the far-end transceivers of that same group is canceled. This cancellation takes place between VDSL2 transceivers, not necessarily of the same profile.

VDSL2 Network Deployment Topology

In standard telephone cables of copper wires, voice signals use only a fraction of the available bandwidth. Like any other DSL technology, the VDSL2 technology utilizes the remaining capacity to carry the data and multimedia on the wire without interrupting the line's ability to carry voice signals.

This example depicts the typical VDSL2 network topology deployed using NFX device.

A VDSL2 link between network devices is set up as follows:

1. Connect an end-user device such as a LAN, hub, or PC through an Ethernet interface to the customer premises equipment (CPE) (for example, an NFX device).

2. Connect the CPE to a DSLAM.

3. The VDSL2 interface uses either Gigabit Ethernet or fiber as second mile to connect to the Broadband Remote Access Server (B-RAS) as shown in Figure 6 on page 41.

4. The ADSL interface uses either Gigabit Ethernet (in case of IP DSLAM] as the “second mile” to connect to the B-RAS or OC3/DS3 ATM as the second mile to connect the B-RAS as shown in Figure 7 on page 42.

   NOTE: The VDSL2 technology is backward compatible with ADSL2 and ADSL2+. VDSL2 provides an ADSL2 and ADSL2+ interface in an ATM DSLAM topology and provides a VDSL2 interface in an IP or VDSL DSLAM topology.

   The DSLAM accepts connections from many customers and aggregates them to a single, high-capacity connection to the Internet.

Figure 6 on page 41 shows a typical VDSL2 network topology.

Figure 6: Typical VDSL2 End-to-End Connectivity and Topology Diagram

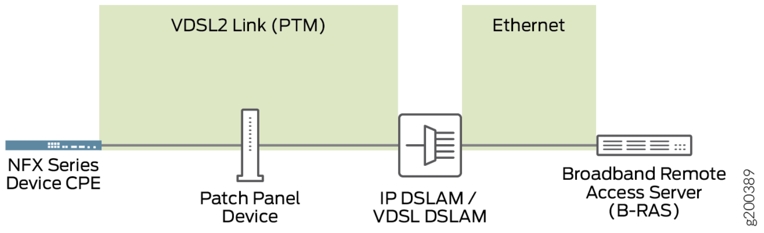
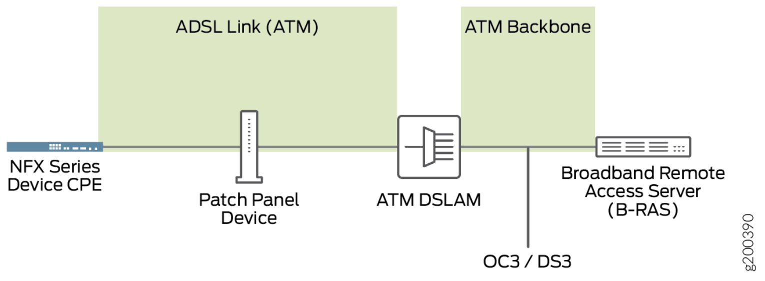


Figure 7 on page 42 shows a backward-compatible ADSL topology using ATM DSLAM.

Figure 7: Backward-Compatible ADSL Topology (ATM DSLAM)



VDSL2 Interface Support on NFX350 Devices

IN THIS SECTION

The VDSL2 interface is supported on the NFX Series devices listed in Table 7 on page 42. (Platform support depends on the Junos OS release in your installation.)

Table 7: VDSL2 Annex A and Annex B Features

Features	POTS
Devices	CPE-SFP-VDSL2
Supported annex operating modes	Annex A and Annex B*
Supported Bandplans	Annex A 998 Annex B 997 and 998
Supported standards	ITU-T G.993.2 and ITU-T G.993.5 (VDSL2)

Table 7: VDSL2 Annex A and Annex B Features *(Continued)*

Features	POTS
Used in	North American network implementations
ADSL backward compatibility	G 992.3 (ADSL2) G 992.5 (ADSL2+)

NOTE: Only one CPE-SFP-VDSL2 device is supported at a time.

VDSL2 Interface Compatibility with ADSL Interfaces

VDSL2 interfaces on NFX Series devices are backward compatible with most ADSL2 and ADSL2+ interface standards. The VDSL2 interface uses Ethernet in the First Mile (EFM) mode or Packet Transfer Mode (PTM) and uses the named interface ge-0/0/10 and ge-0/0/11.

NOTE:

- The VDSL2 interface has backward compatibility with ADSL2 and ADSL2+.
- It requires around 60 seconds to switch from VDSL2 to ADSL2 and ADSL2+ or from ADSL2 and ADSL2+ to VDSL2 operating modes.

VDSL2 Interfaces Supported Profiles

A profile is a table that contains a list of pre-configured VDSL2 settings. Table 8 on page 43 lists the different profiles supported on the VDSL2 interfaces and their properties.

Table 8: Supported Profiles on the VDSL2 Interfaces

Profiles	Data Rate
8a	50
8b	50

Table 8: Supported Profiles on the VDSL2 Interfaces *(Continued)*

Profiles	Data Rate
8c	50
8d	50
12a	68
12b	68
17a	100
Auto	Negotiated (based on operating mode)

Example: Configuring VDSL SFP Interface on NFX350 Devices

IN THIS SECTION

Requirements

This example uses the following hardware and software components:

- NFX350 device running Junos OS Release 20.2R1.

Overview

In this example, you are configuring VDSL SFP interface on an NFX350 device with the following configurations:

- Physical interface - **ge-0/0/11**
- VDSL SFP options **- profile auto and carrier auto**

To configure VDSL SFP interface on NFX250 devices, you must configure JDM, vSRX Virtual Firewall, and vJunos0.

NOTE: Ensure that connectivity to the host is not lost during the configuration process.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To configure VDSL SFP interfaces on NFX350 devices:

1. Connect to the host.

```
user@host> configure
[edit]
user@host#
```

2. Configure virtual interfaces:

```
user@host# set vmhost virtualization-options interfaces ge-1/0/3
user@host# set vmhost virtualization-options interfaces ge-1/0/4
user@host# commit
```

3. Create VLANs using VLAN IDs:

```
user@host# set vlans vlan100 vlan-id 100
user@host# set vlans vlan101 vlan-id 101
user@host# set vlans vlan200 vlan-id 200
user@host# set vlans vlan50 vlan-id 50
```

4. Configure interfaces:

```
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan50
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan100
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan101
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan200
user@host# set interfaces ge-0/0/11 native-vlan-id 50
user@host# set interfaces ge-0/0/11 dsl-sfp-options vdsl-options profile auto
user@host# set interfaces ge-0/0/11 dsl-sfp-options vdsl-options carrier auto
user@host# set interfaces ge-0/0/11 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces ge-0/0/11 unit 0 family ethernet-switching vlan members vlan50
user@host# set interfaces ge-0/0/11 unit 0 family ethernet-switching vlan members vlan101
user@host# set interfaces ge-1/0/3 vlan-tagging
user@host# set interfaces ge-1/0/3 unit 0 vlan-id 50
user@host# set interfaces ge-1/0/3 unit 0 family inet address 130.1.1.11/24
user@host# set interfaces ge-1/0/3 unit 0 family inet6 address 2001::1/64
```

5. Commit the configuration.

```
user@host# commit and-quit
user@host> exit
```

Results

RELATED DOCUMENTATION

NFX250 Overview

JDM Architecture Overview

JDM CLI Overview

5

CHAPTER

Configuring Solid State Disk

IN THIS CHAPTER

Configuring the Solid State Disk on NFX350 Device

NFX350 devices support two external Solid State Disks (SSDs) for customer data, logging and VNF data. These two SSDs work independently.

You can configure the Solid State Disk (SSD) on the NFX350 device for local persistent storage.

NOTE:

- It is highly recommended not to use third-party external SSDs on NFX350 devices.
- You must plug in the external SSD when the NFX350 device is in powered off state. You can use the SSD only after you initialize and add the SSD to the NFX350 device.
- When an external SSD is initialized for a particular NFX350 device, you can use that SSD with that particular device only.
- If an external SSD is present during the installation (USB/clean-install/zeroize), then the SSD is initialized to be used with the NFX350 current device. If the external SSD is not present during the installation and is inserted later, then use the `request vmhost storage external-disk-1 initialize [force]` commands to initialize the external disk.

To initialize and add an SSD:

1. Log in to the JCP CLI and enter configuration mode:

```
root@host% cli
```

2. Initialize an SSD:

```
root@host> request vmhost storage external-ssd initialize slot 0 public-dir-name public-disk0
```

   There are two slots for SSD, slot 0 and slot 1. The disk name should be either *public-disk0* or *public-disk1* based on the SSD slot.

   NOTE: You can store VNF images in the **/var/public-disk1** folder.

3. Add an SSD:

```
root@host> request vmhost storage external-ssd add slot 0
```

4. Verify whether the SSD is added:

```
root@host> show virtual-network-functions storage
Filesystem              Size   Avail   Used   Use%
/var/public             19G    7.4G    11G    60%
/var/public-disk1       734G   697G    69M    1%
```

   In the output message, `/var/public` shows details of the internal SSD and `/var/public-disk1` shows details of the external SSD that is plugged in slot1.

5. (Optional) Re-initialize the SSD:

```
root@host> request vmhost storage external-ssd initialize slot 1 public-dir-name public-disk1
force
```

   NOTE: Upgrade using USB re-initializes and adds the external SSD.

To remove the SSD:

1. Remove the SSD:

```
root@host> request vmhost storage external-ssd remove slot 0
```

2. Power off the device.

3. Remove the SSD.

NOTE: SSD is not formatted when you remove it. To erase the data before removing the SSD, re-initialize the SSD.

6

CHAPTER

Configuring USB Pass-Through on NFX Series Devices

IN THIS CHAPTER

Supporting File Transfer from USB on NFX Series Devices

Starting from Junos OS Release 21.1R1, you can transfer VNF images, NFX software, or any user scripts from USB to NFX devices by enabling the USB pass-through feature. By default, the USB pass-through feature is disabled.

NOTE: Built-in LTE functionality does not work after you enable the USB pass-through feature.

To enable USB pass-through to Junos and mount a USB:

1. Log in to the JCP CLI and enter configuration mode:

   ```
   root@host% cli
   root@host> configure
   ```

2. Configure the USB pass-through feature:

   ```
   root@host# set system services usb-pass-through
   root@host# commit
   ```

3. Restart the device to enable the USB pass-through feature.
4. Verify whether the USB pass-through feature is enabled:

   ```
   root@host# run show system services usb-pass-through
   ```

   USB pass through Information

   Mode: Enabled

5. Mount a USB device on an NFX device. This is helpful if network connectivity is unavailable and you need to copy files to or from the device.

   NOTE: It is recommended to use a USB with the FAT32 format.

Enter the shell prompt as a root user:

```
root@host>
root@host> start shell user root
Password:
root@host%
```

6. Before inserting the USB device, perform the following:

```
root@host:~ # ls -l /dev/da*
```

ls: No match.

7. Insert the USB drive in the USB port. An output similar to the following is displayed:

```
root@% umass1: TOSHIBA TransMemory, rev 2.00/1.00, addr 3
        da2 at umass-sim1 bus 1 target 0 lun 0
        da2: <TOSHIBA TransMemory 5.00> Removable Direct Access SCSI-0 device
        da2: 40.000MB/s transfers
        da2: 983MB (2013184 512 byte sectors: 64H 32S/T 983C)

        root@:~ # ls -l /dev/da*
        crw-r-----  1 root  operator  0x93 Feb  4 04:22 /dev/da0
        crw-r-----  1 root  operator  0x94 Feb  4 04:22 /dev/da0p1
```

In the sample output, `/dev/da0p1` is the USB drive. If the device supports multiple USBs, use the right file that is corresponding to the attached USB. If the console session is not available while inserting the USB, check the **messages** var log file for logs related to `da` (for example, `show log messages | match da`). It logs the same four lines as shown on console if the USB is inserted.

8. Create a directory for the USB drive to mount to:

```
root@host% mkdir /var/tmp/usb
```

9. Mount the USB drive to the **/var/tmp/usb** directory:

NOTE: **ls /var/tmp/usb** directory shows all files that are present in the USB drive.

```
root@host% mount_msdosfs /dev/da0p1 /var/tmp/usb
root@host% ls /var/tmp/usb
```

images.tgz

10. Unmount the USB drive after the file is completely copied:

```
root@host% umount /var/tmp/usb
```

Supporting Faster File Copy or Transfer

IN THIS SECTION

On the NFX series devices with the NFX-3 architecture, a user can log into the vJunos0 through the front panel management port (fxp0) using protocols such as SSH.

The **/var/public/** directory accessible inside the vJunos0 is meant for storing image files required for launching and supporting VNFs. The **/var/public/** is a directory on the hypervisor that is mounted as a virtFS (virtual file system) inside the vJunos0. A file copy or file transfer operation between external network and vJunos0 through fxp0 interface goes through the virtFS to reach the **/var/public/** directory on the Linux hypervisor.

Starting in Junos OS Release 24.2R1, Junos OS allows creating an external network interface directly on the hypervisor and associating an IPv4 address with it. The external network access is through the front panel management interface (out-of-band) or through one of the front panel revenue ports (in-band), depending on the configuration.

NFX series devices support the following types of faster file copy or file transfer:

- Remote file copy—Use `request vmhost remote-file-copy` command to copy a file directly between the hypervisor and an external fileserver.
- Local file copy of a file on the hypervisor—Use `request vmhost local-file-copy` command to copy a file or a directory present under **/var/public/** directory structure on the hypervisor to a different name under **/var/public/** directory structure.

Configuring for in-band External Interface

The topology requirement for in-band external interface configuration, irrespective of the type of SKU is a VNF with SR-IOV interface mapped to a front panel port connected to another front panel port on the same device.

For example, on an NFX150 device, you can map the VNF SR-IOV interface to a front panel port (heth-0-x) directly with a cable connecting heth-0-x port to another heth-0-y port on the same device.

On an NFX250 or an NFX350 device, you can map the VNF SR-IOV interface an internal NIC hsxeX and map to a front panel port ge-0/0/X through a VLAN. Connect the ge-0/0/X with a direct cable to another ge-0/0/Y on the same device. Note that ge-0/0/Y and hsxeY are together in a different VLAN.

```
user@host#set vmhost external-interface in-band mapping interface interface-name
user@host#set vmhost external-interface in-band mapping interface virtual-function

user@host#set vmhost external-interface in-band family inet address ipv4_inet_address

user@host#set vmhost external-interface in-band family inet gateway ipv4_gateway

user@host#set vmhost external-interface in-band vlan-id vlan_id
```

For example, to configure an in-band external interface on an NFX150 device:

```
user@host#set vmhost external-interface in-band mapping interface heth-0-0
user@host#set vmhost external-interface in-band mapping interface virtual-function
user@host#set vmhost external-interface in-band family inet address 10.10.10.10/24 gateway
10.10.10.254
user@host#set vmhost external-interface in-band vlan-id 10
```

For example, to configure an in-band external interface on an NFX250 or an NFX350 device:

```
user@host#set vmhost external-interface in-band mapping interface hsxe0
user@host#set vmhost external-interface in-band mapping interface virtual-function
user@host#set vmhost external-interface in-band family inet address 10.10.10.10/24 gateway
10.10.10.254
user@host#set vmhost external-interface in-band vlan-id 10
```

Configuring for out-of-band External Interface

The topology requirement irrespective of the type of SKU is a VNF with an out-of-band management interface reachable on the same management network as the out-of-band management interface (eth0) of the NFX device.

```
user@host#set vmhost external-interface out-of-band family inet address ipv4_net_addres
user@host#set vmhost external-interface out-of-band family inet gateway ipv4_gateway
```

For example:

```
user@host#set vmhost external-interface out-of-band family inet address 10.204.97.175/20

user@host#set vmhost external-interface out-of-band family inet gateway 10.204.111.254/32
```

Installing Software on NFX Devices Using USB Autoinstallation

IN THIS SECTION

The USB autoinstallation feature simplifies Junos OS image upgrade when there is no console access to the NFX device. This feature allows you to upgrade the Junos OS image by inserting a USB flash drive with the image and configuration into the USB port of the NFX device. To install software on an NFX series device using the USB autoinstallation you must:

1. Enable Junos OS configuration
2. Insert a USB flash drive with Junos OS image and configuration files
3. Enable the configuration to autoinstall the image from the USB.
4. Reboot after installation.

Preparing the USB

IN THIS SECTION

You need to prepare the USB for automatic installation, so that the USB is recognized and works properly.

Precautions

Consider the following precautions before preparing the USB:

- Make sure **/var/public/** has at least 3.5 GB to 4 GB of empty space before attempting `usb-auto-install`.
- Verify any staged installation using `show vmhost version` command. `usb-auto-install` fails to install image if an installation is staged already. To cancel any staged installation, run `request vmhost software rollback` command.
- If the USB is already plugged in, unplug and re-insert the USB to trigger the autoinstallation.
- NFX350 device has 2 USB host or slot. If the device contains any invalid USB (not configured for `usb-auto-install`), the other USB slot can be used for `usb-auto-install`.
- If one connected USB is valid but inactive (not configured for `usb-auto-install`), you can configure if configuration is disabled and use other slot to trigger auto-install without disturbing the first USB.
- If one USB has already triggered `usb-auto-install` and the installation is in progress, you must wait until the installation is complete. Insert the other USB only after the installation is complete. Avoid inserting the second USB to protect the USB and the image inside it from getting corrupted due to improper mount/unmount.

Preparing the USB

1. Format the USB with FAT/FAT32
   - On Windows PC or laptop:
     a. Plug-in the USB key to a Windows PC or a laptop.
     b. From **My Computer** right-click the **Removable Disk** drive.
     c. Format the drive with the FAT/FAT32 file system.
     d. Copy the Junos OS package to the USB key.
   - On Macintosh PC or laptop
     a. Insert the USB to be formatted to a Mac PC.
     b. Navigate to **Applications** > **Utilities**. Double-click to open disk utility.
     c. Select the drive you want to format. Click **Erase**.
     d. Rename the USB drive (optional) and choose the **MS-DOS(FAT)** to format.
     e. Select **Master Boot Record** for scheme, select **Erase** to erase USB drive.

Once the process is complete, the USB drive is ready to reuse with a FAT32 file system to save data.

- On NFX series device or on any Juniper device running Linux:

  a. Run `lsblk` to find the dev node or partition of the USB, possibly **sdb/sdc**.

  b. Format the partition (sdb1/sdc1)

  ```
  mkfs.vfat /dev/sdb1
  ```

  c. If you are not successful, try the following command.

  ```
  mkfs.vfat -I /dev/sdb1
  ```

  NOTE: You can format the partition of the USB, that is, sdb1/sdc1. Do not format sdb/sdc.

2. Copy the file to USB

   - To copy the file from laptop:

     - Copy and paste the files manually if you have a GUI (Windows/Mac OS/Linux).

     - Else, open a command prompt and type `cp <pkg> <drive-name>:\`. For example, if the detected drive is F, type `cp <pkg> F:\`

   - To copy file from an NFX device or any Juniper device running Linux:

     - Mount the dev node to any path and copy the following:

     ```
     mkdir /var/public/tomount
     mount /dev/sdb /var/public/tomount
     cp <pkg> /var/public/tomount
     ```

   NOTE: This feature works with only CLI packaged images (jinstall--***-secure-signed.tgz). The images specifically built for USB (install**usb**.tgz) does not work with this feature.

3. Creating the conf file

- To create the conf file using GUI:
  a. Open the USB partition. Right-click and create an empty file.
  b. Save the file with the name **usb-auto.conf**. Ensure to use the right name and extension.
  c. Else, open a command prompt and type `echo " " > <drive-name>:\usb-auto.conf`

     For example, if the detected drive is F, type `echo " " > F:\usb-auto.conf`.
- To create the conf file using an NFX device or any Juniper device running Linux:
  a. Mount the USB
  b. Edit **/var/public/tomount/usb-auto.conf**

NOTE: After copying a package and creating a conf file, make sure to unmount the USB before removing it using `unmount /var/public/tomount`.

Configuring the NFX Device

After preparing the USB device, you need to configure the NFX device.

1. Ensure `usb-pass-through` is not enabled.
   - Run `show system services usb-pass-through` command. Ensure that the output displays usb-pass-through is disabled.
   - If `usb-pass-through` is enabled, then disable it by removing the configuration.

     Run `delete system services usb-pass-through` command to remove the configuration.
   - Reboot the device once `usb-pass-through` is disabled.
2. Enable `usb-auto-install` configuration
   - Configure `usb-auto-install`.

   ```
   user@host> set system services usb-auto-install
   ```

   - Verify `usb-auto-install` is enabled.

   ```
   user@host> show system services usb-auto-install
   ```

Installing the Image

Once the configuration is applied, the USB and the device are ready for autoinstallation. Following are the steps to install the image on an NFX device.

1. Insert the USB in the USB slot of the device. Wait for at least 5 minutes till the image is copied to the device. You can now remove the USB.

2. Wait for 10 more minutes and observe the LED's. Initially, all LEDs must be up. After the installation is complete and the device reboots, the SYS LED starts to blink green and other LEDs go down. This LED status indicates that the installation is successful and the device is rebooting.

   When all the LEDs are up, it indicates that the bootup is done and the device is ready with new image.

   NOTE:

   - If the LEDs do not change even after 20 minutes, it indicates that the installation has failed.

   - SYS LED turns red if there are any issues during bootup. This indicates that the installation is successful but the bootup after the installation is a failure.

Disabling Autoinstallation

After installation, we recommend disabling `usb-auto-install` to avoid reinstallation or to avoid unintentional upgrade through `usb-auto-install`. To delete the `usb-auto-install`:

```
user@host> delete system services usb-auto-install
```

7

CHAPTER

Configuring Security

IN THIS CHAPTER

IP Security on NFX Devices

IN THIS SECTION

Overview

IPsec provides network-level data integrity, data confidentiality, data origin authentication, and protection from replay. IPsec can protect any protocol running over IP on any medium or a mixture of application protocols running on a complex combination of media. IPsec provides security services at the network layer of the Open Systems Interconnection (OSI) model by enabling a system to select required security protocols, determine the algorithms to use for the security services, and implement any cryptographic keys required to provide the requested services. IPsec is standardized by International Engineering Task Force (IETF).

IPsec protects one or more paths between a pair of hosts or security gateways, or between a security gateway and a host. It achieves this by providing a secure way to authenticate senders/receivers and encrypt IP version 4 (IPv4) and version 6 (IPv6) traffic between network devices.

The key concepts of IPsec include:

- Security associations (SAs)—An SA is a set of IPsec specifications negotiated between devices that are establishing an IPsec relationship. These specifications include preferences for the type of authentication and encryption, and the IPsec protocol that is used to establish the IPsec connection. A security association is uniquely identified by a security parameter index (SPI), an IPv4 or IPv6 destination address, and a security protocol (AH or ESP). IPsec security associations are established either manually through configuration statements, or dynamically by IKE negotiation. For more information about SAs, see Security Associations.

- IPsec key management—VPN tunnels are built using IPsec technology. Virtual private network (VPN) tunnels operate with three kinds of key creation mechanisms such as Manual Key, AutoKey Internet Key Exchange (IKE) , and Diffie-Hellman (DH) Exchange. NFX150 devices support IKEv1 and IKEv2. For more information about IPsec key management, see IPsec Key Management.

- IPsec security protocols—IPsec uses two protocols to secure communications at the IP layer:

    - Authentication Header (AH)—A security protocol for authenticating the source of an IP packet and verifying the integrity of its content.
    - Encapsulating Security Payload (ESP)—A security protocol for encrypting the entire IP packet and authenticating its content.

  For more information about IPsec security protocols, see IPsec Security Protocols.

- IPsec tunnel negotiation—To establish an IKE IPsec tunnel, two phases of negotiation are required:
    - In Phase 1, the participants establish a secure connection to negotiate the IPsec SAs.
    - In Phase 2, the participants negotiate the IPsec SAs for encrypting and authenticating the ensuing exchanges of user data.

  For more information about IPsec tunnel negotiation, see IPsec Tunnel Negotiation.

  Starting with Junos OS Release 19.4 R1, NFX350 devices support IKED by default.

  Starting with Junos OS Release 24.2R1, NFX150 devices and NFX250 devices support IKED.

  NOTE: NFX350 devices have IKED as the default daemon. Starting in Junos OS 24.2R1 IKED is the default daemon on NFX150 and NFX250 devices.

Table 9 on page 63 lists the IPsec features supported on NFX Series devices.

Table 9: IPsec Features Supported on NFX Series Devices

Features	Reference
AutoVPN Spoke	Understanding Spoke Authentication in AutoVPN Deployments
Auto Discovery VPN (ADVPN) Partner **NOTE**: On NFX150 devices, you cannot configure ADVPN Suggester.	Understanding Auto Discovery VPN
Site-to-Site VPN and Dynamic Endpoints	Understanding IPsec VPNs with Dynamic Endpoints

Table 9: IPsec Features Supported on NFX Series Devices *(Continued)*

Features	Reference
Route-based VPN **NOTE**: NFX150 devices do not support policy-based VPNs.	Understanding Route-Based IPsec VPNs
NAT-T	Understanding NAT-T
Dead Peer Detection	Understanding VPN Monitoring

Configuring Security

IN THIS SECTION

On NFX150 devices, security is implemented by using IP security (IPsec). The configuration process of IP security (IPsec) includes the following tasks:

Configuring Interfaces

To enable IPsec on a LAN or WAN, you must configure interfaces to provide network connectivity and data flow.

NOTE: To configure IPsec, use the FPC1 interface.

To configure interfaces, complete the following steps:

1. Log in to the JCP CLI and enter configuration mode:

```
root@host% cli
root@host> configure
```

2. Enable VLAN tagging support on the logical interface:

```
root@host# set interfaces interface-name vlan-tagging
```

3. Assign a VLAN ID to the logical interface:

```
root@host# set interfaces interface-name unit logical-interface-unit-number vlan-id vlan-id
```

4. Assign an IPv4 address to the logical interface:

```
root@host# set interfaces interface-name unit logical-interface-unit-number family inet
address interface-address
```

5. Assign an IPv6 address to the logical interface:

```
root@host# set interfaces interface-name unit interface-logical-unit-number family inet6
address interface-address
```

Configuring Routing Options

Routing capabilities and features that are not specific to any particular routing protocol are collectively called protocol-independent routing properties. These features often interact with routing protocols. In many cases, you combine protocol-independent properties and routing policy to achieve a goal. For example, you define a static route using protocol-independent properties, and then you use a routing policy to re-distribute the static route into a routing protocol, such as BGP, OSPF, or IS-IS.

Protocol-independent routing properties include:

- Static, aggregate, and generated routes

- Global preference
- Martian routes
- Routing tables and routing information base (RIB) groups

To configure the routing table groups into which the interface routes are imported, complete the following steps:

1. Configure RIB and static route:

```
root@host# set routing-options rib rib-name static route ip-address/prefix-length next-hop ip-
address
```

2. Configure static route:

```
root@host# set routing-options static route ip-address/prefix-length next-hop ip-address
```

Configuring Security IKE

IPsec uses the Internet Key Exchange (IKE) protocol to authenticate the IPsec peers, to negotiate the security association (SA) settings, and to exchange IPsec keys. The IKE configuration defines the algorithms and keys used to establish the secure IKE connection with the peer security gateway.

You can configure IKE traceoptions for debugging and managing the IPsec IKE.

To configure IKE traceoptions, complete the following steps:

1. Specify the maximum size of the trace file:

```
root@host# set security ike traceoptions file size file-size
```

2. Specify the parameters to trace information for IKE:

```
root@host# set security ike traceoptions flag all
```

3. Specify the level of trace information for IKE:

```
root@host# set security ike traceoptions level level 7-15
```

You can configure one or more IKE proposals. Each proposal is a list of IKE attributes to protect the IKE connection between the IKE host and its peer.

To configure IKE proposal, complete the following steps:

1. Configure pre-shared-keys as an authentication method for the IPsec IKE proposal:

   NOTE: When you configure IPsec for secure communications in the network, the peer devices in the network must have at least one common authentication method. Only one authentication method can be used between a pair of devices, regardless of the number of authentication methods configured.

   ```
   root@host# set security ike proposal ike-proposal-name authentication-method pre-shared-keys
   ```

2. Define a Diffie-Hellman group (dh-group) for the IKE proposal:

   ```
   root@host# set security ike proposal ike-proposal-name dh-group group14
   ```

3. Configure an authentication algorithm for the IKE proposal:

   ```
   root@host# set security ike proposal ike-proposal-name authentication-algorithm sha-256
   ```

4. Define an encryption algorithm for the IKE proposal:

   ```
   root@host# set security ike proposal ike-proposal-name encryption-algorithm aes-256-cbc
   ```

5. Set a lifetime for the IKE proposal in seconds:

   ```
   root@host# set security ike proposal ike-proposal-name lifetime-seconds 180 to 86400 seconds
   ```

After configuring one or more IKE proposals, you must associate these proposals with an IKE policy. An IKE policy defines a combination of security parameters (IKE proposals) to be used during IKE negotiation. It defines a peer address and the proposals needed for that connection. Depending on which authentication method is used, it defines the preshared key for the given peer. During the IKE negotiation, IKE looks for an IKE policy that is the same on both peers. The peer that initiates the negotiation sends all its policies to the remote peer, and the remote peer tries to find a match.

To configure IKE policy, complete the following steps:

1. Define an IKE policy with first phase mode:

```
root@host# set security ike policy ike-policy-name mode aggressive
```

2. Define a set of IKE proposals:

```
root@host# set security ike policy ike-policy-name proposals proposal-name
```

3. Define a pre-shared key for IKE:

```
root@host# set security ike policy ike-policy-name pre-shared-key ascii-text text-format
```

Configure an IKE gateway to initiate and terminate network connections between a firewall and a security device.

To configure IKE gateway, complete the following steps:

1. Configure an IKE gateway with an IKE policy:

```
root@host# set security ike gateway gateway-name ike-policy ike-policy-name
```

2. Configure an IKE gateway with an address or hostname of the peer:

NOTE: Multiple IKE gateway address redundancy is not supported on NFX350 devices if the deamon is IKED daemon. Only KMD daemon supports this functionality.

```
root@host# set security ike gateway gateway-name address address-or-hostname-of-peer
```

3. Enable dead peer detection (DPD) feature to send DPD messages periodically:

```
root@host# set security ike gateway gateway-name dead-peer-detection always-send
```

4. Configure the local IKE identity:

```
root@host# set security ike gateway gateway-name local-identity <inet | inet6 | key-id |
hostname | user-at-hostname | distinguished-name>
```

5. Configure the remote IKE identity:

```
root@host# set security ike gateway gateway-name remote-identity <inet | inet6 | key-id |
hostname | user-at-hostname | distinguished-name>
```

6. Configure an external interface for IKE negotiations:

```
root@host# set security ike gateway gateway-name external-interface ge-1/0/1.0
```

7. Configure username of the client:

```
root@host# set security ike gateway gateway-name client username client-username
```

8. Configure password of the client:

```
root@host# set security ike gateway gateway-name client password client-password
```

Configuring Security IPsec

IPsec is a suite of related protocols that provides network-level data integrity, data confidentiality, data origin authentication, and protection from replay. IPsec can protect any protocol running over IP on any medium or a mixture of application protocols running on a complex combination of media.

Configure an IPsec proposal, which lists protocols and algorithms or security services to be negotiated with the remote IPsec peer.

To configure an IPsec proposal, complete the following steps:

1. Define an IPsec proposal and protocol for the proposal:

```
root@host# set security ipsec proposal ipsec-proposal-name protocol esp
```

2. Define an authentication algorithm for the IPsec proposal:

   ```
   root@host# set security ipsec proposal ipsec-proposal-name authentication-algorithm hmac-
   sha-256-128
   ```

3. Define an encryption algorithm for the IPsec proposal:

   ```
   root@host# set security ipsec proposal ipsec-proposal-name encryption-algorithm aes-256-cbc
   ```

4. Set a lifetime for the IPsec proposal in seconds:

   ```
   root@host# set security ipsec proposal ipsec-proposal-name lifetime-seconds 180..86400 seconds
   ```

After configuring one or more IPsec proposals, you must associate these proposals with an IPsec policy. An IPsec policy defines a combination of security parameters (IPsec proposals) used during IPsec negotiation. It defines Perfect Forward Secrecy (PFS) and the proposals needed for the connection. During the IPsec negotiation, IPsec searches for a proposal that is the same on both peers. The peer that initiates the negotiation sends all its policies to the remote peer, and the remote peer tries to find a match.

To configure IPsec policies, complete the following steps:

1. Define an IPsec policy, a perfect forward secrecy, and a Diffie-Hellman group for the policy:

   ```
   root@host# set security ipsec policy ipsec-policy-name perfect-forward-secrecy keys group14
   ```

2. Define a set of IPsec proposals for the policy:

   ```
   root@host# set security ipsec policy ipsec-policy-name proposals proposal-name
   ```

Configure an IPsec virtual private network (VPN) to provide a means for securely communicating among remote computers across a public WAN such as the Internet. A VPN connection can link two LANs (site-to-site VPN) or a remote dial-up user and a LAN. The traffic that flows between these two points passes through shared resources such as routers, switches, and other network equipment that make up the public WAN. To secure VPN communication while passing through the WAN, the two participants create an IPsec tunnel. For more information, see IPsec VPN Overview.

