Juniper Networks | Engineering Simplicity

How to Configure the NFX150

Published
2024-06-13

Junos

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

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

How to Configure the NFX150
Copyright © 2024 Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide | x

1 Overview

NFX150 System Overview | 2

- Overview | 2
- NFX150 Models | 3
- Benefits and Uses of NFX150 | 6

NFX150 Feature Overview | 7

- Software Architecture | 7
- Interfaces | 9
- Supported Features | 15
- Performance Modes | 16
- Licensing | 22

Junos OS Releases Supported on NFX Series Hardware | 23

NFX Product Compatibility | 25

2 Initial Configuration

Initial Configuration on NFX150 Devices | 29

- Factory-Default Settings | 29
- Enabling Basic Connectivity | 31
- Establishing the Connection | 33

Zero Touch Provisioning on NFX Series Devices | 34

- Understanding Zero Touch Provisioning | 34
- Pre-staging an NFX Series Device | 35
- Provisioning an NFX Series Device | 38
- Provisioning an NFX Series Device Using Sky Enterprise | 39

3 Generating YANG Files

YANG files on NFX150 Devices | 41

Understanding YANG on NFX150 Devices | 41

Generating YANG Files | 42

4 Configuring Interfaces

Mapping Interfaces on NFX150 Devices | 45

Mapping Physical Interfaces to Virtual Interfaces on NFX150 Devices | 45

Mapping Physical Ports to VNF Interfaces Through SR-IOV | 46

Mapping Layer 3 Dataplane Interfaces to OVS | 46

Configuring the In-Band Management Interface | 47

ADSL2 and ADSL2+ Interfaces on NFX150 Devices | 48

ADSL Interface Overview | 48

Configuring ADSL SFP Interface Using VLANs on NFX150 Network Services Platform | 50

Configuring ADSL SFP Interface Without Using VLANs on NFX150 Network Services Platform | 52

VDSL2 Interfaces on NFX150 Devices | 54

VDSL Interface Overview | 54

VDSL2 Network Deployment Topology | 55

VDSL2 Interface Support on NFX Series Devices | 57

Configuring VDSL SFP Interface Using VLANs on NFX150 Network Services Platform | 59

Configuring VDSL SFP Interface Without Using VLANs on NFX150 Network Services Platform | 61

Configuring the LTE Module on NFX Devices | 63

Configuring the LTE Module for Primary Mode | 64

Configuring the LTE Module for Dial-on-Demand Mode | 66

Configuring the LTE Module for Backup Mode | 68

Configuring the LTE Interface Module in an NFX Chassis Cluster | 70

Upgrading the Modem Firmware on NFX Devices Through Over-the-Air (OTA) | 75

5 Configuring USB Pass-Through on NFX Series Devices

Supporting File Transfer from USB on NFX Series Devices | 80

Installing Software on NFX Devices Using USB Autoinstallation | 82

Preparing the USB | 83

Configuring the NFX Device | 85

Installing the Image | 86

Disabling Autoinstallation | 87

6 Configuring Security

IP Security on NFX Devices | 89

Overview | 89

Configuring Security | 91

Configuring Interfaces | 91

Configuring Routing Options | 92

Configuring Security IKE | 93

Configuring Security IPsec | 96

Configuring Security Policies | 98

Configuring Security Zones | 99

Content Security on NFX Devices | 99

Application Security on NFX Devices | 100

Intrusion Detection and Prevention on NFX Devices | 101

Integrated User Firewall Support on NFX Devices | 102

7 Configuring VNFs

Prerequisites to Onboard Virtual Network Functions on NFX150 Devices | 105

Prerequisites for VNFs | 105

Configuring VNFs on NFX150 Devices | 106

Load the VNF Image | 106

Prepare the Bootstrap Configuration | 107

Allocate CPUs for a VNF | 108

Allocate Memory for a VNF | 110

(Optional) Attach a Config Drive to the VNF | 111

Configure Interfaces and VLANs for a VNF | 113

Configuring VNF Storage Devices | 117

Instantiating a VNF | 118

Verify that the VNF Instantiated Successfully | 119

Managing VNFs on NFX Series Devices | 120

Managing VNF States | 120

Managing VNF MAC Addresses | 121

Managing the MTU of a VNF Interface | 122

Accessing a VNF from the JCP | 123

Viewing the List of VNFs | 123

Displaying the Details of a VNF | 124

Deleting a VNF | 124

Non-Root User Access for VNF Console | 125

Configuring Analyzer VNF and Port-mirroring | 128

Supporting Faster File Copy or Transfer | 129

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

Mapping of Address and Port with Encapsulation on NFX Series Devices | 134

Overview | 134

Benefits of MAP-E | 134

MAP-E Terminology | 135

MAP-E Functionality | 135

Configuring MAP-E on NFX Series Devices | 136

Overview | 137

Requirements | 137

Topology Overview | 137

Configure an NFX Series Device as a MAP-E CE Device | 138

Configure an MX Series Device as a BR Device | 141

Verify the MAP-E Configuration | 143

9 Configuring High Availability

Chassis Cluster on NFX150 Devices | 149

NFX150 Chassis Cluster Overview | 149

Chassis Cluster Interfaces | 150

Chassis Cluster Limitation | 151

Example: Configuring a Chassis Cluster on NFX150 Devices | 151

Requirements | 151

Overview | 152

Configuration | 153

Verification | 159

Upgrading or Disabling a Chassis Cluster on NFX150 Devices | 163

Upgrading Individual Devices in a Chassis Cluster Separately | 163

Disabling a Chassis Cluster | 164

10 Configuring Service Chaining

Service Chaining on NFX150 Devices | 166

Understanding Service Chaining | 166

Configuring Service Chaining Using VLANs | 166

Configuring Service Chaining Using DHCP Services on VLANs | 167

Example: Configuring Service Chaining Using VLANs on NFX150 Network Services Platform | 168

Requirements | 169

Overview | 169

Configuration | 170

Example: Configuring Service Chaining Using SR-IOV on NFX150 Network Services Platform | 175

Requirements | **175**

Overview | **175**

Configuration | **177**

Example: Configuring Service Chaining Using a Custom Bridge | 178

Requirements | **179**

Overview | **179**

Configuration | **180**

Verifying the Configuration | **183**

Example: Configuring Service Chaining for LAN-WAN Routing | 187

Requirements | **187**

Overview | **188**

Configuration | **189**

Verification | **190**

Example: Configuring Cross Connect on NFX150 Devices | 193

Requirements | **193**

Overview | **193**

Configuration | **195**

Verifying the Configuration | **198**

Example: Configuring Service Chaining for LAN Routing | 203

Requirements | **203**

Overview | **204**

Configuration | **205**

Example: Configuring Cross-Connect Using a Custom Bridge on NFX150 Devices | 207

Requirements | **207**

Overview | **207**

Configuration | 209

Verifying the Configuration | 212

11 Monitoring and Troubleshooting

Configuring SNMP on NFX150, NFX250 NextGen, and NFX350 Devices | 219

How to Configure SNMPv2c to Access Libvirt MIB Data | 219

How to Configure SNMPv3 to Access Libvirt MIB Data | 221

How to Query Libvirt MIB Data | 223

Supported Chassis MIBs and Traps | 225

Supported libvirt MIB Traps | 226

Monitor NFX150 Device Health | 228

Recovering the Root Password for NFX150, NFX250 NextGen, and NFX350 Devices | 240

Troubleshooting Interfaces on NFX Devices | 244

Monitoring Interface Status and Traffic on NFX Series Devices | 244

12 Configuration Statements and Operational Commands

Junos CLI Reference Overview | 250

About This Guide

Use this guide to perform initial provisioning, configure Junos OS features, chain multiple virtualized network functions, monitor, and manage the NFX150 Series devices.

1

CHAPTER

Overview

NFX150 System Overview | 2

NFX150 Feature Overview | 7

Junos OS Releases Supported on NFX Series Hardware | 23

NFX Product Compatibility | 25

NFX150 System Overview

IN THIS SECTION

- Overview | 2
- NFX150 Models | 3
- Benefits and Uses of NFX150 | 6

Overview

The Juniper Networks NFX150 Network Services Platform is a secure, automated, software-driven customer premises equipment (CPE) platform that delivers virtualized network and security services on demand. NFX150 is part of the Juniper Cloud CPE solution, which leverages Network Functions Virtualization (NFV). It enables service providers to deploy and chain multiple, secure, high-performance virtualized network functions (VNFs) on a single device.

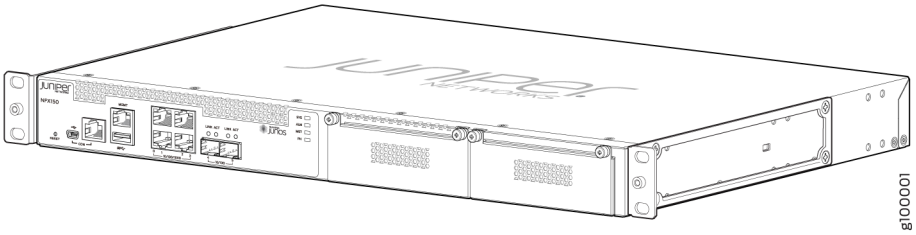
The NFX150 architecture integrates routing, switching, and security functions on a single platform that optimizes the usage of system resources. The architecture enables unified management of all the components through a single CLI. Key components in the NFX150 software include the Junos Control Plane (JCP), Juniper Device Manager (JDM), Layer 2 dataplane, Layer 3 dataplane, and Virtual Network Functions (VNFs).

The NFX150 is suited for small to medium-sized enterprises. With key security features and NFV, the NFX150 can be used in secure SD-WAN and secure router deployments.

The NFX150 portfolio consists of a compact desktop model and a rack-mount model. Both the models are available with or without LTE support. Figure 1 on page 3 shows the compact model without LTE support and Figure 2 on page 3 shows the rack-mount model.

Figure 1: NFX150 Compact Model (Without LTE Support)

Figure 2: NFX150 Rack-Mount Model



For details on the various models, see "NFX150 Models" on page 3.

NFX150 Models

The NFX150 is available in seven models. Table 1 on page 3 lists the NFX150-C models and Table 2 on page 5 lists the NFX150-S1 models.

Table 1: NFX150-C (Compact Models)

	NFX150-C-S1	NFX150-C-S1-AE	NFX150-C-S1-AA	NFX150-C-S1E-AE	NFX150-C-S1E-AA
CPU	2.2-GHz 4-core Intel CPU	2.2-GHz 4-core Intel CPU	2.2-GHz 4-core Intel CPU	2.2-GHz 4-core Intel CPU	2.2-GHz 4-core Intel CPU
RAM	8 GB	8 GB	8 GB	16 GB	16 GB

Table 1: NFX150-C (Compact Models) ***(Continued)***

	NFX150-C-S1	NFX150-C-S1-AE	NFX150-C-S1-AA	NFX150-C-S1E-AE	NFX150-C-S1E-AA
Storage	100 GB SSD	100 GB SSD	100 GB SSD	100 GB SSD	100 GB SSD
Form Factor	Desktop	Desktop	Desktop	Desktop	Desktop
Ports	Four 10/100/1000BASE-T RJ-45 ports which can be used as either access ports or uplinks	Four 10/100/1000BASE-T RJ-45 ports which can be used as either access ports or uplinks	Four 10/100/1000BASE-T RJ-45 ports which can be used as either access ports or uplinks	Four 10/100/1000BASE-T RJ-45 ports which can be used as either access ports or uplinks	Four 10/100/1000BASE-T RJ-45 ports which can be used as either access ports or uplinks
	Two 1-Gigabit Ethernet /10-Gigabit Ethernet SFP+ ports	Two 1-Gigabit Ethernet /10-Gigabit Ethernet SFP+ ports	Two 1-Gigabit Ethernet /10-Gigabit Ethernet SFP+ ports	Two 1-Gigabit Ethernet /10-Gigabit Ethernet SFP+ ports	Two 1-Gigabit Ethernet /10-Gigabit Ethernet SFP+ ports
	One 10/100/1000BASE-T RJ-45 management port	One 10/100/1000BASE-T RJ-45 management port	One 10/100/1000BASE-T RJ-45 management port	One 10/100/1000BASE-T RJ-45 management port	One 10/100/1000BASE-T RJ-45 management port
	Console ports (RJ-45 and mini-USB)	Console ports (RJ-45 and mini-USB)	Console ports (RJ-45 and mini-USB)	Console ports (RJ-45 and mini-USB)	Console ports (RJ-45 and mini-USB)
	One USB 3.0 port	One USB 3.0 port	One USB 3.0 port	One USB 3.0 port	One USB 3.0 port
Expansion module support	No	No	No	No	No

Table 1: NFX150-C (Compact Models) ***(Continued)***

	NFX150-C-S1	NFX150-C-S1-AE	NFX150-C-S1-AA	NFX150-C-S1E-AE	NFX150-C-S1E-AA
LTE support	No	Yes (integrated LTE modem for Europe and North America)	Yes (integrated LTE modem for Asia, Australia, and New Zealand)	Yes (integrated LTE modem for Europe and North America)	Yes (integrated LTE modem for Asia, Australia, and New Zealand)

Table 2: NFX150-S1 Models

	NFX150-S1	NFX150-S1E
CPU	2.2 GHz 8-core Intel CPU	2.2 GHz 8-core Intel CPU
RAM	16 GB	32 GB
Storage	200 GB SSD	200 GB SSD
Form Factor	1 RU	1 RU
Ports	Four 10/100/ 1000BASE-T RJ-45 ports which can be used as either access ports or uplinks	Four 10/100/ 1000BASE-T RJ-45 ports which can be used as either access ports or uplinks
	Two 1-Gigabit Ethernet/10-Gigabit Ethernet SFP+ ports	Two 1-Gigabit Ethernet/10-Gigabit Ethernet SFP+ ports
	One 10/100/ 1000BASE-T RJ-45 management port	One 10/100/ 1000BASE-T RJ-45 management port
	Console ports (RJ-45 and mini-USB)	Console ports (RJ-45 and mini-USB)
	One USB 3.0 port	One USB 3.0 port

Table 2: NFX150-S1 Models *(Continued)*

	NFX150-S1	NFX150-S1E
LTE support	Yes (LTE expansion module)	Yes (LTE expansion module)
Expansion module support	Yes	Yes
Supported expansion modules **NOTE**: You can install only one expansion module on the NFX150-S1 devices. The expansion module must be installed in the first slot, which is next to the chassis LEDs.	• NFX-EM-6T2SFP— Expansion module with six 1-Gigabit Ethernet RJ-45 ports and two 1-Gigabit Ethernet SFP ports • NFX-LTE-AE—Expansion module with a LTE modem supporting the frequency bands in Europe and North America. • NFX-LTE-AA—Expansion module with a LTE modem supporting the frequency bands in Asia, Australia, and New Zealand.	• NFX-EM-6T2SFP— Expansion module with six 1-Gigabit Ethernet RJ-45 ports and two 1-Gigabit Ethernet SFP ports • NFX-LTE-AE—Expansion module with a LTE modem supporting the frequency bands in Europe and North America. • NFX-LTE-AA—Expansion module with a LTE modem supporting the frequency bands in Asia, Australia, and New Zealand.

For more information on the NFX150 models and the expansion modules, see the NFX150 Hardware Guide.

Benefits and Uses of NFX150

The NFX150 Network Services Platform provides these benefits:

- Highly scalable, supporting multiple Juniper 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.
- Supports a variety of flexible deployments. A distributed services deployment model ensures high availability, performance, and compliance. The NFX150 provides an open framework that supports industry standards, protocols, and seamless API integration.

- In addition to Ethernet connections, Wireless WAN support through the LTE module provides more flexibility in deployments.
- Supports advanced security features such as IPsec connectivity, applications detection, and filtering for malicious traffic.
- The 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.
- Support of Open vSwitch (OVS) integrated with Data Plane Development Kit (DPDK) on NFX150 device offers better network packet throughput and lower latencies.