To configure IPsec VPN, complete the following steps:

1. Define an IKE gateway for the IPsec VPN:

```
root@host# set security ipsec vpn vpn-name ike gateway remote-gateway-name
```

2. Define an IPsec policy for the IPsec VPN:

```
root@host# set security ipsec vpn vpn-name ike ipsec-policy ipsec-policy-name
```

3. Define a local traffic selector for the IPsec VPN:

```
root@host# set security ipsec vpn vpn-name traffic-selector traffic-selector-name local-ip
local-traffic-selector-ip-address
```

4. Define a remote traffic selector for the IPsec VPN:

```
root@host# set security ipsec vpn vpn-name traffic-selector traffic-selector-name remote-ip
remote-traffic-selector-ip-address
```

5. Define a criteria to establish IPsec VPN tunnels:

```
root@host# set security ipsec vpn vpn-name establish-tunnels on-traffic
```

Configuring Security Policies

A security policy controls the traffic flow from one zone to another zone by defining the kind of traffic permitted from specified IP sources to specified IP destinations at scheduled times. Policies allow you to deny, permit, reject, encrypt and decrypt, authenticate, prioritize, schedule, filter, and monitor the traffic attempting to cross from one security zone to another. You can decide which users and what data can enter and exit, and when and where they can go.

To configure security policies, complete the following steps:

1. Configure security policy match criteria for the source address:

```
root@host# set security policies from-zone from-zone-name to-zone to-zone-name policy policy-
name match source-address any
```

2. Configure security policy match criteria for the destination address:

```
root@host# set security policies from-zone from-zone-name to-zone to-zone-name policy policy-
name match destination-address any
```

3. Configure security policy application:

```
root@host# set security policies from-zone from-zone-name to-zone to-zone-name policy policy-
name match application any
```

4. Set security policy match criteria:

```
root@host# set security policies from-zone from-zone-name to-zone to-zone-name policy policy-
name match then permit
```

Configuring Security Zones

Security zones are the building blocks for policies. They are logical entities to which one or more interfaces are bound. Security zones provide a means of distinguishing groups of hosts (user systems and other hosts, such as servers) and their resources from one another in order to apply different security measures to them. For information, see Understanding Security Zones.

To configure security zones, complete the following steps:

1. Configure security zones with system services:

```
root@host# set security zones security-zone zone-name host-inbound-traffic system-services all
```

2. Define protocols for security zones:

```
root@host# set security zones security-zone zone-name host-inbound-traffic protocols all
```

3. Configure interfaces for security zones:

```
root@host# set security zones security-zone zone-name interfaces interface-name
```

8

CHAPTER

Configuring Virtual Network Functions

IN THIS CHAPTER

Prerequisites to Onboard Virtual Network Functions on NFX350 Devices

IN THIS SECTION

You can onboard and manage Juniper Virtual Network Functions (VNFs) and third-party VNFs on NFX devices through the Junos Control Plane (JCP).

NOTE: This topic provides general guidelines to qualify VNFs on NFX350 devices. Before onboarding a VNF, you must test the VNF according to your use case scenario.

NFX350 Device Prerequisites to Onboard a VNF

To onboard VNFs on NFX350, the device must be on either Hybrid mode or Compute mode. The number of VNFs that you can onboard on the device depends on the system resources such as CPUs and system memory that are available on the mode that the device is operating. For more information about the performance modes, see "NFX350 Overview" on page 2.

Before you onboard the VNF, check the following NFX350 device capabilities:

- Check the current performance mode of the device by using the `show vmhost mode` command. The NFX350 device must be in either Compute or Hybrid mode when you run the `show vmhost mode` command.
- Check the available system memory by using the `show system visibility memory` command.

  Table 10 on page 75 lists the possible memory availability for VNF usage for the NFX350 models.

Table 10: Memory Availability for VNF Usage (Junos OS 19.4R1 Release)

Model	Total System Memory	Hugepages Availability for VNF Usage
NFX350-S1	32 GB	7 1G hugepages
NFX350-S2	64 GB	23 1G hugepages
NFX350-S3	128 GB	62 1G hugepages

- Check the available CPUs and its status by using the `show system visibility cpu` command. Use the `show vmhost mode` command to check the available CPUs in the current performance mode of the device.

  Table 11 on page 75 lists the CPUs available for VNF usage for the NFX350 models.

Table 11: CPUs Available for VNF Usage (Junos OS 19.4R1 Release)

Model	CPUs Available for VNF Usage		
	Throughput Mode	Hybrid Mode	Compute Mode
NFX350-S1	0	8	10
NFX350-S2	0	10	14
NFX350-S3	0	14	20

NOTE: When you change the performance mode of the device, it is recommended to check the availability of the CPUs for VNFs.

NOTE: On NFX350 devices, it is recommended to use external SSD for storing VNF images or files.

For more information, see "Configuring VNFs on NFX350 Devices" on page 82.

VNF Prerequisites to Onboard on an NFX350 Device

To onboard a VNF on an NFX350 device, the following VNF properties should be met:

NOTE: For VNF production deployment, it is recommended to use external hard disk.

- KVM based hypervisor deployment
- OVS or Virtio interface drivers
- raw or qcow2 VNF file types
- Support of up to a maximum of 8 user interfaces

Following are the optional prerequisites to onboard a VNF:

- (Optional) SR-IOV
- (Optional) CD-ROM and USB configuration drives
- (Optional) Hugepages for memory requirements if VNF wants to access OVS.

Validate the VNFs

To validate and qualify the VNFs, you must ensure the following:

- The configuration commit succeeds for the VNF.
- The `show virtual-network-functions` command output displays the VNF entry.
- The `show system visibility vnf` command output displays the VNF properties and interfaces that are configured.
- The `show vmhost network nfv-back-plane` command displays all interfaces that are connected to the OVS bridges with the state `up/up`. The `show system visibility network` command displays all the VNF interfaces.
- Connection to the console of the VNF succeeds and VNF boot up or login prompt is displayed.
- When you are logged into the VNF, use the `request virtual-network-function console` command for the VNF to display all the interfaces that are configured.
- The `show virtual-network-functions` command lists the VNF that are alive when the internal management interface is configured with DHCP client inside the VNF.

- VNF interfaces on the OVS bridge show `tx/rx` statistics when the traffic is ingressed or egressed from the VNF.
- VNF should restart successfully when a restart is initiated from within the VNF or by using the `request virtual-network-functions restart` *`vnf-name`* command.

For sample configuration of third-party VNFs, see "Example: Configuring Service Chaining for LAN to WAN Routing through Third-party VNFs on NFX350 Devices" on page 210.

Sample Output

- show virtual-network-functions

```
root@host> show virtual-network-functions
ID       Name                                              State      Liveliness
-
5        vsrx                                              Running    down
1        vjunos0                                           Running    alive
```

The `Liveliness` is alive when there is a management connectivity to the VNF. The `State` should be `Running` to show that the VNF is up.

- show system visibility vnf

```
root@host> show system visibility vnf
List of VNFs
-
ID   Name                                  State
- - -
5    vsrx                                  Running

VNF Memory Usage
-
Name                                  Maximum Memory (KiB)  Used Memory (KiB)  Used 1G
Hugepages  Used 2M Hugepages
- - - - -
vsrx                                  4194304               49715
4                  0

VNF CPU Statistics (Time in ms)
```

```
-
Name                                   CPU Time           System Time  User Time
- - - -
vsrx                                   164425446          3214840      197880

VNF MAC Addresses
-
VNF                                       MAC
- -
centos1_ethdef0                           9C:CC:83:BD:8C:40
centos1_ethdef1                           9C:CC:83:BD:8C:46
centos1_eth2                              9C:CC:83:BD:8C:41
vsrx_ethdef0                              9C:CC:83:BD:8C:42
vsrx_ethdef1                              9C:CC:83:BD:8C:43
vsrx_eth2                                 9C:CC:83:BD:8C:45
vsrx_eth3                                 9C:CC:83:BD:8C:44

VNF Internal IP Addresses
-
VNF                                       IP
- -
vsrx                                      192.0.2.100

VNF Interfaces
-
VNF                  Interface Type      Source       Model      MAC               IPv4-
address
- - - - - - -
vsrx                 vnet6     network   default      virtio     9c:cc:83:bd:8c:42 -
vsrx                 vnet7     bridge    eth0br       virtio     9c:cc:83:bd:8c:43 -
vsrx                 vsrx_eth2 vhostuser -           virtio     9c:cc:83:bd:8c:45 -

VNF Disk Information
-
VNF                           Disk        File
- - -
vsrx                          vda         /var/public/junos-vsrx3-x86-64-19.4R1.12.qcow2

VNF Disk Usage
-
VNF                  Disk      Read Req   Read Bytes   Write Req  Write Bytes
- - - - - -
vsrx                 vda       220376     1951876096   24927      185393152
```

```
VNF Port Statistics
-
VNF                  Port      Rcvd Bytes   Rcvd Packets Rcvd Error Rcvd Drop Trxd Bytes
Trxd Packets Trxd Error Trxd Drop
- - - - - - - - - - 
vsrx                 vnet6     4113582      79122        0          0         0
0            0          0
vsrx                 vnet7     3399770129   47653525     0          34631     0
0            0          0
vsrx                 vsrx_eth2 3724         65           0          0         4372
73           0          0
```

- request virtual-network-functions vSRX Virtual Firewall console

```
root@host> **request virtual-network-functions vsrx console**
Internal instance: vsrx
Connected to domain vsrx
Escape character is ^]



FreeBSD/amd64 (Amnesiac) (ttyu0)

login: root
Password:
Last login: Tue Mar 17 16:10:40 on ttyu0

- JUNOS 19.4R1.12 Kernel 64-bit XEN JNPR-11.0-20191115.14c2ad5_buil
root@:~ #
root@:~ # cli
hroot> show interfaces terse
Interface               Admin Link Proto    Local                 Remote
ge-0/0/0                up    up
gr-0/0/0                up    up
ip-0/0/0                up    up
lsq-0/0/0               up    up
lt-0/0/0                up    up
mt-0/0/0                up    up
sp-0/0/0                up    up
sp-0/0/0.0              up    up   inet
                                   inet6
sp-0/0/0.16383          up    up   inet
```

```
ge-0/0/1                up    up
ge-0/0/1.0              up    up   inet     10.10.10.1/24

root> show configuration | display set |match fxp0
set system services web-management http interface fxp0.0
set system services web-management https interface fxp0.0
set interfaces fxp0 unit 0 family inet dhcp

root> show interfaces terse | match fxp0
fxp0                    up    up
fxp0.0                  up    up   inet     192.0.2.100/24
```

- show system visibility memory

```
root@host> show system visibility memory
Memory Information
------------------

Virtual Memory:
---------------
Total       (KiB): 131042784
Used        (KiB): 64842740
Available   (KiB): 66595824
Free        (KiB): 66200044
Percent Used     : 49.2

Huge Pages:
------------
Total 1GiB Huge Pages:      18
Free 1GiB Huge Pages:       0
Configured 1GiB Huge Pages: 0
Total 2MiB Huge Pages:      20481
Free 2MiB Huge Pages:       0
Configured 2MiB Huge Pages: 0

Hugepages Usage:
----------------------------------------------------------------------------------------------
------------
Name                             Type                                Used 1G Hugepages  Used
2M Hugepages
--------------------------------- ---------------------------------- ------------------
------------------
```

```
ovs-vswitchd                      other process                      18                 0
srxpfe                            other process                      6                  20481
```

In the output message, check `Free` and `Configured` fields under `Virtual Memory` and `Huge Pages` sections for the memory availability.

- show vmhost mode

```
root@host> show vmhost mode
Mode:
--------
Current Mode: hybrid

CPU Allocations:
Name                                 Configured                              Used
----------------------------------------------------------------------------------------------
------------------------
Junos Control Plane                  16                                      16,9
Juniper Device Manager               16                                      16
LTE                                  16                                      -
NFV Backplane Control Path           16                                      16
NFV Backplane Data Path              1,2,3,4                                 1,2,3,4
Layer 2 Control Path                 -                                       -
Layer 2 Data Path                    -                                       -
Layer 3 Control Path                 0                                       0
Layer 3 Data Path                    5,6,7,8                                 5,6,7,8
CPUs available for VNFs              9,10,11,12,13,14,15,25,26,27,28,29,30,31 -
CPUs turned off                      17,18,19,20,21,22,23,24                 -

Memory Allocations:
Name                                 Configured                              Used
----------------------------------------------------------------------------------------------
------------------------
Junos Control Plane (mB)             2048                                    2009
NFV Backplane 1G hugepages           12                                      18
NFV Backplane 2M hugepages           -                                       0
Layer 2 1G hugepages                 -                                       -
Layer 2 2M hugepages                 -                                       -
Layer 3 1G hugepages                 6                                       6
Layer 3 2M hugepages                 20481                                   20481
```

In the output message, check the `Current Mode` field under the `Mode` section for the current performance mode of the device. Check the `CPUs available for VNFs` field under the `CPU Allocations` section for the CPU availability.

Configuring VNFs on NFX350 Devices

IN THIS SECTION

The NFX350 devices enable you to instantiate and manage virtualized network functions (VNFs) from the Junos Control Plane (JCP). The JCP supports the creation and management of third-party VNFs.

Load a VNF Image

To configure a VNF, you must log in to the JCP:

```
user@host:~ # cli
user@host>
```

To load a VNF image on the device from a remote location, you can either use the `file-copy` command or copy the image from a USB by using the `usb-pass-through` command.

NOTE: You can save the VNF image in the **/var/public** directory if you are using up to two VNFs. If you are using more than two VNFs, save the files on an external SSD. If you are using an external SSD for VNFs, make sure to initialize and add the SSD to the device. For more information, see "Configuring the Solid State Disk on NFX350 Device" on page 48.

```
user@host> file copy source-address /var/public
```

For example:

```
user@host> file copy scp://192.0.2.0//tftpboot/centos.img /var/public
```

Alternatively, you can load a VNF image by using the NETCONF command, `file-put`.

To copy a VNF image from a USB, see "Supporting File Transfer from USB on NFX Series Devices" on page 51.

Prepare the Bootstrap Configuration

You can bootstrap a VNF by attaching a CD-ROM, a USB storage device, or a config drive that contains a bootstrap-config ISO file.

For an example of creating an ISO file, see the procedure in Creating a vSRX Bootstrap ISO Image. The procedure might differ based on the operating system (for example, Linux, Ubuntu) that you use to create the ISO file.

A bootstrap configuration file must contain an initial configuration that allows the VNF to be accessible from an external controller, and accepts SSH, HTTP, or HTTPS connections from an external controller for further runtime configurations.

NOTE:

- The system saves the bootstrap-config ISO file in the **/var/public** folder. The file is saved only if the available space in the folder is more than double the total size of the contents in the file. If the available space in the folder is not sufficient, an error message is displayed when you commit the configuration.

- When you reboot the system, the system generates a new bootstrap-config ISO file and replaces the existing ISO file with the new ISO file on the VNF.

Allocate CPUs for a VNF

Table 12 on page 84 lists the CPUs available for VNF usage for the NFX350 models.

Table 12: CPUs Available for VNF Usage

Model	CPUs Available for VNF Usage				
	Throughput Mode	Hybrid Mode	Compute Mode	Custom Mode	
				Flex Mode	Perf Mode
NFX350-S1	0	8	10	11	6
NFX350-S2	0	10	14	19	10
NFX350-S3	0	14	20	27	12

NOTE: The resource allocations for *flex* and *perf* custom modes are based on the templates provided in the default Junos configuration.

NOTE: When you change the performance mode of the device, it is recommended to check the availability of the CPUs for VNFs.

To check the CPU availability and its status:

```
user@host> show system visibility cpu
CPU Statistics (Time in sec)
-------------------------------------------------------------------------------
CPU Id User Time System Time Idle Time Nice Time IOWait Time Intr. Service Time
------ --------- ----------- --------- --------- ----------- ------------------
```

```
0      7762      1475        60539     0         84          0
1      191       511         70218     0         10          0
2      102       32          70841     0         12          0
3      0         0           70999     0         0           0
4      0         0           70999     0         0           0
5      0         0           70999     0         0           0
6      70949     0           50        0         0           0
7      9005      532         59602     0         0           0
8      23        7           70966     0         0           0
9      21        7           70969     0         0           0
10     20        6           70969     0         0           0
11     18        6           70970     0         0           0

CPU Usages
----------------
CPU Id CPU Usage
------ ---------
0      17.899999999999999
1      0.0
2      0.0
3      0.0
4      0.0
5      0.0
6      100.0
7      15.199999999999999
8      0.0
9      0.0
10     0.0
11     0.0

CPU Pinning Information
------------------------------------
Virtual Machine             vCPU CPU
--------------------------- ---- ---
vjunos0                     0    0



System Component                CPUs
```

```
------------------------------ --------
ovs-vswitchd                   0, 6
```

```
user@host> show vmhost mode
Starting network management services: snmpd libvirtMib_subagent.
Synchronizing UEFI key-store:
Failed to get revocation list: 2
Juniper Dev keys are not revoked. Doing nothing
cp: cannot stat '/var/platform/lte_vm_xml_params': No such file or directory
rm: cannot remove '/lib/udev/rules.d/lte_usb.rules': No such file or directory

Mode:
--------
Current Mode: compute

CPU Allocations:
Name                                Configured                                 Used
-----------------------------------------------------------------------------------------------
--------------------
Junos Control Plane                 8                                          3,8
Juniper Device Manager              8                                          8
LTE                                 8                                          -
NFV Backplane Control Path          8                                          8
NFV Backplane Data Path             1                                          1
Layer 2 Control Path                -                                          -
Layer 2 Data Path                   -                                          -
Layer 3 Control Path                0                                          0
Layer 3 Data Path                   2                                          2
CPUs available for VNFs             3,4,5,6,7,11,12,13,14,15                   -
CPUs turned off                     9,10                                       -

Memory Allocations:
Name                                Configured                                 Used
-----------------------------------------------------------------------------------------------
--------------------
Junos Control Plane (mB)            2048                                       2011
NFV Backplane 1G hugepages          4                                          8
NFV Backplane 2M hugepages          -                                          0
Layer 2 1G hugepages                -                                          -
Layer 2 2M hugepages                -                                          -
Layer 3 1G hugepages                4                                        4
Layer 3 2M hugepages                5633                                     5377
```

The `CPUs available for VNFs` section in the output message shows the CPUs that are available to onboard VNFs.

NOTE: vjunos0 is a system VNF, you cannot modify the CPU allocation for the vjunos0.

To specify the number of virtual CPUs that are required for a VNF:

1. Specify the number of CPUs required for the VNF:

```
user@host# set virtual-network-functions vnf-name virtual-cpu count number
```

2. Connect a virtual CPU to a physical CPU:

```
user@host# set virtual-network-functions vnf-name virtual-cpu vcpu-number physical-cpu pcpu-
number
```

3. Commit the configuration:

```
user@host# commit
```

The physical CPU number can be either a number or a number range. By default, a VNF is allocated one virtual CPU that is not connected to any physical CPU.

NOTE: You cannot change the CPU configuration of a VNF while the VNF is running. You must restart the VNF for the changes to take effect.

Starting in Junos OS Release 22.1 R1, you can pin the emulator to specific physical CPUs by using the following command:

```
user@host# set virtual-network-functions vnf-name emulator physical-cpu cpu-range
```

You cannot use CPU 0 or offline CPUs for emulator pinning. If you do not pin the emulator to a specific physical CPU, QEMU automatically pins it to a virtual CPU. Changes to emulator pinning take effect immediately on a running VNF.

To enable hardware virtualization or hardware acceleration for VNF CPUs:

```
user@host# set virtual-network-functions vnf-name virtual-cpu features hardware-virtualization
```

Allocate Memory for a VNF

By default, a certain amount of memory is allocated for VNFs. Table 13 on page 88 lists the possible memory availability for VNF usage for the NFX350 models.

Table 13: Memory Availability for VNF Usage

Model	Total Memory Available	Hugepages Availability for VNF Usage in Compute, Hybrid, and Throughput Modes	Hugepages Availability for VNF Usage in Custom Mode	
			Flex Mode	Perf Mode
NFX350-S1	32 GB	7 1G hugepages	24 1G hugepages	22 1G hugepages
NFX350-S2	64 GB	23 1G hugepages	50 1G hugepages	49 1G hugepages
NFX350-S3	128 GB	62 1G hugepages	110 1G hugepages	108 1G hugepages

NOTE: The resource allocations for *flex* and *perf* custom modes are based on the templates provided in the default Junos configuration.

To check the available memory:

```
user@host> show system visibility memory
Memory Information
------------------

Virtual Memory:
```

```
---------------
Total       (KiB): 15914364
Used        (KiB): 13179424
Available   (KiB): 3087076
Free        (KiB): 2734940
Percent Used     : 80.6

Huge Pages:
------------
Total 1GiB Huge Pages:      7
Free 1GiB Huge Pages:       5
Configured 1GiB Huge Pages: 5
Total 2MiB Huge Pages:      1376
Free 2MiB Huge Pages:       1
Configured 2MiB Huge Pages: 0

Hugepages Usage:
-------------------------------------------------------------------------------------------------
---------
Name                              Type                              Used 1G Hugepages  Used 2M
Hugepages
--------------------------------- --------------------------------- ------------------
------------------
srxpfe                            other process                     1                  1375
ovs-vswitchd                      other process                     2                  0
```

NOTE: vjunos0 is a system VNF, you cannot modify the memory allocation for the vjunos0.

To specify the maximum primary memory that the VNF can use:

```
user@host# set virtual-network-functions vnf-name memory size size
```

NOTE: You cannot change the memory configuration of a VNF while the VNF is running. You must restart the VNF for the changes to take effect.

Configure Interfaces and VLANs for a VNF

You can configure a VNF interface, map a VNF interface to a virtual function, and attach the interface to a physical NIC port, a management interface, or VLANs, assign a VLAN ID to it, and enable trust mode on it.

Prior to Junos OS Releases 21.3R1, 21.2R2, 21.2R1, 21.1R2, and 20.4R3, the step to configure an SR-IOV VNF interface and to assign a VLAN ID is as follows:

```
user@host# set virtual-network-functions vnf-name interfaces vnf-interface-name mapping
interface physical-interface-name virtual-function vlan-id vlan-id
```

Starting from Junos OS Releases 21.3R1, 21.2R2, 21.2R1, 21.1R2, and 20.4R3, the steps to configure an SR-IOV VNF interface, to assign a VLAN ID, and to enable trust mode are as follows:

To map a VNF interface to a virtual function:

```
user@host# set virtual-network-functions vnf-name interfaces vnf-interface-name mapping
interface physical-interface-name
```

To attach a VNF interface to a physical NIC port by using the SR-IOV virtual function and assign a VLAN ID:

```
user@host# set virtual-network-functions vnf-name interfaces vnf-interface-name mapping
interface virtual-function vlan-id vlan-id
```

vlan-id is the VLAN ID of the port and is an optional value.

To enable trust mode:

```
user@host# set virtual-network-functions vnf-name interfaces vnf-interface-name mapping
interface virtual-function trust
```

NOTE:

- Trust mode is supported on NFX Series devices from Junos OS Releases 21.3R1, 21.2R2, 21.2R1, 21.1R2, and 20.4R3.

- If you enable trust mode on VNF SR-IOV interface, then the VNF interface goes into promiscuous mode.

To disable spoof check

```
user@host# set virtual-network-functions vnf-name interfaces interface-name mapping interface
virtual-function disable-spoof-check
```

To attach a VNF interface to a VLAN:

- Create a VLAN:

  ```
  user@host# set vmhost vlan vlan-name
  ```

- Attach a VNF interface to a VLAN:

  ```
  user@host# set virtual-network-functions vnf-name interfaces interface-name mapping vlan
  members list-of-vlans [mode trunk|access]
  ```

A VNF interface can be mapped to one or more physical interface .You can enable this functionality by configuring the virtual port peer (VPP) feature. You can configure mappings between an OVS interface of a VNF to one or more front panel interfaces. The VNF interface becomes inactive if all of the mapped physical interfaces are inactive. The VNF interface becomes active even if at least one of the mapped physical interface is active.

NOTE:

- The mapped physical interface does not become inactive if a VNF interface is inactive.
- Before upgrading a software image that does not support trust mode to an image that supports trust mode, it is recommended to delete all VNF interface to virtual-function mappings from the configuration.
- Before downgrading a software image that supports trust mode to an image that does not support trust mode, it is necessary to delete all VNF interface to virtual-function mappings from the configuration. Else, the device goes into **Amnesiac** state after the downgrade.

The interface to the VNF is an OVS port and this mapping is defined in the configuration. If the mapping rules can view multiple physical ports before triggering the action, configuring the VPP feature allows you to manage multiple, redundant physical links.

You can configure a mapping between VNF virtual interfaces and JCP physical interfaces (ge-0/0/*x* and xe-0/0/*x*). One virtual interface can be mapped to one or more physical interfaces. There is no limit on the number of physical interfaces to which a VNF virtual interface can be mapped to. You can map a VNF virtual interface to all the physical interfaces or you can map multiple VNF interfaces to a single physical interface.

To configure VPP:

```
root@host# set virtual-network-functions vnf-name interfaces interface-name mapping peer-
interfaces physical-interface-name
```

For example:

```
root@host# set virtual-network-functions centos1 interfaces eth2 mapping peer-interfaces ge-0/0/6
```

To view mapping of the peer interfaces, run the `show system visibility vnf vnf-name` command.

NOTE:

- The interfaces attached to a VNF are persistent across VNF restarts.
- If the VNF supports hot-plugging, you can attach the interfaces while the VNF is running. Otherwise, you must add the interfaces, and then restart the VNF.
- You cannot change the mapping of a VNF interface while the VNF is running.

NOTE: You can prevent the VNF interface from sending or receiving traffic by using the `deny-forwarding` CLI option.

If the `deny-forwarding` option is enabled on an interface that is a part of cross-connect, then the cross-connect status goes down and drops all traffic.

```
set virtual-network-options vnf-name interface interface-name forwarding-options deny-
forwarding
```

Starting in Junos OS Release 24.2R1, you can set up mapping or peering between the layer 2 interface (xe-0/0/x or ge-0/0/x port) and the layer 3 interface (ge-1/0/0 to ge-1/0/9) by configuring the `virtualization-options interfaces` *`L3-interface-name`* `mapping peer-interfaces` *`physical-interface`* at the `[edit vmhost]` hierearchy.

You can configure the `fail-on-any-peer option` option if a layer 3 interface is routed to several physical interfaces. If any mapped peer physical interface fails, you can use the `fail-on-any-peer option` option to shut down the layer 3 interface.

To configure VPP:

```
root@host# set vmhost virtualization-options interfaces L3-interface-name mapping peer-
interfaces ge-0/0/5
```

```
root@host# set vmhost virtualization-options interfaces L3-interface-name mapping fail-on-any-
peer
```

For example:

```
root@host# set vmhost virtualization-options interfaces ge-1/0/3  mapping peer-interfaces
physical-interface
```

```
root@host# set vmhost virtualization-options interfaces ge-1/0/3  mapping fail-on-any-peer
```

Starting in Junos OS Release 24.2R1, you can configure static MAC addresses on the VNF interface while using VPP feature.

To configure static MAC address on the VNF interface with VPP:

```
root@host# set virtual-network-functions vnf-name interfaces interface-name mac-address mac-
address
```

NOTE: Whenever you add or delete MAC address on VNF interface, the VNF must be restarted for the change to take effect.

To specify the target PCI address for a VNF interface:

```
user@host# set virtual-network-functions vnf-name interfaces interface-name pci-address target-
pci-address
```

You can use the target PCI address to rename or reorganize interfaces within the VNF.

For example, a Linux-based VNF can use udev rules within the VNF to name the interface based on the PCI address.

NOTE:

- The target PCI address string should be in the following format:

  `0000:00:<slot:>:0`, which are the values for domain:bus:slot:function. The value for slot should be different for each VNF interface. The values for domain, bus, and function should be zero.

- You cannot change the target PCI address of VNF interface while the VNF is running.

To delete a VNF interface:

```
user@host# delete virtual-network-functions vnf-name interfaces interface-name
user@host# commit
```

NOTE:

- To delete a VNF interface, you must stop the VNF, delete the interface, and then restart the VNF.
- After attaching or detaching a virtual function, you must restart the VNF for the changes to take effect.
- eth0 and eth1 are reserved for the default VNF interfaces that are connected to the internal network and the out-of-band management network. Therefore, the configurable VNF interface names start from eth2.
- Within a VNF, the interface names can be different, based on guest OS naming conventions. VNF interfaces that are configured in the JCP might not appear in the same order within the VNF.

- You must use the target PCI addresses to map to the VNF interfaces that are configured in the JCP and you must name them accordingly.

Configure Storage Devices for VNFs

An NFX350 device supports the following storage options for VNFs:

- CD-ROM
- Disk
- USB

To add a virtual CD or to update the source file of a virtual CD:

```
user@host# set virtual-network-functions vnf-name storage device-name type cdrom source file
file-name
```

You can specify a valid device name in the format hd*x*, sd*x*, or vd*x*—for example, hdb, sdc, vdb, and so on.

To add a virtual USB storage device:

```
user@host# set virtual-network-functions vnf-name storage device-name type usb source file file-
name
```

To attach an additional hard disk:

```
user@host# set virtual-network-functions vnf-name storage device-name type disk [bus-type virtio
| ide] [file-type raw | qcow2] source file file-name
```

To delete a virtual CD, USB storage device, or hard disk from the VNF:

```
user@host# delete virtual-network-functions vnf-name storage device-name
```

NOTE:

- After attaching or detaching a CD from a VNF, you must restart the device for the changes to take effect. The CD detach operation fails if the device is in use within the VNF.
- A VNF supports one virtual CD, one virtual USB storage device, and multiple virtual hard disks.
- You can update the source file in a CD or USB storage device while the VNF is running.
- You must save the source file in the **/var/public** directory, and the file must have read and write permission for all users.

Instantiate a VNF

You can instantiate a VNF by configuring the VNF name, and by specifying the path of an image.

While instantiating a VNF with an image, two VNF interfaces are added by default. These interfaces are required for management and for the internal network.

NOTE: Only QCOW2, IMG, and RAW image types are supported.

To instantiate a VNF by using an image:

```
user@host# set virtual-network-functions vnf-name image file-path
user@host# set virtual-network-functions vnf-name image image-type image-type
user@host# commit
```

NOTE: When you configure VNFs, do not use VNF names in the format vnf*n*—for example, vnf1, vnf2, and so on. Configurations that contain such names fail to commit.

(Optional) To specify a UUID for the VNF:

```
user@host# set virtual-network-functions vnf-name [uuid vnf-uuid]
```

uuid is an optional parameter. We recommend that you allow the system to allocate a UUID for the VNF.

NOTE: You cannot change the image configuration for a VNF after saving and committing the configuration. To change the image for a VNF, you must delete the VNF and create a VNF again.

Verify the VNF Instantiation

To verify that the VNF is instantiated successfully:

```
user@host> show virtual-network-functions
ID       Name                                               State      Liveliness
--------------------------------------------------------------------------------
1        vjunos0                                            Running    alive
2        centos1                                            Running    alive
3        centos2                                            Running    alive
```

The output in the **Liveliness** field of a VNF indicates whether the IP address of the VNF is reachable over the internal management network. The default IP address of the liveliness bridge is 192.0.2.1/24. Note that this IP address is internal to the device and is used for VNF management.

Managing VNFs on NFX350 Devices

IN THIS SECTION

Managing VNF States

By default, a VNF automatically starts when the VNF configuration is committed.

- To disable autostart of a VNF when the VNF configuration is committed:

```
user@host# set virtual-network-functions vnf-name no-autostart
```

- To manually start a VNF:

```
user@host> request virtual-network-functions vnf-name start
```

- To stop a VNF:

```
user@host> request virtual-network-functions vnf-name stop
```

- To restart a VNF:

```
user@host> request virtual-network-functions vnf-name restart
```

- To access the console of an active VNF:

```
user@host> request virtual-network-functions vnf-name console
```

NOTE: The `request virtual-network-functions` *`vnf-name`* `console` command is supported only for root login over ssh.

- To access a VNF through SSH:

```
user@host> request virtual-network-functions ssh vnf-name
```

- To access a VNF through Telnet:

```
user@host> request virtual-network-functions telnet vnf-name
```

Managing VNF MAC Addresses

VNF interfaces that are defined, either using the CLI, are assigned a globally unique and persistent MAC address. A common pool of 176 MAC addresses is used to assign MAC addresses to VNF interfaces. These MAC addresses are automatically allocated when a VNF is instantiated. You can configure a MAC address other than what is available in the common pool, and this address will not be overwritten.

- To configure a specific MAC address for a VNF interface:

```
user@host# set virtual-network-functions vnf-name interfaces interface-name mac-address mac-
address
```

- To delete the MAC address configuration of a VNF interface:

```
user@host# delete virtual-network-functions vnf-name interfaces interface-name mac-address
mac-address
```

NOTE:

- To delete or modify the MAC address of a VNF interface, you must stop the VNF, make the necessary changes, and then restart the VNF.
- The MAC address specified for a VNF interface can be either a system MAC address or a user-defined MAC address.
- The MAC address specified from the system MAC address pool must be unique for the VNF interfaces.

Managing the MTU of a VNF Interface

The maximum transmission unit (MTU) is the largest data unit that can be forwarded without fragmentation. You can configure either 1500 bytes or 9216 bytes as the MTU size. The default MTU value is 1500 bytes, and the maximum MTU size for both VNF and L3 interface is 9216 bytes.

NOTE: MTU configuration is supported only on VLAN interfaces.

1. To configure the MTU on a VNF interface:

   ```
   user@host# set virtual-network-functions vnf-name interfaces interface-name mtu size
   ```

   NOTE: You must restart the VNF after configuring the MTU, if the VNF does not support hot-plugging functionality.

2. To delete the MTU of a VNF interface:

   ```
   user@host# delete virtual-network-functions vnf-name interfaces interface-name mtu
   ```

   NOTE: After the MTU is deleted, the MTU of the VNF interface is reset to 1500 bytes.

NOTE:

- The maximum number of VLAN interfaces on the OVS that are supported in the system is 64.

Accessing a VNF from the JCP

You can access a VNF from the JCP through SSH or by using the console.

To access a VNF from the JCP through SSH:

```
user@host> request virtual-network-functions ssh vnf-name
```

To access a VNF from the JCP by using the console:

```
user@host> request virtual-network-functions console vnf-name
```

Viewing the List of VNFs

To view the list of VNFs:

```
user@host> show virtual-network-functions
ID       Name                                                  State      Liveliness
--------------------------------------------------------------------------------
1        vjunos0                                               Running    alive
2        centos1                                               Running    alive
3        centos2                                               Running    alive
```

The **Liveliness** field of a VNF indicates whether the IP address of the VNF is reachable from the JCP. The default IP address of the liveliness bridge is 192.0.2.1/24.

Displaying the Details of a VNF

To display the details of a VNF:

```
user@host> show virtual-network-functions vnf-name detail
user@host>show virtual-network-functions centos1 detail
Virtual Network Function Information
------------------------------------


Id:                   2
Name:                 centos1
State:                Running
Liveliness:           Up
IP Address:           192.0.2.101
VCPUs:                1
Maximum Memory:       1048576 KiB
Used Memory:          1048576 KiB
Used 1G Hugepages:    0
Used 2M Hugepages:    0
Error:                None
```

Deleting a VNF

To delete a VNF:

```
user@host# delete virtual-network-functions vnf-name
```

NOTE: The VNF image remains in the disk even after you delete a VNF.

Non-Root User Access for VNF Console

IN THIS SECTION

You can use Junos OS to create, modify, or delete VNF on the NFX Series routers.

Junos OS CLI allows the following management operations on VNFs:

Table 14: VNF Management Operations

Operation	CLI
start	`request virtual-network-functions <vnf-name> start`
stop	`request virtual-network-functions <vnf-name> stop`
restart	`request virtual-network-functions <vnf-name> restart`
console access	`request virtual-network-functions <vnf-name> console [force]`

Table 14: VNF Management Operations *(Continued)*

Operation	CLI
ssh access	`request virtual-network-functions <vnf-name> ssh [user-name <user-name>]`
telnet access	`request virtual-network-functions <vnf-name> ssh [user-name <user-name>]`

The following table lists the user access permissions for the VNF management options:

Table 15: User Access Permissions for VNF Management Operations before Junos OS 24.1R1.

Operation	`root` class user	`super-user` class user	`operator` class user	`read-only` class user
start	command available and works	command available and works	command not available	command not available
stop	command available and works	command available and works	command not available	command not available
restart	command available and works	command available and works	command not available	command not available
console access	command available and works	command available; but not supported	command not available	command not available
ssh access	command available and works	command available; but not supported	command not available	command not available
telnet access	command available and works	command available; but not supported	command not available	command not available

Starting In Junos OS 24.1R1, Junos OS CLI allows the management operations on VNFs for a non-root user.

A new Junos OS user permission, `vnf-operation` allows the `request virtual-network-functions` CLI hierarchy available to Junos OS users that do not belong to the `root` and the `super-user` class.

You can add the user permission to a custom user class using the statement `vnf-operation` at `[edit system login class custom-user permissions]`

The following table lists the VNF management options available for a user belonging to a custom Junos OS user class with `vnf-operation` permission.

Table 16: User Access Permissions for VNF Management Operations after Junos OS 24.1R1.

Operation	`root` user	`super-user` class user	User of a custom Junos OS user class with `vnf-operation` permission
start	command available and works	command available and works	command available and works
stop	command available and works	command available and works	command available and works
restart	command available and works	command available and works	command available and works
console access	command available and works	command available and works	command available and works
ssh access	command available and works	command available and works	command available and works

Accessing the VNF Console

Starting in Junos OS 24.1R1, the following message is displayed when you access the console initially:

```
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.
```

The messages `Trying 192.168.1.1...` and `Connected to 192.168.1.1.` come from telnet client that is launched using the Junos OS CLI command `request virtual-network-functions <vnf-name> console`.

NOTE: The IP addresses present in the message cannot be replaced with the name of the VNF.

Exiting the VNF Console

Starting in Junos OS 24.1R1, when the user uses the escape sequence `^]` the console session terminates, and the telnet command prompt is displayed to the user.

You must enter `quit` or `close` or you must enter `q`or `c` to exit from the terminal command prompt and return to Junos OS command prompt.

```
su-user@host> request virtual-network-functions testvnf1 console
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.

CentOS Linux 7 (Core)
Kernel 3.10.0-1160.49.1.el7.x86_64 on an x86_64

centos2 login:
telnet> q
Connection closed.

su-user@host> exit
```

Configuring Analyzer VNF and Port-mirroring

The **Port-mirroring** feature allows you to monitor network traffic. If the feature is enabled on a VNF interface, the OVS system bridge sends a copy of all network packets of that VNF interface to the analyzer VNF for analysis. You can use the port-mirroring or analyzer commands for analyzing the network traffic.

NOTE:

- Port-mirroring is supported only on VNF interfaces that are connected to an OVS system bridge.
- VNF interfaces must be configured before configuring port-mirroring options.
- If the analyzer VNF is active after you configure, you must restart the VNF for changes to take effect.
- You can configure up to four input ports and only one output port for an analyzer rule.
- Output ports must be unique in all analyzer rules.
- After changing the configuration of the input VNF interfaces, you must de-activate and activate the analyzer rules referencing to it along with the analyzer VNF restart.

To configure the analyzer VNF and enable port-mirroring:

1. Configure the analyzer VNF:

```
[edit]
user@host#set virtual-network-functions analyzer-vnf-name image file-path
user@host#set virtual-network-functions analyzer-vnf-name interfaces interface-name analyzer
```

2. Enable port-mirroring of the network traffic in the input and output ports of the VNF interface and analyzer VNF:

```
user@host# set vmhost forwarding-options analyzer analyzer-instance-name input [ingress |
egress] virtual-network-function vnf-name interface interface-name
user@host# set vmhost forwarding-options analyzer analyzer-rule-name output virtual-network-
function analyzer-vnf-name interface interface-name
```

Supporting Faster File Copy or Transfer

IN THIS SECTION

On the NFX series devices with the NFX-3 architecture, a user can log into the vJunos0 through the front panel management port (fxp0) using protocols such as SSH.

The **/var/public/** directory accessible inside the vJunos0 is meant for storing image files required for launching and supporting VNFs. The **/var/public/** is a directory on the hypervisor that is mounted as a virtFS (virtual file system) inside the vJunos0. A file copy or file transfer operation between external network and vJunos0 through fxp0 interface goes through the virtFS to reach the **/var/public/** directory on the Linux hypervisor.

Starting in Junos OS Release 24.2R1, Junos OS allows creating an external network interface directly on the hypervisor and associating an IPv4 address with it. The external network access is through the front panel management interface (out-of-band) or through one of the front panel revenue ports (in-band), depending on the configuration.

NFX series devices support the following types of faster file copy or file transfer:

- Remote file copy—Use `request vmhost remote-file-copy` command to copy a file directly between the hypervisor and an external fileserver.
- Local file copy of a file on the hypervisor—Use `request vmhost local-file-copy` command to copy a file or a directory present under **/var/public/** directory structure on the hypervisor to a different name under **/var/public/** directory structure.

Configuring for in-band External Interface

The topology requirement for in-band external interface configuration, irrespective of the type of SKU is a VNF with SR-IOV interface mapped to a front panel port connected to another front panel port on the same device.

For example, on an NFX150 device, you can map the VNF SR-IOV interface to a front panel port (heth-0-x) directly with a cable connecting heth-0-x port to another heth-0-y port on the same device.

On an NFX250 or an NFX350 device, you can map the VNF SR-IOV interface an internal NIC hsxeX and map to a front panel port ge-0/0/X through a VLAN. Connect the ge-0/0/X with a direct cable to another ge-0/0/Y on the same device. Note that ge-0/0/Y and hsxeY are together in a different VLAN.

```
user@host#set vmhost external-interface in-band mapping interface interface-name
user@host#set vmhost external-interface in-band mapping interface virtual-function

user@host#set vmhost external-interface in-band family inet address ipv4_inet_address

user@host#set vmhost external-interface in-band family inet gateway ipv4_gateway

user@host#set vmhost external-interface in-band vlan-id vlan_id
```

For example, to configure an in-band external interface on an NFX150 device:

```
user@host#set vmhost external-interface in-band mapping interface heth-0-0
user@host#set vmhost external-interface in-band mapping interface virtual-function
user@host#set vmhost external-interface in-band family inet address 10.10.10.10/24 gateway
10.10.10.254
user@host#set vmhost external-interface in-band vlan-id 10
```

For example, to configure an in-band external interface on an NFX250 or an NFX350 device:

```
user@host#set vmhost external-interface in-band mapping interface hsxe0
user@host#set vmhost external-interface in-band mapping interface virtual-function
user@host#set vmhost external-interface in-band family inet address 10.10.10.10/24 gateway
10.10.10.254
user@host#set vmhost external-interface in-band vlan-id 10
```

Configuring for out-of-band External Interface

The topology requirement irrespective of the type of SKU is a VNF with an out-of-band management interface reachable on the same management network as the out-of-band management interface (eth0) of the NFX device.

```
user@host#set vmhost external-interface out-of-band family inet address ipv4_net_addres
user@host#set vmhost external-interface out-of-band family inet gateway ipv4_gateway
```

For example:

```
user@host#set vmhost external-interface out-of-band family inet address 10.204.97.175/20

user@host#set vmhost external-interface out-of-band family inet gateway 10.204.111.254/32
```

9

CHAPTER

Configuring Mapping of Address and Port with Encapsulation (MAP-E)

IN THIS CHAPTER

Mapping of Address and Port with Encapsulation on NFX Series Devices

IN THIS SECTION

Overview

Mapping of Address and Port with Encapsulation (MAP-E) is an IPv6 transition technique that encapsulates an IPv4 packet in an IPv6 address and carries it over an IPv4-over-IPv6 tunnel from MAP-E customer edge (CE) devices to MAP-E provider edge (PE) devices (also called as border relay [BR] devices) through an IPv6 routing topology, where the packets are detunneled for further processing.

MAP-E uses Network Address Port Translation (NAPT) features for restricting transport protocol ports, Internet Control Message Protocol (ICMP) identifiers, and fragment identifiers to the configured port sets. The existing NAPT features are enhanced to add MAP-E capability.

Benefits of MAP-E

In most cases, during IPv4 to IPv6 migration, only the IPv6 network is available. However, an IPv4 network is required for all residual IPv4 deployment. In scenarios where service providers have an IPv6 network and the LAN subscribers are not IPv6-capable, MAP-E supports IPv4 to IPv6 migration and deployment. MAP-E transports IPv4 packets across an IPv6 network using IP encapsulation. Encapsulation is done based on the mapping of IPv6 addresses to IPv4 addresses and to transport layer ports. Typically, during IPv6 transition, service providers might have a limited pool of public IPv4 addresses. MAP-E enables the sharing of public IPv4 addresses among multiple CE devices.

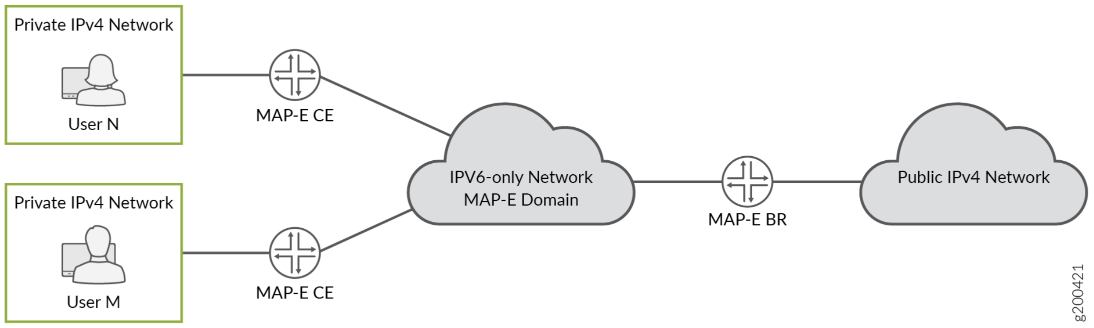
MAP-E Terminology

Terminology	Description
Border relay (BR)	The MAP-E-enabled provider edge device in a MAP domain. A BR device has at least one IPv6-enabled interface and one IPv4 interface connected to the native IPv4 network.
Embedded address (EA) bits	The EA bits in the IPv6 address identify an IPv4 prefix, IPv4 address, or a shared IPv4 address and a PSID.
MAP domain	One or more MAP-E customer edge devices and BR devices connected to the same virtual link.
MAP rule	A set of parameters that describe the mapping of an IPv4 prefix, IPv4 address, or a shared IPv4 address with an IPv6 prefix or IPv6 address. Each domain uses a different mapping rule set. Every MAP node must be provisioned with a basic mapping rule, which is used by the node to configure its IPv4 address, IPv4 prefix, or shared IPv4 address. The basic mapping rule is a forwarding mapping rule that is used for forwarding, where an IPv4 destination address and optionally a destination port is mapped to an IPv6 address.
MAP-E Customer Edge (CE)	The MAP-E-enabled customer edge device in a MAP deployment.
Port set ID (PSID)	Separate part of the transport layer port space that is denoted as the port set ID.
Softwire	Tunnel between two IPv6 endpoints to carry IPv4 packets or between two IPv4 endpoints to carry IPv6 packets.

MAP-E Functionality

Figure 8 on page 113 illustrates a simple MAP-E deployment scenario.

Figure 8: MAP-E Deployment



In a MAP-E network topology, there are two MAP-E CE devices, each connected to a private IPv4 host. The MAP-E CE devices are dual stack and are capable of NAPT. The MAP-E CE devices connect to a MAP-E BR device through an IPv6-only MAP-E network domain. The MAP-E BR device is dual stack and is connected to both a public IPv4 network and an IPv6 MAP-E network.

The MAP-E functionality is as follows:

1. The MAP-E CE devices are capable of NAPT. On receiving an IPv4 packet from the host, the MAP-E CE device performs NAT on the incoming IPv4 packets.

2. After NAT is performed, the IPv4 packets are then encapsulated into IPv6 packets by the MAP-E CE device, and are sent to the MAP-E BR device.

3. The IPv6 packets are transported through the IPv6-only service provider network and reach the MAP-E BR device.

4. The incoming IPv6 packets are decapsulated by the MAP-E BR and are routed to the IPv4 public network.

In the reverse path, the incoming IPv4 packets are encapsulated into IPv6 packets by the MAP-E BR device, and are routed to the MAP-E CE devices.

Configuring MAP-E on NFX Series Devices

IN THIS SECTION

Overview

This example describes how to configure Mapping of Address and Port with Encapsulation (MAP-E) functionality on NFX Series devices. For more information about MAP-E, see *Mapping of Address and Port with Encapsulation on NFX Series Devices*.

Requirements

This example uses the following hardware and software components:

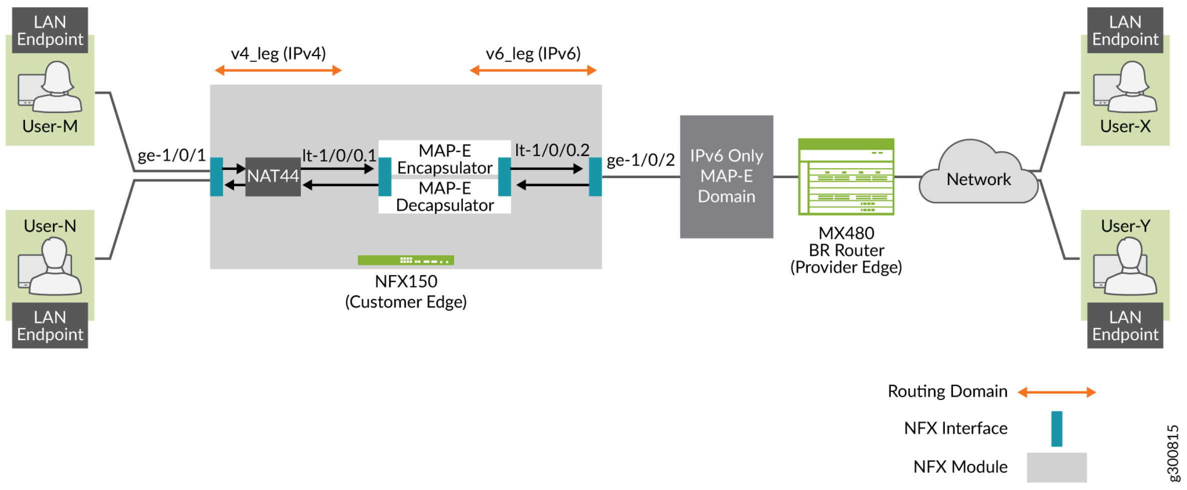
- NFX150 device running Junos OS Release 19.4R1, deployed as a customer edge (CE) device.
- MX480 device, deployed as a border relay (BR) device.
- Map physical interfaces to virtual interfaces. For more information, see Mapping Interfaces on NFX150 Devices.

Topology Overview

This topology shows how to configure MAP-E CE functionality on NFX Series devices. This topology also shows how the IPv4 packets from MAP-E CE devices are encapsulated and transported through an IPv4-over-IPv6 tunnel to MAP-E provider edge (PE) devices (also known as border relay [BR] devices) through an IPv6 routing topology, where the packets are detunneled for further processing. An MX Series device is used as the MAP-E BR device, which is a dual-stack device connected to both a public IPv4 network and an IPv6 MAP-E network.

Figure 9 on page 115 shows the MAP-E deployment on NFX Series devices.

Figure 9: MAP-E Deployment on NFX Series Device



Configure an NFX Series Device as a MAP-E CE Device

To configure an NFX Series device as a MAP-E customer edge device:

1. Configure the security policies and zones for applying different security measures on IPv4-facing interfaces and IPv6-facing interfaces. The following configuration adds LAN interface (ge-1/0/1) and WAN interface on the service provider end (ge-1/0/2) into relevant security zones and configures a policy to permit all traffic between these zones. The configuration also adds corresponding internal logical tunnel (lt) interface units into security zones.

```
user@host# set security policies global policy my_ce match source-address any
user@host# set security policies global policy my_ce match destination-address any
user@host# set security policies global policy my_ce match application any
user@host# set security policies global policy my_ce then permit
user@host# set security policies default-policy permit-all
user@host# set security zones security-zone v4zone host-inbound-traffic system-services all
user@host# set security zones security-zone v4zone host-inbound-traffic protocols all
user@host# set security zones security-zone v4zone interfaces ge-1/0/1.0
user@host# set security zones security-zone v4zone interfaces lt-1/0/0.1
user@host# set security zones security-zone v6zone host-inbound-traffic system-services all
user@host# set security zones security-zone v6zone host-inbound-traffic protocols all
user@host# set security zones security-zone v6zone interfaces ge-1/0/2.0
user@host# set security zones security-zone v6zone interfaces lt-1/0/0.2
```

2. Configure the interfaces to provide network connectivity and data flow. The following configuration assigns IPv4 address on LAN side and IPv6 on WAN side. The MTU on the IPv6 side must support maximum MTU.

```
user@host# set interfaces ge-1/0/1 unit 0 family inet address 10.10.10.1/24
user@host# set interfaces ge-1/0/2 mtu 9192
user@host# set interfaces ge-1/0/2 unit 0 family inet6 address 2001:db8:ffff::1/64
```

3. Configure both the logical tunnel interfaces. The logical tunnel interfaces act as internal endpoints to MAP-E encapsulator or decapsulator block in NFX series box. This separates the network traffic for IPv4 and IPv6. Here, lt-1/0/0 unit 1 terminates IPv4 traffic that is received on ge-1/0/1 and lt-1/0/0 unit 2 initiates IPv6 traffic to be sent out through ge-1/0/2. lt-1/0/0 unit 2 terminates IPv6 traffic that is received on ge-1/0/2 and lt-1/0/0 unit 1 initiates IPv4 traffuc to be sent out through ge-1/0/1.

```
user@host# set interfaces lt-1/0/0 mtu 9192
user@host# set interfaces lt-1/0/0 unit 1 encapsulation ethernet
user@host# set interfaces lt-1/0/0 unit 1 peer-unit 2
user@host# set interfaces lt-1/0/0 unit 1 family inet address 172.16.100.1/24
user@host# set interfaces lt-1/0/0 unit 1 family inet6 address 2001:db8:fffe::1/64
```

```
user@host# set interfaces lt-1/0/0 unit 2 encapsulation ethernet
user@host# set interfaces lt-1/0/0 unit 2 peer-unit 1
user@host# set interfaces lt-1/0/0 unit 2 family inet address 172.16.100.2/24
user@host# set interfaces lt-1/0/0 unit 2 family inet6 address 2001:db8:fffe::2/64
```

4. Configure the routing instances for the IPv4 and IPv6 network traffic domains inside NFX:

```
user@host# set routing-instances v4_leg routing-options rib v4_leg.inet.0 static route
198.51.100.0/24 next-hop 172.16.100.2
user@host# set routing-instances v4_leg routing-options rib v4_leg.inet.0 static route
203.0.113.0/24 next-hop 172.16.100.2
user@host# set routing-instances v4_leg routing-options rib v4_leg.inet.0 static route
192.0.2.0/24 next-hop 172.16.100.2
```

```
user@host# set routing-instances v4_leg instance-type virtual-router
user@host# set routing-instances v4_leg interface lt-1/0/0.1
```

```
user@host# set routing-instances v4_leg interface ge-1/0/1.0
user@host# set routing-instances v6_leg routing-options rib v6_leg.inet.0 static route
10.10.10.0/24 next-hop 172.16.100.1
user@host# set routing-instances v6_leg routing-options rib v6_leg.inet6.0 static route
2001:db8::a/128 next-hop 2001:db8:ffff::9
user@host# set routing-instances v6_leg routing-options rib v6_leg.inet6.0 static route
2001:db8:0012:3500::/56 next-hop 2001:db8:ffff::2
user@host# set routing-instances v6_leg routing-options rib v6_leg.inet6.0 static route
2001:db8:0012:3400::/56 next-hop 2001:db8:fffe::1
user@host# set routing-instances v6_leg instance-type virtual-router
user@host# set routing-instances v6_leg interface lt-1/0/0.2
user@host# set routing-instances v6_leg interface ge-1/0/2.0
```

5. Configure the MAP-E BMR and FMR rules to provide mapping between the IPv4 network and IPv6 network:

```
user@host# set security softwires map-e mapce1 br-address 2001:db8::a/128
user@host# set security softwires map-e mapce1 end-user-prefix 2001:db8:0012:3400::/56
user@host# set security softwires map-e mapce1 rule bmr rule-type BMR
user@host# set security softwires map-e mapce1 rule bmr ipv4-prefix 192.0.2.0/24
user@host# set security softwires map-e mapce1 rule bmr ipv6-prefix 2001:db8::/40
user@host# set security softwires map-e mapce1 rule bmr ea-bits-length 16
user@host# set security softwires map-e mapce1 rule bmr psid-offset 6
user@host# set security softwires map-e mapce1 role CE
user@host# set security softwires map-e mapce1 version 3
```

6. (Optional) Configure the `confidentiality` option for MAP-E if you want to hide the MAP-E parameters in show command output for non-super users:

```
user@host# set security softwires map-e confidentiality
```

For more information, see map-e.

7. Configure source NAT rule and NAT pool:

```
user@host# set security nat source pool my_mape allocation-domain mapce1
user@host# set security nat source pool my_mape allocation-domain allocation-rule bmr
```

```
user@host# set security nat source rule-set mape from zone v4zone
user@host# set security nat source rule-set mape to interface lt-1/0/0.1
user@host# set security nat source rule-set mape to interface ge-1/0/1.0
user@host# set security nat source rule-set mape rule r1 match source-address 10.10.10.0/24
user@host# set security nat source rule-set mape rule r1 match destination-address
10.10.10.0/24
user@host# set security nat source rule-set mape rule r1 match destination-address
198.51.100.0/24
user@host# set security nat source rule-set mape rule r1 match destination-address
203.0.113.0/24
user@host# set security nat source rule-set mape rule r1 match destination-address
192.0.2.0/24
user@host# set security nat source rule-set mape rule r1 then source-nat pool my_mape
user@host# set security nat source rule-set mape rule r1 then source-nat pool persistent-nat
permit any-remote-host
```

8. Commit the configuration:

```
user@host# commit
```

Configure an MX Series Device as a BR Device

To configure an MX Series device as a border relay device:

1. Configure the service set for MAP-E on the MX Series device:

```
user@host# set services service-set ss1 softwire-rules sw-rule1
user@host# set services service-set ss1 next-hop-service inside-service-interface si-1/0/0.1
user@host# set services service-set ss1 next-hop-service outside-service-interface si-1/0/0.2
```

2. Configure the MAP-E softwire concentrator and associated parameters. This creates a tunnel between two IPv6 endpoints to carry IPv4 packets or between two IPv4 endpoints to carry IPv6 packets.

```
user@host# set services softwire softwire-concentrator map-e mape-domain-1 softwire-address
2001:db8::a
user@host# set services softwire softwire-concentrator map-e mape-domain-1 ipv4-prefix
192.0.2.0/24
user@host# set services softwire softwire-concentrator map-e mape-domain-1 mape-prefix
```

```
2001:db8::/40
user@host# set services softwire softwire-concentrator map-e mape-domain-1 ea-bits-len 16
user@host# set services softwire softwire-concentrator map-e mape-domain-1 psid-offset 6
user@host# set services softwire softwire-concentrator map-e mape-domain-1 psid-length 8
user@host# set services softwire softwire-concentrator map-e mape-domain-1 mtu-v6 9192
user@host# set services softwire softwire-concentrator map-e mape-domain-1 version-03
user@host# set services softwire softwire-concentrator map-e mape-domain-1 v4-reassembly
user@host# set services softwire softwire-concentrator map-e mape-domain-1 v6-reassembly
user@host# set services softwire softwire-concentrator map-e mape-domain-1 disable-auto-route
```

3. Configure a softwire rule to specify the direction of traffic to be tunneled and the MAP-E softwire concentrator to be used:

```
user@host# set services softwire rule sw-rule1 match-direction input
user@host# set services softwire rule sw-rule1 term t1 then map-e mape-domain-1
```

4. Configure a service interface inside the dual-stack domain:

```
user@host# set interfaces si-1/0/0 unit 1 family inet6
user@host# set interfaces si-1/0/0 unit 1 service-domain inside
```

5. Configure a service interface outside the dual-stack domain:

```
user@host# set interfaces si-1/0/0 unit 2 family inet
user@host# set interfaces si-1/0/0 unit 2 service-domain outside
```

6. Configure the maximum transmission unit (MTU) on the BR interface:

```
user@host# set interfaces ge-1/1/2 mtu 9192
```

7. Configure the logical interfaces and assign the IPv4 and IPv6 addresses:

```
user@host# set interfaces ge-1/1/2 unit 0 family inet6 address 2001:db8:ffff::9/64
user@host# set interfaces ge-1/1/3 unit 0 family inet address 203.0.113.1/24
```

8. Configure the routing instances:

```
user@host# set routing-options rib inet6.0 static route 2001:db8::/40 next-hop si-1/0/0.1
user@host# set routing-options rib inet6.0 static route 2001:db8:0012:3400::/56 next-hop
2001:db8:ffff::1
```

```
user@host# set routing-options rib inet6.0 static route 2001:db8:0012:3500::/56 next-hop
2001:db8:ffff::2
user@host# set routing-options static route 192.0.2.0/24 next-hop si-1/0/0.2
user@host# set routing-options static route 198.51.100.0/24 next-hop si-1/0/0.2
user@host# set routing-options static route 203.0.113.0/24 next-hop si-1/0/0.2
```

9. Commit the configuration:

```
user@host# commit
```

Verify the MAP-E Configuration

IN THIS SECTION

Purpose

After completing the MAP-E configuration on an NFX Series device, you can verify the status of the MAP-E configuration.

Action

- Verify the status of the packet flow:

```
user@host> show security flow session
Session ID: 134218806, Policy name: my_ce/4, Timeout: 1800, Valid
  In: 10.10.10.2/57630 --> 203.0.113.2/22;tcp, Conn Tag: 0x0, If: ge-1/0/1.0, Pkts: 50,
Bytes: 5797,
  Out: 203.0.113.2/22 --> 192.0.2.18/20691;tcp, Conn Tag: 0x0, If: lt-1/0/0.1, Pkts: 33,
Bytes: 5697,
Session ID: 134218807, Policy name: my_ce/4, Timeout: 1800, Valid
  In: 2001:db8:12:3400:c0:2:1200:3400/1 --> 2001:db8::a/1;ipip, Conn Tag: 0x0, If:
lt-1/0/0.2, Pkts: 50, Bytes: 7797,
  Out: 2001:db8::a/1 --> 2001:db8:12:3400:c0:2:1200:3400/1;ipip, Conn Tag: 0x0, If:
ge-1/0/2.0, Pkts: 33, Bytes: 7017,
Total sessions: 2
```

- Verify whether the IPv4 and IPv6 addresses are configured correctly:

```
user@host> show security softwires map-e domain mapce1
Role                   : CE
Version                : 3
Domain Name            : mapce1
BR Address             : 2001:db8::a/128
End User Ipv6 prefix   : 2001:db8:12:3400::/56
BMR Mapping Rule :
    Rule Name             : bmr
    Rule Ipv4 Prefix      : 192.0.2.0/24
    Rule Ipv6 Prefix      : 2001:db8::/40
    PSID offset           : 6
    PSID length           : 8
    EA bit length         : 16
    Port SetID            : 0x34
    MAP-E Ipv4 address    : 192.0.2.18/32
    MAP-E Ipv6 address    : 2001:db8:12:3400:c0:2:1200:3400
```

- Verify the map rule statistics:

```
user@host> show security softwires map-e domain mapce1 statistics rule bmr
BMR Rule Name                :bmr
Encapsulated packets          :289
Decapsulated packets          :269
Encapsulation errors          :0
Decapsulation errors          :0
Encapsulated fragmentation :0
Decapsulated fragmentation :0
Invalid port set              :0
IPv6 address mismatch         :0
```

- View the details of the NAT source rule:

```
user@host> show security nat source rule all
Total rules: 1
Total referenced IPv4/IPv6 ip-prefixes: 5/0
source NAT rule: r1                     Rule-set: mape
  Rule-Id                    : 1
  Rule position              : 1
  From zone                  : v4zone
  To interface               : lt-1/0/0.1
                             : ge-1/0/1.0
  Match
    Source addresses         : 10.10.10.0      - 10.10.10.255
    Destination addresses    : 10.10.10.0      - 10.10.10.255
                               198.51.100.0    - 198.51.100.255
                               203.0.113.0     - 203.0.113.255
                               192.0.2.0       - 192.0.2.255
  Action                        : my_mape
    Persistent NAT type         : any-remote-host
    Persistent NAT mapping type : address-port-mapping
    Inactivity timeout          : 300
    Max session number          : 30
  Translation hits           : 1
    Successful sessions      : 1
    Failed sessions          : 0
  Number of sessions         : 1
```

- View the details of the NAT source pool:

```
user@host> show security nat source pool all
Total pools: 1
Pool name          : my_mape
Pool id            : 4
Routing instance   : default
Host address base  : 0.0.0.0
Map-e domain name  : mapce1
Map-e rule name    : bmr
PSID offset        : 6
PSID length        : 8
PSID               : 0x34
Port overloading   : 1
```

```
Address assignment : no-paired
Total addresses    : 1
Translation hits   : 1
Address range                       Single Ports   Twin Ports
         192.0.2.18 - 192.0.2.18        1              0
Total used ports   :                    1              0
```

- View the NAT source summary:

```
user@host> **show security nat source summary**
show security nat source summary
Total port number usage for port translation pool: 252
Maximum port number for port translation pool: 33554432
Total pools: 1
Pool                 Address                  Routing             PAT  Total
Name                 Range                    Instance                 Address
my_mape              192.0.2.18-192.0.2.18    default             yes  1

Total rules: 1
Rule name          Rule set       From              To                  Action
r1                 mape           v4zone            lt-1/0/0.1          my_mape
r1                                                  ge-1/0/1.0
```

- View the persistent NAT table:

```
user@host> **show security nat source persistent-nat-table all**
Internal                       Reflective                 Source     Type
Left_time/  Curr_Sess_Num/  Source
In_IP          In_Port I_Proto Ref_IP          Ref_Port R_Proto NAT Pool
Conf_time   Max_Sess_Num    NAT Rule
10.10.10.2      57630   tcp    192.0.2.18      20691     tcp      my_mape    any-remote-
host     -/300      1/30          r1
```

- View the softwire statistics on the MX Series device:

```
user@host> **show services inline softwire statistics mape**
Service PIC Name                                      si-1/0/0

Control Plane Statistics
     MAPE ICMPv6 echo requests to softwire concentrator          0
     MAPE ICMPv6 echo responses from softwire concentrator       0
     MAPE Dropped ICMPv6 packets to softwire concentrator        0

Data Plane Statistics (v6-to-v4)      Packets                  Bytes
     MAPE decaps                        15034                    1388760
     MAPE ICMP decap errors             0                        0
     MAPE decap spoof errors            0                        0
     MAPE v6 reassembled                0                        0
     MAPE dropped v6 fragments          0                        0
     MAPE v6 unsupp protocol drops      0                        0

Data Plane Statistics (v4-to-v6)      Packets                  Bytes
     MAPE encaps                        149544                   223527457
     MAPE ICMP encap errors             0                        0
     MAPE v6 mtu errors                 0                        0
     MAPE v4 reassembled                0                        0
     MAPE dropped v4 fragments          0                        0
```

Meaning

This section describes the output fields for the MAP-E configuration on NFX Series devices.