RELATED DOCUMENTATION

NFX150 Feature Overview | 7

NFX150 Feature Overview

IN THIS SECTION

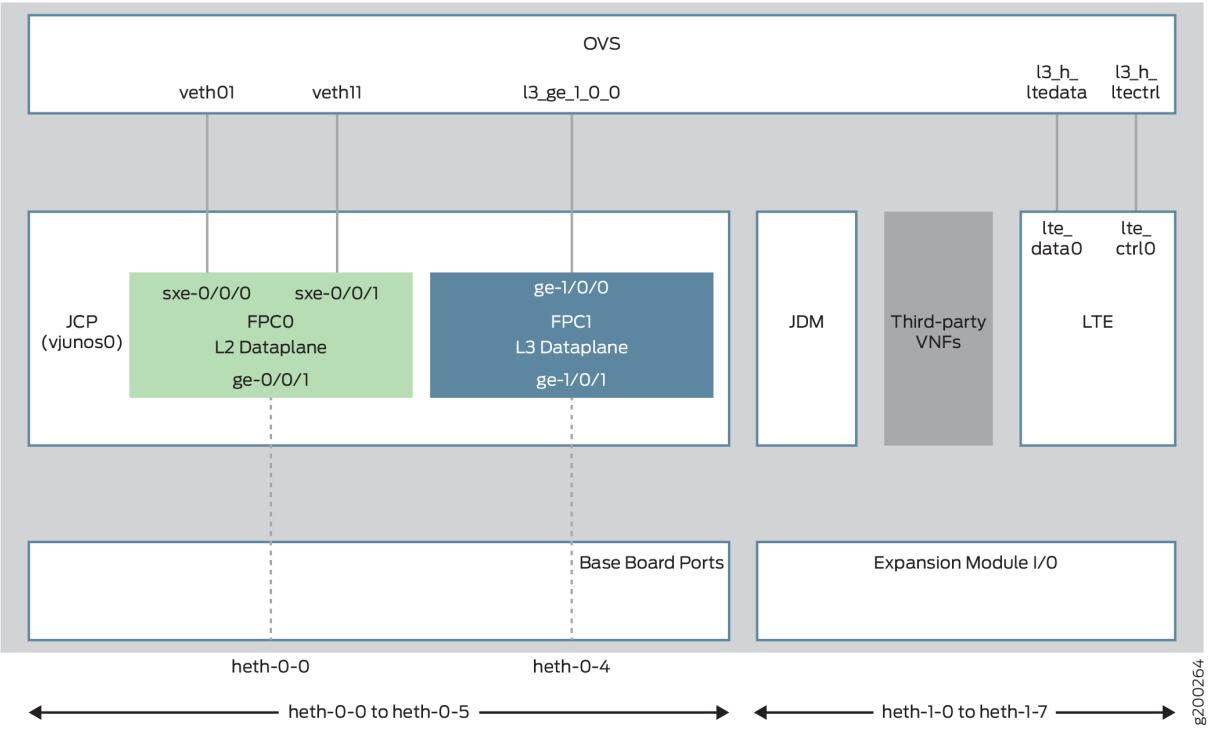
- Software Architecture | 7
- Interfaces | 9
- Supported Features | 15
- Performance Modes | 16
- Licensing | 22

Software Architecture

The software architecture for the NFX150 is designed to provide a unified control plane that functions as a single management point.

Figure 3 on page 8 illustrates the architecture of the NFX150.

Figure 3: NFX150 Architecture



Key components of the system software include:

- VNF—A VNF is a consolidated offering that contains all the components required for supporting a fully virtualized networking environment. You can configure and use third-party VNFs in service chains.

- Junos Control Plane (JCP)—The JCP is the Junos VM running on the host OS, Wind River Linux. The JCP functions as the single point of management for all the components. The JCP controls the Layer 2 dataplane, which provide the Layer 2 services and the Layer 3 dataplane, which provides the Layer 3 to Layer 7 services.

  In addition to chassis management, JCP enables:

  - Configuration of advanced security features.
  - Management of guest virtualized network functions (VNFs) during their life cycle.
  - Installation of third-party VNFs.
  - Creation of VNF service chains.
  - Management of guest VNF images (their binary files).
  - Management of the system inventory and resource usage.

  - Management of the LTE interface.

- Juniper Device Manager (JDM)—An application container that manages VNFs and provides infrastructure services. The JDM functions in the background and users cannot access JDM directly.
- L2 Dataplane—The Layer 2 dataplane that manages the Layer 2 traffic. The Layer 2 dataplane forwards the LAN traffic to the NFV backplane, Open vSwitch (OVS). The Layer 2 dataplane is mapped to the virtual FPC0 on the JCP. By default, all the 1-Gigabit Ethernet physical ports are mapped to the virtual interfaces on the Layer 2 dataplane.
- L3 Dataplane—The Layer 3 dataplane that provides datapath functions for the Layer 3 to Layer 7 services. The Layer 3 dataplane is mapped to the virtual FPC1 on the JCP. By default, the two SFP+ ports on the NFX150 chassis are mapped to the virtual interfaces on the Layer 3 dataplane.
- Linux—The host OS, WindRiver Linux. In Junos OS Release 18.1R1, the WindRiver Linux version is 8.
- Open vSwitch (OVS) bridge—The OVS bridge is a VLAN-aware system bridge, which acts as the NFV backplane to which the VNFs and FPCs connect. Additionally, you can create custom OVS bridges to isolate connectivity between different VNFs.
- LTE—A containerized driver that provides 4G LTE connectivity management. The LTE container is bound to the FPC1 for management.

Interfaces

IN THIS SECTION

- Physical Interfaces | **10**
- Virtual Interfaces | **10**
- LTE Interface | **11**
- Interface Mapping | **11**

The interfaces on the NFX150 devices comprise of physical interfaces, virtual interfaces, and the LTE interface.

Physical Interfaces

The physical interfaces represent the physical ports on the NFX150 chassis and expansion module. The physical interfaces comprise of network and management ports:

- Network ports—Four 1-Gigabit Ethernet ports and two 10-Gigabit Ethernet SFP+ ports function as network ports on the NFX150 chassis. The expansion modules consists of six 1-Gigabit Ethernet ports and two 1-Gigabit Ethernet SFP ports.

  The network ports follow the naming convention **heth-*slot number-port number***, where:

  - heth denotes host Ethernet
  - *slot number* is 0 for the chassis ports and 1 for the expansion module ports. The ports on the chassis are named as heth-0-x and the ports on the expansion module are named heth-1-x.
  - *port number* is the number of the port on the chassis or expansion module

  Each physical port has four virtual functions (VFs) enabled by default.

  > **NOTE**: You cannot map a VF from a port which is mapped to the Layer 2 dataplane.

- Management port—The NFX150 device has a dedicated management port labeled **MGMT** (fxp0), which functions as the out-of-band management interface. The fxp0 interface is assigned an IP address in the 192.168.1.1/24 network.

Virtual Interfaces

The virtual FPCs running within the JCP, contain the virtual interfaces. The virtual interfaces on the NFX150 devices are categorized as follows:

- Virtual Layer 2 interfaces (FPC0)—Denoted as ge-0/0/x, where the value of x ranges from:

  - 0 to 3 for NFX150 devices without an expansion module
  - 0 to 11 for NFX150 devices with an expansion module

  These interfaces are used to configure the following Ethernet switching features:

  - Layer 2 switching of traffic, including support for both trunk and access ports
  - Link Layer Discovery Protocol (LLDP)
  - IGMP snooping
  - Port Security features (MAC limiting, Persistent MAC learning)

- MVRP
- Ethernet OAM, CFM, and LFM

All the 1-Gigabit Ethernet physical ports (heth ports) are mapped to FPC0, by default.

- Virtual Layer 3 interfaces (FPC1)—Denoted as ge-1/0/x, where value of x ranges from 0 to 9. These interfaces are used to configure Layer 3 features such as routing protocols and QoS.

  In an NFX150 device, you can configure any of the ge-1/0/x interfaces as in-band management interfaces. In in-band management, you configure a network interface as a management interface and connect it to the management device. You can configure any number of interfaces for in-band management by assigning an IPv4 or IPv6 address to each of the ports, and an in-band management VLAN.

  > **NOTE**: The NFX150 devices do not support integrated routing and bridging (IRB) interfaces. The IRB functionality is provided by ge-1/0/0, which is always mapped to the service chaining backplane (OVS). Note that this mapping cannot be changed.

- Virtual SXE Interfaces—Two static interfaces, sxe-0/0/0 and sxe-0/0/1, connect the FPC0 (Layer 2 dataplane) to the OVS backplane.

LTE Interface

The NFX150 device models with LTE support can be configured for wireless WAN connectivity over 3G or 4G networks. The LTE physical interface uses the name cl-1/1/0. The dialer interface, dl0, is a logical interface, which is used to trigger calls.

Interface Mapping

Table 3 on page 12 summarizes the interfaces on the NFX150.

Table 3: Interfaces on the NFX150

Interface Name	Description
heth-0-0 to heth-0-5	Physical ports on the front panel of the NFX150 device, which can be mapped to Layer 2 or Layer 3 interfaces, or VNFs. Ports heth-0-0 to heth-0-3 are 10 Mbps/100 Mbps/1 Gbps tri-speed copper ports. Ports heth-0-4 and heth-0-5 are 10 Gbps SFP+ ports **For Junos OS Releases 18.1, 18.2 R1, and 18.3 R1:** • Ports heth-0-0 to heth-0-3 are mapped to the LAN ports ge-0/0/0 to ge-0/0/3, respectively. • Ports heth-0-4 and heth-0-5 are mapped to the WAN ports ge-1/0/1 and ge-1/0/2, respectively. **For Junos OS Release 18.2 R2** • Ports heth-0-0, heth-0-1, and heth-0-2 are mapped to the LAN ports ge-0/0/0 to ge-0/0/2, respectively. • Port heth-0-4 is mapped to the LAN port ge-0/0/3. Ports heth-0-3 and heth-0-5 are mapped to the WAN ports ge-1/0/1 and ge-1/0/2, respectively.
heth-1-0 to heth-1-7	Physical ports on the expansion module of the NFX150-S1 device. These ports are mapped to the ge-0/0/n ports by default. Ports heth-1-0 to heth-1-5 are 10 Mbps/100 Mbps/1 Gbps tri-speed copper ports mapped to the LAN ports ge-0/0/4 to ge-0/0/9, respectively. Ports heth-1-6 and heth-1-7 are 1 Gbps SFP ports mapped to the LAN ports ge-0/0/10 and ge-0/0/11 respectively.
ge-0/0/x	Logical Layer 2 interfaces, which can be used for LAN connectivity. The values of x ranges from: • 0 to 3 for NFX150 devices without an expansion module • 0 to 11 for NFX150 devices with an expansion module

Table 3: Interfaces on the NFX150 *(Continued)*

Interface Name	Description
ge-1/0/x	A set of up to 10 logical Layer 3 interfaces. Each of these interfaces can have 4k sub-interfaces. The value of x ranges from 0 to 9.
cl-1/1/0	The LTE cellular interface, which carries the physical layer attributes.
dl0	The LTE dialer interface, which carries Layer 3 and security services. The security flow session contains the dl0 interface as the ingress or egress interface.
st0	Secure tunnel interface used for IPsec VPNs.
fxp0	The out-of-band management interface.

The list of supported transceivers for the NFX150 is located at https://pathfinder.juniper.net/hct/product/.

Table 5 on page 14 illustrates the default mapping between the physical and virtual interfaces on a NFX150 device.

Table 4: Default Mapping of Physical Ports to Virtual Ports on NFX150 (for Junos OS Releases 18.1, 18.2 R1, and 18.3 R1)

Physical Port	Virtual Interface (Layer 2 dataplane)	Virtual Interface (Layer 3 dataplane)
heth-0-0	ge-0/0/0	NA
heth-0-1	ge-0/0/1	NA
heth-0-2	ge-0/0/2	NA
heth-0-3	ge-0/0/3	NA
heth-0-4	NA	ge-1/0/1

Table 4: Default Mapping of Physical Ports to Virtual Ports on NFX150 (for Junos OS Releases 18.1, 18.2 R1, and 18.3 R1) ***(Continued)***

Physical Port	Virtual Interface (Layer 2 dataplane)	Virtual Interface (Layer 3 dataplane)
heth-0-5	NA	ge-1/0/2

Table 5: Default Mapping of Physical Ports to Virtual Ports on NFX150 (for Junos OS Releases 18.2 R2)

Physical Port	Virtual Interface (Layer 2 dataplane)	Virtual Interface (Layer 3 dataplane)
heth-0-0	ge-0/0/0	NA
heth-0-1	ge-0/0/1	NA
heth-0-2	ge-0/0/2	NA
heth-0-3	NA	ge-1/0/1
heth-0-4	ge-0/0/3	NA
heth-0-5	NA	ge-1/0/2

Table 6 on page 14 illustrates the default mapping between the physical ports on the expansion module and the virtual interfaces.

Table 6: Default Mapping of Physical Ports to Virtual Ports for the Expansion Module

Physical Port	Virtual Port (Layer 2 dataplane)
heth-1-0	ge-0/0/4
heth-1-1	ge-0/0/5
heth-1-2	ge-0/0/6

Table 6: Default Mapping of Physical Ports to Virtual Ports for the Expansion Module *(Continued)*

Physical Port	Virtual Port (Layer 2 dataplane)
heth-1-3	ge-0/0/7
heth-1-4	ge-0/0/8
heth-1-5	ge-0/0/9
heth-1-6	ge-0/0/10
heth-1-7	ge-0/0/11

NOTE: The expansion module ports are mapped to the Layer 2 dataplane interfaces by default. You can change the mapping to suit your requirement. Any of the ports on the chassis and expansion module can be mapped to the ge-1/0/x or ge-0/0/x interfaces. Any change in port mapping configuration will automatically reset the affected FPC.

Supported Features

Table 7 on page 16 lists the Junos features supported on NFX150.

Table 7: Features Supported on NFX150

Junos OS Release	Routing	Security	Switching
18.1R1	• BGP, OSPF, RIP, IS-IS, • MVRP	• NAT • ALG • IPSec • IPv6 NTP • IPv6 TACACS • CoS • Firewall filters	• LLDP • Port mirroring • IGMP/MLD snooping • MLD snooping • Persistent MAC learning • L2Rewrite • Native VLAN
18.2 R1		• Application Security • IDP • Integrated User Firewall • UTM	

For more details on supported features, see Feature Explorer.

Performance Modes

IN THIS SECTION

- How to Define a Custom Mode Template | 18
- Core to CPU Mapping on NFX150 | 21

NFX150 devices provide the following operational modes:

- Throughput mode—Provides maximum resources (CPU and memory) for Junos software and remaining resources, if any, for third-party VNFs.

> **NOTE**: You cannot create VNFs in throughput mode.
> Starting in Junos OS Release 21.1R1, mapping OVS to Layer 3 data plane interface is not supported in throughput mode on NFX150-S1 and NFX150-S1E 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 the system components:
  - Layer 2 data plane, Layer 3 data plane, and NFV backplane for NFX150-S1 and NFX150-S1E models
  - Layer 2 data plane and Layer 3 data plane for NFX150-C-S1, NFX150-C-S1-AE/AA, and NFX150-C-S1E-AE/AA models

> **NOTE**: Compute, hybrid, and throughput modes are supported in Junos OS Release 19.1R1 or later. Custom mode is supported starting in Junos OS Release 22.1R1.
> 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 hybrid and compute modes, you can map Layer 3 data plane interfaces to either SR-IOV or OVS. In throughput mode, you can map Layer 3 data plane interfaces to only SR-IOV.

For example:

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

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

- Map Layer 3 data plane interfaces to 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 supports a maximum of 10 interfaces (eth0 through eth9), including the two management interfaces eth0 and eth1.

> **NOTE**: You cannot attach a single VNF interface to both SR-IOV and OVS. However, you can attach different interfaces from the same VNF to SR-IOV and OVS.
>
> When the mapping to a particular Layer 3 data plane interface changes between SR-IOV NICs (for example, heth-0-0) or from heth-x-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 data plane is not mapped to SR-IOV, then switching from hybrid or compute mode to throughput mode results in an error.

If you pin a virtual CPU to physical CPUs for a VNF, then ensure that the physical CPUs do not overlap with the CPUs being used for Juniper system components, including the physical CPU 0.

Physical CPUs used to pin an emulator can overlap with the CPUs being used for Juniper system components, except the physical CPU 0. This overlap can impact the performance of one or more Juniper system components and VNFs.

How to Define a Custom Mode Template

You can use a custom mode template if you need to allocate maximum resources to third-party VNFs. In custom mode, you must configure both CPU count and amount of memory for:

- Layer 2 data plane, Layer 3 data plane, and NFV backplane for NFX150-S1 and NFX150-S1E models
- Layer 2 data plane and Layer 3 data plane for NFX150-C-S1, NFX150-C-S1-AE/AA, NFX150-C-S1E-AE/AA models

The absence of any of the configuration causes a commit failure.

NOTE: You can opt to disable the Layer 2 data plane to free up CPU and memory resources in deployments that do not require Layer 2 software PFE services.

```
user@host# set vmhost mode custom custom-mode-name layer-2-infrastructure offline
```

If you disable the the Layer 2 data plane, you cannot configure the virtual interface mappings of the Layer 2 dataplane. For example:

```
set vmhost virtualization-options interfaces ge-0/0/0 mapping interface heth-0-0
```

Before you configure custom mode, note the following:

- If you disable the Layer 2 dataplane, then you cannot configure `cpu count` and `memory size` for the Layer 2 dataplane.

  If you do not disable the Layer 2 dataplane, then you must configure the `cpu count` and `memory size` for it. The CPU count and memory must not exceed the total CPU count and memory available on the system.

- 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).

To define a custom mode template on NFX150-C-S1, NFX150-C-S1-AE/AA, NFX150-C-S1E-AE/AA models, use the following configuration. Configuring `cpu colocation quota` is optional.

```
user@host# set vmhost mode custom custom-mode-name layer-2-infrastructure cpu count count
user@host# set vmhost mode custom custom-mode-name layer-2-infrastructure memory size memG
user@host# set vmhost mode custom custom-mode-name layer-3-infrastructure cpu count count
user@host# set vmhost mode custom custom-mode-name layer-3-infrastructure memory size memG
```

To define a custom mode template on NFX150-S1 and NFX150-S1E models, use the following configuration. Configuring `cpu colocation quota` is optional.

```
user@host# set vmhost mode custom custom-mode-name layer-2-infrastructure cpu count count
user@host# set vmhost mode custom custom-mode-name layer-2-infrastructure memory size memG
user@host# set vmhost mode custom custom-mode-name layer-3-infrastructure cpu count count
user@host# set vmhost mode custom custom-mode-name layer-3-infrastructure memory size memG
user@host# set vmhost mode custom custom-mode-name nfv-back-plane cpu count count
user@host# set vmhost mode custom custom-mode-name nfv-back-plane memory size memG
```

The memory specified through a custom mode is created and backed by 1G huge pages for NFV backplane and Layer 2 data plane usage and 2M huge pages for Layer 3 data plane usage.

The *flex* template is the custom mode template that is present in the default Junos configuration. This template supports a keyword MIN, which is a device-specific pre-defined value for allocating minimal resources. The flex template uses the MIN keyword for allocating resources to system components such as Layer 3 data plane and NFV backplane. In this mode, the device provides maximum memory and CPUs to third-party VNFs.

To allocate resources in flex mode, use the following commands:

- For NFX150-C-S1, NFX150-C-S1-AE/AA, NFX150-C-S1E-AE/AA models:

  ```
  set vmhost mode custom flex layer-2-infrastructure cpu count MIN
  set vmhost mode custom flex layer-2-infrastructure memory size MIN
  set vmhost mode custom flex layer-3-infrastructure cpu count MIN
  set vmhost mode custom flex layer-3-infrastructure memory size MIN
  ```

- For NFX150-S1/S1E models:

  ```
  set vmhost mode custom flex layer-2-infrastructure cpu count MIN
  set vmhost mode custom flex layer-2-infrastructure memory size MIN
  set vmhost mode custom flex layer-3-infrastructure cpu count MIN
  ```

```
set vmhost mode custom flex layer-3-infrastructure memory size MIN
set vmhost mode custom flex nfv-back-plane cpu count MIN
set vmhost mode custom flex nfv-back-plane memory size MIN
```

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. The configuration of Layer 2 virtual interfaces, Layer 3 virtual interfaces, VNF virtual CPU to physical CPU mapping, VNF emulator to physical CPU mapping, and VNF memory size is validated during commit check against the currently active custom mode's configuration parameters and the modified custom mode's configuration parameters that take effect after a reboot.

When the device is in custom mode only basic firewall features are supported. In flex mode, you can configure a maximum of:

- 8 IPSec VPN tunnels
- 16 IFL
- 4 IFD

Core to CPU Mapping on NFX150

The following tables list the CPU to core mappings for the NFX150 models:

NFX150-S1 and NFX150-S1E								
Core	0	1	2	3	4	5	6	7
CPU	0	1	2	3	4	5	6	7

NFX150-C-S1				
Core	0	1	2	3
CPU	0	1	2	3

Licensing

For features or scaling levels that require a license, you must install and properly configure the license to meet the requirements for using the licensable feature or scale level. The device enables you to commit a configuration that specifies a licensable feature or scale without a license for a 30-day grace period. The grace period is a short-term grant that enables you to start using features in the pack or scale up to the system limits (regardless of the license key limit) without a license key installed. The grace period begins when the licensable feature or scaling level is actually used by the device (not when it is first committed). In other words, you can commit licensable features or scaling limits to the device configuration, but the grace period does not begin until the device uses the licensable feature or exceeds a licensable scaling level.

For information about how to purchase software licenses, contact your Juniper Networks sales representative. Junos OS software implements an honor-based licensing structure and provides you with a 30-day grace period to use the feature without a license key installed. The grace period begins when you configure the feature and your device uses the licensed feature for the first time, but not necessarily when you install the license. After the grace period expires, the system generates system log messages saying that the feature requires a license. To clear the error message and use the licensed feature properly, you must install and verify the required license.

NOTE: Configurations might include both licensed and nonlicensed features. For these situations, the license is enforced up to the point where the license can be clearly distinguished. For example, an authentication-order configuration is shared by both Authentication, Authorization, and Accounting (AAA), which is licensed, and by Layer 2 Tunneling Protocol (L2TP), which is not licensed. When the configuration is committed, the device does not issue any license warnings, because it is not yet known whether AAA or L2TP is using the configuration. However, at runtime, the device checks for a license when AAA authenticates clients, but does not check when L2TP authenticates clients.

The device reports any license breach as a warning log message whenever a configuration is committed that contains a feature or scale limit usage that requires a license. Following the 30-day grace period, the device periodically reports the breach to syslog messages until a license is installed and properly configured on the device to resolve the breach.

NOTE: Successful commitment of a licensable feature or scaling configuration does not imply that the required licenses are installed or not required. If a required license is not present, the system issues a warning message after it commits the configuration.

Table 8: NFX150 Junos Software Licenses

License	Features	License SKU	Device Model
Base software (STD)	Layer 2 services, Layer 3 services, NAT, IPsec, stateful firewall	NFX150-C-STD	NFX150-C-S1 and NFX150-C-S1E
		NFX150-S-STD	NFX150-S1 and NFX150-S1E
Advanced software (ADV)	Features in the base software plus AppFW, AppID, AppTrack, AppRoute	NFX150-C-ADV	NFX150-C-S1 and NFX150-C-S1E
		NFX150-S-ADV	NFX150-S1 and NFX150-S1E

RELATED DOCUMENTATION

Initial Configuration on NFX150 Devices | 29

Mapping Interfaces on NFX150 Devices | 45

Junos OS Releases Supported on NFX Series Hardware

The Table 9 on page 24 provides details of Junos OS software releases supported on the NFX Series devices.

NOTE: Support for Linux bridge mode on NFX250 devices ended in Junos OS Release 18.4.

NOTE: Support for nfx-2 software architecture on NFX250 devices ended in Junos OS Release 19.1R1.

Table 9: 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
NFX250	15.1X53-D45, 15.1X53-D47, 15.1X53-D470, and 15.1X53-D471	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
	17.2R1 through 19.1R1		
	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

RELATED DOCUMENTATION

NFX250 Overview

NFX Product Compatibility

IN THIS SECTION

- Hardware Compatibility | 25
- Software Version Compatibility | 25

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 10 on page 26.
- The Linux Bridge mode is supported only up to Junos OS Release 18.4 on NFX250 devices.

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 10: 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
17.3R1	15.1X49-D78.3	Not applicable

Table 10: Software Compatibility Details with vSRX Virtual Firewall and Cloud CPE Solution *(Continued)*

NFX250 Junos OS Release	vSRX Virtual Firewall	Cloud CPE Solution
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

Initial Configuration on NFX150 Devices | 29

Zero Touch Provisioning on NFX Series Devices | 34

Initial Configuration on NFX150 Devices

IN THIS SECTION

- Factory-Default Settings | 29
- Enabling Basic Connectivity | 31
- Establishing the Connection | 33

Factory-Default Settings

The NFX150 device is shipped with the following factory-default settings:

Table 11: Security Policies

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

Table 12: Interface Mapping (for Junos OS Releases 18.1, 18.2 R1, and 18.3 R1)

Port Label	Interface	Virtual Interface	Security Zone	DHCP State	IP Address
0/0 to 0/3	heth-0-0 to heth-0-3	ge-0/0/0 to ge-0/0/3	trust	Server	192.168.2.1/24
0/4	heth-0-4	ge-1/0/1	untrust	Client	ISP assigned
0/5	heth-0-5	ge-1/0/2	untrust	Client	ISP assigned

Table 12: Interface Mapping (for Junos OS Releases 18.1, 18.2 R1, and 18.3 R1) *(Continued)*

Port Label	Interface	Virtual Interface	Security Zone	DHCP State	IP Address
MGMT	fxp0	N/A	N/A	N/A	192.168.1.1/24

Table 13: Interface Mapping (for Junos OS Releases 18.2 R2 and 18.4R1)

Port Label	Interface	Virtual Interface	Security Zone	DHCP State	IP Address
0/0 to 0/2	heth-0-0 to heth-0-2	ge-0/0/0 to ge-0/0/2	trust	Server	192.168.2.1/24
0/3	heth-0-3	ge-1/0/1	untrust	Client	ISP assigned
0/4	heth-0-4	ge-0/0/3	trust	Server	192.168.2.1/24
0/5	heth-0-5	ge-1/0/2	untrust	Client	ISP assigned
MGMT	fxp0	N/A	N/A	N/A	192.168.1.1/24

Table 14: Interface Mapping (for Junos OS Releases 19.1R1 or later)

Port Label	Interface	Virtual Interface	Security Zone	DHCP State	IP Address
0/0	heth-0-0	ge-1/0/1	untrust	Client	ISP assigned
0/1 to 0/4	heth-0-1 to heth-0-4	ge-0/0/1 to ge-0/0/4	trust	Server	192.168.2.1/24
0/5	heth-0-5	ge-1/0/2	untrust	Client	ISP assigned
MGMT	fxp0	N/A	N/A	N/A	192.168.1.1/24

Table 15: LTE Interfaces

Interface	Security Zone	IP Address
cl-1/1/0	N/A	N/A
dl0 (logical)	untrust	ISP assigned

The NFX150 device is shipped with the following services enabled by default: DHCP, HTTPS, and TFTP.

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

Enabling Basic Connectivity

1. Ensure that the NFX150 device is powered on.
2. Connect to the console port:
   a. Plug one end of the Ethernet cable into the console port on your NFX150 device.
   b. Connect the other end of the Ethernet cable to the RJ-45—to—DB-9 serial port adapter.
   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:

      Baud rate—9600; Parity—N; Data bits—8; Stop bits—1; Flow control—None.

   > **NOTE**: We no longer include a DB-9 to RJ-45 cable or a DB-9 to RJ-45 adapter with a CAT5E copper cable as part of the device package. If you require a console cable, you can order it separately with the part number JNP-CBL-RJ45-DB9 (DB-9 to RJ-45 adapter with a CAT5E copper cable).

   > **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 it 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** and enter the password **juniper123**. If the software completes booting before you connect to the console, you might need to press the **Enter** key for the prompt to appear:

> **NOTE**: Starting with Junos OS Release 18.1R2 or later, the root password is not configured for initial configuration of the NFX150 devices.

```
login: root
password: juniper123
```

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 the Internet connection for devices connected on LAN by setting the DNS IP:

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

10. Commit the configuration:

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

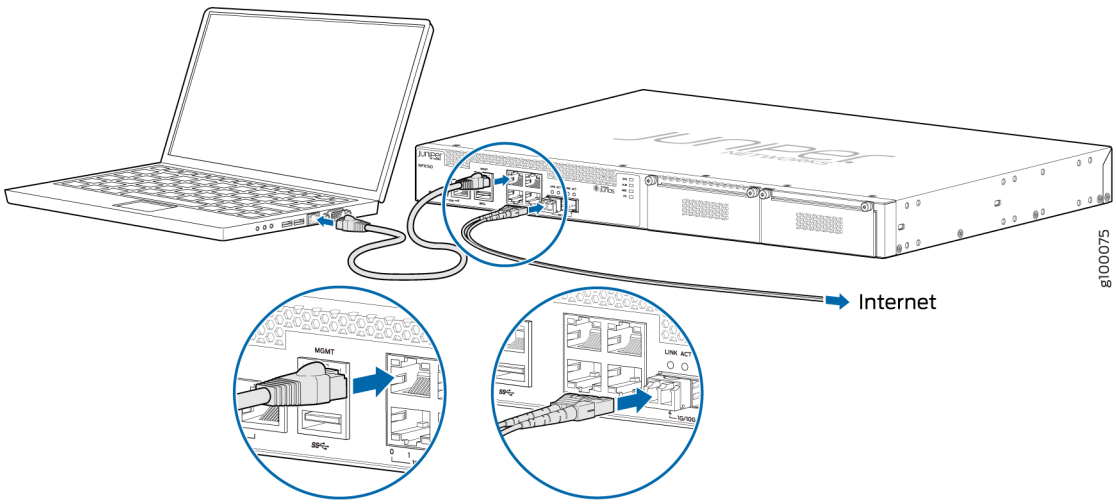
Establishing the Connection

1. Connect the device to the Internet Service Provider (ISP) by using the following step:

> **NOTE**: For information on interface mapping, see "Factory-Default Settings" on page 29 and "Factory-Default Settings" on page 29.

Connect one of the WAN ports to the ISP. The device is assigned an IP address by the ISP through DHCP.

Figure 4: Connecting the Interfaces on an NFX150-S1 Device



Optionally, you can obtain a SIM card from the ISP and connect the device through LTE.

> **NOTE**: The LTE expansion module must be purchased separately.

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

RELATED DOCUMENTATION

Installing and Configuring the LTE Module

Zero Touch Provisioning on NFX Series Devices

IN THIS SECTION

- Understanding Zero Touch Provisioning | 34
- Pre-staging an NFX Series Device | 35
- Provisioning an NFX Series Device | 38
- Provisioning an NFX Series Device Using Sky Enterprise | 39

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.

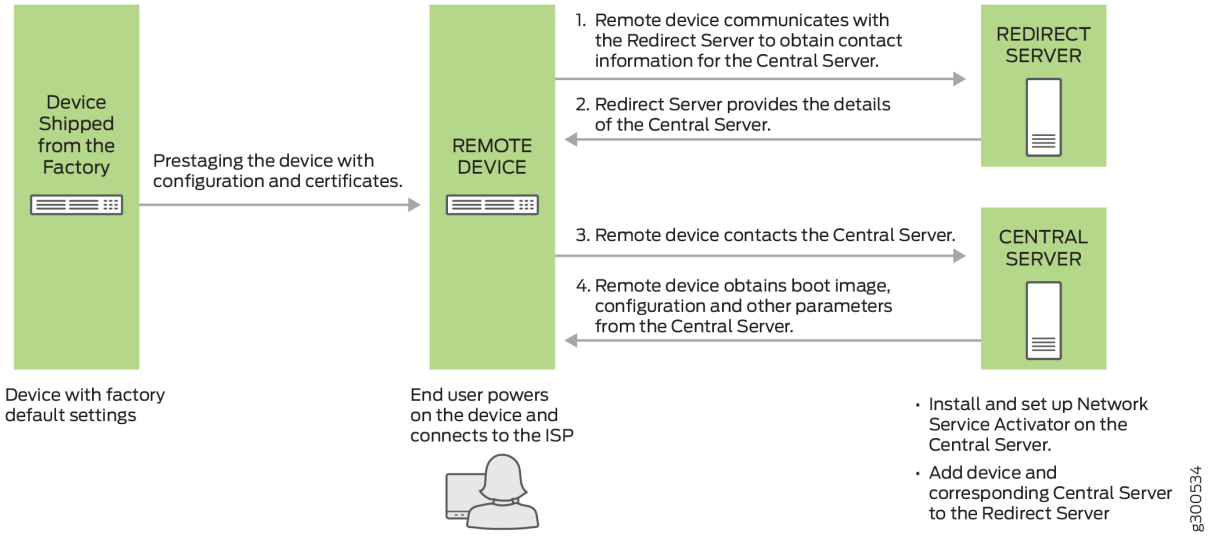
Pre-staging an NFX Series Device

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 5 on page 36 illustrates the workflow of prestaging the NFX Series devices.