Role	MAP-E is deployed on a CE device. Currently, only the CE role is supported.
Version	MAP-E version: MAP-E draft-3.
BR address	Border router address to be used as the destination address in the absence of a matching FMR rule.
Rule name	Name of the BMR or FMR rule configured.
Rule IPv4 prefix	IPv4 prefix in the BMR or FMR rule.
Rule IPv6 prefix	IPv6 prefix in the BMR or FMR rule.
Port set ID	Port set identifier, used to algorithmically identify a set of ports exclusively assigned to a CE device.
PSID offset	Port set identifier offset, used to specify the range of excluded ports.
PSID length	Port set identifier length, used to specify the sharing ratio.
EA bit length	Embedded address bit length, used to specify part of the IPv4 address or the PSID.

10

CHAPTER

Configuring Cross-Connect

IN THIS CHAPTER

Configuring Cross-Connect on NFX Series Devices

You can configure cross-connect feature on NFX Series devices when there is a requirement to switch the traffic from one interface to the other interface without using the MAC-based forwarding rule. The cross-connect feature allows you to connect any two NFV backplane interfaces, which can be either two VNF interfaces or one VNF and one NIC interface to switch traffic between the interfaces. You can either switch all traffic or traffic belonging to a particular VLAN, unidirectionally or bidirectionally.

NOTE:

- On NFX150 and NFX250 NextGen devices, virtual interfaces can be hsxe0, hsxe1, or ge-1/0/x connected to NFV backplane.
- On NFX350 devices, virtual interfaces can be hsxe0, hsxe1, hsxe2, hsxe3, or ge-1/0/x connected to NFV backplane.
- On NFX250 devices, virtual interfaces can be hsxe0 or hsxe1.

NOTE: Cross-connect feature is not supported on VNF interfaces that are SR-IOV interfaces.

You can configure the following types of cross-connect on your device:

- Port cross-connect—You can configure port cross-connect between two OVS interfaces. All traffic is switched between two OVS interfaces in the cross-connect configuration.
- VLAN cross-connect—You can specify a VLAN ID in the cross-connect rule to redirect the traffic from a particular VLAN on an OVS interface to a different interface. All VLAN cross-connects have higher precedence over port cross-connects.
- Untagged cross-connect—You can choose to redirect the untagged frames out of a trunk port to a different destination port by specifying the VLAN ID as *none* in the cross-connect CLI. During this process, you can additionally add a VLAN tag by mentioning the VLAN ID in the other entry of the cross connect rule.

Single leg cross-connect feature allows configuration of single entry on either VNF interface or virtual interface and configure other entry at any later point of time. Single leg cross-connect status is down until the other entry is configured and the interface status of both the entries is up. In a single leg cross-connect configuration, traffic flow is not present until the other entry of the cross-connect is configured.

Unidirectional cross-connect feature allows the traffic to be forwarded conditionally or unconditionally in a single direction. Traffic flow in the opposite direction follows the MAC-based forwarding rule.

The cross-connect feature supports the following:

- Unconditional and conditional cross-connect between two interfaces on the NFV backplane.
- VLAN-based traffic forwarding between two interfaces on the NFV backplane support the following functions:
  - Allows to switch traffic based on a VLAN ID.
  - Allows to switch traffic flow from trunk to access interfaces.
  - Allows to switch traffic flow from access to trunk interfaces.
  - Allows to add a VLAN tag to the traffic, remove the VLAN tag from the traffic, and rewrite the existing VLAN tag to a different tag while switching the traffic between the interfaces.

NOTE:

- When two VNF interfaces are part of cross-connect configuration, and if one of the VNF interfaces is disabled, then all traffic from the VNF interface which is up and is part of cross-connect, is dropped.

Example: Configuring Cross-Connect on NFX350 Devices

IN THIS SECTION

This example shows how to configure the cross-connect feature on NFX350 devices.

Requirements

This example uses an NFX350 device running Junos OS Release 19.4R1.

Overview

IN THIS SECTION

The cross-connect feature enables traffic switching between any two VNF interfaces. You can bidirectionally switch either all traffic or traffic belonging to a particular VLAN between any two VNF interfaces.

NOTE: This feature does not support unidirectional traffic flow.

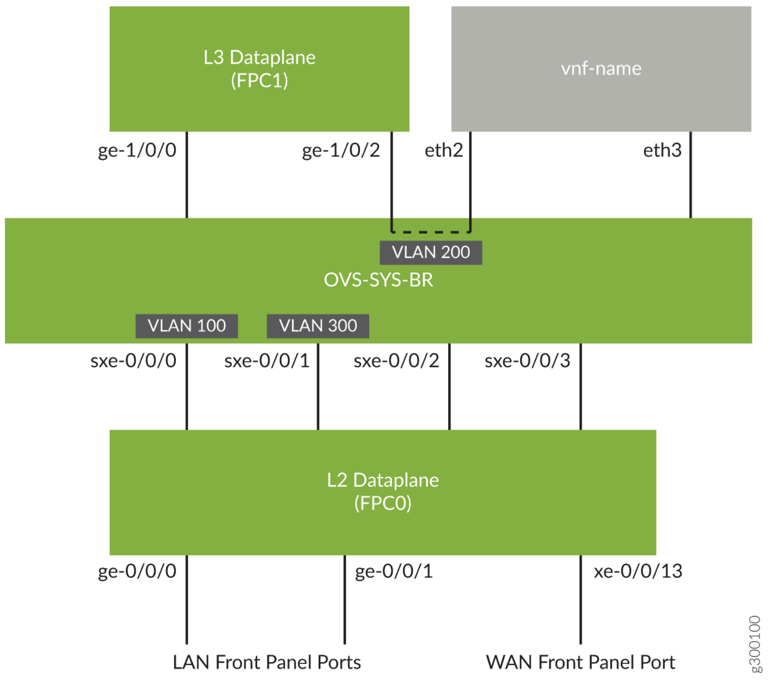
The cross-connect feature supports the following:

- Port cross-connect between two VNF interfaces for all network traffic.
- VLAN-based traffic forwarding between VNF interfaces that support the following functions:
  - Provides an option to switch traffic based on a VLAN ID.
  - Supports VLAN PUSH, POP, and SWAP operations.
  - Supports network traffic flow from trunk to access port through the POP operation.
  - Supports network traffic flow from access to trunk ports through the PUSH operation.

Topology

This example uses the topology shown in Figure 10 on page 129.

Figure 10: Configuring Cross-Connect



Configuration

IN THIS SECTION

Configure VLANs

Step-by-Step Procedure

1. Configure VLANs for the LAN-side interfaces.

```
user@host# set vlans vlan100 vlan-id 100
```

2. Configure a VLAN for the WAN-side interface.

```
user@host# set vlans vlan300 vlan-id 300
```

Configure the Layer 2 Datapath

Step-by-Step Procedure

1. Configure the LAN-side front panel ports and add them to the LAN-side VLAN.

```
user@host# set interfaces ge-0/0/0 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces ge-0/0/0 unit 0 family ethernet-switching vlan members vlan100
user@host# set interfaces ge-0/0/1 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces ge-0/0/1 unit 0 family ethernet-switching vlan members vlan100
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan100
```

2. Configure the internal-facing interfaces as trunk ports and add them to the WAN-side VLAN. The internal-facing interfaces are typically trunk ports as they must support traffic from multiple front panel ports and VLANs.

```
user@host# set interfaces xe-0/0/13 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces xe-0/0/13 unit 0 family ethernet-switching vlan members vlan300
user@host# set interfaces sxe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/1 unit 0 family ethernet-switching vlan members vlan300
```

Configure the Layer 3 Datapath

Step-by-Step Procedure

1. Configure VLAN tagging on ge-1/0/0:

```
user@host# set interfaces ge-1/0/0 vlan-tagging
user@host# set interfaces ge-1/0/0 unit 0 vlan-id 100
user@host# set interfaces ge-1/0/0 unit 0 family inet address 192.0.2.1/24
```

2. Configure VLAN tagging on ge-1/0/2:

```
user@host# set interfaces ge-1/0/2 vlan-tagging
user@host# set interfaces ge-1/0/2 unit 0 vlan-id 200
user@host# set interfaces ge-1/0/2 unit 0 family inet address 203.0.113.2/24
```

Configure the VNF

Step-by-Step Procedure

1. Launch the VNF:

```
user@host# set virtual-network-functions vnf-name image /var/public/centos-updated_1.img
user@host# set virtual-network-functions vnf-name image image-type raw
```

2. Specify the number of CPUs required for the VNF:

```
user@host# set virtual-network-functions vnf-name virtual-cpu count 1
```

3. Pin a virtual CPU to a physical CPU:

```
user@host# set virtual-network-functions vnf-name virtual-cpu 0 physical-cpu 2
```

4. Create host VLANs:

```
user@host# set vmhost vlans vlan200 vlan-id 200
user@host# set vmhost vlans vlan300 vlan-id 300
```

5. Configure the VNF interfaces as trunk ports and add them to the LAN-side VLAN:

```
user@host# set virtual-network-functions vnf-name interfaces eth2 mapping vlan mode trunk
user@host# set virtual-network-functions vnf-name interfaces eth2 mapping vlan members vlan200
user@host# set virtual-network-functions vnf-name interfaces eth3 mapping vlan members vlan300
```

6. Specify the memory allocation for the VNF:

```
user@host# set virtual-network-functions vnf-name memory size 1048576
```

Configure Cross-Connect

Step-by-Step Procedure

1. Configure cross-connect:

```
user@host# set vmhost cross-connect c1 virtual-interface ge-1/0/2
user@host# set vmhost cross-connect c1 virtual-network-function vnf-name interface eth2
```

Verify the Configuration

IN THIS SECTION

Verifying the Control Plane Configuration

Purpose

Verify the control plane configuration:

Action

- Verify the VLANs configured.

```
user@host > show vlans
Routing instance        VLAN name           Tag          Interfaces
default-switch          default             1

default-switch          vlan100             100
                                                         ge-0/0/0.0*
                                                         ge-0/0/1.0*
                                                         sxe-0/0/0.0*
default-switch          vlan200             200
                                                         sxe-0/0/1.0*
                                                         xe-0/0/12.0*
default-switch          vlan300             300
                                                         sxe-0/0/1.0*
                                                         xe-0/0/13.0*
```

- Verify that the VLANs and VLAN memberships are correct by using the `show vmhost vlans` command.

```
user@host> show vmhost vlans
Routing instance        VLAN name           Tag          Interfaces
vmhost                  vlan200             200
                                                         vnf-name_eth2.0
vmhost                  vlan300             300
                                                         vnf-name_eth3.0
```

- Verify that the VNF is operational. The `State` field shows `Running` for VNFs that are up.

```
user@host> show virtual-network-functions vnf-name
ID       Name                                            State      Liveliness
```

```
--------------------------------------------------------------------------------
3        vnf-name                                            Running    alive
```

The `Liveliness` field of the VNF indicates whether the internal management IP address of the VNF is accessible from the Junos Control Plane (JCP).

To view more details of the VNF:

```
user@host> show virtual-network-functions vnf-name detail
Virtual Network Function Information
------------------------------------


Id:                   3
Name:                 vnf-name
State:                Running
Liveliness:           alive
IP Address:           192.0.2.100
VCPUs:                1
Maximum Memory:       1048576 KiB
Used Memory:          1048576 KiB
Used 1G Hugepages:    0
Used 2M Hugepages:    0
Error:                None
```

Verifying the Data Plane Configuration

Purpose

Verify the data plane configuration.

Action

- Verify the status of the Layer 2 (ge-0/0/x) and Layer 3 (ge-1/0/x) interfaces.

```
user@host> show interfaces interface-name statistics
```

For example:

```
user@host> show interfaces ge-0/0/0 statistics
Physical interface: ge-0/0/0, Enabled, Physical link is Up
```

```
  Interface index: 149, SNMP ifIndex: 517
  Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Link-mode: Full-duplex, Speed: 
1000mbps, Duplex: Full-Duplex, BPDU Error: None,
  Loop Detect PDU Error: None, Ethernet-Switching Error: None, MAC-REWRITE Error: None, 
Loopback: Disabled, Source filtering: Disabled,
  Flow control: Enabled, Auto-negotiation: Enabled, Remote fault: Online, IEEE 802.3az Energy 
Efficient Ethernet: Disabled, Auto-MDIX: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 12 supported, 12 maximum usable queues
  Current address: 30:7c:5e:4c:78:03, Hardware address: 30:7c:5e:4c:78:03
  Last flapped   : 2018-11-26 11:03:32 UTC (04:15:32 ago)
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)
  Active alarms  : None
  Active defects : None
  PCS statistics                      Seconds
    Bit errors                            0
    Errored blocks                        0
  Ethernet FEC statistics              Errors
    FEC Corrected Errors                   0
    FEC Uncorrected Errors                 0
    FEC Corrected Errors Rate              0
    FEC Uncorrected Errors Rate            0
  PRBS Statistics : Disabled
  Interface transmit statistics: Disabled

  Logical interface ge-0/0/0.0 (Index 330) (SNMP ifIndex 519)
    Flags: Up SNMP-Traps 0x24024000 Encapsulation: Ethernet-Bridge
    Input packets : 0
    Output packets: 0
    Protocol eth-switch, MTU: 1514
      Flags: Trunk-Mode
```

```
user@host> show interfaces ge-1/0/2 statistics
Physical interface: ge-1/0/2, Enabled, Physical link is Up
  Interface index: 167, SNMP ifIndex: 547
  Link-level type: Ethernet, MTU: 1518, LAN-PHY mode, Link-mode: Half-duplex, Speed: 
1000mbps, BPDU Error: None, Loop Detect PDU Error: None,
  Ethernet-Switching Error: None, MAC-REWRITE Error: None, Loopback: Disabled, Source 
```

```
filtering: Disabled, Flow control: Enabled, Auto-negotiation: Enabled,
  Remote fault: Online
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Current address: 30:7c:5e:4c:78:1d, Hardware address: 30:7c:5e:4c:78:1d
  Last flapped   : 2018-11-26 11:03:45 UTC (04:19:57 ago)
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)
  Active alarms  : None
  Active defects : None
  PCS statistics                      Seconds
    Bit errors                             0
    Errored blocks                         0
  Ethernet FEC statistics              Errors
    FEC Corrected Errors                    0
    FEC Uncorrected Errors                  0
    FEC Corrected Errors Rate               0
    FEC Uncorrected Errors Rate             0
  PRBS Statistics : Disabled
  Interface transmit statistics: Disabled

  Logical interface ge-1/0/2.0 (Index 334) (SNMP ifIndex 550)
    Flags: Up SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.200 ]  Encapsulation: ENET2
    Input packets : 0
    Output packets: 0
    Security: Zone: Null
    Protocol inet, MTU: 1500
    Max nh cache: 75000, New hold nh limit: 75000, Curr nh cnt: 0, Curr new hold cnt: 0, NH
drop cnt: 0
      Flags: Sendbcast-pkt-to-re
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 203.0.113/24, Local: 203.0.113.2, Broadcast: 203.0.113.255

  Logical interface ge-1/0/2.32767 (Index 335) (SNMP ifIndex 551)
    Flags: Up SNMP-Traps 0x4004000 VLAN-Tag [ 0x0000.0 ]  Encapsulation: ENET2
    Input packets : 0
    Output packets: 0
    Security: Zone: Null
```

- Verify the status of the OVS interfaces.

```
user@host> show vmhost network nfv-back-plane
Network Name : ovs-sys-br

   Interface : ovs-sys-br
   Type : internal, Link type : Full-Duplex, MAC : 52:86:3c:df:9c:44
   MTU : [], Link State :down, Admin State : down
   IPV4 : None, Netmask : None
   IPV6 : None, IPV6 netmask : None
       Rx-packets :       0
       Rx-drops   :       0
       Rx-errors  :       0
       Tx-packets :       1
       Tx-drops   :       1
       Tx-errors  :       0

   Interface : dpdk0
   Type : dpdk, Link type : Full-Duplex, MAC : 02:09:c0:e2:b9:08
   MTU : [], Link State :up, Admin State : up
   IPV4 : None, Netmask : None
   IPV6 : None, IPV6 netmask : None
       Rx-packets :       0
       Rx-drops   :       0
       Rx-errors  :       0
       Tx-packets :       1
       Tx-drops   :       0
       Tx-errors  :       0

   Interface : dpdk1
   Type : dpdk, Link type : Full-Duplex, MAC : 02:09:c0:83:39:72
   MTU : [], Link State :up, Admin State : up
   IPV4 : None, Netmask : None
   IPV6 : None, IPV6 netmask : None
       Rx-packets :       0
       Rx-drops   :       0
       Rx-errors  :       0
       Tx-packets :       0
       Tx-drops   :       0
       Tx-errors  :       0

   Interface : l3_h_ge_1_0_0
```

```
Type : dpdkvhostuser, Link type : Full-Duplex, MAC : 00:00:00:00:00:00
MTU : [], Link State :up, Admin State : up
IPV4 : None, Netmask : None
IPV6 : None, IPV6 netmask : None
    Rx-packets :       0
    Rx-drops   :       0
    Rx-errors  :       0
    Tx-packets :       0
    Tx-drops   :       0
    Tx-errors  :       0

Interface : l3_h_ge_1_0_2
Type : dpdkvhostuser, Link type : Full-Duplex, MAC : 00:00:00:00:00:00
MTU : [], Link State :down, Admin State : up
IPV4 : None, Netmask : None
IPV6 : None, IPV6 netmask : None
    Rx-packets :       0
    Rx-drops   :       0
    Rx-errors  :       0
    Tx-packets :       0
    Tx-drops   :       0
    Tx-errors  :       0

Interface : vnf-name_eth2
Type : dpdkvhostuser, Link type : Full-Duplex, MAC : 00:00:00:00:00:00
MTU : 1500, Link State :down, Admin State : up
IPV4 : None, Netmask : None
IPV6 : None, IPV6 netmask : None
    Rx-packets :       0
    Rx-drops   :       0
    Rx-errors  :       0
    Tx-packets :       0
    Tx-drops   :       0
    Tx-errors  :       0

Interface : vnf-name_eth3
Type : dpdkvhostuser, Link type : Full-Duplex, MAC : 00:00:00:00:00:00
MTU : 1500, Link State :down, Admin State : up
IPV4 : None, Netmask : None
IPV6 : None, IPV6 netmask : None
    Rx-packets :       0
    Rx-drops   :       0
    Rx-errors  :       0
```

```
        Tx-packets :       0
        Tx-drops   :       0
        Tx-errors  :       0
```

11

CHAPTER

Configuring High Availability

IN THIS CHAPTER

Chassis Cluster on NFX350 Devices

IN THIS SECTION

A chassis cluster, where two devices operate as a single device, provides high availability (HA) on NFX350 devices. Chassis clustering involves the synchronizing of configuration files and the dynamic runtime session states between the devices, which are part of the chassis cluster setup.

NFX350 Chassis Cluster Overview

IN THIS SECTION

You can configure NFX350 devices to operate in cluster mode by connecting and configuring a pair of devices to operate like a single node, providing redundancy at the device, interface, and service level.

When two devices are configured to operate as a *chassis cluster*, each device becomes a node of that cluster. The two nodes back up each other, with one node acting as the primary device and the other node acting as the secondary device, ensuring stateful failover of processes and services when the system or hardware fails. If the primary device fails, the secondary device takes over the processing of traffic.

The nodes of a cluster are connected together through two links called control link and fabric link. The devices in a chassis cluster synchronize the configuration, kernel, and PFE session states across the cluster to facilitate high availability, failover of stateful services, and load balancing.

- Control link—Synchronizes the configuration between the nodes. When you submit configuration statements to the cluster, the configuration is automatically synchronized over the control interface.

  To create a control link in a chassis cluster, connect the ge-0/0/0 interface on one node to the ge-0/0/0 interface on the second node.

  NOTE: You can use only the ge-0/0/0 interface to create a control link.

- Fabric link (data link)—Forwards traffic between the nodes. Traffic arriving on a node that needs to be processed on the other node is forwarded over the fabric link. Similarly, traffic processed on a node that needs to exit through an interface on the other node is forwarded over the fabric link.

  You can use any interface except the ge-0/0/0 to create a fabric link.

Chassis Cluster Modes

The chassis cluster can be configured in active/passive or active/active mode.

- **Active/passive mode**—In active/passive mode, the transit traffic passes through the primary node while the backup node is used only in the event of a failure. When a failure occurs, the backup device becomes the primary and takes over all forwarding tasks.

- **Active/active mode**—In active/active mode, the transit traffic passes through both nodes all the time.

Chassis Cluster Interfaces

The chassis cluster interfaces include:

- Redundant Ethernet (reth) interface—A pseudo-interface that includes a physical interface from each node of a cluster. The reth interface of the active node is responsible for passing the traffic in a chassis cluster setup.

  A reth interface must contain, at minimum a pair of Gigabit Ethernet interfaces that are referred to as child interfaces of the redundant Ethernet interface (the redundant parent). If two or more child interfaces from each node are assigned to the redundant Ethernet interface, a redundant Ethernet interface link aggregation group can be formed.

  NOTE: You can configure a maximum of 128 reth interfaces on NFX350 devices.

- Control interface—An interface that provides the control link between the two nodes in the cluster. This interface is used for routing updates and for control plane signal traffic, such as heartbeat and threshold information that trigger node failover.

  NOTE: By default, the ge-0/0/0 interface is configured as the dedicated control interface on NFX350 devices. Therefore, you cannot apply any configuration to ge-0/0/0 in HA mode.

- Fabric interface—An interface that provides the physical connection between two nodes of a cluster. A fabric interface is formed by connecting a pair of Ethernet interfaces back-to-back (one from each node). The Packet Forwarding Engines of the cluster uses this interface to transmit transit traffic and to synchronize the runtime state of the data plane software. You must specify the physical interfaces to be used for the fabric interface in the configuration.

Chassis Cluster Limitation

Redundant LAG (RLAG) of reth member interfaces of the same node is not supported. A reth interface with more than one child interface per node is called RLAG.

Example: Configuring a Chassis Cluster on NFX350 Devices

IN THIS SECTION

This example shows how to set up chassis clustering on NFX350 devices.

Requirements

Before you begin:

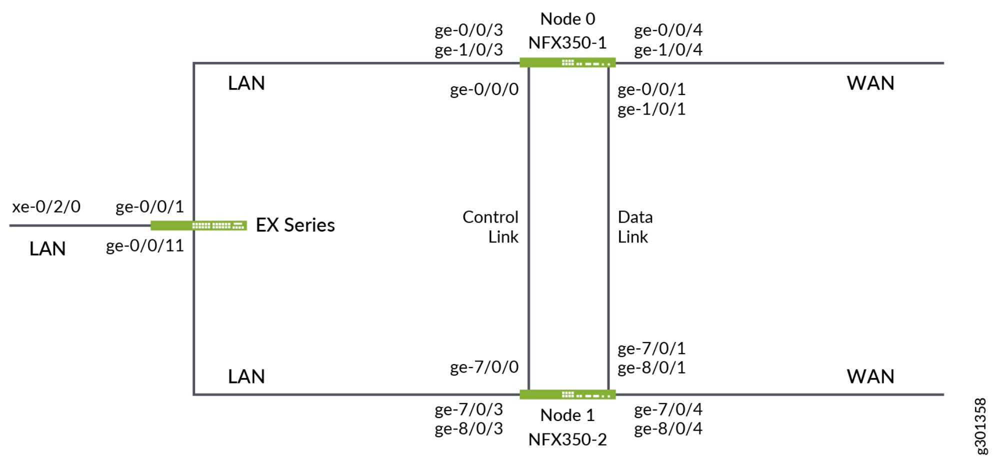
- Physically connect the two devices and ensure that they are the same NFX350 model.
- Ensure that both devices are running the same Junos OS version
- Remove all interface mapping for the control port ge-0/0/0 on both the nodes.
- Connect the dedicated control port ge-0/0/0 on node 0 to the ge-0/0/0 port on node 1.
- Connect the fabric port on node 0 to the fabric port on node 1.

Overview

Figure 11 on page 144 shows the topology used in this example. This example shows how to set up basic active/passive chassis clustering. One device actively maintains control of the chassis cluster. The other device passively maintains its state for cluster failover capabilities in case the active device becomes inactive.

NOTE: This example does not describe in detail miscellaneous configurations such as how to configure security features. They are essentially the same as they would be for standalone configurations.

Figure 11: NFX350 Chassis Cluster



Configuration

IN THIS SECTION

Configuring a Chassis Cluster

Step-by-Step Procedure

1. Configure the cluster ID on both the nodes and reboot the devices. A reboot is required to enter into cluster mode after the cluster ID and node ID are set.

   NOTE: You must enter the operational mode to issue the commands on both devices.

```
user@host1> set chassis cluster cluster-id 1 node 0 reboot
user@host2> set chassis cluster cluster-id 1 node 1 reboot
```

   The cluster-id is the same on both devices, but the node ID must be different because one device is node 0 and the other device is node 1. The range for the cluster-id is 0 through 255 and setting it to 0 is equivalent to disabling cluster mode.

2. Verify that the chassis cluster is configured successfully:

-
```
user@host1> show chassis cluster status
Monitor Failure codes:
    CS  Cold Sync monitoring        FL  Fabric Connection monitoring
    GR  GRES monitoring             HW  Hardware monitoring
    IF  Interface monitoring        IP  IP monitoring
    LB  Loopback monitoring         MB  Mbuf monitoring
    NH  Nexthop monitoring          NP  NPC monitoring
    SP  SPU monitoring              SM  Schedule monitoring
    CF  Config Sync monitoring      RE  Relinquish monitoring

Cluster ID: 1
```

```
Node   Priority Status               Preempt Manual   Monitor-failures

Redundancy group: 0 , Failover count: 0
node0  1        primary              no      no       None
node1  1        secondary            no      no       None
```

-
  ```
  root@host1> show chassis cluster information
  node0:
  --------------------------------------------------------------------------
  Redundancy Group Information:

      Redundancy Group 0 , Current State: primary, Weight: 255

          Time            From                 To                   Reason
          Mar 15 11:33:47 hold                 secondary            Hold timer expired
          Mar 15 11:34:03 secondary            primary              Only node present

  Chassis cluster LED information:
      Current LED color: Green
      Last LED change reason: No failures

  node1:
  --------------------------------------------------------------------------
  Redundancy Group Information:

      Redundancy Group 0 , Current State: secondary, Weight: 255

          Time            From                 To                   Reason
          Mar 15 12:14:49 hold                 secondary            Hold timer expired

  Chassis cluster LED information:
      Current LED color: Green
      Last LED change reason: No failures
  ```

After the chassis cluster is set up, you can enter the configuration mode and perform all the configurations on the primary node, node0.

3. Configure the host names and the out-of-band management IP addresses for nodes 0 and 1:

```
user@host1# set groups node0 system host-name NFX250NG-1
user@host1# set groups node0 interfaces fxp0 unit 0 family inet address 172.16.100.1/24
```

```
user@host2# set groups node1 system host-name NFX250NG-2
user@host2# set groups node1 interfaces fxp0 unit 0 family inet address 172.16.100.2/24
```

   If you are accessing the device from a different subnet other than the one configured for the out-of-band management, then set up a static route:

```
user@host1# set routing-options static route 198.51.100.0/24 next-hop 172.16.0.0
user@host1# set routing-options static route 203.0.113.0/24 next-hop 172.16.0.0
```

4. Configure a backup router to access the router from an external network for the out-of-band management

```
user@host1# set groups node0 system backup-router 172.16.0.0
user@host1# set groups node0 system backup-router destination 172.0.0.0/8
user@host1# set groups node0 system backup-router destination 203.0.0.0/8
user@host1# set groups node1 system backup-router 172.16.0.0
user@host1# set groups node1 system backup-router destination 172.0.0.0/8
user@host1# set groups node1 system backup-router destination 203.0.0.0/8
```

Configure Fabric interfaces

Step-by-Step Procedure

The ge-0/0/0 interface is a pre-defined control link. Therefore, you should select any other interface on the device to configure a fabric interface. For example, in the below configuration, ge-0/0/1 is used as the fabric interface.

1. Connect one end of the Ethernet cable to ge-0/0/1 on NFX250NG-1 device and the other end of the cable to ge-0/0/1 on NFX250NG-2 device.

2. Map physical LAN to virtual WAN port:

```
user@host1> set vmhost virtualization-options interfaces ge-8/0/1
user@host1> set vmhost virtualization-options interfaces ge-1/0/1
```

3. Configure front panel (L2) interfaces corresponding to fabric interface:

```
user@host1# set interfaces ge-0/0/1 mtu 9192
user@host1# set interfaces ge-0/0/1 unit 0 family ethernet-switching interface-mode access
user@host1# set interfaces ge-0/0/1 unit 0 family ethernet-switching vlan members vlan100
```

```
user@host1# set interfaces sxe-0/0/0 mtu 9192
user@host1# set interfaces sxe-0/0/0 unit 0 family ethernet-switching interface-mode trunk
user@host1# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan100
user@host1# set vlans vlan100 vlan-id 100
```

4. Configure L3 interfaces as fabric member:

```
user@host1# set chassis cluster fabric-member ge-1/0/1 vlan-id 100
user@host1# set interfaces fab0 fabric-options member-interfaces ge-1/0/1
user@host1# set groups fab chassis cluster fabric-member ge-1/0/1 vlan-id 100
user@host1# set groups fab chassis cluster fabric-member ge-8/0/1 vlan-id 100
user@host1# set groups fab interfaces fab0 fabric-options member-interfaces ge-1/0/1
user@host1# set groups fab interfaces fab1 fabric-options member-interfaces ge-8/0/1
user@host1# set groups fab vmhost virtualization-options interfaces ge-1/0/1
user@host1# set groups fab vmhost virtualization-options interfaces ge-8/0/1
```

5. Configure data path for fabric interfaces:

```
user@host1# set groups fab interfaces sxe-7/0/0 unit 0 family ethernet-switching vlan members
vlan100
user@host1# set groups fab interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members
vlan100
user@host1# set groups fab interfaces ge-0/0/9 mtu 9000
user@host1# set groups fab interfaces ge-0/0/9 unit 0 family ethernet-switching interface-
mode access
user@host1# set groups fab interfaces ge-0/0/9 unit 0 family ethernet-switching vlan members
vlan100
```

```
user@host1# set groups fab interfaces ge-7/0/9 mtu 9000
user@host1# set groups fab interfaces ge-7/0/9 unit 0 family ethernet-switching interface-
mode access
user@host1# set groups fab interfaces ge-7/0/9 unit 0 family ethernet-switching vlan members
vlan100
user@host1# set groups fab vlan vlan100 vlan-id 100
user@host1# set apply-groups fab
```

6. Configure port peering for fabric and reth members. Port peering ensures that when a LAN interface controlled by the Layer 2 dataplane (FPC0) fails, the corresponding interface on the Layer 3 dataplane (FPC1) is marked down and vice versa. This helps in the failover of the corresponding redundant group to the secondary node.

```
user@host1# set groups node1 chassis cluster redundant-interface ge-8/0/1 mapping-interface
ge-7/0/1
user@host1# set groups node0 chassis cluster redundant-interface ge-1/0/1 mapping-interface
ge-0/0/1
```

7. Enable the system to perform control link recovery automatically. After it determines that the control link is healthy, the system issues an automatic reboot on the node that was disabled when the control link failed. When the disabled node reboots, it rejoins the cluster.

```
user@host1# set chassis cluster control-link-recovery
```

Configure Redundant Groups and Redundant Interfaces

Step-by-Step Procedure

1. Configure redundancy groups 1 and 2. The **redundancy-group 1** RG controls the control plane and it determines the primary node. The **redundancy-group 2** RG controls the data plane and includes the data plane ports. Each node has interfaces in a redundancy group.