Figure 5: Workflow for Prestaging an NFX Series Device



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]
```

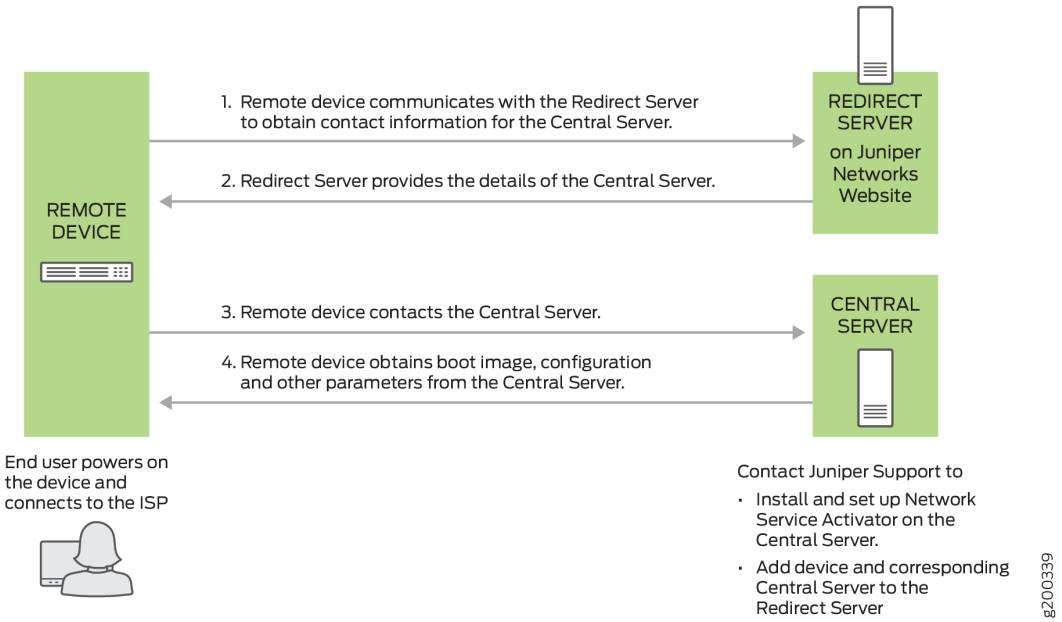
To disable trace operation:

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

Provisioning an NFX Series Device

Figure 6 on page 38 illustrates the workflow of the initial provisioning of NFX Series devices.

Figure 6: Workflow for Initial Provisioning of an NFX Series Device



> **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 6 on page 38 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

YANG files on NFX150 Devices | 41

YANG files on NFX150 Devices

IN THIS SECTION

- Understanding YANG on NFX150 Devices | 41
- Generating YANG Files | 42

Understanding YANG on NFX150 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 NFX150 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

Mapping Interfaces on NFX150 Devices | 45

Configuring the In-Band Management Interface | 47

ADSL2 and ADSL2+ Interfaces on NFX150 Devices | 48

VDSL2 Interfaces on NFX150 Devices | 54

Configuring the LTE Module on NFX Devices | 63

Upgrading the Modem Firmware on NFX Devices Through Over-the-Air (OTA) | 75

Mapping Interfaces on NFX150 Devices

IN THIS SECTION

- Mapping Physical Interfaces to Virtual Interfaces on NFX150 Devices | 45
- Mapping Physical Ports to VNF Interfaces Through SR-IOV | 46
- Mapping Layer 3 Dataplane Interfaces to OVS | 46

Mapping Physical Interfaces to Virtual Interfaces on NFX150 Devices

The NFX150 devices are shipped with the default mapping as described in "NFX150 Feature Overview" on page 7. You can change the mapping to suit your needs. Any of the physical ports can be mapped to the ge-0/0/x or ge-1/0/x interfaces. Any change in port mapping configuration will automatically reset the affected FPC.

> **NOTE**: On NFX150 devices, the link does not come up if 1G SFP port is connected from heth-0-4 and heth-0-5 ports to a peer device. As a workaround, disable the auto-negotiation for the interface connected to the NFX150 device on the remote device.

- To map a physical port to a virtual interface:

```
root# set vmhost virtualization-options interfaces virtual-interface-name mapping interface
physical-interface-name
```

For example, to map the physical port heth-0-2 to the Layer 2 dataplane (FPC0) interface ge-0/0/3, the command is as follows:

```
root# set vmhost virtualization-options interfaces ge-0/0/3 mapping interface heth-0-2
```

For example, to map the physical port heth-0-5 to the Layer 3 dataplane (FPC1) interface ge-1/0/2, the command is as follows

```
root# set vmhost virtualization-options interfaces ge-1/0/2 mapping interface heth-0-5
```

- To map a physical port to two FPC1 interfaces:

```
root# set vmhost virtualization-options interfaces virtual-interface-name1 mapping interface
physical-interface-name
root# set vmhost virtualization-options interfaces virtual-interface-name2 mapping interface
physical-interface-name
```

For example, to map the physical port heth-0-1 to the Layer 3 data plane(FPC1) interfaces ge-1/0/1 and ge-1/0/2:

```
root# set vmhost virtualization-options interfaces ge-1/0/1 mapping interface heth-0-1
root# set vmhost virtualization-options interfaces ge-1/0/2 mapping interface heth-0-1
```

Mapping Physical Ports to VNF Interfaces Through SR-IOV

You can configure Layer 3 services using single-root I/O virtualization (SR-IOV). To map a VNF interface to a physical interface by using the SR-IOV virtual function, use the following command:

```
root# set virtual-network-functions  vnf-name interfaces interface-name mapping interface
physical-interface-name virtual-function
```

Mapping Layer 3 Dataplane Interfaces to OVS

The ge-1/0/0 interface is always mapped to the service chaining backplane (OVS). To map additional Layer 3 dataplane interfaces to the system bridge, use the following command:

```
root# set vmhost virtualization-options interfaces interface-name
```

RELATED DOCUMENTATION

Configuring the In-Band Management Interface | 47

Configuring the In-Band Management Interface

In in-band management, you configure a network interface as a management interface and connect it to the management device. By default. ports ge-1/0/0, ge-1/0/1, and ge-1/0/2 are configured as network interfaces. In addition, you can configure network interfaces from ge-1/0/3 to ge-1/0/9.

You can configure any of the ge-1/0/x ports, where x ranges from 0 to 9, as in-band management interfaces. In-band management can be configured using either a LAN port (FPC0) or a WAN port (FPC1).

To configure a WAN port for in-band management:

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

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

2. Configure the IP address for the in-band management interface:

```
root@host# set interfaces interface-name unit 0 family inet address address/prefix-length
```

> **NOTE**: The ge-1/0/x port selected for configuration must be the same port that is mapped to the physical port (heth) being used for management connectivity.

3. 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
root@host# set interfaces sxe-0/0/[01] unit 0 family ethernet-switching vlan members mgmt-vlan
```

3. Configure VLAN tagging:

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

ADSL2 and ADSL2+ Interfaces on NFX150 Devices

IN THIS SECTION

- ADSL Interface Overview | 48
- Configuring ADSL SFP Interface Using VLANs on NFX150 Network Services Platform | 50
- Configuring ADSL SFP Interface Without Using VLANs on NFX150 Network Services Platform | 52

ADSL Interface Overview

IN THIS SECTION

- ADSL2 and ADSL2+ | 49

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 16 on page 49.

Table 16: 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

The ADSL Mini-PIM facilitates a maximum of 10 virtual circuits on supported security devices.

Supported security devices with Mini-PIMs can use PPP over Ethernet over ATM (PPPoEoA) and PPP over ATM (PPPoA) 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.

Configuring ADSL SFP Interface Using VLANs on NFX150 Network Services Platform

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

To configure ADSL SFP interfaces on NFX150 devices:

1. Connect to the host.

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

2. Configure the WAN side front panel port with vlan-tagging.

```
user@host# set interfaces virtual-interface-name vlan-tagging
```

```
user@host# set interfaces ge-1/0/1 vlan-tagging
```

3. Configure a VLAN for the WAN side front panel port.

```
user@host# set interfaces virtual-interface-name unit 0 vlan-id vlan-id
```

```
user@host# set interfaces ge-1/0/1 unit 0 vlan-id 152
```

4. Configure the WAN side front panel port with an IP address.

```
set interfaces virtual-interface-name unit 0 family inet address ip-address
```

```
user@host# set interfaces ge-1/0/1 unit 0 family inet address 152.1.1.152/24
```

5. Configure the physical (heth) interface with ADSL SFP options.

```
user@host# set vmhost interfaces physical-interface-name dsl-sfp-options adsl-options vpi vpi
vci vci encap encapsulation
```

```
user@host# set vmhost interfaces heth-0-4 dsl-sfp-options adsl-options vpi 8 vci 36 encap
llcsnap-bridged-802.1q
```

NOTE:

- The default value for `encap` is `llcsnap-bridged-802.1q`.
- The ADSL SFP interface is enabled only if you configure `vci` and `vpi`. By default, `vci` and `vpi` are disabled.

6. Map the physical (heth) interfaces to the virtual (ge) interfaces.

```
user@host# set vmhost virtualization-options interfaces virtual-interface-name mapping
interface physical-interface-name
```

```
user@host# set vmhost virtualization-options interfaces ge-1/0/1 mapping interface heth-0-4
```

7. Commit the configuration.

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

To verify the configuration, enter the **show interfaces heth-0-4** command.

```
[edit]
user@host# show interfaces heth-0-4
Physical interface: heth-0-4, Enabled, Physical link is Up
  Link-level type: Ethernet, Media type: Fiber, MTU: 9192, ADSL2P mode, Speed: 1Gbps, Duplex:
Full-duplex, Auto-negotiation: Disabled
  ADSL status:
    Modem status  : Showtime (Adsl2plus)
    DSL mode      :     Auto    Annex A
```

```
Device flags   : Present Running
Current address: 58:00:bb:ac:c8:51, Hardware address: 58:00:bb:ac:c8:51
```

Configuring ADSL SFP Interface Without Using VLANs on NFX150 Network Services Platform

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

To configure ADSL SFP interfaces on NFX150 devices:

1. Connect to the host.

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

2. Configure the WAN side front panel port with an IP address.

```
set interfaces virtual-interface-name unit 0 family inet address ip-address
```

```
user@host# set interfaces ge-1/0/1 unit 0 family inet address 153.1.1.153/24
```

3. Configure the physical (heth) interface with ADSL SFP options.

```
user@host# set vmhost interfaces physical-interface-name dsl-sfp-options adsl-options vpi vpi
vci vci encap encapsulation
```

```
user@host# set vmhost interfaces heth-0-4 dsl-sfp-options adsl-options vpi 8 vci 36 encap
llcsnap-bridged-802.1q
```

NOTE:

- The default value for `encap` is `llcsnap-bridged-802.1q`.
- The ADSL SFP interface is enabled only if you configure `vci` and `vpi`. By default, `vci` and `vpi` are disabled.

4. Map the physical (heth) interfaces to the virtual (ge) interfaces.

```
user@host# set vmhost virtualization-options interfaces virtual-interface-name mapping
interface physical-interface-name
```

```
user@host# set vmhost virtualization-options interfaces ge-1/0/1 mapping interface heth-0-4
```

5. Commit the configuration.

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

To verify the configuration, enter the **show interfaces heth-0-4** command.

```
[edit]
user@host# show interfaces heth-0-4
Physical interface: heth-0-4, Enabled, Physical link is Up
  Link-level type: Ethernet, Media type: Fiber, MTU: 9192, ADSL2P mode, Speed: 1Gbps, Duplex:
Full-duplex, Auto-negotiation: Disabled
  ADSL status:
    Modem status  : Showtime (Adsl2plus)
    DSL mode      :     Auto    Annex A
  Device flags   : Present Running
  Current address: 58:00:bb:ac:c8:51, Hardware address: 58:00:bb:ac:c8:51
```

VDSL2 Interfaces on NFX150 Devices

IN THIS SECTION

- VDSL Interface Overview | 54
- VDSL2 Network Deployment Topology | 55
- VDSL2 Interface Support on NFX Series Devices | 57
- Configuring VDSL SFP Interface Using VLANs on NFX150 Network Services Platform | 59
- Configuring VDSL SFP Interface Without Using VLANs on NFX150 Network Services Platform | 61

VDSL Interface Overview

IN THIS SECTION

- VDSL2 Vectoring Overview | 55

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 7 on page 56.
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 8 on page 56.

> **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 7 on page 56 shows a typical VDSL2 network topology.

Figure 7: Typical VDSL2 End-to-End Connectivity and Topology Diagram

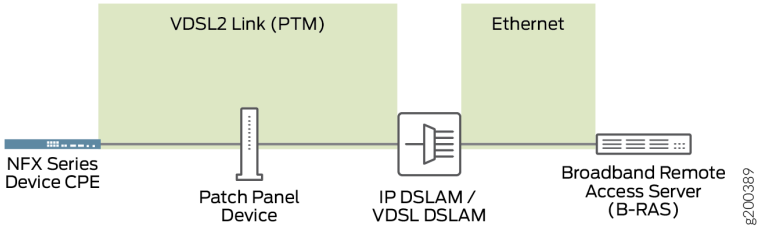
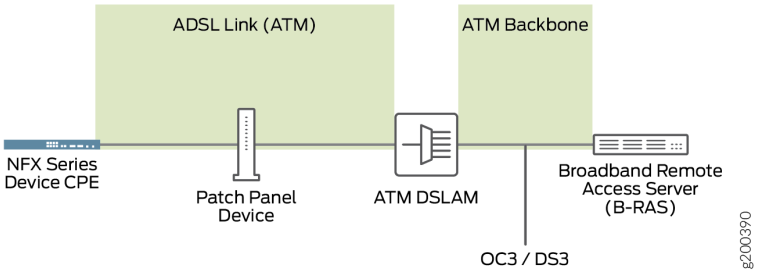


Figure 8 on page 56 shows a backward-compatible ADSL topology using ATM DSLAM.

Figure 8: Backward-Compatible ADSL Topology (ATM DSLAM)



VDSL2 Interface Support on NFX Series Devices

IN THIS SECTION

- VDSL2 Interface Compatibility with ADSL Interfaces | 58
- VDSL2 Interfaces Supported Profiles | 58
- VDSL2 Interfaces Supported Features | 59

The VDSL2 interface is supported on the NFX Series devices listed in Table 17 on page 57. (Platform support depends on the Junos OS release in your installation.)

Table 17: VDSL2 Annex A and Annex B Features

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

NOTE: Only one JNP-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 heth-0-4 and heth-0-5.

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 18 on page 58 lists the different profiles supported on the VDSL2 interfaces and their properties.

Table 18: Supported Profiles on the VDSL2 Interfaces

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

VDSL2 Interfaces Supported Features

The following features are supported on the VDSL2 interfaces:

- ADSL2 and ADSL2+ backward compatibility with Annex A, Annex M support
- PTM or EFM (802.3ah) support
- Operation, Administration, and Maintenance (OAM) support for ADSL2 and ADSL2+ modes
- Multilink Point-to-Point Protocol (MLPPP) (supported only when the VDSL2 Mini-PIM is operating in ADSL2 mode)
- MTU size of 1514 bytes (maximum) in VDSL2 mode and 1496 bytes in ADSL mode.
- Support for maximum of 10 permanent virtual connections (PVCs) (only in ADSL2 and ADSL2+ mode)

Configuring VDSL SFP Interface Using VLANs on NFX150 Network Services Platform

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

To configure VDSL SFP interfaces on NFX150 devices:

1. Connect to the host.

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

2. Configure the WAN side front panel port with vlan-tagging.

```
user@host# set interfaces virtual-interface-name vlan-tagging
```

```
user@host# set interfaces ge-1/0/1 vlan-tagging
```

3. Configure a VLAN for the WAN side front panel port.

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

```
user@host# set interfaces ge-1/0/1 unit 0 vlan-id 151
```

4. Configure the WAN side front panel port with an IP address.

```
user@host# set interfaces virtual-interface-name unit logical-unit-number family inet address
ip-address
```

```
user@host# set interfaces ge-1/0/1 unit 0 family inet address 151.1.1.151/24
```

5. Configure the physical (heth) interface with VDSL SFP options, VDSL profile, and carrier settings on the VDSL SFP interface.

```
user@host# vmhost interfaces physical-interface-name dsl-sfp-options vdsl-options profile
profile carrier carrier
```

```
user@host# set vmhost interfaces heth-0-4 dsl-sfp-options vdsl-options profile auto carrier
auto
```

NOTE:

- The default value for `vdsl-options profile` is `auto`. The value `auto` supports all profiles ranging from 8a to 17a.
- The default value for `vdsl-options carrier` is `auto`. The value `auto` includes a43 and b43.

6. Map the physical (heth) interfaces to the virtual (ge) interfaces.

```
user@host# set vmhost virtualization-options interfaces virtual-interface-name mapping
interface physical-interface-name
```

```
user@host# set vmhost virtualization-options interfaces ge-1/0/1 mapping interface heth-0-4
```

7. Commit the configuration.

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

To verify the configuration, enter the **show interfaces heth-0-4** command.

```
[edit]
user@host# show interfaces heth-0-4
Physical interface: heth-0-4, Enabled, Physical link is Up
  Link-level type: Ethernet, Media type: Fiber, MTU: 9192, VDSL2 mode, Speed: 1Gbps, Duplex:
Full-duplex, Auto-negotiation: Disabled
  VDSL status:
    Modem status  : Showtime (Auto)
    VDSL profile  : Profile-8b
  Device flags   : Present Running
  Current address: d8:b1:22:33:5e:51, Hardware address: d8:b1:22:33:5e:51
```

Configuring VDSL SFP Interface Without Using VLANs on NFX150 Network Services Platform

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

To configure VDSL SFP interfaces on NFX150 devices:

1. Connect to the host.

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

2. Configure the WAN side front panel port with an IP address.

```
user@host# set interfaces virtual-interface-name unit logical-unit-number family inet address
ip-address
```

```
user@host# set interfaces ge-1/0/1 unit 0 family inet address 151.1.1.151/24
```

3. Configure the physical (heth) interface with VDSL SFP options, VDSL profile, and carrier settings on the VDSL SFP interface.

```
user@host# vmhost interfaces physical-interface-name dsl-sfp-options vdsl-options profile
profile carrier carrier
```

```
user@host# set vmhost interfaces heth-0-4 dsl-sfp-options vdsl-options profile auto carrier
auto
```

NOTE:

- The default value for `vdsl-options profile` is `auto`. The value `auto` supports all profiles ranging from 8a to 17a.
- The default value for `vdsl-options carrier` is `auto`. The value `auto` includes a43 and b43.

4. Map the physical (heth) interfaces to the virtual (ge) interfaces.

```
user@host# set vmhost virtualization-options interfaces virtual-interface-name mapping
interface physical-interface-name
```

```
user@host# set vmhost virtualization-options interfaces ge-1/0/1 mapping interface heth-0-4
```

5. Commit the configuration.

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

To verify the configuration, enter the **show interfaces heth-0-4** command.

```
[edit]
user@host# show interfaces heth-0-4
Physical interface: heth-0-4, Enabled, Physical link is Up
  Link-level type: Ethernet, Media type: Fiber, MTU: 9192, VDSL2 mode, Speed: 1Gbps, Duplex:
Full-duplex, Auto-negotiation: Disabled
  VDSL status:
    Modem status  : Showtime (Auto)
    VDSL profile  : Profile-8b
  Device flags   : Present Running
  Current address: d8:b1:22:33:5e:51, Hardware address: d8:b1:22:33:5e:51
```

Configuring the LTE Module on NFX Devices

IN THIS SECTION

- Configuring the LTE Module for Primary Mode | 64
- Configuring the LTE Module for Dial-on-Demand Mode | 66
- Configuring the LTE Module for Backup Mode | 68
- Configuring the LTE Interface Module in an NFX Chassis Cluster | 70

The LTE module can be configured in three modes:

- Always-on—The LTE module connects to the 3G/4G network after booting. The connection is always maintained, as long as there are no network or connectivity problems.

> **NOTE**: The default mode for LTE module is always-on. For the LTE module to be operational, you only need to install one SIM card on the LTE module before powering on the device. There is no additional configuration required.

- Dial-on-demand—The LTE module initiates a connection when it receives interesting traffic. You define interesting traffic using the dialer filter. To configure dial-on-demand using a dialer filter, you first configure the dialer filter and then apply the filter to the dialer interface.
- Backup—The LTE module connects to the 3G/4G network when the primary connection fails.

You can configure the LTE module either as a primary interface or as a backup interface. When configured as the primary interface, the LTE module supports both the always-on and dial-on-demand modes. When configured as the backup interface, the LTE module connects to the network only when the primary interface fails.

> **NOTE**: Starting in Junos OS Release 19.1R1, you can configure LTE modules on both nodes in a chassis cluster to provide backup WAN support.

Profile configuration is not needed in most scenarios, as LTE has a built-in database of many service providers and can automatically select the profile to use. Occasionally, you might need to specify profiles explicitly in the configuration, in which case, the automatic profile selection is disabled.

Before you begin the configuration, insert the Subscriber Identity Module (SIM) in the LTE module. The SIM uses a profile to establish a connection with the network. You can configure up to 16 profiles for each SIM card. The LTE module supports two SIM cards and so you can configure a total of 32 profiles, although only one profile can be active at a time. To configure the SIM profile, you will require the following information from the service provider:

- Username and password
- Access Point Name (APN)
- Authentication (Challenge Handshake Authentication Protocol (CHAP) or Password Authentication Protocol (PAP))

Configuring the LTE Module for Primary Mode

Before you begin the procedure, ensure that the logical interface (dl0.0) is not configured as a backup. If dl0.0 is configured as a backup option for any interface on the device, then this configuration overrides the configuration outlined in this procedure, and the LTE module will function as a backup interface.

Use the `show interfaces | display set | match backup-option | match dl0.0` command to check whether any interface uses dl0.0 as a backup interface. If dl0.0 is configured as a backup interface, then delete the configuration by issuing the following command:

`delete interfaces` *`interface-name`* `unit 0 backup-options interface dl0.0`

To configure the LTE module as a primary interface:

1. Configure the dialer interface:

```
user@host# set interfaces dl0 unit 0 family inet negotiate-address
user@host# set interfaces dl0 unit 0 family inet6 negotiate-address
user@host# set interfaces dl0 unit 0 dialer-options pool dialer-pool-number
user@host# set interfaces dl0 unit 0 dialer-options dial-string dial-number
user@host# set interfaces dl0 unit 0 dialer-options always-on
```

2. Configure the dialer pool for the LTE physical interface:

```
user@host# set interfaces cl-1/1/0 dialer-options pool dialer-pool-number
```

   The *`dialer-pool-number`* is always 1 as there is only one LTE interface on the NFX150.

3. Configure the profile.

```
user@host# run request modem wireless create-profile profile-id profile-id cl-1/1/0 slot sim-slot-number
access-point-name apn-name authentication-method none
```

> **NOTE**: `sim-slot-number` is the slot on the module in which the SIM card is inserted.

4. Verify that the profile is configured successfully:

```
user@host# run show modem wireless profiles cl-1/1/0 slot 1
```

5. Activate the SIM card:

```
user@host# set interfaces cl-1/1/0 act-sim sim-slot-number
```

6. Select the profile and configure the radio access type for the SIM card:

```
user@host# set interfaces cl-1/1/0 cellular-options sim sim-slot-number select-profile profile-id profile-id
user@host# set interfaces cl-1/1/0 cellular-options sim sim-slot-number radio-access automatic
```

> **NOTE**: If a SIM card is installed in the second slot, then select the profile and configure the radio access type for the SIM card in the second slot as well.

7. Verify the status of the wireless network and dialer interface:

```
user@host#  run show modem wireless network
user@host#  run show interfaces dl0.0
```

Configuring the LTE Module for Dial-on-Demand Mode

When the LTE module is configured as a primary interface, it can function either in always-on mode or in dial-on-demand mode. In always-on mode, the interface remains connected to the network whereas In dial-on-demand mode, the connection is established only when needed.

In dial-on-demand mode, the dialer interface is enabled only when network traffic configured as an “interesting traffic” arrives on the network. Interesting traffic triggers or activates the wireless WAN connection. You define an interesting packet by using the dialer filter. To configure dial-on-demand by using a dialer filter, you first configure the dialer filter and then apply the filter to the dialer interface.

Once the traffic is sent over the network, an inactivity timer is triggered and the connection is closed after the timer expires.

> **NOTE**: The dial-on-demand mode is supported only if the LTE module is configured as a primary interface.

To configure the LTE module as a dial-on-demand interface:

1. Configure the dialer interface:

```
user@host# set interfaces dl0 unit 0 family inet negotiate-address
user@host# set interfaces dl0 unit 0 family inet6 negotiate-address
user@host# set interfaces dl0 unit 0 family inet filter dialer dialer-filter-name
user@host# set interfaces dl0 unit 0 dialer-options pool dialer-pool-number
user@host# set interfaces dl0 unit 0 dialer-options dial-string dial-number
```

2. (Optional) Configure the `idle-timeout` value, which determines the duration for which the connection will remain enabled in the absence of interesting traffic.

```
user@host# set interfaces dl0 unit 0 dialer-options idle-timeout idle-timeout-value
```

3. Configure the dialer pool for the LTE physical interface:

```
user@host# set interfaces cl-1/1/0 dialer-options pool dialer-pool-number
```

The *dialer-pool-number* is always 1 as there is only one LTE interface on the NFX150.

4. Create the dialer filter rule:

```
user@host# set firewall family inet dialer-filter dialer-filter-name term term1 from destination-
address ip-address then note
```

5. Set the default route:

```
user@host# set routing-options static route ip-address next-hop dl0.0
```

6. Configure the profile.

```
user@host# run request modem wireless create-profile profile-id profile-id cl-1/1/0 slot sim-slot-
number access-point-name apn-name authentication-method none
```

NOTE: *sim-slot-number* is the slot on the module in which the SIM card is inserted.

7. Verify that the profile is configured successfully:

```
user@host# run show modem wireless profiles cl-1/1/0 slot 1
```

8. Activate the SIM card:

```
user@host# set interfaces cl-1/1/0 act-sim sim-slot-number
```

9. Select the profile and configure the radio access type for the SIM card:

```
user@host# set interfaces cl-1/1/0 cellular-options sim sim-slot-number select-profile profile-id
profile-id
user@host# set interfaces cl-1/1/0 cellular-options sim sim-slot-number radio-access automatic
```

NOTE: If a SIM card is installed in the second slot, then select the profile and configure the radio access type for the SIM card in the second slot as well.

10. Verify the configuration by sending traffic to the destination address. The traffic is routed to the dl0 interface and if it matches the dialer filter rule, then the dl0 is triggered to dial.
11. Verify the status of the wireless network and dialer interface:

```
user@host#  run show modem wireless network
user@host#  run show interfaces dl0.0
```

Configuring the LTE Module for Backup Mode

You can configure the LTE module as a backup interface. If the primary interface fails, the LTE module connects to the network and remains online only until the primary interface becomes functional. The dialer interface is enabled only when the primary interface fails.

To configure the LTE module as a backup interface:

1. Configure the dialer interface:

```
user@host# set interfaces dl0 unit 0 family inet negotiate-address
user@host# set interfaces dl0 unit 0 family inet6 negotiate-address
```

```
user@host# set interfaces dl0 unit 0 dialer-options pool dialer-pool-number
user@host# set interfaces dl0 unit 0 dialer-options dial-string dial-number
```

2. Configure the dialer pool for the LTE physical interface:

```
user@host# set interfaces cl-1/1/0 dialer-options pool dialer-pool-number
```

The *dialer-pool-number* is always 1 as there is only one LTE interface on the NFX150.

3. Configure the profile.

```
user@host# run request modem wireless create-profile profile-id profile-id cl-1/1/0 slot sim-slot-number
access-point-name l3vpn.corp authentication-method none
```

NOTE: `sim-slot-number` is the slot on the LTE module in which the SIM card is inserted.

4. Verify that the profile is configured successfully:

```
user@host# run show modem wireless profiles cl-1/1/0 slot 1
```

5. Activate the SIM card:

```
user@host# set interfaces cl-1/1/0 act-sim sim-slot-number
```

6. Select the profile and configure the radio access type for the SIM card:

```
user@host# set interfaces cl-1/1/0 cellular-options sim sim-slot-number select-profile profile-id profile-
id
user@host# set interfaces cl-1/1/0 cellular-options sim sim-slot-number radio-access automatic
```

NOTE: If a SIM card is installed in the second slot, then select the profile and configure the radio access type for the SIM card in the second slot as well.

7. Configure the Ethernet interface as the primary interface, which connects to the wireless network. Configure the dl0 interface as the backup interface.

```
user@host# set interfaces ge-1/0/2 unit 0 family inet address 192.168.2.1/24
user@host# set interfaces ge-1/0/2 unit 0 backup-options interface dl0.0
```

8. Verify the status of the wireless network and dialer interface:

```
user@host#  run show modem wireless network
user@host#  run show interfaces dl0.0
```

Configuring the LTE Interface Module in an NFX Chassis Cluster

An NFX150 chassis cluster supports two `cl` interfaces, `cl-1/1/0` (primary node) and `cl-8/1/0` (secondary node).

To configure the LTE modules in a chassis cluster:

1. Configure the dialer interface (`dl0`):

```
{primary:node0}[edit]
user@host# set interfaces dl0 unit 0 family inet negotiate-address
user@host# set interfaces dl0 unit 0 family inet6 negotiate-address
user@host# set interfaces dl0 unit 0 dialer-options pool dialer-pool-number
user@host# set interfaces dl0 unit 0 dialer-options dial-string dial-number
user@host# set interfaces dl0 unit 0 dialer-options always-on
```

Sample configuration for the dl0 interface:

```
set interfaces dl0 unit 0 family inet negotiate-address
set interfaces dl0 unit 0 dialer-options pool 1
set interfaces dl0 unit 0 dialer-options always-on
set interfaces dl0 unit 0 dialer-options dial-string 1234
```

2. Configure the LTE interface (`cl-1/1/0`) on the primary node:

a. Configure the dialer pool for the LTE physical interface:

```
{primary:node0}[edit]
user@host# set interfaces cl-1/1/0 dialer-options pool dialer-pool-number
```

b. Specify the priority for the interface. The interface with the higher priority becomes the active interface.

```
{primary:node0}[edit]
user@host# set interfaces cl-1/1/0 dialer-options pool dialer-pool-number priority priority
```

c. Configure the profile:

```
{primary:node0}[edit]
user@host# run request modem wireless create-profile profile-id profile-id cl-1/1/0 slot
sim-slot-number access-point-name apn-name
```

d. Verify that the profile is configured successfully:

```
{primary:node0}[edit]
user@host# run show modem wireless profiles cl-1/1/0 slot 1
Profile details
  Max profiles: 16
  Default profile Id: 1

Profile 1: ACTIVE
  Valid: TRUE
  Username: user1
  Password: ********
  Access point name (APN): 3gnet
  Authentication: CHAP
  IP Version: IPV4V6
Profile 2: Invalid
Profile 3: Invalid
Profile 4: Invalid
Profile 5: Invalid
Profile 6: Invalid
Profile 7: Invalid
Profile 8: Invalid
```

```
Profile 9: Invalid
Profile 10: Invalid
Profile 11: Invalid
Profile 12: Invalid
Profile 13: Invalid
Profile 14: Invalid
Profile 15: Invalid
Profile 16: Inactive
  Valid: TRUE
  Access point name (APN): 3gnet
  Authentication: None
  IP Version: IPV4V6
```

e. Activate the SIM card:

```
{primary:node0}[edit]
user@host# set interfaces cl-1/1/0 act-sim sim-slot-number
```

f. Select the profile and configure the radio access type for the SIM card:

```
{primary:node0}[edit]
user@host# set interfaces cl-1/1/0 cellular-options sim sim-slot-number select-profile
profile-id profile-id
user@host# set interfaces cl-1/1/0 cellular-options sim sim-slot-number radio-access
automatic
```

> **NOTE**: If a SIM card is installed in the second slot, then select the profile and configure the radio access type for the SIM card in the second slot as well.

Sample configuration for the cl-1/1/0 interface:

```
set interfaces cl-1/1/0 act-sim 1
set interfaces cl-1/1/0 cellular-options sim 1 select-profile profile-id 1
set interfaces cl-1/1/0 cellular-options sim 1 radio-access automatic
set interfaces cl-1/1/0 cellular-options sim 2 select-profile profile-id 1
set interfaces cl-1/1/0 cellular-options sim 2 radio-access automatic
set interfaces cl-1/1/0 dialer-options pool 1 priority 1
```

3. Repeat Step 2 to configure the LTE interface (`cl-8/1/0`) for the secondary node.

   If you assign the same priority to both interfaces, then the interface that is listed first in the configuration becomes the active interface.