   As part of redundancy group configuration, you must also define the priority for control plane and data plane—which device is preferred for the control plane, and which device is preferred for the data plane. For chassis clustering, higher priority is preferred. The higher number takes precedence.

In this configuration, node 0 is the active node as it is associated with **redundancy-group 1**. reth0 is member of **redundancy-group 1** and reth1 is member of **redundancy-group 2**. You must configure all changes in the cluster through node 0. If node 0 fails, then node 1 will be the active node.

```
user@host1# set chassis cluster reth-count 4
user@host1# set chassis cluster redundancy-group 1 node 0 priority 200
user@host1# set chassis cluster redundancy-group 1 node 1 priority 100
user@host1# set chassis cluster redundancy-group 2 node 0 priority 200
user@host1# set chassis cluster redundancy-group 2 node 1 priority 100
user@host1# set chassis cluster redundancy-group 1 preempt
user@host1# set chassis cluster redundancy-group 2 preempt
```

2. Map physical LAN to virtual WAN port for reth members:

```
user@host1# set vmhost virtualization-options interfaces ge-1/0/3
user@host1# set vmhost virtualization-options interfaces ge-1/0/4
user@host1# set vmhost virtualization-options interfaces ge-8/0/3
user@host1# set vmhost virtualization-options interfaces ge-8/0/4
```

3. Configure front panel (L2) interfaces corresponding to reth interface:

```
user@host1# set interfaces ge-0/0/3 unit 0 family ethernet-switching interface-mode access
user@host1# set interfaces ge-0/0/3 unit 0 family ethernet-switching vlan members vlan300
```

```
user@host1# set interfaces ge-0/0/4 unit 0 family ethernet-switching interface-mode access
user@host1# set interfaces ge-0/0/4 unit 0 family ethernet-switching vlan members vlan400
```

```
user@host1# set interfaces ge-7/0/3 unit 0 family ethernet-switching interface-mode access
user@host1# set interfaces ge-7/0/3 unit 0 family ethernet-switching vlan members vlan300
```

```
user@host1# set interfaces ge-7/0/4 unit 0 family ethernet-switching interface-mode access
user@host1# set interfaces ge-7/0/4 unit 0 family ethernet-switching vlan members vlan400
```

```
user@host1# set interfaces sxe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
user@host1# set interfaces sxe-0/0/1 unit 0 family ethernet-switching vlan members vlan300
user@host1# set interfaces sxe-0/0/1 unit 0 family ethernet-switching vlan members vlan400
```

```
user@host1# set interfaces sxe-7/0/1 unit 0 family ethernet-switching interface-mode trunk
user@host1# set interfaces sxe-7/0/1 unit 0 family ethernet-switching vlan members vlan300
user@host1# set interfaces sxe-7/0/1 unit 0 family ethernet-switching vlan members vlan400
```

```
user@host1# set vlans vlan300 vlan-id 300
user@host1# set vlans vlan400 vlan-id 400
```

4. Configure WAN (L3) interfaces as reth member:

```
user@host1# set interfaces ge-1/0/3 gigether-options redundant-parent reth0
user@host1# set interfaces ge-8/0/3 gigether-options redundant-parent reth0
user@host1# set interfaces ge-1/0/4 gigether-options redundant-parent reth1
user@host1# set interfaces ge-8/0/4 gigether-options redundant-parent reth1
```

5. Configure reth interfaces:

- Configure reth0:

```
user@host1# set interfaces reth0 vlan-tagging
user@host1# set interfaces reth0 redundant-ether-options redundancy-group 1
user@host1# set interfaces reth0 unit 0 vlan-id 300
user@host1# set interfaces reth0 unit 0 family inet address 192.0.2.0/24
```

- Configure reth1:

```
user@host1# set interfaces reth1 vlan-tagging
user@host1# set interfaces reth1 redundant-ether-options redundancy-group 2
user@host1# set interfaces reth1 unit 0 vlan-id 400
user@host1# set interfaces reth1 unit 0 family inet address 198.51.100.0/24
```

6. Configure interface monitoring for reth interfaces members:

```
user@host1# set chassis cluster redundancy-group 1 interface-monitor ge-1/0/3 weight 255
user@host1# set chassis cluster redundancy-group 1 interface-monitor ge-8/0/3 weight 255
user@host1# set chassis cluster redundancy-group 2 interface-monitor ge-1/0/4 weight 255
user@host1# set chassis cluster redundancy-group 2 interface-monitor ge-8/0/4 weight 255
```

7. Configure port peering for reth interface members:

```
user@host1# set groups node1 chassis cluster redundant-interface ge-8/0/3 mapping-interface
ge-7/0/3
user@host1# set groups node1 chassis cluster redundant-interface ge-8/0/4 mapping-interface
ge-7/0/4
user@host1# set groups node0 chassis cluster redundant-interface ge-1/0/3 mapping-interface
ge-0/0/3
user@host1# set groups node0 chassis cluster redundant-interface ge-1/0/4 mapping-interface
ge-0/0/4
```

8. Configure security policies to allow traffic from LAN to WAN, and from WAN to LAN:

```
user@host1# set security policies default-policy permit-all
user@host1# set security zones security-zone trust host-inbound-traffic system-services all
```

```
user@host1# set security zones security-zone trust host-inbound-traffic protocols all
user@host1# set security zones security-zone trust interfaces all
```

Verification

IN THIS SECTION

Verifying Chassis Cluster Status

Purpose

Verify the status of the chassis cluster and its interfaces.

Action

From operational mode, issue the following commands:

- Verify the status of the cluster:

```
root@host1> show chassis cluster status
Monitor Failure codes:
    CS  Cold Sync monitoring        FL  Fabric Connection monitoring
    GR  GRES monitoring             HW  Hardware monitoring
    IF  Interface monitoring        IP  IP monitoring
    LB  Loopback monitoring         MB  Mbuf monitoring
    NH  Nexthop monitoring          NP  NPC monitoring
    SP  SPU monitoring              SM  Schedule monitoring
    CF  Config Sync monitoring      RE  Relinquish monitoring
    IS  IRQ storm

Cluster ID: 1
Node   Priority Status              Preempt Manual   Monitor-failures

Redundancy group: 0 , Failover count: 1
node0  1        primary             no      no       None
node1  1        secondary           no      no       None
```

```
Redundancy group: 1 , Failover count: 1
node0  200      primary             yes     no       None
node1  100      secondary           yes     no       None

Redundancy group: 2 , Failover count: 1
node0  200      primary             yes     no       None
node1  100      secondary           yes     no       None
```

- Verify the status of the redundancy groups:

```
root@host1> show chassis cluster information
node0:
--------------------------------------------------------------------------
Redundancy Group Information:

    Redundancy Group 0 , Current State: primary, Weight: 255

        Time            From                 To                   Reason
        Jun  8 11:24:14 hold                 secondary            Hold timer expired
        Jun  8 11:24:30 secondary            primary              Only node present

    Redundancy Group 1 , Current State: primary, Weight: 255

        Time            From                 To                   Reason
        Jun  8 11:24:14 hold                 secondary            Hold timer expired
        Jun  8 11:24:30 secondary            primary              Only node present

    Redundancy Group 2 , Current State: primary, Weight: 255

        Time            From                 To                   Reason
        Jun  8 11:24:14 hold                 secondary            Hold timer expired
        Jun  8 11:24:30 secondary            primary              Only node present

Chassis cluster LED information:
    Current LED color: Green
    Last LED change reason: No failures

node1:
--------------------------------------------------------------------------
Redundancy Group Information:

    Redundancy Group 0 , Current State: secondary, Weight: 255
```

```
        Time            From                 To                   Reason
        Jun  8 11:25:24 hold                 secondary            Hold timer expired

    Redundancy Group 1 , Current State: secondary, Weight: 255

        Time            From                 To                   Reason
        Jun  8 11:25:24 hold                 secondary            Hold timer expired

    Redundancy Group 2 , Current State: secondary, Weight: 255

        Time            From                 To                   Reason
        Jun  8 11:25:23 hold                 secondary            Hold timer expired

Chassis cluster LED information:
    Current LED color: Green
    Last LED change reason: No failures
```

- Verify the status of the interfaces:

```
root@host1> **show chassis cluster interfaces**
Control link status: Up

Control interfaces:
    Index   Interface   Monitored-Status   Internal-SA   Security
    0       em1         Up                 Disabled      Disabled

Fabric link status: Up

Fabric interfaces:
    Name    Child-interface    Status                    Security
                               (Physical/Monitored)
    fab0    ge-1/0/1           Up   / Up                 Disabled
    fab0
    fab1    ge-8/0/1           Up   / Up                 Disabled
    fab1

Redundant-ethernet Information:
    Name         Status      Redundancy-group
    reth0        Up          1
    reth1        Up          2
    reth2        Down        Not configured
    reth3        Down        Not configured

Redundant-pseudo-interface Information:
    Name         Status      Redundancy-group
    lo0          Up          0

Interface Monitoring:
    Interface         Weight    Status                    Redundancy-group
                                (Physical/Monitored)
    ge-8/0/3          255       Up  /  Up                 1
    ge-1/0/3          255       Up  /  Up                 1
    ge-8/0/4          255       Up  /  Up                 2
    ge-1/0/4          255       Up  /  Up                 2
```

- Verify the status of the port-peering interfaces:

```
root@host1> show chassis cluster port-peering
node0:
--------------------------------------------------------------------------


Port peering interfaces:
    Backend L3                 Mapped Peer L2
    Interface  Status          Interface  Status
    ge-1/0/3    Up             ge-0/0/3   Up
    ge-1/0/4    Up             ge-0/0/4   Up
    ge-1/0/1    Up             ge-0/0/1   Up

node1:
--------------------------------------------------------------------------


Port peering interfaces:
    Backend L3                 Mapped Peer L2
    Interface  Status          Interface  Status
    ge-8/0/3    Up             ge-7/0/3   Up
    ge-8/0/4    Up             ge-7/0/4   Up
    ge-8/0/1    Up             ge-7/0/1   Up
```

RELATED DOCUMENTATION

Monitoring of Global-Level Objects in a Chassis Cluster

Upgrading or Disabling a Chassis Cluster on NFX350 Devices

IN THIS SECTION

Upgrading Individual Devices in a Chassis Cluster Separately

Devices in a chassis cluster can be upgraded separately one at a time.

NOTE: During this type of chassis cluster upgrade, a service disruption of about 3 to 5 minutes occurs.

To upgrade each device in a chassis cluster separately:

1. Load the new image file on node 0.
2. Perform the image upgrade without rebooting the node by entering:

```
user@host> request vmhost software add image_name
```

3. Load the new image file on node 1.
4. Repeat Step 2.
5. Reboot both nodes simultaneously.

Disabling a Chassis Cluster

If you want to operate the device as a standalone device or remove a node from a chassis cluster, you must disable the chassis cluster.

To disable a chassis cluster, enter the following command:

```
{primary:node1}
user@host> set chassis cluster disable reboot
```

After the system reboots, the chassis cluster is disabled.

NOTE: You can also disable the chassis cluster by setting the cluster-id to zero on both the nodes:

```
user@host>set chassis cluster cluster-id 0 node 0 reboot
user@host>set chassis cluster cluster-id 0 node 1 reboot
```

12

CHAPTER

Configuring Media Access Control Security (MACsec)

IN THIS CHAPTER

Configuring MACsec on NFX350 Devices

IN THIS SECTION

Media Access Control Security (MACsec) is an industry-standard security technology that provides secure communication for almost all types of traffic on Ethernet links. MACsec provides point-to-point security on Ethernet links between directly-connected nodes and is capable of identifying and preventing most security threats, including denial of service, intrusion, man-in-the-middle, masquerading, passive wiretapping, and playback attacks. MACsec is standardized in IEEE 802.1AE.

You can configure MACsec to secure point-to-point Ethernet links connecting switches, or on Ethernet links connecting a switch to a host device such as a PC, phone, or server. Each point-to-point Ethernet link that you want to secure using MACsec must be configured independently. You can enable MACsec on switch-to-switch links using static secure association key (SAK) security mode or static connectivity association key (CAK) security mode. Both processes are provided in this document.

BEST PRACTICE: We recommend enabling MACsec using static CAK security mode on switch-to-switch links. Static CAK security mode ensures security by frequently refreshing to a new random secure association key (SAK) and by only sharing the SAK between the two devices on the MACsec-secured point-to-point link. Additionally, some optional MACsec features—replay protection, SCI tagging, and the ability to exclude traffic from MACsec—are only available in static CAK security mode.

BEST PRACTICE: When enabling MACsec, we recommend that you examine your interface MTU, adjusting it for MACsec overhead, which is 32 bytes.

Configuring MACsec Using Static Connectivity Association Key (CAK) Mode (Recommended for Enabling MACsec on Switch-to-Switch Links)

You can enable MACsec using static connectivity association key (CAK) security mode or static secure association keys (SAK) security mode on a point-to-point Ethernet link connecting switches. This procedure shows you how to configure MACsec using static CAK security mode.

BEST PRACTICE: We recommend enabling MACsec using static CAK security mode on switch-to-switch links. Static CAK security mode ensures security by frequently refreshing to a new random secure association key (SAK) and by only sharing the SAK between the two devices on the MACsec-secured point-to-point link. Additionally, some optional MACsec features—replay protection, SCI tagging, and the ability to exclude traffic from MACsec—are only available for MACsec-secured switch-to-switch connections that are enabled using static CAK security mode.

When you enable MACsec using static CAK security mode, a pre-shared key is exchanged between the switches on each end of the point-to-point Ethernet link. The pre-shared key includes a connectivity association name (CKN) and a connectivity association key (CAK). The CKN and CAK are configured by the user in the connectivity association and must match on both ends of the link to initially enable MACsec.

After the pre-shared keys are exchanged and verified, the MACsec Key Agreement (MKA) protocol, which enables and maintains MACsec on the link, is enabled. The MKA is responsible for selecting one of the two switches on the point-to-point link as the key server. The key server then creates a randomized security key that is shared only with the other device over the MACsec-secured link. The randomized security key enables and maintains MACsec on the point-to-point link. The key server will continue to periodically create and share a randomly-created security key over the point-to-point link for as long as MACsec is enabled.

NOTE: If the MACsec session is terminated due to a link failure, when the link is restored, the MKA key server elects a key server and generates a new SAK.

You enable MACsec using static CAK security mode by configuring a connectivity association on both ends of the link. All configuration is done within the connectivity association but outside of the secure channel. Two secure channels—one for inbound traffic and one for outbound traffic—are automatically created when using static CAK security mode. The automatically-created secure channels do not have any user-configurable parameters that cannot already be configured in the connectivity association.

NOTE: MACsec functions only when it is configured from switch to switch. The behavior of MACsec functionality is not determinable if it is configured from switch to host.

To configure MACsec using static CAK security mode to secure a switch-to-switch Ethernet link:

1. Create a connectivity association. You can skip this step if you are configuring an existing connectivity association.

   ```
   [edit security macsec]
   user@host# set connectivity-association connectivity-association-name
   ```

   For instance, to create a connectivity association named `ca1`, enter:

   ```
   [edit security macsec]
   user@host# set connectivity-association ca1
   ```

2. Configure the MACsec security mode as `static-cak` for the connectivity association:

   ```
   [edit security macsec]
   user@host# set connectivity-association connectivity-association-name security-mode static-
   cak
   ```

   For instance, to configure the MACsec security mode to `static-cak` on connectivity association ca1:

   ```
   [edit security macsec]
   user@host# set connectivity-association ca1 security-mode static-cak
   ```

3. Create the pre-shared key by configuring the connectivity association key name (CKN) and connectivity association key (CAK):

   ```
   [edit security macsec]
   user@host# set connectivity-association connectivity-association-name pre-shared-key ckn
   hexadecimal-number
   user@host# set connectivity-association connectivity-association-name pre-shared-key cak
   hexadecimal-number
   ```

   A pre-shared key is exchanged between directly-connected links to establish a MACsec-secure link. The pre-shared-key includes the CKN and the CAK. The CKN is a 64-digit hexadecimal number and the CAK is a 32-digit hexadecimal number. The CKN and the CAK must match on both ends of a link to create a MACsec-secured link.

NOTE: To maximize security, we recommend configuring all 64 digits of a CKN and all 32 digits of a CAK.

If you do not configure all 64 digits of a CKN or all 32 digits of a CAK, all remaining digits will be auto-configured to 0. However, you will receive a warning message when you commit the configuration.

After the pre-shared keys are successfully exchanged and verified by both ends of the link, the MACsec Key Agreement (MKA) protocol is enabled and manages the secure link. The MKA protocol then elects one of the two directly-connected devices as the key server. The key server then shares a random security with the other device over the MACsec-secure point-to-point link. The key server will continue to periodically create and share a random security key with the other device over the MACsec-secured point-to-point link as long as MACsec is enabled.

To configure a CKN of `37c9c2c45ddd012aa5bc8ef284aa23ff6729ee2e4acb66e91fe34ba2cd9fe311` and CAK of `228ef255aa23ff6729ee664acb66e91f` on connectivity association ca1:

```
[edit security macsec]
user@host# set connectivity-association ca1 pre-shared-key ckn
37c9c2c45ddd012aa5bc8ef284aa23ff6729ee2e4acb66e91fe34ba2cd9fe311
user@host# set connectivity-association ca1 pre-shared-key cak
228ef255aa23ff6729ee664acb66e91f
```

NOTE: MACsec is not enabled until a connectivity association is attached to an interface. See the final step of this procedure to attach a connectivity association to an interface.

4. (Optional) Set the MKA key server priority:

```
[edit security macsec connectivity-association connectivity-association-name]
user@host# set mka key-server-priority priority-number
```

Specifies the key server priority used by the MKA protocol to select the key server. The device with the lower *priority-number* is selected as the key server.

The default *priority-number* is 16.

If the `key-server-priority` is identical on both sides of the point-to-point link, the MKA protocol selects the interface with the lower MAC address as the key server. Therefore, if this statement is not configured in the connectivity associations at each end of a MACsec-secured point-to-point link, the interface with the lower MAC address becomes the key server.

To change the key server priority to 0 to increase the likelihood that the current device is selected as the key server when MACsec is enabled on the interface using connectivity association `ca1`:

```
[edit security macsec connectivity-association ca1]
user@host# set mka key-server-priority 0
```

To change the key server priority to 255 to decrease the likelihood that the current device is selected as the key server in connectivity association ca1:

```
[edit security macsec connectivity-association ca1]
user@host# set mka key-server-priority 255
```

5. (Optional) Set the MKA transmit interval:

```
[edit security macsec connectivity-association connectivity-association-name]
user@host# set mka transmit-interval interval
```

The MKA transmit interval setting sets the frequency for how often the MKA protocol data unit (PDU) is sent to the directly connected device to maintain MACsec connectivity on the link. A lower *interval* increases bandwidth overhead on the link; a higher *interval* optimizes MKA protocol communication.

The default *interval* is 2000ms. We recommend increasing the interval to 6000 ms in high-traffic load environments. The transmit interval settings must be identical on both ends of the link when MACsec using static CAK security mode is enabled.

For instance, if you wanted to increase the MKA transmit interval to 6000 milliseconds when connectivity association ca1 is attached to an interface:

```
[edit security macsec connectivity-association ca1]
user@host# set mka transmit-interval 6000
```

6. (Optional) Disable MACsec encryption:

```
[edit security macsec connectivity-association connectivity-association-name]
user@host# set no-encryption
```

   Encryption is enabled for all traffic entering or leaving the interface when MACsec is enabled using static CAK security mode, by default.

   When encryption is disabled, traffic is forwarded across the Ethernet link in clear text. You are able to view unencrypted data in the Ethernet frame traversing the link when you are monitoring it. The MACsec header is still applied to the frame, however, and all MACsec data integrity checks are run on both ends of the link to ensure the traffic sent or received on the link has not been tampered with and does not represent a security threat.

7. Assign an encryption algorithm:

   You can encrypt all traffic entering or leaving the interface using any of the following MACsec encryption algorithms:

   - gcm-aes-128—GCM-AES-128 cipher suite without extended packet numbering (XPN) mode
   - gcm-aes-256—GCM-AES-256 cipher suite without XPN
   - gcm-aes-xpn-128—GCM-AES-XPN_128 cipher suite with XPN mode
   - gcm-aes-xpn-256—GCM-AES-XPN_256 cipher suite with XPN mode

   If MACsec encryption is enabled and if no encryption algorithm is specified, the default (gcm-aes-128) encryption algorithm is used without XPN mode.

```
[edit security macsec connectivity-association connectivity-association-name]
user@host# set cipher-suite (gcm-aes-128 | gcm-aes-256 | gcm-aes-xpn-128 | gcm-aes-xpn-256)
```

   For instance, if you wanted to encrypt using gcm-aes-xpn-128 algorithm in the connectivity association named ca1:

```
[edit security macsec connectivity-association ca1]
user@host# set cipher-suite gcm-aes-xpn-128
```

8. (Optional) Set an offset for all packets traversing the link:

```
[edit security macsec connectivity-association connectivity-association-name]
user@host# set offset (0 | 30 | 50)
```

For instance, if you wanted to set the offset to 30 in the connectivity association named `ca1`:

```
[edit security macsec connectivity-association ca1]
user@host# set offset 30
```

The default offset is 0. All traffic in the connectivity association is encrypted when encryption is enabled and an `offset` is not set.

When the offset is set to 30, the IPv4 header and the TCP/UDP header are unencrypted while encrypting the rest of the traffic. When the offset is set to 50, the IPv6 header and the TCP/UDP header are unencrypted while encrypting the rest of the traffic.

You would typically forward traffic with the first 30 or 50 octets unencrypted if a feature needed to see the data in the octets to perform a function, but you otherwise prefer to encrypt the remaining data in the frames traversing the link. Load balancing features, in particular, typically need to see the IP and TCP/UDP headers in the first 30 or 50 octets to properly load balance traffic.

9. (Optional) Enable replay protection.

```
[edit security macsec connectivity-association connectivity-association-name]
user@host# set replay-protect replay-window-size number-of-packets
```

When MACsec is enabled on a link, an ID number is assigned to each packet on the MACsec-secured link.

When replay protection is enabled, the receiving interface checks the ID number of all packets that have traversed the MACsec-secured link. If a packet arrives out of sequence and the difference between the packet numbers exceeds the replay protection window size, the packet is dropped by the receiving interface. For instance, if the replay protection window size is set to five and a packet assigned the ID of 1006 arrives on the receiving link immediately after the packet assigned the ID of 1000, the packet that is assigned the ID of 1006 is dropped because it falls outside the parameters of the replay protection window.

Replay protection is especially useful for fighting man-in-the-middle attacks. A packet that is replayed by a man-in-the-middle attacker on the Ethernet link will arrive on the receiving link out of sequence, so replay protection helps ensure the replayed packet is dropped instead of forwarded through the network.

Replay protection should not be enabled in cases where packets are expected to arrive out of order.

You can require that all packets arrive in order by setting the replay window size to 0.

To enable replay protection with a window size of five on connectivity association `ca1`:

```
[edit security macsec connectivity-association ca1]
user@host# set replay-protect replay-window-size 5
```

10. (Optional) Exclude a protocol from MACsec:

```
[edit security macsec connectivity-association connectivity-association-name]
user@host# set exclude-protocol protocol-name
```

For instance, if you did not want Link Level Discovery Protocol (LLDP) to be secured using MACsec:

```
[edit security macsec connectivity-association connectivity-association-name]
user@host# set exclude-protocol lldp
```

When this option is enabled, MACsec is disabled for all packets of the specified protocol—in this case, LLDP—that are sent or received on the link.

11. Assign the connectivity association to an interface:

NOTE: When you configure MACsec on an NFX350 device and the connectivity is Layer 3, the IP address is assigned on the virtual interface and MACsec is configured on the front panel of the device.

```
[edit security macsec]
user@host# set interfaces interface-names connectivity-association connectivity-association-
name
```

Assigning the connectivity association to an interface is the final configuration step to enabling MACsec on an interface.

For instance, to assign connectivity association ca1 to interface xe-0/0/1:

```
[edit security macsec]
user@host# set interfaces xe-0/1/0 connectivity-association ca1
```

MACsec using static CAK security mode is not enabled until a connectivity association on the opposite end of the link is also configured, and contains pre-shared keys that match on both ends of the link.

13

CHAPTER

Configuring Link Aggregation Control Protocol

IN THIS CHAPTER

Example: Configuring Link Aggregation Control Protocol on NFX Series Devices

IN THIS SECTION

This example shows how to configure Link Aggregation Control Protocol (LACP) on NFX Series devices.

Requirements

This example uses an NFX Series device running Junos OS Release 20.4R1.

Before you configure LACP, be sure you have:

- An NFX250 NextGen or NFX350 device running Junos OS Release 20.4R1, which is not in a Chassis Cluster.

Overview

Starting in Junos OS Release 20.4, LACP is supported on NFX250 NextGen and NFX350 devices. LACP is a method to bundle several physical interfaces to form one logical interface. You can configure LACP on the front panel interfaces of the devices.

NOTE: Only Layer 2 LACP is supported on NFX250 NextGen and NFX350 devices.

Configuration

IN THIS SECTION

Configure LACP

CLI Quick Configuration

To quickly configure this section of the example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from the configuration mode.

```
set interfaces ge-0/0/6 gigether-options 802.3ad ae0
set interfaces ge-0/0/7 gigether-options 802.3ad ae0
set interfaces ae0 aggregated-ether-options lacp active periodic fast
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set vlan vlan1000 vlan-id 1000
set interfaces ae0 unit 0 family ethernet-switching vlan members vlan1000
set chassis aggregated-devices ethernet device-count 5
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy.

NOTE:

- Static LAG and LACP are supported on NFX250 NextGen and NFX350 devices.
- You can add a minimum of one link and maximum of eight links to a LAG.

To configure LACP:

1. Configure the interfaces for ae0.

```
[edit]
```

```
user@host# set interfaces ge-0/0/6 gigether-options 802.3ad ae0
```

```
user@host# set interfaces ge-0/0/7 gigether-options 802.3ad ae0
```

2. Configure LACP for ae0 and configure periodic transmission of LACP packets.

```
[edit]
```

```
user@host# set interfaces ae0 aggregated-ether-options lacp active periodic fast
```

3. Configure ae0 as a trunk port.

```
[edit]
```

```
user@host# set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
```

4. Configure the VLAN.

```
[edit]
```

```
user@host# set vlan vlan1000 vlan-id 1000
```

5. Add the ae0 interface to the VLAN.

```
[edit]
```

```
user@host# set interfaces ae0 unit 0 family ethernet-switching vlan members vlan1000
```

6. Create LAG members.

```
[edit]
```

```
user@host# set chassis aggregated-devices ethernet device-count 5
```

7. If you are done configuring the device, commit the configuration.

```
[edit]
```

```
user@host# commit
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
```

```
user@host# show interfaces
ge-0/0/6 {
   ether-options {
     802.3ad ae0;
   }
}
ge-0/0/7 {
   ether-options {
      802.3ad ae0;
   }
}
ae0 {
  vlan- tagging;
  aggregated-ether-options {
     lacp {
        active;
        periodic fast;
      }
  }
  unit 0 {
     family ethernet-switching {
        interface-mode trunk;
        vlan {
          members vlan1000;
        }
      }
    }
}
```

Verification

IN THIS SECTION

Verifying the LACP configuration

Purpose

Display LACP statistics for aggregated Ethernet interfaces.

Action

From operational mode, enter the `show lacp statistics interfaces ae0` command.

```
user@host> show lacp statistics interfaces ae0
Aggregated interface: ae0
    LACP Statistics:       LACP Rx     LACP Tx   Unknown Rx   Illegal Rx
      ge-0/0/6                1352        2035            0            0
      ge-0/0/7                1352        2056            0            0
```

Meaning

The output shows LACP statistics for each physical interface associated with the aggregated Ethernet interface, such as the following:

- The LACP received counter that increments for each normal hello packet received
- The number of LACP transmit packet errors logged
- The number of unrecognized packet errors logged
- The number of invalid packets received

Use the following command to clear the statistics and see only new changes:

```
user@host# clear lacp statistics interfaces ae0
```

Verifying LACP Aggregated Ethernet Interfaces

Purpose

Display LACP status information for aggregated Ethernet interfaces.

Action

From operational mode, enter the `show lacp interfaces ae0` command.

```
user@host> show lacp interfaces ae0
Aggregated interface: ae0
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      ge-0/0/6       Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      ge-0/0/6     Partner    No    No   Yes  Yes  Yes   Yes     Fast    Passive
      ge-0/0/7       Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      ge-0/0/7     Partner    No    No   Yes  Yes  Yes   Yes     Fast    Passive
    LACP protocol:        Receive State  Transmit State          Mux State
      ge-0/0/6                  Current   Fast periodic Collecting distributing
      ge-0/0/7                  Current   Fast periodic Collecting distributing
```

Meaning

The output shows aggregated Ethernet interface information, including the following information:

- The LACP state—Indicates whether the link in the bundle is an actor (local or near-end of the link) or a partner (remote or far-end of the link).
- The LACP mode—Indicates whether both ends of the aggregated Ethernet interface are enabled (active or passive)—at least one end of the bundle must be active.
- The periodic link aggregation control PDU transmit rate.
- The LACP protocol state—Indicates the link is up if it is collecting and distributing packets.

14

CHAPTER

Configuring Service Chaining

IN THIS CHAPTER

Example: Configuring Service Chaining Using VLANs on NFX350 Devices

IN THIS SECTION

This example shows how to configure service chaining using VLANs on the host bridge.

Requirements

This example uses an NFX350 device running Junos OS Release 19.4R1.

Before you configure service chaining, ensure that you have installed and instantiated the relevant virtual network functions (VNFs), assigned the corresponding interfaces, and configured the resources.

Overview

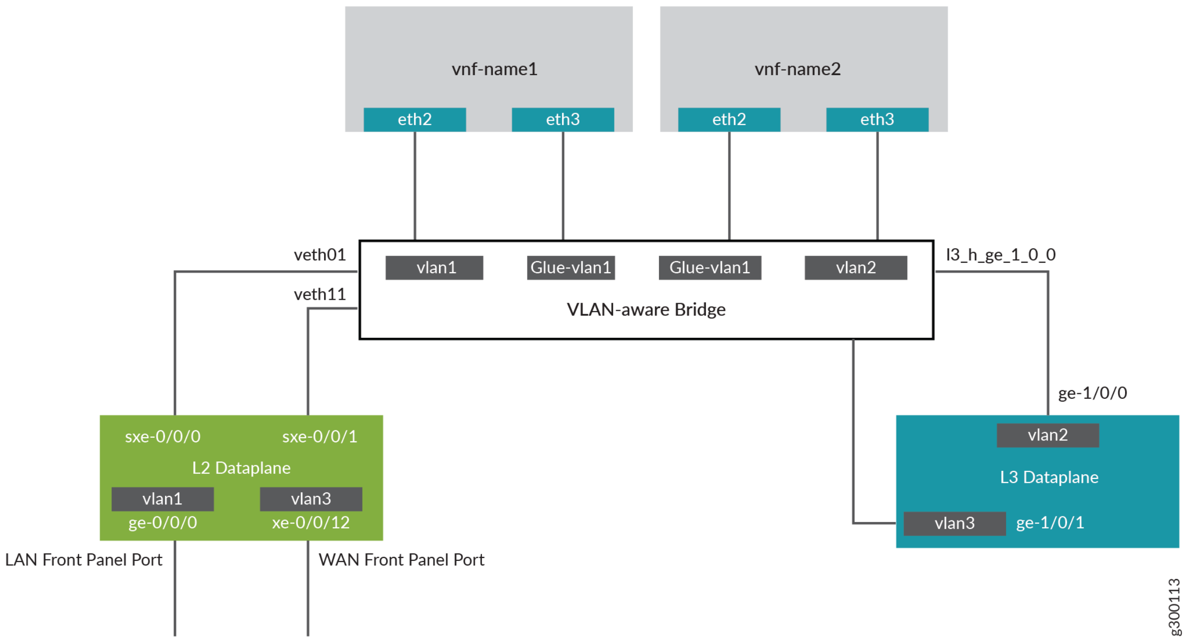
IN THIS SECTION

Service chaining on a device enables multiple services or VNFs on the traffic that flows through the device. This example explains how to configure the various layers of the device to enable traffic to enter the device, flow through two service VNFs, and exit the device.

Topology

This example uses a single NFX350 device running Junos OS, as shown in Figure 12 on page 178.

Figure 12: Configuring Service Chaining Using VLANs



This example is configured using the Junos Control Plane (JCP). The key configuration elements include:

- Front panel ports
- Internal-facing ports
- VNF interfaces, which use the naming format eth*#*(where *#* ranges from 0 through 9)
- VLANs to provide bridging between the static interfaces (sxe) and VNF interfaces

Configuration

IN THIS SECTION

Configuring the JCP Interfaces

Step-by-Step Procedure

To configure the interfaces:

1. Log in to the CLI.

```
user@host:~ # cli
user@host>
user@host> configure
[edit]
user@host#
```

2. Map the Layer 3 interface to the Open vSwitch (OVS).

```
user@host# set vmhost virtualization-options interfaces ge-1/0/1
```

3. Configure a VLAN for the LAN-side interfaces.

```
user@host# set vlans vlan1 vlan-id 77
```

4. Configure the LAN-side front panel port and add it to the LAN-side VLAN.

   The LAN-side port is typically an access port, but can be a trunk port if required.

```
user@host# set interfaces ge-0/0/0.0 family ethernet-switching vlan members vlan1
```

5. Configure the LAN-side internal-facing interface as a trunk port and add it to the LAN-side VLAN.

The internal-facing interfaces are typically trunk ports as they must support traffic from multiple front panel ports and VLANs.

```
user@host# set interfaces sxe-0/0/0.0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/0.0 family ethernet-switching vlan members vlan1
```

6. Configure the WAN-side internal-facing interface as a trunk port and add it to the WAN-side VLAN.

```
user@host# set interfaces sxe-0/0/1.0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/1.0 family ethernet-switching vlan members vlan3
```

7. Configure the WAN-side front panel port and add it to the WAN-side VLAN.

```
user@host# set interfaces xe-0/0/12.0 family ethernet-switching interface-mode access
user@host#  set interfaces xe-0/0/12.0 family ethernet-switching vlan members vlan3
```

8. Configure a VLAN for the WAN-side interface.

```
user@host# set vlans vlan3 vlan-id 1178
```

9. Configure VLAN tagging on the WAN-side external facing interface and assign an IP address.

```
user@host# set interfaces ge-1/0/1 vlan-tagging
user@host# set interfaces ge-1/0/1.0 vlan-id 1178
user@host# set interfaces ge-1/0/1.0 family inet address 192.0.2.1/24
```

10. Configure the WAN-side internal facing interface as a VLAN-tagged interface and assign an IP address.

```
user@host# set interfaces ge-1/0/0 vlan-tagging
user@host# set interfaces ge-1/0/0.0 vlan-id 1177
user@host# set interfaces ge-1/0/0.0 family inet address 203.0.113.2/24
```

11. Commit the configuration.

```
user@host# commit
```

Results

From configuration mode, check the results of your configuration by entering the following **show** commands:

```
[edit]
user@host# show interfaces ge-0/0/0
mtu 9192;
unit 0 {
family ethernet-switching {
vlan {
members [ vlan1 ];
}
}
}
```

```
[edit]
user@host# show interfaces ge-1/0/0
vlan-tagging;
unit 0 {
    vlan-id 1177;
    family inet {
        address 203.0.113.2/24;
    }
}
```

```
[edit]
user@host# show interfaces ge-1/0/1
vlan-tagging;
unit 0 {
    vlan-id 1178;
    family inet {
          address 192.0.2.1/24;
    }
}
```

```
[edit]
user@host# show interfaces sxe-0/0/0
mtu 9192;
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members [ default vlan1 ];
        }
    }
}
```

```
[edit]
user@host# show interfaces sxe-0/0/1
mtu 9192;
unit 0 {
family ethernet-switching {
interface-mode trunk;
vlan {
members [ vlan3 ];
}
}
}
```

```
[edit]
user@host# show interfaces xe-0/0/12
mtu 9192;
unit 0 {
family ethernet-switching {
vlan {
members [ vlan3 ];
}
}
}
```

```
[edit]
user@host# show vlans
default {
vlan-id 1;
}
vlan1 {
vlan-id 77;
}
Vlan3 {
vlan-id 1178;
}
```

Configuring the VNF Interfaces and Creating the Service Chain

Step-by-Step Procedure

Configure the VNF interfaces.

1. Configure the vmhost instance with the LAN, WAN, or the glue VLANs to be used for service chaining:

   ```
   user@host# set vmhost vlans vlan1 vlan-id 77
   user@host# set vmhost vlans vlan2 vlan-id 1177
   user@host# set vmhost vlans glue-vlan1 vlan-id 123
   ```

2. Instantiate the VNF (vnf-name1) with one virtio interface mapped to the VLAN vlan1 and the other virtio interface mapped to the VLAN glue-vlan1.

   ```
   user@host# set virtual-network-functions vnf-name1 interfaces eth2 mapping vlan members vlan1
   user@host# set virtual-network-functions vnf-name1 interfaces eth3 mapping vlan members glue-
   vlan1
   ```

3. Instantiate the second VNF (vnf-name2) with one interface mapped to the VLAN vlan2 and the second interface mapped to the same glue-vlan1.

```
user@host# set virtual-network-functions vnf-name2 interfaces eth2 mapping vlan members glue-
vlan1
user@host# set virtual-network-functions vnf-name2 interfaces eth3 mapping vlan members vlan2
```

4. Configure the IP addresses and static routes for each interface of the VNFs as shown in Figure 12 on page 178.

Example: Configuring Service Chaining Using SR-IOV on NFX350 Devices

IN THIS SECTION

This example shows how to configure service chaining using single-root I/O virtualization (SR-IOV). For information about SR-IOV, see Understanding SR-IOV Usage.

Requirements

This example uses an NFX350 device running Junos OS Release 19.4R1.

Before you configure service chaining, ensure that you have installed and started the relevant VNFs.

Overview

IN THIS SECTION

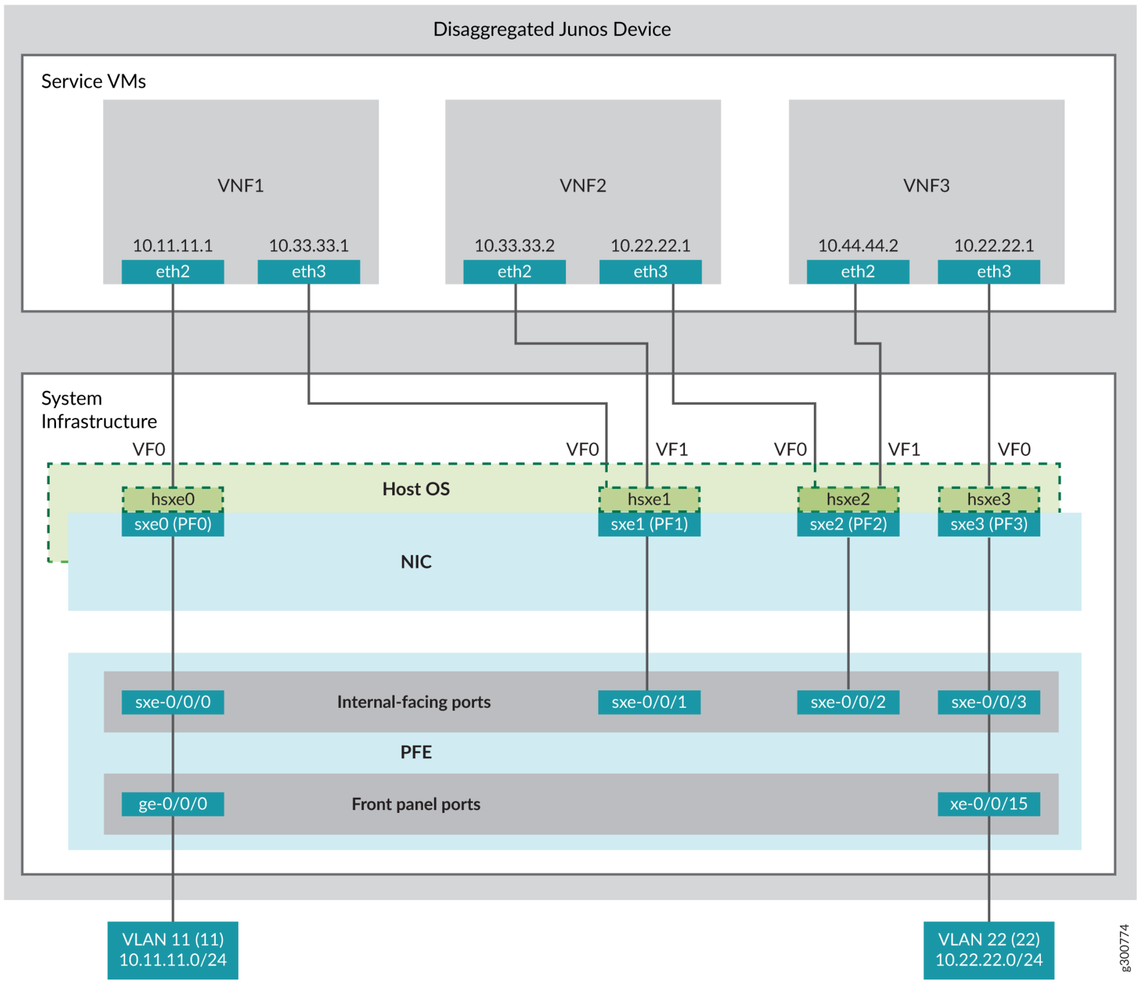
This example uses the front panel ports ge-0/0/0 and xe-0/0/15 associated with the PFE, and its internal-facing ports, sxe-0/0/0 and sxe-0/0/3. The internal NIC ports, sxe0 and sxe3, are not configured directly; instead, they are abstracted at the host OS layer and configured as interfaces hsxe0 and hsxe3. The VNFs use two interfaces, eth2 and eth3. These elements are generally separated into a LAN side and a WAN side. For information on configuring VNFs, see "Configuring VNFs on NFX350 Devices" on page 82.

As this example uses SR-IOV, the virtual functions (VFs) of the NIC ports are used to bypass the host OS and provide direct NIC-to-VM connectivity.

Topology

Figure 13 on page 186 shows the topology for this example.

Figure 13: Service Chaining Using SR-IOV



This example is configured using the Junos Control Plane (JCP). The key configuration elements include:

- Front panel ports associated with the Packet Forwarding Engine
- Internal-facing ports associated with the Packet Forwarding Engine
- NIC ports

> **NOTE**: You must use the host OS interface (hsxe) for these ports because the NIC interfaces (sxe ports) cannot be configured directly.

- VNF interfaces, which use the format eth*#*(where *#* ranges from 2 to 9)
- Virtual function settings, which indicate that SR-IOV is being used to provide direct access between the hsxe and VNF interfaces

Configuration

IN THIS SECTION

Configuring the Packet Forwarding Engine Interfaces

CLI Quick Configuration

To quickly configure the Packet Forwarding Engine interfaces, enter the following configuration statements from the JCP:

```
[edit]
user@host#

set vlans Vlan11 vlan-id 11
set interfaces ge-0/0/0.0 family ethernet-switching vlan member Vlan11
set interfaces sxe-0/0/0.0 family ethernet-switching interface-mode trunk
set interfaces sxe-0/0/0.0 family ethernet-switching vlan member Vlan11
set vlans Vlan22 vlan-id 22
set interfaces xe-0/0/15.0 family ethernet-switching interface-mode trunk
set interfaces xe-0/0/15.0 family ethernet-switching vlan member Vlan22
set interfaces sxe-0/0/3.0 family ethernet-switching interface-mode trunk
set interfaces sxe-0/0/3.0 family ethernet-switching vlan member Vlan22
```

Step-by-Step Procedure

To configure the Packet Forwarding Engine interfaces:

1. Configure a VLAN for the LAN-side interfaces.

   ```
   user@host# set vlans Vlan11 vlan-id 11
   ```

2. Configure the PFE LAN-side front panel port and add it to the LAN-side VLAN.

   The LAN-side port is typically an access port, but can be a trunk port if required.

   ```
   user@host# set interfaces ge-0/0/0.0 family ethernet-switching vlan members Vlan11
   ```

3. Configure the PFE LAN-side internal-facing interface as a trunk port and add it to the LAN-side VLAN.

   The internal-facing interfaces are typically trunk ports as they must support traffic from multiple front panel ports and VLANs.

   ```
   user@host# set interfaces sxe-0/0/0.0 family ethernet-switching interface-mode trunk
   user@host# set interfaces sxe-0/0/0.0 family ethernet-switching vlan member Vlan11
   ```

4. Configure a VLAN for the WAN-side interfaces.

   ```
   user@host# set vlans Vlan22 vlan-id 22
   ```

5. Configure the PFE WAN-side front panel port as a trunk port and add it to the WAN-side VLAN.

   The WAN-side front panel port is typically a trunk port as it might be required to support multiple VLANs.

   ```
   user@host# set interfaces xe-0/0/15.0 family ethernet-switching interface-mode trunk
   user@host# set interfaces xe-0/0/15.0 family ethernet-switching vlan members Vlan22
   ```

6. Configure the PFE WAN-side internal-facing interface as a trunk port and add it to the WAN-side VLAN.

   ```
   user@host# set interfaces sxe-0/0/3.0 family ethernet-switching interface-mode trunk
   user@host# set interfaces sxe-0/0/3.0 family ethernet-switching vlan members Vlan22
   ```

7. Commit the configuration.

   ```
   user@host# commit
   ```

Results

From configuration mode, check the results of your configuration by entering the following **show** commands:

```
user@host> show interfaces ge-0/0/0
unit 0 {
    family ethernet-switching {
        vlan {
            members Vlan11;
        }
    }
}

user@host> show interfaces xe-0/0/15
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members Vlan22;
        }
    }
}

user@host> show interfaces sxe-0/0/0
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members Vlan11;
        }
    }
}

user@host> show interfaces sxe-0/0/3
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members Vlan22;
        }
    }
}

user@host> show vlans
Vlan11 {
    vlan-id 11;
}
Vlan22 {
    vlan-id 22;
}
```

Configuring the VNF Interfaces and Creating the Service Chain

Step-by-Step Procedure

To configure the VNF interfaces and create the service chain:

1. Configure VNF1's LAN-side interface as a Layer 3 interface, and map it to the LAN-side NIC interface. Include the virtual function (VF) setting to specify direct NIC-to-VM connectivity. VNFs must use the interfaces from eth2 through eth9.

   The hsxe interface is the configurable representation of the related NIC (sxe) interface.

   ```
   user@host> configure
   [edit]
   user@host# set virtual-network-functions vm1 interfaces eth2 mapping hsxe0 virtual-function
   ```

2. Configure VNF1's WAN-side interface from sxe1.

   ```
   user@host# set virtual-network-functions vm1 interfaces eth3 mapping hsxe1 virtual-function
   ```

3. Instantiate VNF2 with the interfaces eth2 on sxe1 and eth3 on sxe2.

   ```
   user@host# set virtual-network-functions vm2 interfaces eth2 mapping hsxe1 virtual-function
   user@host# set virtual-network-functions vm2 interfaces eth3 mapping hsxe2 virtual-function
   ```

4. Instantiate VNF3 with the interfaces eth2 on sxe2 and eth3 on sxe3.

```
user@host# set virtual-network-functions vm2 interfaces eth2 mapping hsxe2 virtual-function
user@host# set virtual-network-functions vm2 interfaces eth3 mapping hsxe3 virtual-function
```

5. Configure the IP addresses and static routes for each interface of the VNFs, and add routes to achieve a complete bidirectional path for the service chain.

RELATED DOCUMENTATION

Understanding Service Chaining on Disaggregated Junos OS Platforms

Disaggregated Junos OS VMs

Understanding SR-IOV Usage

Example: Configuring Service Chaining Using a Custom Bridge on NFX350 Devices

IN THIS SECTION

This example shows how to configure service chaining using a custom bridge.

Requirements

This example uses an NFX350 device running Junos OS Release 19.4R1.

Overview

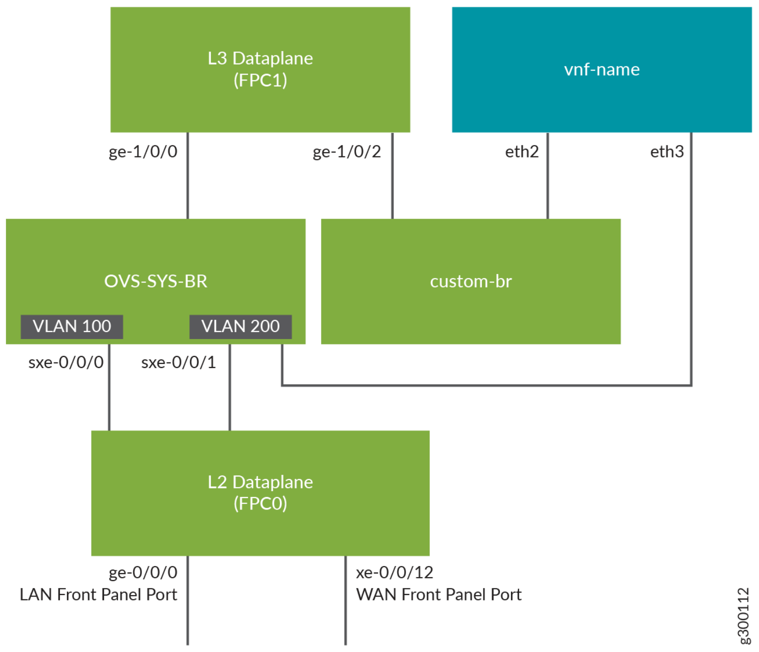
IN THIS SECTION

The default system bridge is Open vSwitch (OVS). The OVS bridge is a VLAN-aware system bridge, which acts as the Network Functions Virtualization (NFV) backplane to which the VNFs and FPCs connect. However, you can choose to create a custom bridge based on your requirement. This example explains how to configure service chaining using a custom bridge.

Topology

This example uses the topology shown in Figure 14 on page 192.

Figure 14: Service Chaining Using a Custom Bridge



Configuration

IN THIS SECTION

Configuring VLANs and Creating the Custom Bridge

Step-by-Step Procedure

1. Configure VLANs for the LAN-side interfaces:

```
user@host# set vlans vlan100 vlan-id 100
user@host# set vlans vlan200 vlan-id 200
```

2. Create a custom bridge:

```
user@host# set vmhost vlans custom-br vlan-id none
```

3. Map the Layer 3 interface to the custom bridge:

```
user@host# set vmhost virtualization-options interfaces ge-1/0/2 mapping vlan custom-br
```

Configuring the Layer 2 Datapath

Step-by-Step Procedure

1. Configure the LAN-side front panel ports and add them to the LAN-side VLAN.

```
user@host# set interfaces ge-0/0/0 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces ge-0/0/0 unit 0 family ethernet-switching vlan members vlan100
user@host# set interfaces xe-0/0/12 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces xe-0/0/12 unit 0 family ethernet-switching vlan members vlan200
```

2. Configure the internal-facing interfaces as trunk ports and add them to the LAN-side VLAN. The internal-facing interfaces are typically trunk ports as they must support traffic from multiple front panel ports and VLANs.

```
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan100
user@host# set interfaces sxe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/1 unit 0 family ethernet-switching vlan members vlan200
```

Configuring the Layer 3 Datapath

Step-by-Step Procedure

1. Configure VLAN tagging on ge-1/0/0:

```
user@host# set interfaces ge-1/0/0 vlan-tagging
user@host# set interfaces ge-1/0/0 unit 0 vlan-id 100
user@host# set interfaces ge-1/0/0 unit 0 family inet address 192.0.2.1/24
```

2. Configure VLAN tagging on ge-1/0/2:

```
user@host# set interfaces ge-1/0/2 vlan-tagging
user@host# set interfaces ge-1/0/2 unit 0 vlan-id 200
user@host# set interfaces ge-1/0/2 unit 0 family inet address 203.0.113.2/24
```

Configuring the VNF

Step-by-Step Procedure

NOTE: This example uses a Layer 2 VNF.

1. Launch the VNF:

```
user@host# set virtual-network-functions vnf-name image /var/public/centos-updated1.img
user@host# set virtual-network-functions vnf-name image image-type raw
```

2. Specify the number of CPUs required for the VNF:

```
user@host# set virtual-network-functions vnf-name virtual-cpu count 1
```

3. Pin a virtual CPU to a physical CPU:

```
user@host# set virtual-network-functions vnf-name virtual-cpu 0 physical-cpu 2
```

4. Configure the vmhost instance:

```
user@host# set vmhost vlans vlan200 vlan-id 200
```

5. Create a VNF interface on the custom OVS bridge:

```
user@host# set virtual-network-functions vnf-name interfaces eth2 mapping vlan members custom-
br
```

6. Create a VNF interface on the OVS bridge:

```
user@host# set virtual-network-functions vnf-name interfaces eth3 mapping vlan members vlan200
```

7. Specify the memory allocation for the VNF:

```
user@host# set virtual-network-functions vnf-name memory size 1048576
```

NOTE: When a VNF interface is mapped to a custom bridge, you should restart the VNF for the mapping to take effect.

Verifying the Configuration

IN THIS SECTION

Verify the Control Plane Configuration

Purpose

Verify the control plane configuration:

Action

- Verify that the VLANs are configured:

```
user@host > show vlans
Routing instance        VLAN name           Tag          Interfaces
default-switch          default             1

default-switch          vlan100             100
                                                         ge-0/0/0.0*
                                                         sxe-0/0/0.0*
default-switch          vlan200             200
                                                         sxe-0/0/1.0*
                                                         xe-0/0/12.0*
```

- Verify the vmhost VLANs:

```
user@host> show vmhost vlans
Routing instance        VLAN name             Tag          Interfaces
vmhost                  custom-br                            vnf-name_eth2.0
vmhost                  vlan200               200            vnf-name_eth3.0
```

- Verify that the VNF is operational. The `State` field shows `Running` for VNFs that are up.

```
user@host> show virtual-network-functions
ID       Name                                              State      Liveliness
--------------------------------------------------------------------------------
4        vnf-name                                          Running    alive
1        vjunos0                                           Running    alive
```

The `Liveliness` field of the VNF indicates whether the internal management IP address of the VNF is reachable from the Junos Control Plane (JCP).

To view more details of the VNF:

```
user@host> show virtual-network-functions vnf-name detail
Virtual Network Function Information
------------------------------------


Id:                   4
Name:                 vnf-name
State:                Running
Liveliness:           alive
IP Address:           192.0.2.100
VCPUs:                1
Maximum Memory:       1048576 KiB
Used Memory:          1048576 KiB
Used 1G Hugepages:    0
Used 2M Hugepages:    0
Error:                None
```

Verifying the Data Plane Configuration

Purpose

Verify the data plane configuration.

Action

- Verify the status of the Layer 2 (ge-0/0/x) and Layer 3 (ge-1/0/x) interfaces.

```
user@host > show interfaces interface-name statistics
```

For example:

```
user@host > show interfaces ge-0/0/0 statistics
Physical interface: ge-0/0/0, Enabled, Physical link is Up
  Interface index: 149, SNMP ifIndex: 517
  Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Link-mode: Full-duplex, Speed:
1000mbps, Duplex: Full-Duplex, BPDU Error: None,
  Loop Detect PDU Error: None, Ethernet-Switching Error: None, MAC-REWRITE Error: None,
Loopback: Disabled, Source filtering: Disabled,
  Flow control: Enabled, Auto-negotiation: Enabled, Remote fault: Online, IEEE 802.3az Energy
Efficient Ethernet: Disabled, Auto-MDIX: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 12 supported, 12 maximum usable queues
  Current address: 30:7c:5e:4c:78:03, Hardware address: 30:7c:5e:4c:78:03
  Last flapped   : 2018-11-26 11:03:32 UTC (04:25:39 ago)
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)
  Active alarms  : None
  Active defects : None
  PCS statistics                      Seconds
    Bit errors                             0
    Errored blocks                         0
  Ethernet FEC statistics              Errors
    FEC Corrected Errors                    0
    FEC Uncorrected Errors                  0
    FEC Corrected Errors Rate               0
    FEC Uncorrected Errors Rate             0
  PRBS Statistics : Disabled
  Interface transmit statistics: Disabled

  Logical interface ge-0/0/0.0 (Index 330) (SNMP ifIndex 519)
    Flags: Up SNMP-Traps 0x24024000 Encapsulation: Ethernet-Bridge
    Input packets : 0
    Output packets: 0
    Protocol eth-switch, MTU: 1514
      Flags: Trunk-Mode
```

- Verify the status of the interfaces on the OVS and the custom bridge:

```
user@host > show vmhost network nfv-back-plane
Network Name : custom-br

   Interface : custom-br
   Type : internal, Link type : Full-Duplex, MAC : 2e:8e:a3:e3:e5:40
   MTU : [], Link State :down, Admin State : down
   IPV4 : None, Netmask : None
   IPV6 : None, IPV6 netmask : None
       Rx-packets :       0
       Rx-drops   :       0
       Rx-errors  :       0
       Tx-packets :       0
       Tx-drops   :       0
       Tx-errors  :       0

   Interface : vnf-name_eth2
   Type : dpdkvhostuser, Link type : Full-Duplex, MAC : 00:00:00:00:00:00
   MTU : 1500, Link State :down, Admin State : up
   IPV4 : None, Netmask : None
   IPV6 : None, IPV6 netmask : None
       Rx-packets :       0
       Rx-drops   :       0
       Rx-errors  :       0
       Tx-packets :       0
       Tx-drops   :       0
       Tx-errors  :       0


Network Name : ovs-sys-br
```

```
Interface : ovs-sys-br
Type : internal, Link type : Full-Duplex, MAC : 66:9c:3f:25:04:40
MTU : [], Link State :down, Admin State : down
IPV4 : None, Netmask : None
IPV6 : None, IPV6 netmask : None
    Rx-packets :       0
    Rx-drops   :       0
    Rx-errors  :       0
    Tx-packets :       0
    Tx-drops   :       0
    Tx-errors  :       0

Interface : dpdk0
Type : dpdk, Link type : Full-Duplex, MAC : 02:09:c0:1a:c6:ee
MTU : [], Link State :up, Admin State : up
IPV4 : None, Netmask : None
IPV6 : None, IPV6 netmask : None
    Rx-packets :       0
    Rx-drops   :       0
    Rx-errors  :       0
    Tx-packets :       0
    Tx-drops   :       0
    Tx-errors  :       0

Interface : dpdk1
Type : dpdk, Link type : Full-Duplex, MAC : 02:09:c0:7b:6c:47
MTU : [], Link State :up, Admin State : up
IPV4 : None, Netmask : None
IPV6 : None, IPV6 netmask : None
    Rx-packets :       0
    Rx-drops   :       0
    Rx-errors  :       0
    Tx-packets :       0
    Tx-drops   :       0
    Tx-errors  :       0

Interface : l3_h_ge_1_0_0
Type : dpdkvhostuser, Link type : Full-Duplex, MAC : 00:00:00:00:00:00
MTU : [], Link State :down, Admin State : up
IPV4 : None, Netmask : None
IPV6 : None, IPV6 netmask : None
    Rx-packets :       0
    Rx-drops   :       0
    Rx-errors  :       0
    Tx-packets :       0
    Tx-drops   :       0
    Tx-errors  :       0

Interface : l3_h_ge_1_0_1
Type : dpdkvhostuser, Link type : Full-Duplex, MAC : 00:00:00:00:00:00
MTU : [], Link State :down, Admin State : up
IPV4 : None, Netmask : None
IPV6 : None, IPV6 netmask : None
    Rx-packets :       0
    Rx-drops   :       0
    Rx-errors  :       0
    Tx-packets :       0
    Tx-drops   :       0
    Tx-errors  :       0

Interface : l3_h_ge_1_0_2
Type : dpdkvhostuser, Link type : Full-Duplex, MAC : 00:00:00:00:00:00
MTU : [], Link State :down, Admin State : up
IPV4 : None, Netmask : None
IPV6 : None, IPV6 netmask : None
    Rx-packets :       0
    Rx-drops   :       0
    Rx-errors  :       0
    Tx-packets :       0
    Tx-drops   :       0
    Tx-errors  :       0

Interface : vnf-name_eth3
Type : dpdkvhostuser, Link type : Full-Duplex, MAC : 00:00:00:00:00:00
MTU : 1500, Link State :down, Admin State : up
IPV4 : None, Netmask : None
IPV6 : None, IPV6 netmask : None
    Rx-packets :       0
    Rx-drops   :       0
    Rx-errors  :       0
    Tx-packets :       0
    Tx-drops   :       0
    Tx-errors  :       0
```

Example: Configuring Service Chaining for LAN Routing on NFX350 Devices

IN THIS SECTION

This example shows how to configure service chaining for LAN routing.

Requirements

This example uses an NFX350 device running Junos OS Release 19.4R1.

Overview

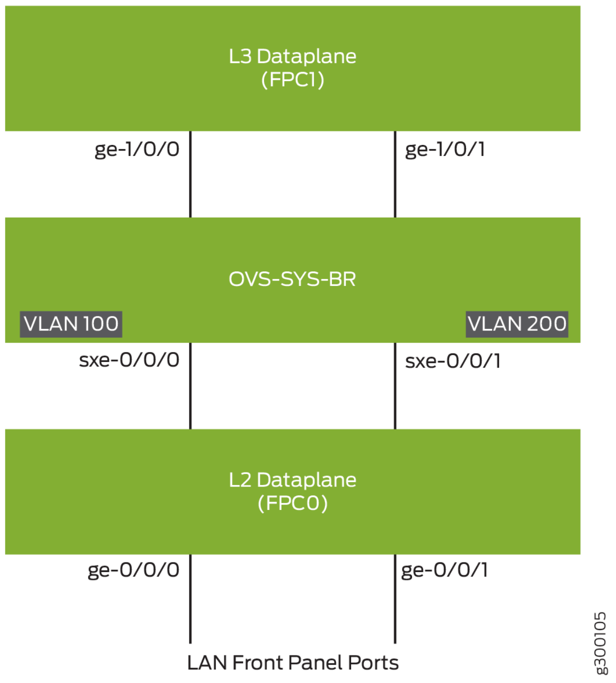
IN THIS SECTION

This example explains how to configure the various layers of the device to enable traffic flow within a LAN network.

Topology

This example uses the topology shown in Figure 15 on page 203.

Figure 15: Service Chaining for LAN Routing



Configuration

IN THIS SECTION

Configuring the Layer 2 Datapath

Step-by-Step Procedure

1. Configure VLANs for the LAN-side interfaces.

```
user@host# set vlans vlan100 vlan-id 100
user@host# set vlans vlan200 vlan-id 200
```

2. Configure the LAN-side front panel ports and add them to the LAN-side VLAN.

```
user@host# set interfaces ge-0/0/0 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces ge-0/0/0 unit 0 family ethernet-switching vlan members vlan100
user@jcp# set interfaces ge-0/0/1 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces ge-0/0/1 unit 0 family ethernet-switching vlan members vlan200
```

3. Configure the internal-facing interfaces as trunk ports and add them to the LAN-side VLAN. The internal-facing interfaces are typically trunk ports as they must support traffic from multiple front panel ports and VLANs.

```
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan100
user@host# set interfaces sxe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/1 unit 0 family ethernet-switching vlan members vlan200
```

Configuring the Layer 3 Datapath

Step-by-Step Procedure

1. Configure VLAN tagging on ge-1/0/0:

```
user@host# set interfaces ge-1/0/0 vlan-tagging
user@host# set interfaces ge-1/0/0 unit 0 vlan-id 100
user@host# set interfaces ge-1/0/0 unit 0 family inet address 192.0.2.1/24
```

2. Configure VLAN tagging on ge-1/0/1:

```
user@host# set interfaces ge-1/0/1 vlan-tagging
user@host# set interfaces ge-1/0/1 unit 0 vlan-id 200
user@host# set interfaces ge-1/0/1 unit 0 family inet address 203.0.113.2/24
```

RELATED DOCUMENTATION

Example: Configuring Service Chaining for LAN-WAN Routing

Example: Configuring Service Chaining for LAN to WAN Routing on NFX350 Devices

IN THIS SECTION

This example shows how to configure service chaining for LAN to WAN routing.

Requirements

This example uses an NFX350 device running Junos OS Release 19.4R1.

Overview

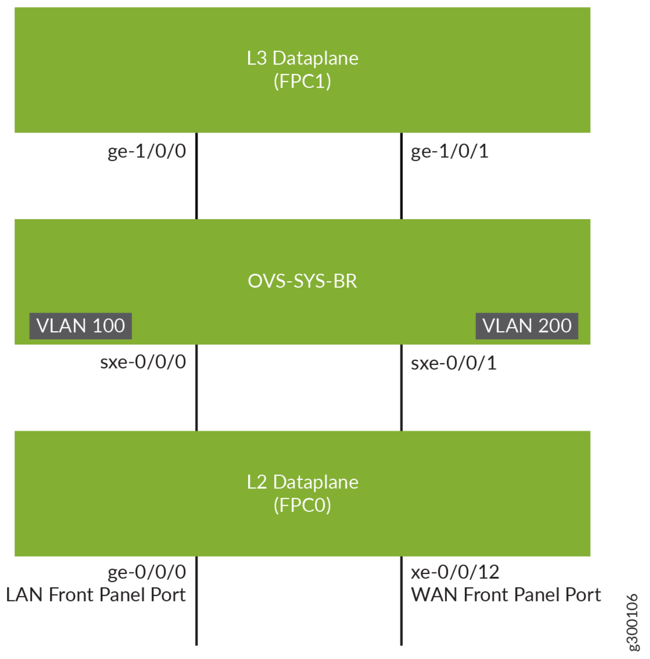
IN THIS SECTION

This example explains how to configure the various layers of the device to enable traffic from the LAN network to enter the device, flow through the OVS, exit the device, and enter the WAN network.

Topology

This example uses the topology shown in Figure 16 on page 206.

Figure 16: Service Chaining for LAN to WAN Routing



Configuration

IN THIS SECTION

Configuring the Layer 2 Datapath

Step-by-Step Procedure

1. Configure VLANs for the LAN-side interfaces.