   To verify which interface is the active interface:

```
root@host> show dialer pools
Pool: 1
  Dialer interfaces:      Name            State
                          dl0.0           Active
  Subordinate interfaces: Name            Flags         Priority
                          cl-1/1/0        Active        100
                          cl-8/1/0        Inactive      1
```

   Sample configuration for the cl-8/1/0 interface:

```
set interfaces cl-8/1/0 act-sim 1
set interfaces cl-8/1/0 cellular-options sim 1 select-profile profile-id 1
set interfaces cl-8/1/0 cellular-options sim 1 radio-access automatic
set interfaces cl-8/1/0 cellular-options sim 2 select-profile profile-id 1
set interfaces cl-8/1/0 cellular-options sim 2 radio-access automatic
set interfaces cl-8/1/0 dialer-options pool 1 priority 254
```

4. Verify the status of the wireless network and dialer interface:

```
{primary:node0}[edit]
user@host# run show modem wireless network
LTE Connection details
  Connected time: 210
  IP: 10.90.51.234
  Gateway: 10.90.51.233
  DNS: 123.123.123.123
  IPv6: ::
  Gatewayv6: ::
  DNSv6: ::
  Input bps: 0
  Output bps: 14
  Bytes Received: 7236
  Bytes Transferred: 25468
  Packets Received: 89
```

```
  Packets Transferred: 316
Wireless Modem Network Info
  Current Modem Status: Connected
  Current Service Status: Normal
  Current Service Type: CS
  Current Service Mode: LTE
  Network: CHN-UNICOM
  Mobile Country Code (MCC): 0
  Mobile Network Code (MNC): 0
  Location Area Code (LAC): 0
  Routing Area Code (RAC): 0
  Cell Identification: 0
  Access Point Name (APN): ctnet
  Public Land Mobile Network (PLMN): CHN-UNICOM
  Physical Cell ID (PCI): N/A
  International Mobile Subscriber Identification (IMSI): ****************
  International Mobile Equipment Identification (IMEI/MEID): ****************
  Integrate Circuit Card Identity (ICCID): 89860118802425942389
  Reference Signal Receiving Power (RSRP): N/A
  Reference Signal Receiving Quality (RSRQ): N/A
  Signal to Interference-plus-Noise Ratio (SiNR): N/A
  Signal Noise Ratio (SNR): N/A
  Energy per Chip to Interference (ECIO): 0
```

```
{primary:node0}[edit]
user@host#  run show interfaces dl0.0
Physical interface: dl0, Enabled, Physical link is Up
  Interface index: 522, SNMP ifIndex: 0
  Type: 27, Link-level type: Ethernet, MTU: 1504
  Device flags   : Present Running
  Interface flags: SNMP-Traps
  Link type      : Full-Duplex
  Link flags     : None
  Current address: 00:00:5e:00:53:82, Hardware address: 00:00:5e:00:53:82
  Last flapped   : Never
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)

  Logical interface dl0.0 (Index 101) (SNMP ifIndex 0)
    Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
    Dialer:
```

```
      State: Active, Dial pool: 1
      Dial strings: 1234
      Subordinate interfaces: cl-8/1/0 (Index 519)
      Activation delay: 0, Deactivation delay: 0
      Initial route check delay: 120
      Redial delay: 255
      Callback wait period: 5
      Load threshold: 0, Load interval: 60
    Bandwidth: 300mbps
    Input packets : 1
    Output packets: 4
    Protocol inet, MTU: 1490
    Max nh cache: 0, New hold nh limit: 0, Curr nh cnt: 0, Curr new hold cnt: 0, NH drop cnt:
0
      Flags: Sendbcast-pkt-to-re, Negotiate-Address
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 10.34.163.0/26, Local: 10.34.163.31, Broadcast: 10.34.163.63
```

By default, the time interval taken to switch to the secondary cl interface when the active cl interface times out is 120 seconds. You can change the time interval by configuring the `redial-delay` option:

```
{primary:node0}[edit]
user@host# user@host# set interfaces dl0 unit 0 dialer-options redial-delay time-in-seconds
```

RELATED DOCUMENTATION

Upgrading the Modem Firmware on NFX Devices Through Over-the-Air (OTA) | 75

Upgrading the Modem Firmware on NFX Devices Through Over-the-Air (OTA)

Over-the-Air (OTA) firmware upgrade enables automatic and timely upgrade of modem firmware when new firmware versions are available. The OTA upgrade can be enabled or disabled on the LTE module. OTA is disabled by default.

NOTE: When upgrading the software on the NFX devices, the LTE firmware is also upgraded if the software contains a newer firmware version.

1. Enable OTA upgrade on the LTE module:

   ```
   user@host > request modem wireless fota cl-1/1/0 enable
   ```

   ```
   Set FOTA on modem succeeded
   ```

2. Initiate the firmware upgrade:

   ```
   user@host > request modem wireless upgrade cl-1/1/0
   ```

   ```
   Launch FOTA upgrade succeeded
   ```

3. Verify the firmware upgrade status:

   ```
   user@host > show modem wireless firmware cl-1/1/0
   ```

   ```
   LTE mPIM firmware details
     Product name: Junos LTE mPIM
     Serial number: D23F4349-10FA-41AA-A538-03648DE
     Hardware version: AcceleratedConcepts/porter
     Firmware version: 17.11.13
     MAC: 00:00:5e:00:53:82
     System uptime: 4632 seconds
   Wireless modem firmware details
     Modem firmware version: 9999999_9904609_SWI9X30C_02.24.05.06_00_GENERIC_002.026_000
     Modem Firmware build date: 19/05/2017
     Card type: MC7455
     Modem manufacturer: Sierra Wireless, Inc
     Hardware version: 1.0
     Power & Temperature: Normal 3368 mV, Normal 29.00 C
   OTA status
     State: Enabled
     New firmware available: No
   Number of SIM: 1
   Slot of active: 1
   Status of SIM 1
     SIM state: SIM present
     Modem PIN security status: Disabled
   ```

```
  SIM status: SIM Okay
  SIM user operation needed: No Op
  Retries remaining: 3
```

4. Check the LTE module connection status:

```
user@host > show modem wireless network cl-1/1/0
```

```
LTE Connection details
  Connected time: 2880
  IP: 10.12.219.210
  Gateway: 10.12.219.209
  DNS: 123.123.123.123
  IPv6: ::
  Gatewayv6: ::
  DNSv6: ::
  Input bps: 0
  Output bps: 0
  Bytes Received: 1952
  Bytes Transferred: 2164
  Packets Received: 10
  Packets Transferred: 20
Wireless Modem Network Info
  Current Modem Status: Connected
  Current Service Status: Normal
  Current Service Type: PS
  Current Service Mode: LTE
  Current Band: B3
  Network: UNICOM
  Mobile Country Code (MCC): 460
  Mobile Network Code (MNC): 1
  Location Area Code (LAC): 65534
  Routing Area Code (RAC): 0
  Cell Identification: 239907605
  Access Point Name (APN): 3gnet
  Public Land Mobile Network (PLMN): CHN-UNICOM
  Physical Cell ID (PCI): 452
  International Mobile Subscriber Identification (IMSI): ***************
  International Mobile Equipment Identification (IMEI/MEID): ***************
  Integrate Circuit Card Identity (ICCID): 89860117811046631207
  Reference Signal Receiving Power (RSRP): -71
  Reference Signal Receiving Quality (RSRQ): -8
```

```
Signal to Interference-plus-Noise Ratio (SiNR): 19
Signal Noise Ratio (SNR): 22
Energy per Chip to Interference (ECIO): 0
```

RELATED DOCUMENTATION

Configuring the LTE Module on NFX Devices | 63

5

CHAPTER

Configuring USB Pass-Through on NFX Series Devices

Supporting File Transfer from USB on NFX Series Devices | 80

Installing Software on NFX Devices Using USB Autoinstallation | 82

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
```

Installing Software on NFX Devices Using USB Autoinstallation

IN THIS SECTION

- Preparing the USB | 83
- Configuring the NFX Device | 85
- Installing the Image | 86
- Disabling Autoinstallation | 87

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

- Precautions | 83
- Preparing the USB | 83

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
```

6

CHAPTER

Configuring Security

IP Security on NFX Devices | 89

Content Security on NFX Devices | 99

Application Security on NFX Devices | 100

Intrusion Detection and Prevention on NFX Devices | 101

Integrated User Firewall Support on NFX Devices | 102

IP Security on NFX Devices

IN THIS SECTION

- Overview | 89
- Configuring Security | 91

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 19 on page 90 lists the IPsec features supported on NFX Series devices.

Table 19: 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 19: 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

- Configuring Interfaces | 91
- Configuring Routing Options | 92
- Configuring Security IKE | 93
- Configuring Security IPsec | 96
- Configuring Security Policies | 98
- Configuring Security Zones | 99

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
```

Content Security on NFX Devices

The Content Security solution consolidates several security features to protect against multiple threat types. The Content Security solution for NFX devices consists of the following security features:

- Antispam—Examines e-mail messages to identify spam. When the device detects an e-mail spam, it drops the message or tags the message header or subject field with a preprogrammed string. For more information, see *Antispam Filtering Overview*.
- Antivirus—Offers a less CPU-intensive alternative to the full file-based antivirus feature. Sophos uses a scanning engine and virus signature databases to protect against virus-infected files, worms, trojans, spyware, and other malware over POP3, HTTP, SMTP, IMAP, and FTP protocols. The virus pattern and malware database is located on external servers maintained by Sophos (Sophos Extensible List) servers. For more information, see Sophos Antivirus Protection on NFX Devices (OBSOLETE).
- Content filtering—Blocks or permits certain types of traffic based on the MIME type, file extension, protocol command, and embedded object type. For more information, see *Content Filtering*.
- Web filtering—Allows you to manage Internet usage by preventing access to inappropriate Web content. The Web filtering solution consists of the following types:
  - Redirect web filtering
  - Local web filtering
  - Enhanced Web filtering

  For more information, see *Web Filtering Overview*.

NOTE: Antispam, Sophos antivirus, and enhanced web filtering are licensed features and will not function until you install the respective licenses.

RELATED DOCUMENTATION

Intrusion Detection and Prevention on NFX Devices | **101**

Integrated User Firewall Support on NFX Devices | **102**

Application Security on NFX Devices

The NFX150 devices support the AppSecure feature, which is a suite of application-aware security services that deliver security services to provide visibility and control over the types of applications traversing in the networks. AppSecure uses a sophisticated classification engine to accurately identify

applications regardless of port or protocol, including nested applications that reside within trusted network services.

The AppSecure feature comprises of the following services:

- Application identification (AppID)- Recognizes traffic at different network layers using characteristics other than port number. Once the application is determined, AppSecure service modules can be configured to monitor and control traffic for tracking, prioritization, access control, detection, and prevention based on the application ID of the traffic. For more information, see Application Identification.
- Application Tracking (AppTrack)—Tracks and reports applications passing through the device. For more information, see Application Tracking on NFX Devices.
- Application Firewall (AppFW)—Implements an application firewall using application-based rules. For more information, see Application Firewall on NFX Devices.
- Application Quality of Service (AppQoS)—Provides quality-of-service prioritization based on application awareness. For more information, see Application QoS.
- Advanced policy-based routing (APBR)— Classifies session based on applications and applies the configured rules to reroute the traffic. For more information, see Advanced Policy-Based Routing on NFX Devices.

AppSecure works with additional content security on the device through integrated Content Security , intrusion prevention systems (IPS), and Juniper Networks Juniper Advanced Threat Prevention Cloud (ATP Cloud) for deeper protection against malware, spam, phishing, and application exploits.

RELATED DOCUMENTATION

Integrated User Firewall Support on NFX Devices | 102

Intrusion Detection and Prevention on NFX Devices

Intrusion detection is the process of monitoring the events occurring in your network and analyzing them for signs of possible incidents, violations, or imminent threats to your security policies. Intrusion prevention is the process of performing intrusion detection and then stopping the detected incidents. These security measures are available as intrusion detection systems (IDS) and intrusion prevention systems (IPS), which become part of your network to detect and stop potential incidents.

An Intrusion Detection and Prevention (IDP) policy lets you selectively enforce various attack detection and prevention techniques on the network traffic passing through your device. Juniper devices offer the

same set of IDP signatures that are available on Juniper Networks IDP Series Intrusion Detection and Prevention Appliances to secure networks against attacks. The basic IDP configuration involves the following tasks:

- Download and install the IDP license.
- Download and install the signature database—You must download and install the IDP signature database. The signature databases are available as a security package on the Juniper Networks website. This database includes attack object and attack object groups that you can use in IDP policies to match traffic against known attacks.
- Configure recommended policy as the IDP policy—Juniper Networks provides predefined policy templates to use as a starting point for creating your own policies. Each template is a set of rules of a specific rulebase type that you can copy and then update according to your requirements.

  To get started, we recommend you use the predefined policy named “Recommended”.
- Enable a security policy for IDP inspection—For transit traffic to pass through IDP inspection, you configure a security policy and enable IDP application services on all traffic that you want to inspect.

For information on configuring IDP on NFX Series devices, see the Intrusion Detection and Prevention User Guide.

RELATED DOCUMENTATION

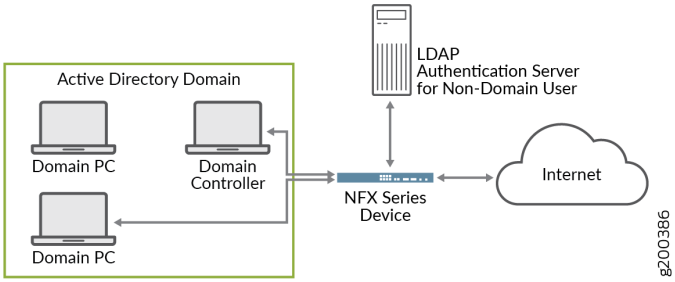
Content Security on NFX Devices | 99

Integrated User Firewall Support on NFX Devices

The integrated user firewall feature introduces an authentication source via integration with Microsoft Active Directory. This feature consists of the device polling the event log of the Active Directory controller to determine, by username and source IP address, who has logged in to the device. Then the username and group information are queried from the LDAP service in the Active Directory controller. Once the device has the IP address, username, and group relationship information, it generates authentication entries. With the authentication entries, the device user firewall module enforces user-based and group-based policy control over traffic.

Figure 9 on page 103 illustrates a typical scenario where the integrated user firewall feature is deployed. Users in the Active Directory domain and users outside the Active Directory domain want access to the Internet through the device. The domain controller might also act as the LDAP server.

Figure 9: Scenario for Integrated User Firewall



The device reads and analyzes the event log of the domain controller and generates an authentication table as an Active Directory authentication source for this feature. The user firewall is aware of any domain user on an Active Directory domain device via the Active Directory authentication source. The device administrator configures a user firewall policy that enforces the desired user-based or group-based access control.

For information on configuring the integrated user firewall on NFX Series devices, see Authentication and Integrated User Firewalls User Guide.

RELATED DOCUMENTATION

Content Security on NFX Devices | 99

7

CHAPTER

Configuring VNFs

Prerequisites to Onboard Virtual Network Functions on NFX150 Devices | 105

Configuring VNFs on NFX150 Devices | 106

Managing VNFs on NFX Series Devices | 120

Configuring Analyzer VNF and Port-mirroring | 128

Supporting Faster File Copy or Transfer | 129

Prerequisites to Onboard Virtual Network Functions on NFX150 Devices

IN THIS SECTION

- Prerequisites for VNFs | 105

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

The number of VNFs that you can onboard on the device depends on the availability of system resources such as the number of CPUs and system memory.

Before you onboard the VNFs, it is recommended to check the available system resources such as CPUs, memory, and storage for VNFs. For more information, see "Configuring VNFs on NFX150 Devices" on page 106.

Prerequisites for VNFs

To instantiate VNFs, the NFX device supports:

- KVM based hypervisor deployment
- OVS or Virtio interface drivers
- raw or qcow2 VNF file types
- (Optional) SR-IOV
- (Optional) CD-ROM and USB configuration drives
- (Optional) Hugepages for memory requirements

Configuring VNFs on NFX150 Devices

IN THIS SECTION

- Load the VNF Image | 106
- Prepare the Bootstrap Configuration | 107
- Allocate CPUs for a VNF | 108
- Allocate Memory for a VNF | 110
- (Optional) Attach a Config Drive to the VNF | 111
- Configure Interfaces and VLANs for a VNF | 113
- Configuring VNF Storage Devices | 117
- Instantiating a VNF | 118
- Verify that the VNF Instantiated Successfully | 119

The NFX150 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.

To configure a VNF, log in to the JCP and perform the following tasks:

Load the VNF Image

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

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.

To copy a VNF image from a USB, see "Supporting File Transfer from USB on NFX Series Devices" on page 80.

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

NOTE: You must save the VNF image in the **/var/public** directory.

Prepare the Bootstrap Configuration

You can bootstrap a VNF using an attached config drive that contains a bootstrap-config ISO file. The config drive is a virtual drive, which can be a CD-ROM, USB drive or Disk drive associated to a VNF with the configuration data. Configuration data can be files or folders, which are bundled in the ISO file that makes a virtual CD-ROM, USB drive, or Disk drive.

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.

By attaching a config drive, you can pass the networking configurations such as the IP address, subnet mask, and gateway to the VNFs through a CLI. After receiving the configuration inputs, the device generates a bootstrap-config ISO file, and attaches the file to the VNF as a CD-ROM, USB drive, or Disk drive.

For more information about configuring and attaching a config drive, see "(Optional) Attach a Config Drive to the VNF " on page 111.

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.
- The config drive is a read-only drive. Based on the VNF, you can specify the config drive as a read-only CD-ROM drive, USB drive, or a Disk drive.

The config drive supports the following data for VNFs:

- Static content as files—The device accepts one or more file paths through a CLI, converts these files to an ISO image, and attaches it to the VNF. The config drive supports multiple static files in a VNF configuration.
- Jinja2 template and parameters—Jinja2 parameters consist of key-value pairs. The key is specified in the template and the value replaces the key when the template is rendered. The system adds the rendered output file to the ISO image, and attaches it to the VNF. The maximum number of parameters for a template is 256 key-value pairs. The config drive supports multiple templates and its parameters in a VNF configuration.

> **NOTE**: The config drive supports only Jinja2 templates.

- Directory—The device accepts the specific directory contents, converts the folder structure in the given folder to an ISO image, and attaches it to the VNF. The config drive accepts only one folder. That folder becomes the root directory in the ISO image, and all the subsequent folders and files are added to the ISO image.

> **NOTE**:
>
> - You can add multiple source templates and source files in a VNF configuration.
> - To add multiple source templates and one source folder in a VNF configuration, the target template file must be inside the source folder.
> - You can add only one source folder in a VNF configuration.
> - If two VNFs share the same set of files, separate bootstrap-config ISO files are generated for each VNF. Deleting one VNF will not affect the other VNF.

Allocate CPUs for a VNF

Table 20 on page 108 lists the CPUs available for VNF usage for the NFX150 models.

Table 20: CPUs Available for VNF Usage

Model	CPUs Available for VNF Usage			
	Throughput Mode	Hybrid Mode	Compute Mode	**Custom Mode (Flex Mode)**
NFX150-C-S1	0	1	2	1
NFX150-C-S1-AE	0	1	2	1
NFX150-C-S1-AA	0	1	2	1

Table 20: CPUs Available for VNF Usage *(Continued)*

Model	CPUs Available for VNF Usage			
	Throughput Mode	Hybrid Mode	Compute Mode	**Custom Mode (Flex Mode)**
NFX150-C-S1E-AE	0	1	2	1
NFX150-C-S1E-AA	0	1	2	1
NFX150-S1	0	2	4	4
NFX150-S1E	0	2	4	4

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 current operational mode of the device:

```
user@host> show vmhost mode
```

For more information, see *show vmhost mode*.

To check the CPU availability and its status:

```
user@host> show system visibility cpu
```

For more information, see *show system visibility cpu*.

To specify the number of virtual CPUs that are required for a VNF, use the following commands:

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

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

2. Pin 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 either be a number or a range. By default, a VNF is allocated with one virtual CPU that is not pinned to any physical CPU.

> **NOTE**: You cannot change the CPU configuration of a VNF when the VNF is in **running** state. Restart the VNF for 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, type the following command:

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

Allocate Memory for a VNF

On NFX150 devices running Junos OS Release 18.1R1, enabling hugepages for VNFs and pre-reserving of hugepages are not supported.

Table 21 on page 111 lists the possible memory availability for VNF usage for the NFX150 models.

Table 21: Memory Availability for VNF Usage

Model	**Total Memory Available**	Memory/Hugepages Availability for VNF Usage in Compute, Hybrid, and Throughput Modes	**Memory/Hugepages Availability for VNF Usage in Custom Mode (Flex Mode)**
		NOTE: Hugepages are not required for NFX150-C-S1 and NFX150-C-S1E models.	
NFX150-C-S1	8	1 GB	2 GB
NFX150-C-S1-AE	8	1 GB	2 GB
NFX150-C-S1-AA	8	1 GB	2 GB
NFX150-C-S1E-AE	16	9 GB	10 GB
NFX150-C-S1E-AA	16	9 GB	10 GB
NFX150-S1	16	7 1G Hugepages	8 1G Hugepages
NFX150-S1E	32	23 1G Hugepages	24 1G Hugepages

To specify the maximum primary memory that the VNF can use, enter the following command:

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

NOTE: You cannot change the memory configuration of a VNF if the VNF is in the running state. Restart the VNF for changes to take effect.

(Optional) Attach a Config Drive to the VNF

Add files and template to the config drive.

1. Specify the source file to add in the config drive:

```
user@host# set virtual-network-functions vnf-name config-data source file source-file1
user@host# set virtual-network-functions vnf-name config-data source file source-file2
```

2. Specify the template file to add in the config drive:

```
user@host# set virtual-network-functions vnf-name config-data source template template-name
file template-file
user@host# set virtual-network-functions vnf-name config-data source template template-name
parameters image_type image-type
user@host# set virtual-network-functions vnf-name config-data source template template-name
parameters memory-size memory-size
user@host# set virtual-network-functions vnf-name config-data source template template-name
target target-filename
```

3. Specify the device name and type to add in the config drive:

```
user@host# set virtual-network-functions vnf-name config-data target device-name target-
device-name
```

```
user@host# set virtual-network-functions vnf-name config-data target device-type target-
device-type
```

The `target device-type` is optional. If you do not specify, it takes the device type as `cd-rom`.

```
user@host# set virtual-network-functions vnf-name config-data target device-label target-
device-label
```

The `target device-label` is optional. If you do not specify, it takes the device label as `config-data`.

4. Commit the configuration:

```
user@host# commit
```

Add a directory to the config drive.

1. Specify the source directory to add in the config drive:

```
user@host# set virtual-network-functions vnf-name config-data source directory directory-name
```

2. Specify the device name and type to add in the config drive:

```
user@host# set virtual-network-functions vnf-name config-data target device-name target-
device-name
```

```
user@host# set virtual-network-functions vnf-name config-data target device-type device-type
user@host# set virtual-network-functions vnf-name config-data target device-label device-label
```

3. Commit the configuration:

```
user@host# commit
```

To verify whether the config drive is attached to the VNF, see the `VNF Disk Information` section in the *show system visibility vnf* command output message.

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 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
```

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.

Configuring VNF Storage Devices

The NFX150 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, enter the following command:

```
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 hdx or sdx or vdx. For example, hdb, sdc, vdb and so on.

To add a virtual USB storage device, enter the following command:

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

To attach an additional hard disk, enter the following command:

```
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 a hard disk from the VNF, enter the following command:

```
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 changes to take effect. The CD detach operation fails if the device is in use within the VNF.
- 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 in **running** state.
- You must save the source file in the **/var/public** directory and the file must have read and write permission for all users.

Instantiating 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 internal network. The target Peripheral Component Interconnect (PCI) addresses, such as 0000:00:03:0 and 0000:00:04:0 are reserved for these interfaces.

- To instantiate a VNF using an image:

> **NOTE**: Only Qcow2 and Raw image types are supported.

```
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 configuring VNFs, do not use VNF names in the format vnf*n*—for example, vnf1, vnf2, and so on. Configurations containing 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, and it is recommended to allow the system to allocate a UUID for the VNF.

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

Verify that the VNF Instantiated Successfully

Verify that the VNF instantiated successfully by using the following command:

```
user@host> show virtual-network-functions
ID       Name                                          State      Liveliness
-------------------------------------------------------------------------------
1        vjunos0                                       Running    alive
2        centos1                                       Running    alive
3        centos2                                       Running    alive
11057    LTE                                           Running    alive
```

The **Liveliness** output field of a VNF indicates whether the IP address of the VNF is reachable or not reachable over the internal management network. The default IP address of the liveliness bridge is 192.0.2.1/24.

Managing VNFs on NFX Series Devices

IN THIS SECTION

- Managing VNF States | 120
- Managing VNF MAC Addresses | 121
- Managing the MTU of a VNF Interface | 122
- Accessing a VNF from the JCP | 123
- Viewing the List of VNFs | 123
- Displaying the Details of a VNF | 124
- Deleting a VNF | 124
- Non-Root User Access for VNF Console | 125

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 or specified in an init-descriptor XML file, are assigned a globally unique and persistent MAC address. A common pool of 64 MAC addresses is used to assign MAC addresses to VNF interfaces. You can configure a MAC address other than what is available in the common pool, and this address will not be overwritten.

There are 160 MAC addresses for the network interfaces on the VNF. These MAC addresses are automatically allocated when a VNF is instantiated.

- 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 2048 bytes as the MTU size. The default MTU value is 1500 bytes, and the maximum MTU size for a VNF interface is 2048 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 25.

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 vnf-name ssh
```

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

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

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

- Accessing the VNF Console | **127**
- Exiting the VNF Console | **128**

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 22: 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]`
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>]`

Table 2 lists the user access permissions for the VNF management options:

Table 23: 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]`

Table 3 on page 127 lists the VNF management options available for a user belonging to a custom Junos OS user class with `vnf-operation` permission.

Table 24: 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

- Configuring for in-band External Interface | 130
- Configuring for out-of-band External Interface | 131

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
```

8

CHAPTER

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

Mapping of Address and Port with Encapsulation on NFX Series Devices | 134

Configuring MAP-E on NFX Series Devices | 136

Mapping of Address and Port with Encapsulation on NFX Series Devices

IN THIS SECTION

- Overview | 134
- Benefits of MAP-E | 134
- MAP-E Terminology | 135
- MAP-E Functionality | 135

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.

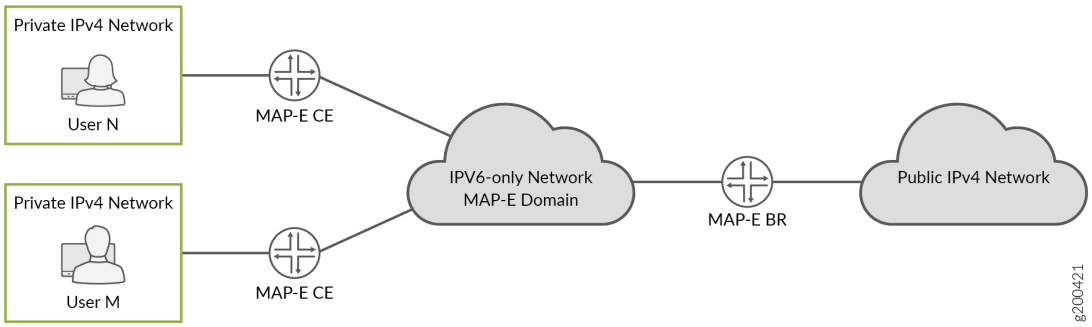
MAP-E Terminology

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 10 on page 136 illustrates a simple MAP-E deployment scenario.

Figure 10: 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 | 137

- Requirements | 137
- Topology Overview | 137
- Configure an NFX Series Device as a MAP-E CE Device | 138
- Configure an MX Series Device as a BR Device | 141
- Verify the MAP-E Configuration | 143

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" on page 134.

Requirements

This example uses the following hardware and software components:

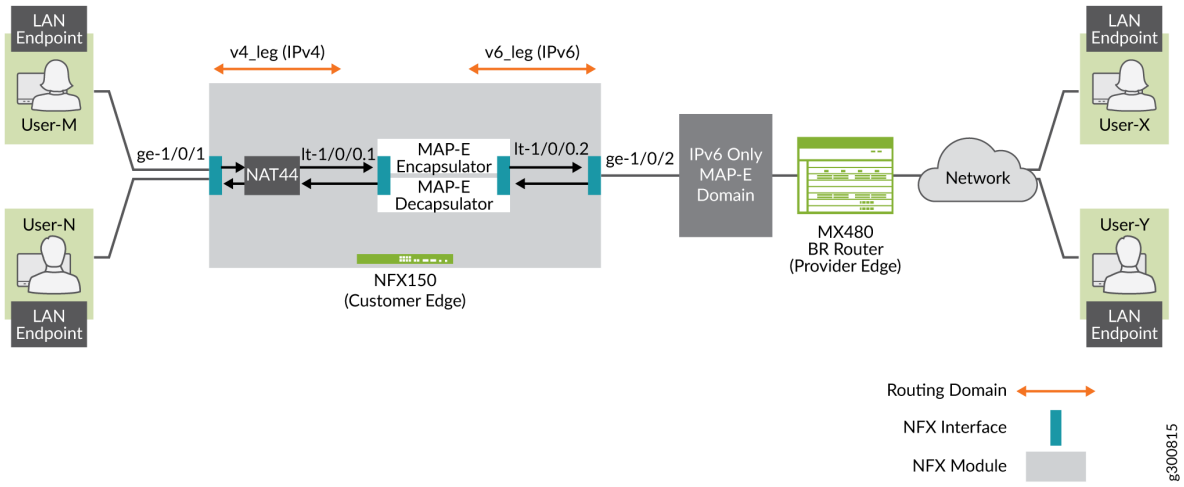
- NFX150 device running Junos OS Release 19.4R1, deployed as a customer edge (CE) device.
- MX480 device, deployed as a border relay (BR) device.
- Map physical interfaces to virtual interfaces. For more information, see Mapping Interfaces on NFX150 Devices.

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 11 on page 138 shows the MAP-E deployment on NFX Series devices.

Figure 11: 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 confidentiality and show security softwires map-e confidentiality status.

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 | 143
- Action | 143
- Meaning | 147

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.

9

CHAPTER

Configuring High Availability

Chassis Cluster on NFX150 Devices | 149

Upgrading or Disabling a Chassis Cluster on NFX150 Devices | 163

Chassis Cluster on NFX150 Devices

IN THIS SECTION

- NFX150 Chassis Cluster Overview | 149
- Chassis Cluster Interfaces | 150
- Chassis Cluster Limitation | 151
- Example: Configuring a Chassis Cluster on NFX150 Devices | 151

A chassis cluster, where two devices operate as a single device, provides high availability on NFX150 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.

NFX150 Chassis Cluster Overview

IN THIS SECTION

- Chassis Cluster Modes | 150

You can configure NFX150 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 heth-0-0 port on one node to the heth-0-0 port on the second node.

  > **NOTE**: You can use only the heth-0-0 port 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 port except the heth-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 device 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 Fast Ethernet interfaces or 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 NFX150 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 heth-0-0 port is configured as the dedicated control interface on NFX150 devices. Therefore, you cannot map the heth-0-0 port to any other virtual interface if the device is part of a chassis cluster.

- 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 NFX150 Devices

IN THIS SECTION

- Requirements | 151
- Overview | 152
- Configuration | 153
- Verification | 159

This example shows how to set up chassis clustering on NFX150 devices.

Requirements

Before you begin:

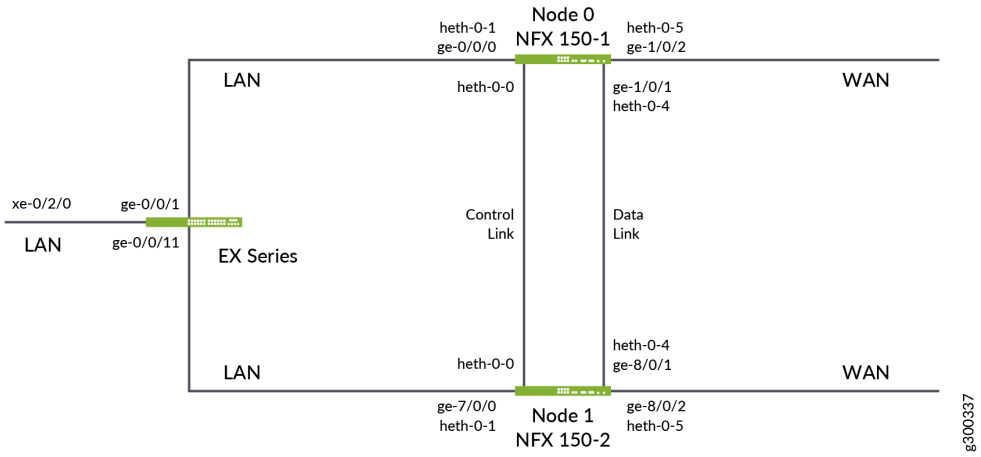
- Physically connect the two devices and ensure that they are the same NFX150 model.
- Ensure that both devices are running the same Junos OS version
- Remove all interface mapping for the control port heth-0-0 on both the nodes.
- Connect the dedicated control port heth-0-0 on node 0 to the heth-0-0 port on node 1.
- Connect the fabric port on node 0 to the fabric port on node 1.