```
user@host# set vlans vlan100 vlan-id 100
user@host# set vlans vlan200 vlan-id 200
```

2. Configure the LAN-side front panel ports and add them to the LAN-side and WAN-side VLANs.

```
user@host# set interfaces ge-0/0/0 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces ge-0/0/0 unit 0 family ethernet-switching vlan members vlan100
user@host# set interfaces xe-0/0/12 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces xe-0/0/12 unit 0 family ethernet-switching vlan members vlan200
```

3. Configure the internal-facing interface, sxe-0/0/0, as a trunk port and add it to the LAN-side VLAN. The internal-facing interfaces are typically trunk ports as they must support traffic from multiple front panel ports and VLANs.

```
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/0 unit 0 family ethernet-switching vlan members vlan100
```

4. Configure the internal-facing interface, sxe-0/0/1, as a trunk port and add it to the WAN-side VLAN.

```
user@host# set interfaces sxe-0/0/1 unit 0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/1 unit 0 family ethernet-switching vlan members vlan200
```

Configuring the Layer 3 Datapath

Step-by-Step Procedure

1. Configure VLAN tagging on ge-1/0/0:

```
user@host# set interfaces ge-1/0/0 vlan-tagging
user@host# set interfaces ge-1/0/0 unit 0 vlan-id 100
user@host# set interfaces ge-1/0/0 unit 0 family inet address 192.0.2.1/24
```

2. Configure VLAN tagging on ge-1/0/1:

```
user@host# set interfaces ge-1/0/1 vlan-tagging
user@host# set interfaces ge-1/0/1 unit 0 vlan-id 200
user@host# set interfaces ge-1/0/1 unit 0 family inet address 203.0.113.2/24
```

Verification

IN THIS SECTION

Verifying the Status of the Interfaces

Purpose

Verify the status of the Layer 2 and Layer 3 interfaces.

Action

- Verify the status of the Layer 2 (ge-0/0/x) and Layer 3 (ge-1/0/x) interfaces.

```
user@host> show interfaces interface-name statistics
```

For example:

```
user@host> show interfaces ge-0/0/0 statistics
Physical interface: ge-0/0/0, Enabled, Physical link is Up
  Interface index: 144, SNMP ifIndex: 518
  Link-level type: Ethernet, MTU: 9192, LAN-PHY mode, Speed: 1000mbps,
  BPDU Error: None, Loop Detect PDU Error: None, Ethernet-Switching Error: None,
  MAC-REWRITE Error: None, Loopback: Disabled, Source filtering: Disabled,
  Flow control: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Current address: 00:00:5e:00:53:43, Hardware address: 00:00:5e:00:53:43
  Last flapped   : 2018-04-18 05:38:22 UTC (2d 10:07 ago)
  Statistics last cleared: Never
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)
  Input errors: 0, Output errors: 0
  Active alarms  : None
  Active defects : None
  PCS statistics                      Seconds
    Bit errors                             0
    Errored blocks                         0
  Ethernet FEC statistics              Errors
    FEC Corrected Errors                    0
    FEC Uncorrected Errors                  0
    FEC Corrected Errors Rate               0
    FEC Uncorrected Errors Rate             0
  PRBS Statistics : Disabled
  Interface transmit statistics: Disabled

  Logical interface ge-0/0/0.0 (Index 333) (SNMP ifIndex 524)
    Flags: Up SNMP-Traps 0x24024000 Encapsulation: Ethernet-Bridge
    Input packets : 147888
    Output packets: 22
    Protocol eth-switch, MTU: 9192
      Flags: Is-Primary
```

Example: Configuring Service Chaining for LAN to WAN Routing through Third-party VNFs on NFX350 Devices

IN THIS SECTION

This example shows how to configure service chaining for LAN to WAN routing through third-party VNFs on NFX350 devices.

Requirements

This example uses an NFX350 device running Junos OS Release 19.4R1.

Overview

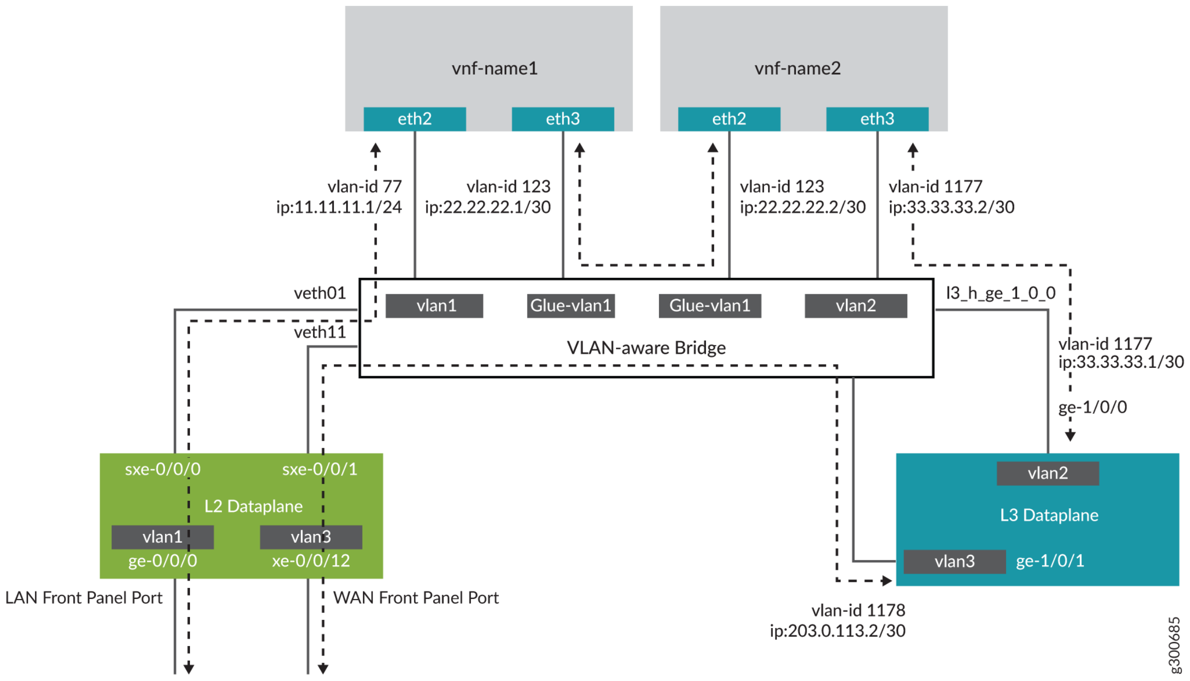
IN THIS SECTION

This example explains how to configure the various layers of the device to enable traffic from the LAN network to enter the device, flow through the OVS bridge and third-party VNFs, exit the device, and enter the WAN network.

Topology

This example uses the topology shown in Figure 17 on page 211.

Figure 17: Service Chaining for LAN to WAN Routing through Third-party VNFs



Configuration

IN THIS SECTION

Configuring the Layer 2 Datapath (JCP LAN Interfaces)

Step-by-Step Procedure

1. Connect to the JCP.

```
user@host:~ # cli
user@host>
user@host> configure
[edit]
user@host#
```

2. Configure VLANs for the LAN-side interfaces.

```
user@host# set vlans vlan1 vlan-id 77
```

3. Configure the LAN-side front panel ports and add them to the LAN-side VLANs. The LAN-side port is typically an access port, and can be a trunk port if required

```
user@host# set interfaces ge-0/0/0.0 family ethernet-switching vlan members vlan1
```

4. Configure the internal-facing interface, sxe-0/0/0, as a trunk port and add it to the LAN-side VLAN. The internal-facing interfaces are typically trunk ports as they must support traffic from multiple front panel ports and VLANs.

```
user@host# set interfaces sxe-0/0/0.0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/0.0 family ethernet-switching vlan members vlan1
```

Verifying the Performance Mode of the NFX350 Device

Purpose

Verify the performance mode of the NFX350 device and check the CPU availability. If the NFX350 device is operating in throughput mode, you must change it to either compute or hybrid mode by using the `request vmhost mode` command.

For more information about the device performance modes, see "NFX350 Overview" on page 2.

Action

```
user@host> show vmhost mode | no-more
Mode:
--------
Current Mode: compute

CPU Allocations:
Name                                 Configured                                 Used
-------------------------------------------------------------------------------------------------
---------------------
Junos Control Plane                  16                                         16,6
Juniper Device Manager               16                                         16
LTE                                  16                                         -
NFV Backplane Control Path           16                                         16
NFV Backplane Data Path              1,2,3                                      1,2,3
Layer 2 Control Path                 -                                          -
Layer 2 Data Path                    -                                          -
Layer 3 Control Path                 0                                          0
Layer 3 Data Path                    4,5                                        4,5
CPUs available for VNFs              6,7,8,9,10,11,12,13,14,15,22,23,24,25,26,27,28,29,30,31 -
CPUs turned off                      17,18,19,20,21                             -

Memory Allocations:
Name                                 Configured                                 Used
-------------------------------------------------------------------------------------------------
---------------------
Junos Control Plane (mB)             2048                                       2002
NFV Backplane 1G hugepages           12                                         18
NFV Backplane 2M hugepages           -                                          0
Layer 2 1G hugepages                 -                                          -
Layer 2 2M hugepages                 -                                          -
Layer 3 1G hugepages                 6                                        6
Layer 3 2M hugepages                 20481                                    20481
```

Configuring the Hugepages for VNF

Step-by-Step Procedure

NOTE: It is recommended to reboot the device if the configured number of hugepages are not allocated.

1. Check the memory availability:

```
user@host> show system visibility memory | no-more
Memory Information
------------------


Virtual Memory:
---------------
Total       (KiB): 131042784
Used        (KiB): 67141828
Available   (KiB): 66151972
Free        (KiB): 63900956
Percent Used     : 49.5

Huge Pages:
------------
Total 1GiB Huge Pages:      18
Free 1GiB Huge Pages:       0
Configured 1GiB Huge Pages: 0
Total 2MiB Huge Pages:      20481
Free 2MiB Huge Pages:       0
Configured 2MiB Huge Pages: 0
```

2. Configure hugepages:

```
user@host> configure
[edit]
user@host#
```

```
user@host# set system memory hugepages page-size 1024 page-count 10
user@host# commit
```

3. Verify whether hugepages is configured:

```
user@host# run show system visibility memory | no-more
Memory Information
------------------


Virtual Memory:
---------------
Total       (KiB): 131042784
Used        (KiB): 77624220
Available   (KiB): 55670868
Free        (KiB): 53418564
Percent Used     : 57.5

Huge Pages:
------------
Total 1GiB Huge Pages:      28
Free 1GiB Huge Pages:       10
Configured 1GiB Huge Pages: 10
Total 2MiB Huge Pages:      20481
Free 2MiB Huge Pages:       0
Configured 2MiB Huge Pages: 0

Hugepages Usage:
----------------------------------------------------------------------------------------------
------------
Name                             Type                              Used 1G Hugepages  Used
2M Hugepages
-------------------------------- --------------------------------- ------------------
------------------
ovs-vswitchd                     other process                     18                 0
srxpfe                           other process                     6                  20481
```

Configuring VNFs

Step-by-Step Procedure

Configure VNF-1:

1. Load the VNF image on the device from the remote location:

   NOTE: You can save the VNF image in the **/var/public** directory if you are using up to two VNFs. If you are using more than two VNFs, save the files on an external SSD. If you are using an external SSD for VNFs, make sure to initialize and add the SSD to the device. For more information, see "Configuring the Solid State Disk on NFX350 Device" on page 48.

   ```
   user@host> file copy source-address /var/public/vnf-1_junos-vsrx3-x86-64-19.1R1-S1.3.qcow2
   ```

2. Launch the VNF:

   ```
   user@host> set virtual-network-functions VNF-1 image /var/public/vnf-1_junos-vsrx3-
   x86-64-19.1R1-S1.3.qcow2
   ```

3. Connect a virtual CPUs to physical CPUs:

   ```
   user@host> set virtual-network-functions VNF-1 virtual-cpu 0 physical-cpu 6
   user@host> set virtual-network-functions VNF-1 virtual-cpu 1 physical-cpu 7
   ```

4. Specify the number of CPUs required for the VNF:

   ```
   user@host> set virtual-network-functions VNF-1 virtual-cpu count 2
   ```

5. Enable hardware virtualization or hardware acceleration for VNF CPUs:

   ```
   user@host> set virtual-network-functions VNF-1 virtual-cpu features hardware-virtualization
   ```

6. Configure the VNF interfaces as trunk ports and add them to the LAN-side VLAN:

   ```
   user@host> set virtual-network-functions VNF-1 interfaces eth2 mapping vlan mode trunk
   user@host> set virtual-network-functions VNF-1 interfaces eth2 mapping vlan members vlan1
   user@host> set virtual-network-functions VNF-1 interfaces eth3 mapping vlan mode trunk
   user@host> set virtual-network-functions VNF-1 interfaces eth3 mapping vlan members glue-vlan1
   ```

7. Specify the memory allocation for the VNF:

```
user@host> set virtual-network-functions VNF-1 memory size 4194304
user@host> set virtual-network-functions VNF-1 memory features hugepages
```

Step-by-Step Procedure

Configure VNF-2:

1. Load the VNF image on the device from the remote location:

```
user@host> file copy source-address /var/public/vnf-2-junos-vsrx3-x86-64-19.1R1-S1.3.qcow2
```

2. Launch the VNF:

```
user@host> set virtual-network-functions VNF-2 image /var/public/vnf-2-junos-vsrx3-
x86-64-19.1R1-S1.3.qcow2
```

3. Connect a virtual CPUs to physical CPUs:

```
user@host> set virtual-network-functions VNF-2 virtual-cpu 0 physical-cpu 8
user@host> set virtual-network-functions VNF-2 virtual-cpu 1 physical-cpu 9
```

4. Specify the number of CPUs required for the VNF:

```
user@host> set virtual-network-functions VNF-2 virtual-cpu count 2
```

5. Enable hardware virtualization or hardware acceleration for VNF CPUs:

```
user@host> set virtual-network-functions VNF-2 virtual-cpu features hardware-virtualization
```

6. Configure the VNF interfaces as trunk ports and add them to the LAN-side VLAN:

```
user@host> set virtual-network-functions VNF-2 interfaces eth2 mapping vlan mode trunk
user@host> set virtual-network-functions VNF-2 interfaces eth2 mapping vlan members glue-vlan1
```

```
user@host> set virtual-network-functions VNF-2 interfaces eth3 mapping vlan mode trunk
user@host> set virtual-network-functions VNF-2 interfaces eth3 mapping vlan members vlan2
```

7. Specify the memory allocation for the VNF:

```
user@host> set virtual-network-functions VNF-2 memory size 4194304
user@host> set virtual-network-functions VNF-2 memory features hugepages
```

Configuring the Layer 3 Datapath (WAN Interfaces)

Step-by-Step Procedure

1. Configure the internal-facing L3 Dataplane interface as a VLAN-tagged interface and assign an IP address to it:

```
user@host# set interfaces ge-1/0/0 vlan-tagging
user@host# set interfaces ge-1/0/0.0 vlan-id 1177
user@host# set interfaces ge-1/0/0.0 family inet address 33.33.33.1/30
```

2. Map the Layer 3 interface to the Open vSwitch (OVS) and commit the configuration:

```
user@host# set vmhost virtualization-options interfaces ge-1/0/1
user@host# commit
```

3. Configure the external-facing L3 Dataplane interface as a VLAN-tagged interface and assign an IP address to it:

```
user@host# set interfaces ge-1/0/1 vlan-tagging
user@host# set interfaces ge-1/0/1.0 vlan-id 1178
user@host# set interfaces ge-1/0/1.0 family inet address 203.0.113.2/30
```

4. Configure a VLAN for the WAN-side JCP interfaces:

```
user@host# set vlans vlan3 vlan-id 1178
```

5. Configure the WAN-side internal-facing interface as a trunk port and add it to the WAN-side VLAN:

```
user@host# set interfaces sxe-0/0/1.0 family ethernet-switching interface-mode trunk
user@host# set interfaces sxe-0/0/1.0 family ethernet-switching vlan members vlan3
```

6. Configure the WAN-side front panel port and add it to the WAN-side VLAN:

```
user@host# set interfaces xe-0/0/12.0 family ethernet-switching interface-mode access
user@host# set interfaces xe-0/0/12.0 family ethernet-switching vlan members vlan3
```

7. Commit the configuration:

```
user@host# commit
```

Configuring the VNF Interfaces for Creating the Service Chain

Step-by-Step Procedure

1. Check the MAC addresses of the VNF interfaces:

```
user@host# run show system visibility network
VNF MAC Addresses
-----------------------------------------------------------
VNF                                       MAC
----------------------------------------- -----------------
VNF-1_ethdef0                             D0:DD:49:E8:B6:CA
VNF-1_ethdef1                             D0:DD:49:E8:B6:CB
VNF-1_eth2                                D0:DD:49:E8:B6:CC
VNF-1_eth3                                D0:DD:49:E8:B6:C7
VNF-2_ethdef0                             D0:DD:49:E8:B6:C8
VNF-2_ethdef1                             D0:DD:49:E8:B6:C9
VNF-2_eth2                                D0:DD:49:E8:B6:CD
VNF-2_eth3                                D0:DD:49:E8:B6:CE

VNF Internal IP Addresses
---------------------------------------------------------
VNF                                       IP
----------------------------------------- ---------------
```

```
VNF-1                                       192.0.2.100
VNF-2                                       192.0.2.101

Free Virtual Functions
----------------------
PF        VF
--------- ------------
hsxe0     0000:b7:03.6
hsxe0     0000:b7:03.4
hsxe0     0000:b7:03.5
hsxe0     0000:b7:02.3
hsxe0     0000:b7:02.2
hsxe0     0000:b7:02.1
hsxe0     0000:b7:02.7
hsxe0     0000:b7:02.6
hsxe0     0000:b7:02.5
hsxe0     0000:b7:02.4
hsxe1     0000:b7:07.4
hsxe1     0000:b7:06.7
hsxe1     0000:b7:06.6
hsxe1     0000:b7:06.5
hsxe1     0000:b7:06.4
hsxe1     0000:b7:06.3
hsxe1     0000:b7:06.2
hsxe1     0000:b7:06.1
hsxe1     0000:b7:07.5
hsxe1     0000:b7:07.6
hsxe2     0000:b7:0b.6
hsxe2     0000:b7:0b.5
hsxe2     0000:b7:0b.4
hsxe2     0000:b7:0a.4
hsxe2     0000:b7:0a.5
hsxe2     0000:b7:0a.6
hsxe2     0000:b7:0a.7
hsxe2     0000:b7:0a.1
hsxe2     0000:b7:0a.2
hsxe2     0000:b7:0a.3
hsxe3     0000:b7:0f.6
hsxe3     0000:b7:0f.5
hsxe3     0000:b7:0f.4
hsxe3     0000:b7:0e.1
hsxe3     0000:b7:0e.2
hsxe3     0000:b7:0e.3
```

```
hsxe3     0000:b7:0e.4
hsxe3     0000:b7:0e.5
hsxe3     0000:b7:0e.6
hsxe3     0000:b7:0e.7

VNF Interfaces
---------------------------------------------------------------------------------------------
-
VNF                  Interface Type      Source       Model      MAC                  VLAN-
ID
-------------------- --------- --------- ------------ ---------- --------------------
-------
VNF-1                vnet4     network   default      virtio     d0:dd:49:e8:b6:ca    --
VNF-1                vnet5     bridge    eth0br       virtio     d0:dd:49:e8:b6:cb    --
VNF-1                VNF-1_eth2 vhostuser --          virtio     d0:dd:49:e8:b6:cc    --
VNF-1                VNF-1_eth3 vhostuser --          virtio     d0:dd:49:e8:b6:c7    --
VNF-2                vnet6     network   default      virtio     d0:dd:49:e8:b6:c8    --
VNF-2                vnet7     bridge    eth0br       virtio     d0:dd:49:e8:b6:c9    --
VNF-2                VNF-2_eth2 vhostuser --          virtio     d0:dd:49:e8:b6:cd    --
VNF-2                VNF-2_eth3 vhostuser --          virtio     d0:dd:49:e8:b6:ce    --

OVS Interfaces
------------------------
NAME              MTU
----------------- ------
ovs-sys-br        1500
dpdk2             9216
xdsl_eth0         9192
l3_h_ge_1_0_1     9216
l3_h_ge_1_0_0     1500
dpdk0             9216
VNF-2_eth2        1500
dpdk1             9216
VNF-1_eth3        1500
dpdk3             9216
VNF-1_eth2        1500
VNF-2_eth3        1500
```

2. Access the VNF (VNF-1) from the JCP through the console:

```
user@host> request virtual-network-functions console VNF-1
Internal instance: VNF-1
Connected to domain VNF-1
```

3. Log in to the console:

```
user@host:~ # cli
user@host>
```

4. Check the status of the interfaces:

   -
     ```
     user@host# show interfaces terse | no-more
     Interface               Admin Link Proto    Local                 Remote
     ge-0/0/0                up    up
     gr-0/0/0                up    up
     ip-0/0/0                up    up
     lsq-0/0/0               up    up
     lt-0/0/0                up    up
     mt-0/0/0                up    up
     sp-0/0/0                up    up
     sp-0/0/0.0              up    up   inet
                                        inet6
     sp-0/0/0.16383          up    up   inet
     ge-0/0/1                up    up
     ge-0/0/2                up    up
     dsc                     up    up
     fti0                    up    up
     fxp0                    up    up
     fxp0.0                  up    up
     gre                     up    up
     ipip                    up    up
     irb                     up    up
     lo0                     up    up
     lo0.16384               up    up   inet     127.0.0.1           --> 0/0
     lo0.16385               up    up   inet     10.0.0.1            --> 0/0
                                                 10.0.0.16           --> 0/0
                                                 128.0.0.1           --> 0/0
                                                 128.0.0.4           --> 0/0
     ```

```
                                                128.0.1.16          --> 0/0
lo0.32768               up    up
lsi                     up    up
mtun                    up    up
pimd                    up    up
pime                    up    up
pp0                     up    up
ppd0                    up    up
ppe0                    up    up
st0                     up    up
tap                     up    up
vlan                    up    down
```

-
  ```
  user@host> show interfaces ge-0/0/0 | no-more
  Physical interface: ge-0/0/0, Enabled, Physical link is Up
    Interface index: 135, SNMP ifIndex: 508
    Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Link-mode: Half-duplex,
    Speed: 1000mbps, BPDU Error: None, Loop Detect PDU Error: None,
    Ethernet-Switching Error: None, MAC-REWRITE Error: None, Loopback: Disabled,
    Source filtering: Disabled, Flow control: Enabled, Auto-negotiation: Enabled,
    Remote fault: Online
    Device flags   : Present Running
    Interface flags: SNMP-Traps Internal: 0x4000
    Link flags     : None
    CoS queues     : 8 supported, 8 maximum usable queues
    Current address: d0:dd:49:e8:b6:cb, Hardware address: d0:dd:49:e8:b6:cb
    Last flapped   : 2020-05-11 10:22:06 UTC (00:46:40 ago)
    Input rate     : 0 bps (0 pps)
    Output rate    : 0 bps (0 pps)
    Active alarms  : None
    Active defects : None
   PCS statistics                      Seconds
     Bit errors                             0
     Errored blocks                         0
    Ethernet FEC statistics              Errors
     FEC Corrected Errors                    0
     FEC Uncorrected Errors                  0
     FEC Corrected Errors Rate               0
     FEC Uncorrected Errors Rate             0
    Interface transmit statistics: Disabled
  ```

-
  ```
  user@host> show interfaces fxp0 | no-more
  Physical interface: fxp0, Enabled, Physical link is Up
    Interface index: 65, SNMP ifIndex: 1
    Type: Ethernet, Link-level type: Ethernet, MTU: 1514, Speed: 1000mbps
    Device flags   : Present Running
    Interface flags: SNMP-Traps
    Link type      : Full-Duplex
    Current address: d0:dd:49:e8:b6:ca, Hardware address: d0:dd:49:e8:b6:ca
    Last flapped   : 2020-05-11 10:21:26 UTC (00:47:53 ago)
      Input packets : 1484
      Output packets: 0

    Logical interface fxp0.0 (Index 3) (SNMP ifIndex 13)
      Flags: Up SNMP-Traps Encapsulation: ENET2
      Input packets : 1452
      Output packets: 0
  ```

-
  ```
  user@host> show interfaces ge-0/0/1 | no-more
  Physical interface: ge-0/0/1, Enabled, Physical link is Up
    Interface index: 136, SNMP ifIndex: 517
    Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Link-mode: Half-duplex,
    Speed: 1000mbps, BPDU Error: None, Loop Detect PDU Error: None,
    Ethernet-Switching Error: None, MAC-REWRITE Error: None, Loopback: Disabled,
    Source filtering: Disabled, Flow control: Enabled, Auto-negotiation: Enabled,
    Remote fault: Online
    Device flags   : Present Running
    Interface flags: SNMP-Traps Internal: 0x4000
    Link flags     : None
    CoS queues     : 8 supported, 8 maximum usable queues
    Current address: d0:dd:49:e8:b6:cc, Hardware address: d0:dd:49:e8:b6:cc
    Last flapped   : 2020-05-11 10:22:06 UTC (00:47:39 ago)
    Input rate     : 0 bps (0 pps)
    Output rate    : 0 bps (0 pps)
    Active alarms  : None
    Active defects : None
    PCS statistics                      Seconds
      Bit errors                             0
      Errored blocks                         0
    Ethernet FEC statistics              Errors
      FEC Corrected Errors                    0
      FEC Uncorrected Errors                  0
  ```

```
    FEC Corrected Errors Rate               0
    FEC Uncorrected Errors Rate             0
  Interface transmit statistics: Disabled
```

-
  ```
  user@host> show interfaces ge-0/0/2 | no-more
  Physical interface: ge-0/0/2, Enabled, Physical link is Up
    Interface index: 137, SNMP ifIndex: 518
    Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Link-mode: Half-duplex,
    Speed: 1000mbps, BPDU Error: None, Loop Detect PDU Error: None,
    Ethernet-Switching Error: None, MAC-REWRITE Error: None, Loopback: Disabled,
    Source filtering: Disabled, Flow control: Enabled, Auto-negotiation: Enabled,
    Remote fault: Online
    Device flags   : Present Running
    Interface flags: SNMP-Traps Internal: 0x4000
    Link flags     : None
    CoS queues     : 8 supported, 8 maximum usable queues
    Current address: d0:dd:49:e8:b6:c7, Hardware address: d0:dd:49:e8:b6:c7
    Last flapped   : 2020-05-11 10:22:06 UTC (00:47:52 ago)
    Input rate     : 0 bps (0 pps)
    Output rate    : 0 bps (0 pps)
    Active alarms  : None
    Active defects : None
    PCS statistics                      Seconds
      Bit errors                             0
      Errored blocks                         0
    Ethernet FEC statistics              Errors
      FEC Corrected Errors                    0
      FEC Uncorrected Errors                  0
      FEC Corrected Errors Rate               0
      FEC Uncorrected Errors Rate             0
    Interface transmit statistics: Disabled
  ```

5. Set the root password:

```
user@host# set system root-authentication plain-text-password
```

6. At the first prompt, enter the new root password. At the second prompt, reenter the new root password:

```
New password:
Retype new password:
```

7. After you have finished configuring the password, commit the configuration:

```
user@host# commit
commit complete
```

8. Configure the WAN-side internal-facing interface (ge-0/0/1) as a VLAN-tagged interface and assign an IP address to it:

```
user@host# set interfaces ge-0/0/1 vlan-tagging
user@host# set interfaces ge-0/0/1 unit 0 vlan-id 77
user@host# set interfaces ge-0/0/1 unit 0 family inet address 11.11.11.1/24
user@host# commit
commit complete
```

9. Configure the WAN-side internal-facing interface (ge-0/0/2) as a VLAN-tagged interface and assign an IP address to it:

```
user@host# set interfaces ge-0/0/2 vlan-tagging
user@host# set interfaces ge-0/0/2 unit 0 vlan-id 123
user@host# set interfaces ge-0/0/2 unit 0 family inet address 22.22.22.1/30
user@host# commit
commit complete
```

10. Access the VNF (VNF-2) from the JCP through the console:

```
user@host> request virtual-network-functions console VNF-2
Internal instance: VNF-2
Connected to domain VNF-2
```

11. Log in to the console:

```
user@host:~ # cli
user@host>
```

12. Check the status of the interfaces:

-
  ```
  user@host# show interfaces terse | no-more
  Interface               Admin Link Proto    Local                 Remote
  ge-0/0/0                up    up
  gr-0/0/0                up    up
  ip-0/0/0                up    up
  lsq-0/0/0               up    up
  lt-0/0/0                up    up
  mt-0/0/0                up    up
  sp-0/0/0                up    up
  sp-0/0/0.0              up    up   inet
                                     inet6
  sp-0/0/0.16383          up    up   inet
  ge-0/0/1                up    up
  ge-0/0/2                up    up
  dsc                     up    up
  fti0                    up    up
  fxp0                    up    up
  fxp0.0                  up    up
  gre                     up    up
  ipip                    up    up
  irb                     up    up
  lo0                     up    up
  lo0.16384               up    up   inet     127.0.0.1           --> 0/0
  lo0.16385               up    up   inet     10.0.0.1            --> 0/0
                                              10.0.0.16           --> 0/0
                                              128.0.0.1           --> 0/0
                                              128.0.0.4           --> 0/0
                                              128.0.1.16          --> 0/0
  lo0.32768               up    up
  lsi                     up    up
  mtun                    up    up
  pimd                    up    up
  pime                    up    up
  pp0                     up    up
  ```

```
ppd0                    up    up
ppe0                    up    up
st0                     up    up
tap                     up    up
vlan                    up    down
```

-
  ```
  user@host> show interfaces ge-0/0/0 | no-more
  Physical interface: ge-0/0/0, Enabled, Physical link is Up
    Interface index: 135, SNMP ifIndex: 508
    Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Link-mode: Half-duplex,
    Speed: 1000mbps, BPDU Error: None, Loop Detect PDU Error: None,
    Ethernet-Switching Error: None, MAC-REWRITE Error: None, Loopback: Disabled,
    Source filtering: Disabled, Flow control: Enabled, Auto-negotiation: Enabled,
    Remote fault: Online
    Device flags   : Present Running
    Interface flags: SNMP-Traps Internal: 0x4000
    Link flags     : None
    CoS queues     : 8 supported, 8 maximum usable queues
    Current address: d0:dd:49:e8:b6:c9, Hardware address: d0:dd:49:e8:b6:c9
    Last flapped   : 2020-05-11 10:26:20 UTC (22:53:57 ago)
    Input rate     : 0 bps (0 pps)
    Output rate    : 0 bps (0 pps)
    Active alarms  : None
    Active defects : None
    PCS statistics                      Seconds
      Bit errors                             0
      Errored blocks                         0
    Ethernet FEC statistics              Errors
      FEC Corrected Errors                    0
      FEC Uncorrected Errors                  0
      FEC Corrected Errors Rate               0
      FEC Uncorrected Errors Rate             0
    Interface transmit statistics: Disabled
  ```

-
  ```
  user@host> show interfaces fxp0 | no-more
  Physical interface: fxp0, Enabled, Physical link is Up
    Interface index: 65, SNMP ifIndex: 1
    Type: Ethernet, Link-level type: Ethernet, MTU: 1514, Speed: 1000mbps
    Device flags   : Present Running
    Interface flags: SNMP-Traps
    Link type      : Full-Duplex
  ```

```
  Current address: d0:dd:49:e8:b6:c8, Hardware address: d0:dd:49:e8:b6:c8
  Last flapped   : 2020-05-11 10:25:39 UTC (22:54:38 ago)
    Input packets : 41363
    Output packets: 0

  Logical interface fxp0.0 (Index 3) (SNMP ifIndex 13)
    Flags: Up SNMP-Traps Encapsulation: ENET2
    Input packets : 41320
    Output packets: 0
```

-
  ```
  user@host> show interfaces ge-0/0/1 | no-more
  Physical interface: ge-0/0/1, Enabled, Physical link is Up
    Interface index: 136, SNMP ifIndex: 509
    Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Link-mode: Half-duplex,
    Speed: 1000mbps, BPDU Error: None, Loop Detect PDU Error: None,
    Ethernet-Switching Error: None, MAC-REWRITE Error: None, Loopback: Disabled,
    Source filtering: Disabled, Flow control: Enabled, Auto-negotiation: Enabled,
    Remote fault: Online
    Device flags   : Present Running
    Interface flags: SNMP-Traps Internal: 0x4000
    Link flags     : None
    CoS queues     : 8 supported, 8 maximum usable queues
    Current address: d0:dd:49:e8:b6:cd, Hardware address: d0:dd:49:e8:b6:cd
    Last flapped   : 2020-05-11 10:26:20 UTC (22:53:57 ago)
    Input rate     : 0 bps (0 pps)
    Output rate    : 0 bps (0 pps)
    Active alarms  : None
    Active defects : None
    PCS statistics                      Seconds
      Bit errors                             0
      Errored blocks                         0
    Ethernet FEC statistics              Errors
      FEC Corrected Errors                    0
      FEC Uncorrected Errors                  0
      FEC Corrected Errors Rate               0
      FEC Uncorrected Errors Rate             0
    Interface transmit statistics: Disabled
  ```

-
  ```
  user@host> show interfaces ge-0/0/2 | no-more
  Physical interface: ge-0/0/2, Enabled, Physical link is Up
    Interface index: 137, SNMP ifIndex: 510
  ```

```
    Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Link-mode: Half-duplex,
    Speed: 1000mbps, BPDU Error: None, Loop Detect PDU Error: None,
    Ethernet-Switching Error: None, MAC-REWRITE Error: None, Loopback: Disabled,
    Source filtering: Disabled, Flow control: Enabled, Auto-negotiation: Enabled,
    Remote fault: Online
    Device flags   : Present Running
    Interface flags: SNMP-Traps Internal: 0x4000
    Link flags     : None
    CoS queues     : 8 supported, 8 maximum usable queues
    Current address: d0:dd:49:e8:b6:ce, Hardware address: d0:dd:49:e8:b6:ce
    Last flapped   : 2020-05-11 10:26:20 UTC (22:53:57 ago)
    Input rate     : 0 bps (0 pps)
    Output rate    : 0 bps (0 pps)
    Active alarms  : None
    Active defects : None
    PCS statistics                      Seconds
      Bit errors                            0
      Errored blocks                        0
    Ethernet FEC statistics              Errors
      FEC Corrected Errors                   0
      FEC Uncorrected Errors                 0
      FEC Corrected Errors Rate              0
      FEC Uncorrected Errors Rate            0
    Interface transmit statistics: Disabled
```

13. Set the root password:

```
user@host# set system root-authentication plain-text-password
```

14. At the first prompt, enter the new root password. At the second prompt, reenter the new root password:

```
New password:
Retype new password:
```

15. After you have finished configuring the password, commit the configuration:

```
user@host# commit
commit complete
```

16. Configure the WAN-side internal-facing interface (ge-0/0/1) as a VLAN-tagged interface and assign an IP address to it:

```
user@host# set interfaces ge-0/0/1 vlan-tagging
user@host# set interfaces ge-0/0/1 unit 0 vlan-id 123
user@host# set interfaces ge-0/0/1 unit 0 family inet address 22.22.22.2/30
user@host# commit
commit complete
```

17. Configure the WAN-side internal-facing interface (ge-0/0/2) as a VLAN-tagged interface and assign an IP address to it:

```
user@host# set interfaces ge-0/0/2 vlan-tagging
user@host# set interfaces ge-0/0/2 unit 0 vlan-id 1177
user@host# set interfaces ge-0/0/2 unit 0 family inet address 33.33.33.2/30
user@host# commit
commit complete
```

Configuring Security in NFX350

Step-by-Step Procedure

1. Clear the current security settings:

```
user@host# delete security
```

2. Configure security options:

```
user@host# set security forwarding-options family inet6 mode flow-based
```

3. Configure security policies:

```
user@host# set security policies default-policy permit-all
```

4. Configure security zones:

```
user@host# set security zones security-zone trust host-inbound-traffic system-services all
user@host# set security zones security-zone trust host-inbound-traffic protocols all
user@host# set security zones security-zone trust interfaces all
```

Configuring Security in vSRX Virtual Firewall VNFs

Step-by-Step Procedure

1. Clear the current security settings:

```
user@host# delete security
```

2. Configure security options:

```
user@host# set security forwarding-options family inet6 mode flow-based
```

3. Configure security policies:

```
user@host# set security policies default-policy permit-all
```

4. Configure security zones:

```
user@host# set security zones security-zone trust host-inbound-traffic system-services all
user@host# set security zones security-zone trust host-inbound-traffic protocols all
user@host# set security zones security-zone trust interfaces all
```

15

CHAPTER

Monitoring and Troubleshooting

IN THIS CHAPTER

Configuring SNMP on NFX150, NFX250 NextGen, and NFX350 Devices

IN THIS SECTION

SNMP monitors network devices from a central location. NFX Series (NFX150, NFX250 NextGen, and NFX350) devices support querying of MIB data by using SNMPv2c and SNMPv3. Separate SNMP agents (known as the SNMP process or snmpd) reside on the vjunos0 and Host OS. The vjunos0 acts as the proxy for the Host OS. The system and chassis related MIB data is available in vjunos0.

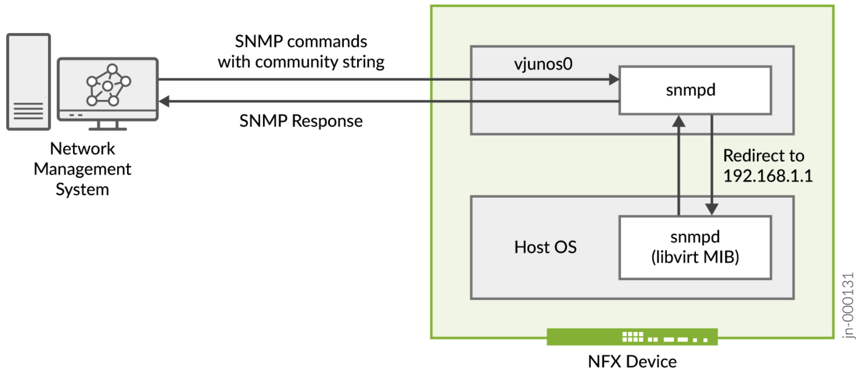
Starting in Junos OS Release 21.4R1, NFX Series devices support LM-SENSORS-MIB, ENTITY-SENSORS-MIB, and libvirt MIB. The LM-SENSORS-MIB and ENTITY-SENSORS-MIB data is available on vjunos0 whereas the libvirt MIB data is available on the Host OS. You can use the libvirt MIB to monitor virtual machines. This topic discusses the SNMP implementation for the libvirt MIB.

How to Configure SNMPv2c to Access Libvirt MIB Data

SNMPv2c uses community strings, which act as passwords when determining the SNMP clients and how clients can access the data in the SNMP agent. The community string is not pre-configured on NFX Series devices. To access MIBs data using SNMPv2c, you must configure a community string and an SNMP proxy for the Host OS. The community string is added to the Host OS.

Figure 18 on page 235 illustrates the communication flow for SNMPv2c on NFX Series devices.

Figure 18: Communication Flow for SNMPv2c on NFX Series Devices



When a user issues SNMP commands like `snmpwalk`, `snmpget` with the community string from the network management server:

- The request goes to the SNMP daemon in vjunos0. The SNMPD reads the community string in the SNMP request and redirects the request to the Host OS using the internal routing instance nfx-host.
- The SNMPD in the Host OS processes the request and sends the response to vjunos0, which then sends it to the network management server.

To configure SNMPv2c:

1. Configure the SNMPv2c community string in the Host OS:

```
root@host# set vmhost snmp v2c community community-name
```

NOTE: Ensure that a community with the same name does not already exist on the device.

2. Configure the proxy in vjunos0:

```
root@host# set snmp proxy snmp-proxy-name device-name 192.168.1.1
root@host# set snmp proxy snmp-proxy-name version-v2c snmp-community community-name
root@host# set snmp proxy snmp-proxy-name nfx-host
```

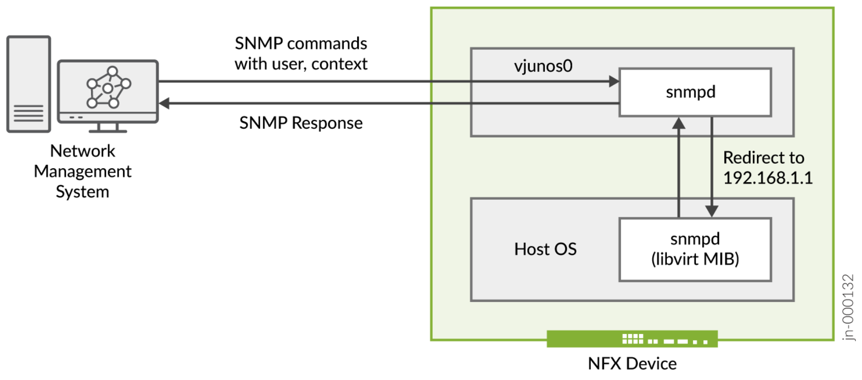
To enable traps, see "How to Enable libvirt SNMPv2c Trap Support" on page 242.

How to Configure SNMPv3 to Access Libvirt MIB Data

SNMPv3 provides a secure way to access MIB data as it supports authentication and encryption. SNMPv3 uses the user-based security model (USM) for message security and the view-based access control model (VACM) for access control. USM specifies authentication and encryption, and VACM specifies access-control rules. Figure 19 on page 236 illustrates the communication flow for SNMPv3 on NFX Series devices. For SNMPv3, you must create:

- An SNMPv3 user under the vmhost hierarchy in Host OS with the authentication type and privacy
- An SNMPv3 proxy with the user name and context

Figure 19: Communication Flow for SNMPv3 on NFX Series Devices



When a user issues SNMP commands like (`snmpwalk`, `snmpget`) with the user name and authentication credentials from the network management server:

- The request goes to the SNMP daemon in vjunos0. The SNMPD reads the context for the Host OS in the SNMP request and redirects the request to the Host OS using the internal routing instance nfx-host.
- The SNMPD in the Host OS processes the request and sends the response to vjunos0, which then sends it to the network management server.

To configure SNMPv3:

1. Configure the local engine information for USM:

```
root@host# set snmp v3 usm local-engine user snmpv3-user-name  authentication-type
authentication-password Authentication_Password
```

```
root@host# set snmp v3 usm local-engine user snmpv3-user-name  privacy-type privacy-password
Privacy_Password
```

2. Configure the remote engine and remote user. You must configure the remote-engine id as 80001f8804686f7374. The remote engine ID is used to compute the security digest for authenticating and encrypting packets sent to a user on the remote host. When sending an inform message, the agent uses the credentials of the user configured on the remote engine (inform target).

```
root@host# set snmp v3 usm remote-engine 80001f8804686f7374 user snmpv3-user-name
authentication-type authentication-password Authentication_Password
root@host# set snmp v3 usm remote-engine 80001f8804686f7374 user snmpv3-user-name  privacy-
type privacy-password Privacy_Password
```

3. Configure VACM:

```
root@host# set snmp v3 vacm security-to-group security-model usm security-name snmpv3-user-
name  group Group-Name
root@host# set snmp v3 vacm access group Group-Name context-prefix snmpv3-context-name
security-model usm security-level authentication read-view iso
```

4. Configure SNMPv3 on the Host OS:

```
root@host# set vmhost snmp v3 user snmpv3-user-name authentication-type authentication-
password Authentication_Password
root@host# set vmhost snmp v3 user snmpv3-user-name privacy-type privacy-password
Privacy_Password
```

5. Configure SNMP v3 proxy:

```
root@host# set snmp proxy snmpv3-proxy-name device-name 192.168.1.1
root@host# set snmp proxy snmpv3-proxy-name version-v3 security-name snmpv3-user-name
root@host# set snmp proxy snmpv3-proxy-name version-v3 context snmpv3-context-name
root@host# set snmp proxy snmpv3-proxy-name nfx-host
```

To enable traps, see "How to Enable libvirt SNMPv3 Trap Support" on page 243.

How to Query Libvirt MIB Data

You can use the **snmpget**, **snmpgetnext**, and **snmpwalk** commands to read the MIB information. Note that you cannot use **snmpset** to configure the libvirt MIB.

The libvirt MIB provides the following information:

- Name of active virtual guest (domain name)
- Current state of the active guest (state of domain)
- Number of virtual CPUs the virtual guest uses (cpu count defined for domain)
- Current amount of memory (in MiB) used by the virtual guest (current allocated memory)
- Memory limit for the domain (the maximum amount of memory (in MiB) that can be used by the virtual guest)
- CPU time used by the virtual guest, in nanoseconds (CPU time)
- Status of the virtual guest (row status)

The following are sample outputs of the **snmpwalk** command when you execute it on NMS:

- SNMPv2c:

```
nms_server# snmpwalk -v2c -Obs -c community-name device_A LIBVIRT-MIB::libvirtMIB
libvirtGuestName.51.145.57.25.141.222.70.137.181.75.129.190.178.93.132.174 = STRING: "centos1"
libvirtGuestName.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = STRING: "vjunos0"
libvirtGuestState.51.145.57.25.141.222.70.137.181.75.129.190.178.93.132.174 = INTEGER:
running(1)
libvirtGuestState.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = INTEGER: running(1)
libvirtGuestCpuCount.51.145.57.25.141.222.70.137.181.75.129.190.178.93.132.174 = Gauge32: 1
libvirtGuestCpuCount.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = Gauge32: 1
libvirtGuestMemoryCurrent.51.145.57.25.141.222.70.137.181.75.129.190.178.93.132.174 =
Gauge32: 512
libvirtGuestMemoryCurrent.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = Gauge32: 1954
libvirtGuestMemoryLimit.51.145.57.25.141.222.70.137.181.75.129.190.178.93.132.174 = Gauge32:
512
libvirtGuestMemoryLimit.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = Gauge32: 1954
libvirtGuestCpuTime.51.145.57.25.141.222.70.137.181.75.129.190.178.93.132.174 = Counter64:
12430000000
libvirtGuestCpuTime.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = Counter64:
808830000000
libvirtGuestRowStatus.51.145.57.25.141.222.70.137.181.75.129.190.178.93.132.174 = INTEGER:
```

```
active(1)
libvirtGuestRowStatus.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = INTEGER:
active(1)
libvirtGuestRowStatus.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = No more
variables left in this MIB View (It is past the end of the MIB tree)
```

- SNMPv3:

```
nms-server# snmpwalk -Obs -v3 -a authentication-type -A Authentication_Password -x privacy-
type -X Privacy-Password -l authPriv-level -u snmpv3-user-name -n snmpv3-context-name
device_B LIBVIRT-MIB::libvirtMIB
libvirtGuestName.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = STRING: "vjunos0"
libvirtGuestName.185.178.34.217.212.63.76.189.191.225.54.201.166.88.167.118 = STRING: "vnf0"
libvirtGuestState.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = INTEGER: running(1)
libvirtGuestState.185.178.34.217.212.63.76.189.191.225.54.201.166.88.167.118 = INTEGER:
running(1)
libvirtGuestCpuCount.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = Gauge32: 1
libvirtGuestCpuCount.185.178.34.217.212.63.76.189.191.225.54.201.166.88.167.118 = Gauge32: 1
libvirtGuestMemoryCurrent.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = Gauge32: 1954
libvirtGuestMemoryCurrent.185.178.34.217.212.63.76.189.191.225.54.201.166.88.167.118 =
Gauge32: 512
libvirtGuestMemoryLimit.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = Gauge32: 1954
libvirtGuestMemoryLimit.185.178.34.217.212.63.76.189.191.225.54.201.166.88.167.118 = Gauge32:
512
libvirtGuestCpuTime.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = Counter64:
959760000000
libvirtGuestCpuTime.185.178.34.217.212.63.76.189.191.225.54.201.166.88.167.118 = Counter64:
19830000000
libvirtGuestRowStatus.170.187.204.221.238.255.66.227.0.0.86.74.85.78.79.83 = INTEGER:
active(1)
libvirtGuestRowStatus.185.178.34.217.212.63.76.189.191.225.54.201.166.88.167.118 = INTEGER:
active(1)
```

The following is a sample output of the **snmpwalk** command when you exexute it on the NFX Series device:

```
root@host> show vmhost snmp mib walk LIBVIRT-MIB::libvirtMIB
LIBVIRT-MIB::libvirtGuestName[STRING: 33913919-8dde-4689-b54b-81beb25d84ae] = STRING: "centos1"
LIBVIRT-MIB::libvirtGuestName[STRING: aabbccdd-eeff-42e3-0-564a554e4f53] = STRING: "vjunos0"
LIBVIRT-MIB::libvirtGuestState[STRING: 33913919-8dde-4689-b54b-81beb25d84ae] = INTEGER:
running(1)
```

```
LIBVIRT-MIB::libvirtGuestState[STRING: aabbccdd-eeff-42e3-0-564a554e4f53] = INTEGER: running(1)
LIBVIRT-MIB::libvirtGuestCpuCount[STRING: 33913919-8dde-4689-b54b-81beb25d84ae] = Gauge32: 1
LIBVIRT-MIB::libvirtGuestCpuCount[STRING: aabbccdd-eeff-42e3-0-564a554e4f53] = Gauge32: 1
LIBVIRT-MIB::libvirtGuestMemoryCurrent[STRING: 33913919-8dde-4689-b54b-81beb25d84ae] = Gauge32:
512
LIBVIRT-MIB::libvirtGuestMemoryCurrent[STRING: aabbccdd-eeff-42e3-0-564a554e4f53] = Gauge32: 1954
LIBVIRT-MIB::libvirtGuestMemoryLimit[STRING: 33913919-8dde-4689-b54b-81beb25d84ae] = Gauge32: 512
LIBVIRT-MIB::libvirtGuestMemoryLimit[STRING: aabbccdd-eeff-42e3-0-564a554e4f53] = Gauge32: 1954
LIBVIRT-MIB::libvirtGuestCpuTime[STRING: 33913919-8dde-4689-b54b-81beb25d84ae] = Counter64:
12300000000
LIBVIRT-MIB::libvirtGuestCpuTime[STRING: aabbccdd-eeff-42e3-0-564a554e4f53] = Counter64:
734900000000
LIBVIRT-MIB::libvirtGuestRowStatus[STRING: 33913919-8dde-4689-b54b-81beb25d84ae] = INTEGER:
active(1)
LIBVIRT-MIB::libvirtGuestRowStatus[STRING: aabbccdd-eeff-42e3-0-564a554e4f53] = INTEGER:
active(1)
```

Supported Chassis MIBs and Traps

NFX Series devices support the following chassis MIBs:

- jnxFruContentsIndex
- jnxFruL1Index
- jnxFruL2Index
- jnxFruL3Index
- jnxFruName
- jnxFruType
- jnxFruSlot
- jnxFruTemp
- jnxFruOfflineReason
- jnxFruLastPowerOff
- jnxFruLastPowerOn
- jnxFruPowerUpTime

- jnxFruChassisId
- jnxFruChassisDescr
- jnxFruPsdAssignment

NFX Series devices support the following traps:

- jnxFanFailure
- jnxFanOK
- jnxPowerSupplyFailure
- jnxPowerSupplyOK
- jnxOverTemperature
- jnxTemperatureOK
- jnxPowerSupplyRemoved (only for NFX350)

Supported libvirt MIB Traps

IN THIS SECTION

The libvirt MIB monitors the virtual machines and sends asynchronous traps to the network management server. For example, if a domain (VNF) crashes unexpectedly, a notification is sent to the network management server. The traps are generated in the Host OS and sent to the snmptrapd daemon on vjunos0. The snmptrapd daemon forwards the traps to the network management server.

The libvirt trap has the following definition structure:

```
libvirtGuestNotif NOTIFICATION-TYPE
      OBJECTS { libvirtGuestName,
                  libvirtGuestUUID,
                  libvirtGuestState,
                 libvirtGuestRowStatus }
          STATUS current
          DESCRIPTION
             "Guest lifecycle notification."
          ::= { libvirtNotifications 1 }
```

Here is a sample output of an snmp libvirt trap:

```
SNMPv2-MIB::snmpTrapOID.0 = OID: LIBVIRT-MIB::libvirtGuestNotif,
 LIBVIRT-MIB::libvirtGuestName.0 = STRING: "test1",
 LIBVIRT-MIB::libvirtGuestUUID.1 = STRING: 7ad4bc2a-16db-d8c0-1f5a-6cb777e17cd8,
 LIBVIRT-MIB::libvirtGuestState.2 = INTEGER: running(1),
 LIBVIRT-MIB::libvirtGuestRowStatus.3 = INTEGER: active(1)
```

The current state of the active guest can be one of the following:

- running(1)
- blocked(2)
- paused(3)
- shutdown(4)
- shutoff(5)
- crashed(6)

How to Enable libvirt SNMPv2c Trap Support

To enable SNMPv2c trap support:

1. Configure the community name for the trap:

   ```
   root@host# set vmhost snmp v2c-trap trap-community community-name
   ```

2. Configure the client-address, which is the source address from which the trap originates. If you do not configure the source address, the hypervisor address (192.168.1.1) is used as the client address.

   ```
   root@host# set vmhost snmp client-address client-ip
   ```

3. Configure port-forwarding. You can configure multiple IP adresses.

```
root@host#  set forwarding-options helpers port 162 server IP-address-of-trap-target
```

How to Enable libvirt SNMPv3 Trap Support

To enable SNMPv3 trap support:

1. Configure the user name for the trap:

```
root@host# set vmhost snmp v3-trap trap-user user-name
```

2. Configure the client-address, which is the source address from which the trap originates. If you do not configure the source address, the hypervisor address (192.168.1.1) is used as the client address.

```
root@host# set vmhost snmp client-address client-ip
```

3. Configure the AES and SHA passwords for the user:

```
root@host# set vmhost snmp v3 user user-name authentication-sha authentication-password
password
root@host# set vmhost snmp v3 user user-name privacy-aes128 privacy-password password
```

4. Configure port-forwarding for libVirtMIB trap support. You can configure multiple IP adresses.

```
root@host#  set forwarding-options helpers port 162 server IP-address-of-trap-target
```

Recovering the Root Password for NFX150, NFX250 NextGen, and NFX350 Devices

The root password on your Junos OS-enabled device helps to prevent unauthorized users from making changes to your network.

If you forget the root password, you can use the password recovery procedure to reset the root password.

NOTE: You need console access to the device to recover the root password.

To recover the root password:

1. Power off the device by switching off the AC power outlet of the device or, if necessary, by pulling the power cords out of the device's power supplies.
2. Turn off the power to the management device, such as a PC or laptop computer, that you want to use to access the CLI.
3. Plug one end of the Ethernet rollover cable supplied with the device into the RJ-45 to DB-9 serial port adapter supplied with the device.
4. Plug the RJ-45 to DB-9 serial port adapter into the serial port on the management device.
5. Connect the other end of the Ethernet rollover cable to the console port on the device.
6. Turn on the power to the management device.
7. On the management device, start any asynchronous terminal emulation application (such as Microsoft Windows HyperTerminal), and select the port to be used.
8. Configure the port settings as follows:
   - Bits per second—9600
   - Data bits—8
   - Parity—None
   - Stop bits—1
   - Flow control—None
9. Power on the device by plugging the power cords into the device's power supply (if necessary), or by turning on the power to the device by switching on the AC power outlet that the device is plugged into.

   The terminal emulation screen on your management device displays the device's boot sequence.

```
i2cset -y 5 0x19 0xff 0x05
i2cset -y 5 0x19 0x2d 0x81
i2cset -y 5 0x19 0x15 0x12
i2cset -y 5 0x18 0xff 0x05
i2cset -y 5 0x18 0x2d 0x82
i2cset -y 5 0x18 0x15 0x12
 * Stopping virtualization library daemon: libvirtd
```

[This message is truncated...]

```
Checking Prerequisites
jdm docker container is in Exit state, required to cleanup, please wait...
9dba6935234b
[  OK  ]
Launching jdm container 'jdm'...
```

10. When the prompt shows `Launching jdm container 'jdm'`, press **Ctrl+C**. The **Main Menu** appears.

```
Main Menu

1.  Boot [J]unos volume
2.  Boot Junos volume in [S]afe mode
3.  [R]eboot
4.  [B]oot menu
5.  [M]ore options
```

11. From the **Main Menu**, select **5. [M]ore options**. The **Options Menu** appears.

```
Options Menu

1.  Recover [J]unos volume
2.  Recovery mode - [C]LI
3.  Check [F]ile system
4.  Enable [V]erbose boot
5.  [B]oot prompt
6.  [M]ain menu
```

12. From the **Options Menu**, select **2. Recovery mode - [C]LI**. The device reboots into CLI recovery mode.

```
Booting Junos in CLI recovery mode ...

it will boot in recovery mode and will get MGD cli

/packages/sets/active/boot/os-kernel/kernel text=0x444c38 data=0x82348+0x2909a0
syms=[0x8+0x94c50+0x8+0x8165b]
/packages/sets/active/boot/os-kernel/contents.izo size=0x84d200
/packages/sets/active/boot/os-kernel/miibus.ko size 0x40778 at 0x14bc000
```

```
loading required module 'netstack'
/packages/sets/active/boot/netstack/netstack.ko size 0x1386b08 at 0x14fd000
loading required module 'crypto'
```

[This message is truncated...]

```
Starting MGD
mgd: error: could not open database: /var/run/db/schema.db: No such file or directory
mgd: error: could not open database schema: /var/run/db/schema.db
mgd: error: could not open database schema
mgd: error: database schema is out of date, rebuilding it
mgd: error: could not open database: /var/run/db/juniper.data: No such file or directory
mgd: error: Cannot read configuration: Could not open configuration database
mgd: warning: schema: dbs_remap_daemon_index: could not find daemon name 'isdnd'
 Starting CLI ...
```

13. Enter configuration mode in the CLI.

```
root> configure
Entering configuration mode
```

14. Set the root password.

```
[edit]
root# set system root-authentication plain-text-password
```

15. At the first prompt, enter the new root password:

```
New password:
```

16. At the second prompt, reenter the new root password.

```
Retype new password:
```

17. After you have finished configuring the password, commit the configuration.

```
[edit]
root# commit
commit complete
```

18. Exit configuration mode in the CLI.

```
[edit]
root@host# exit
root@host>
```

19. Exit operational mode in the CLI.

```
root@host> exit
root@host%
```

20. At the shell prompt, type **exit** to reboot the device.

```
root@host% exit
```

RELATED DOCUMENTATION

Configuring the Root Password

Troubleshooting Interfaces on NFX Devices

IN THIS SECTION

Monitoring Interface Status and Traffic on NFX Series Devices

IN THIS SECTION

Purpose

View the interface status to monitor bandwidth utilization and traffic statistics of an interface.

Action

To view the status of an interface:

```
user@host> show interfaces interface-name
```

For example:

- To view the status of an interface for an NFX350 device:

```
user@host> show interfaces ge-0/0/0 | no-more
Physical interface: ge-0/0/0, Enabled, Physical link is Down
  Interface index: 150, SNMP ifIndex: 514
  Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Link-mode: Unknown,
  Speed: 1000mbps, Duplex: Full-Duplex, BPDU Error: None,
  Loop Detect PDU Error: None, Ethernet-Switching Error: None,
  MAC-REWRITE Error: None, Loopback: Disabled, Source filtering: Disabled,
  Flow control: Enabled, Auto-negotiation: Enabled, Remote fault: Online,
  IEEE 802.3az Energy Efficient Ethernet: Disabled, Auto-MDIX: Enabled
  Device flags   : Present Running Down
  Interface flags: Hardware-Down SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 12 supported, 12 maximum usable queues
  Current address: d0:dd:49:e8:6e:7d, Hardware address: d0:dd:49:e8:6e:7d
  Last flapped   : 2020-02-19 06:17:42 UTC (00:25:17 ago)
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)
  Active alarms  : LINK
  Active defects : LINK
  PCS statistics                    Seconds
    Bit errors                           0
    Errored blocks                       0
  Ethernet FEC statistics              Errors
    FEC Corrected Errors                    0
    FEC Uncorrected Errors                  0
    FEC Corrected Errors Rate               0
    FEC Uncorrected Errors Rate             0
  PRBS Statistics : Disabled
  Interface transmit statistics: Disabled

  Logical interface ge-0/0/0.0 (Index 74) (SNMP ifIndex 523)
    Flags: Device-Down SNMP-Traps 0x24024000 Encapsulation: Ethernet-Bridge
    Input packets : 0
    Output packets: 0
    Protocol eth-switch, MTU: 1514
```

```
user@host> show interfaces xe-0/0/15 | no-more
Physical interface: xe-0/0/15, Enabled, Physical link is Up
  Interface index: 145, SNMP ifIndex: 557
  Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Speed: 10Gbps,
  BPDU Error: None, Loop Detect PDU Error: None, Ethernet-Switching Error: None,
  MAC-REWRITE Error: None, Loopback: Disabled, Source filtering: Disabled,
  Flow control: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 12 supported, 12 maximum usable queues
  Current address: d0:dd:49:e8:6e:8c, Hardware address: d0:dd:49:e8:6e:8c
  Last flapped   : 2020-02-19 06:17:43 UTC (00:25:32 ago)
  Input rate     : 0 bps (0 pps)
  Output rate    : 232 bps (0 pps)
  Active alarms  : None
  Active defects : None
  PCS statistics                      Seconds
    Bit errors                             0
    Errored blocks                         0
  Ethernet FEC statistics              Errors
    FEC Corrected Errors                    0
    FEC Uncorrected Errors                  0
    FEC Corrected Errors Rate               0
    FEC Uncorrected Errors Rate             0
  PRBS Statistics : Disabled
  Interface transmit statistics: Disabled
  Logical interface xe-0/0/15.0 (Index 72) (SNMP ifIndex 558)
    Flags: Up SNMP-Traps 0x24024000 Encapsulation: Ethernet-Bridge
    Input packets : 0
    Output packets: 57
    Protocol eth-switch, MTU: 1514
      Flags: Is-Primary
```

```
user@host> **show interfaces ge-1/0/1 | no-more**
Physical interface: ge-1/0/1, Enabled, Physical link is Up
  Interface index: 168, SNMP ifIndex: 538
  Link-level type: Ethernet, MTU: 1518, LAN-PHY mode, Link-mode: Half-duplex,
  Speed: 1000mbps, BPDU Error: None, Loop Detect PDU Error: None,
  Ethernet-Switching Error: None, MAC-REWRITE Error: None, Loopback: Disabled,
  Source filtering: Disabled, Flow control: Enabled, Auto-negotiation: Enabled,
  Remote fault: Online
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues     : 8 supported, 8 maximum usable queues
  Current address: d0:dd:49:e8:6e:96, Hardware address: d0:dd:49:e8:6e:96
  Last flapped   : 2020-02-19 06:18:30 UTC (00:24:55 ago)
  Input rate     : 0 bps (0 pps)
  Output rate    : 208 bps (0 pps)
  Active alarms  : None
  Active defects : None
  PCS statistics                      Seconds
    Bit errors                             0
    Errored blocks                         0
  Ethernet FEC statistics              Errors
    FEC Corrected Errors                    0
    FEC Uncorrected Errors                  0
    FEC Corrected Errors Rate               0
    FEC Uncorrected Errors Rate             0
  PRBS Statistics : Disabled
  Interface transmit statistics: Disabled
 Logical interface ge-1/0/1.2 (Index 85) (SNMP ifIndex 544)
    Flags: Up SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.2 ]  Encapsulation: ENET2
    Input packets : 0
    Output packets: 19
    Security: Zone: Null
    Protocol inet, MTU: 1500
    Max nh cache: 75000, New hold nh limit: 75000, Curr nh cnt: 0,
    Curr new hold cnt: 0, NH drop cnt: 0
      Flags: Sendbcast-pkt-to-re
    Protocol inet6, MTU: 1500
    Max nh c
```

- To view the status of an interface for an NFX150 device:

```
user@host> show interfaces heth-0-1
Physical interface: heth-0-1, Enabled, Physical link is Up
  Link-level type: Ethernet, Media type: Copper, MTU: 9192, Speed: 1Gbps, Duplex: Full-
duplex, Auto-negotiation: Enabled
  Device flags   : Present Running
  Current address: 00:00:5e:00:53:8e, Hardware address: 00:00:5e:00:53:8e
```

- To view the status of the interface for an NFX250 device:

```
user@host> show interfaces xe-0/0/12
Physical interface: xe-0/0/12, Enabled, Physical link is Up
Interface index: 145, SNMP ifIndex: 509
Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Speed: 10Gbps, BPDU Error: None, Loop
Detect PDU Error: None,
Ethernet-Switching Error: None, MAC-REWRITE Error: None, Loopback: Disabled, Source
filtering: Disabled, Flow control: Enabled
Device flags : Present Running
Interface flags: SNMP-Traps Internal: 0x4000
Link flags : None
CoS queues : 12 supported, 12 maximum usable queues
Current address: 30:7c:5e:4c:78:0f, Hardware address: 30:7c:5e:4c:78:0f
Last flapped : 2018-12-10 19:53:35 UTC (2d 03:08 ago)
Input rate : 0 bps (0 pps)
Output rate : 0 bps (0 pps)
Active alarms : None
Active defects : None
PCS statistics Seconds
Bit errors 0
Errored blocks 0
Ethernet FEC statistics Errors
FEC Corrected Errors 0
FEC Uncorrected Errors 0
FEC Corrected Errors Rate 0
FEC Uncorrected Errors Rate 0
```

```
PRBS Statistics : Disabled
Interface transmit statistics: Disabled
```

16

CHAPTER

Configuration Statements and Operational Commands

IN THIS CHAPTER

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Read this guide to learn about the syntax and options that make up the statements and commands. Also understand the contexts in which you'll use these CLI elements in your network configurations and operations.

- Junos CLI Reference

Click the links to access Junos OS and Junos OS Evolved configuration statement and command summary topics.

- Configuration Statements
- Operational Commands