Overview

Figure 12 on page 152 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 12: NFX150 Chassis Cluster



Configuration

IN THIS SECTION

- Configuring a Chassis Cluster | 153
- Configuring Redundant Groups and Redundant Interfaces | 156

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@NFX150-1> 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 NFX150-1
user@host1# set groups node0 interfaces fxp0 unit 0 family inet address 10.204.41.80/20
```

```
user@host1# set groups node1 system host-name NFX150-2
user@host1# set groups node1 interfaces fxp0 unit 0 family inet address 10.204.41.90/20
```

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 groups node0 routing-options static route 0.0.0.0/0 next-hop 10.204.47.254
user@host1# set groups node1 routing-options static route 0.0.0.0/0 next-hop 10.204.47.254
```

4. Map the physical LAN port to the virtual LAN interface on FPC0:

```
user@host1# set groups node0 vmhost virtualization-options interfaces ge-0/0/0 mapping
interface heth-0-1
user@host1# set groups node1 vmhost virtualization-options interfaces ge-7/0/0 mapping
interface heth-0-1
```

NOTE: In a chassis cluster, the FPC1 ports on the secondary node are denoted as ge-8/0/x, and the FPC0 ports are denoted as ge-7/0/x.

5. Map the physical WAN port to the virtual WAN interface on FPC1:

```
user@host1# set groups node0 vmhost virtualization-options interfaces ge-1/0/2 mapping
interface heth-0-5
user@host1# set groups node1 vmhost virtualization-options interfaces ge-8/0/2 mapping
interface heth-0-5
```

6. Configure port peering between the FPC0 and FPC1 on nodes 0 and 1. 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@host# set groups node0 chassis cluster redundant-interface ge-1/0/0 mapping-interface
ge-0/0/0
user@host# set groups node1 chassis cluster redundant-interface ge-8/0/0 mapping-interface
ge-7/0/0
```

7. Configure the fabric ports:

```
user@host1# set groups node0 vmhost virtualization-options interfaces ge-1/0/1 mapping
interface heth-0-4
user@host1# set groups node1 vmhost virtualization-options interfaces ge-8/0/1 mapping
interface heth-0-4
user@host1# set interfaces fab0 fabric-options member-interfaces ge-1/0/1
user@host1# set interfaces fab1 fabric-options member-interfaces ge-8/0/1
```

8. Apply the node-specific configurations on nodes 0 and 1:

```
user@host1# set apply-groups "${node}"
```

9. 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
```

10. Verify the interfaces:

```
user@host1# run show chassis cluster interfaces
```

Configuring Redundant Groups and Redundant Interfaces

Step-by-Step Procedure

1. Configure redundancy groups 1 and 2. Both `redundancy-group 1` and `redundancy-group 2` control the data plane and include 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 redundancy-group 1 node 0 priority 100
user@host1# set chassis cluster redundancy-group 1 node 1 priority 100
user@host1# set chassis cluster redundancy-group 2 node 0 priority 100
user@host1# set chassis cluster redundancy-group 2 node 1 priority 100
```

2. Enable preempt for `redundancy-group 1`.

```
user@host1# set chassis cluster redundancy-group 1 preempt
```

> **NOTE**: If preempt is added to a redundancy group configuration, the device with the higher priority in the group can initiate a failover to become the primary device. By default, preemption is disabled.

3. Configure the interfaces that the redundancy groups need to monitor to determine whether an interface is up or down.

   By default, redundancy groups have a threshold tolerance value of 255. When an interface monitored by a redundancy group becomes unavailable, its weight is subtracted from the redundancy group's threshold. When a redundancy group's threshold reaches 0, it fails over to the other node.

```
user@host# set chassis cluster redundancy-group 1 interface-monitor ge-1/0/0 weight 255
user@host# set chassis cluster redundancy-group 1 interface-monitor ge-8/0/0 weight 255
user@host# set chassis cluster redundancy-group 2 interface-monitor ge-1/0/2 weight 255
user@host# set chassis cluster redundancy-group 2 interface-monitor ge-8/0/2 weight 255
```

4. Configure the data interfaces so that in the event of a data plane failover, the other chassis cluster member can take over the connection seamlessly.

   Define the following parameters:

- The maximum number of reth interfaces for the cluster, so that the system can allocate the appropriate resources for them.

```
user@host1# set chassis cluster reth-count 5
```

- The heartbeat interval and threshold, which define the wait time before failover is triggered in the chassis cluster.

```
user@host1# set chassis cluster heartbeat-interval 1000
user@host1# set chassis cluster heartbeat-threshold 5
```

- Membership information of the member interfaces to reth interfaces.

```
user@host# set interfaces ge-1/0/0 gigether-options redundant-parent reth1
user@host# set interfaces ge-1/0/2 gigether-options redundant-parent reth2
user@host# set interfaces ge-8/0/0 gigether-options redundant-parent reth1
user@host# set interfaces ge-8/0/2 gigether-options redundant-parent reth2
```

5. Configure the reth interfaces:

- Configure reth1:

```
user@host1# set interfaces reth1 vlan-tagging
user@host1# set interfaces reth1 redundant-ether-options redundancy-group 1
user@host1# set interfaces reth1 unit 0 vlan-id 100
user@host1# set interfaces reth1 unit 0 family inet address 192.0.3.1/24
user@host1# set interfaces reth1 unit 0 family inet6 address      2001:db8:1:1::/64
```

- Configure reth2:

```
user@host1# set interfaces reth2 vlan-tagging
user@host1# set interfaces reth2 redundant-ether-options redundancy-group 2
user@host1# set interfaces reth2 unit 0 vlan-id 200
user@host1# set interfaces reth2 unit 0 family inet address •    203.0.113.1/24
user@host1# set interfaces reth2 unit 0 family inet6 address      2001:db8:2:1::/64
```

6. 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 | 159

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@NFX150-1> 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

Redundancy group: 1 , Failover count: 0
node0  100      primary              yes     no       None
node1  100      secondary            yes     no       None

Redundancy group: 2 , Failover count: 0
node0  100      primary              no      no       None
node1  100      secondary            no      no       None
```

- Verify the status of the redundancy groups:

```
root@NFX150-1> show chassis cluster information
node0:
--------------------------------------------------------------------------
Redundancy Group Information:

    Redundancy Group 0 , Current State: primary, Weight: 255

        Time            From                 To                   Reason
        Mar 21 12:06:18 hold                 secondary            Hold timer expired
        Mar 21 12:18:46 secondary            primary              Remote node reboot

    Redundancy Group 1 , Current State: primary, Weight: 255

        Time            From                 To                   Reason
        Mar 21 12:06:19 hold                 secondary            Hold timer expired
        Mar 21 12:08:51 secondary            primary              Remote is in secondary hold

    Redundancy Group 2 , Current State: primary, Weight: 255

        Time            From                 To                   Reason
        Mar 21 12:06:19 hold                 secondary            Hold timer expired
        Mar 21 12:18:46 secondary            primary              Remote node reboot

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 21 13:02:05 hold                 secondary            Hold timer expired

    Redundancy Group 1 , Current State: secondary, Weight: 255

        Time            From                 To                   Reason
        Mar 21 13:02:05 hold                 secondary            Hold timer expired

    Redundancy Group 2 , Current State: secondary, Weight: 255

        Time            From                 To                   Reason
        Mar 21 13:02:06 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@NFX150-1> 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        Down        Not configured
    reth1        Down        1
    reth2        Down        2
    reth3        Down        Not configured
    reth4        Down        Not configured

Redundant-pseudo-interface Information:
    Name         Status      Redundancy-group
    lo0          Up          0

Interface Monitoring:
    Interface         Weight    Status                    Redundancy-group
                                (Physical/Monitored)
    ge-8/0/0          255       Up  /  Up                 1
    ge-1/0/0          255       Up  /  Up                 1
    ge-8/0/2          255       Up  /  Up                 2
    ge-1/0/2          255       Up  /  Up                 2
```

- Verify the status of the port-peering interfaces:

```
root@NFX150-1> show chassis cluster port-peering
node0:
--------------------------------------------------------------------------

Port peering interfaces:
    Backend L3                Mapped Peer L2
    Interface  Status         Interface  Status
    ge-1/0/0    Up            ge-0/0/0   Up

node1:
--------------------------------------------------------------------------

Port peering interfaces:
    Backend L3                Mapped Peer L2
    Interface  Status         Interface  Status
    ge-8/0/0    Up            ge-7/0/0   Up
```

RELATED DOCUMENTATION

- Monitoring of Global-Level Objects in a Chassis Cluster
- Monitoring Chassis Cluster Interfaces
- Monitoring IP Addresses on a Chassis Cluster
- Configuring Cluster Failover Parameters
- Chassis Cluster Redundancy Group Failover

Upgrading or Disabling a Chassis Cluster on NFX150 Devices

IN THIS SECTION

- Upgrading Individual Devices in a Chassis Cluster Separately | 163
- Disabling a Chassis Cluster | 164

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
```

10

CHAPTER

Configuring Service Chaining

Service Chaining on NFX150 Devices | 166

Example: Configuring Service Chaining Using VLANs on NFX150 Network Services Platform | 168

Example: Configuring Service Chaining Using SR-IOV on NFX150 Network Services Platform | 175

Example: Configuring Service Chaining Using a Custom Bridge | 178

Example: Configuring Service Chaining for LAN-WAN Routing | 187

Example: Configuring Cross Connect on NFX150 Devices | 193

Example: Configuring Service Chaining for LAN Routing | 203

Example: Configuring Cross-Connect Using a Custom Bridge on NFX150 Devices | 207

Service Chaining on NFX150 Devices

IN THIS SECTION

- Understanding Service Chaining | 166
- Configuring Service Chaining Using VLANs | 166
- Configuring Service Chaining Using DHCP Services on VLANs | 167

Understanding Service Chaining

In many network environments, it is common for traffic to flow through several *network services* on the way to its destination. These services—firewalls, Network Address Translators (NAT), load balancers, and so on—are generally spread across multiple network elements. Each device is a separate piece of hardware, providing a different service, and requiring separate operation and management. This method of linking together multiple network functions could be thought of as *physical service chaining*.

A more efficient model for service chaining is to virtualize and consolidate network functions onto a single device.

Virtualized service chaining is supported on NFX150 devices starting with Junos OS Release 18.1. Virtual network functions (VNFs) can be installed and linked together to provide L2, L3, and L4-L7 services for traffic flowing through the device.

Configuring Service Chaining Using VLANs

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

To configure service chaining:

1. Create a VLAN. Use one of the following commands:

- Create a VLAN without a VLAN ID. You can add only access ports to this VLAN:

```
set vmhost vlans vlan-name vlan-id none
```

- Create a VLAN with a VLAN ID:

```
set vmhost vlans vlan-name vlan-id vlan-id
```

- Create a VLAN using a list of VLAN IDs:

```
set vmhost vlans vlan-name vlan-id-list vlan-id range | comma-separated list
```

2. Attach an interface on the VNF to the VLAN:

```
set virtual-network-functions vnf-name interfaces ethx mapping vlan mode [access|trunk]
set virtual-network-functions vnf-name interfaces ethx mapping vlan members list
```

3. Attach a native VLAN ID to the VNF interface:

```
set virtual-network-functions vnf-name interfaces ethx mapping vlan native-vlan-id vlan-id
```

Configuring Service Chaining Using DHCP Services on VLANs

Using DHCP services, you need not manually configure the IP addresses on the VNF interfaces to achieve service-chaining. Enable DHCP clients on the glue bridge interfaces within the VNF for an IP address to be assigned from the DHCP pool.

To configure service chaining:

1. Create a VLAN with a VLAN ID `none`.

```
user@host# set vmhost vlans vlan-name vlan-id none
```

> **NOTE**: To use the DHCP pooling feature, the VLAN ID must be set to `none`.

2. Specify the IP address pool to be used:

```
user@host# set access address-assignment pool p4 family inet network network-address
user@host# set access address-assignment pool p4 family inet range r4 low start-IP-address
user@host# set access address-assignment pool p4 family inet range r4 high end-IP-address
```

3. Attach an interface from FPC1 to the VLAN:

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

4. Configure an IP address on the interface and enable dhcp-server on it.

```
user@host# set interface ge-1/0/x unit 0 family inet address address/prefix-length
user@host# set system services dhcp-local-server group grp1 interface ge-1/0/x
```

5. Attach an interface on the VNF to the VLAN to complete the service chain:

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

6. Enable the DHCP client on the VNF.

To check the assigned IP address, use the show system visibility vnf command.

Example: Configuring Service Chaining Using VLANs on NFX150 Network Services Platform

IN THIS SECTION

- Requirements | 169
- Overview | 169
- Configuration | 170

This example shows how to configure service chaining using VLANs on the host bridge.

Requirements

This example uses the following hardware and software components:

- NFX150 running Junos OS Release 18.1R1

Before you configure service chaining, be sure you have:

- Installed and launched the relevant VNFs, assigned the corresponding interfaces, and configured the resources.

Overview

IN THIS SECTION

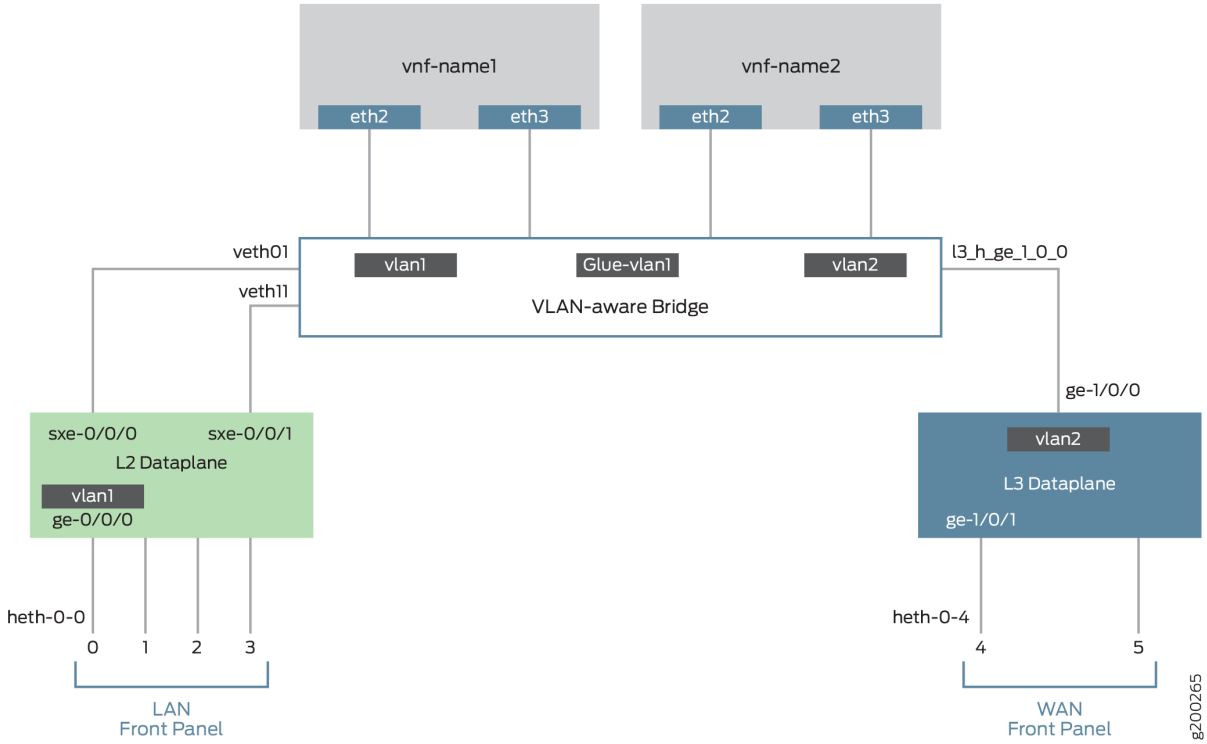
- Topology | 169

Service chaining on a device running the disaggregated Junos OS allows multiple services, or virtual network functions (VNFs), to be applied to traffic as it 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 device running the disaggregated Junos OS, as shown in Figure 13 on page 170.

Figure 13: Service Chaining Using VLANs



The key configuration elements include:

- The front panel ports.
- The internal-facing ports.
- The VM interfaces. VNF interfaces must use the format eth#, where #is from 0 through 9.
- VLANs, to provide bridging between the sxe and VM interfaces.

Configuration

IN THIS SECTION

- Configuring the Interfaces | **171**
- Configuring the VNF Interfaces and Creating the Service Chain | **174**

Configuring the Interfaces

Step-by-Step Procedure

To configure the interfaces:

1. Log in to the CLI:

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

2. Map the physical (heth) interfaces to the virtual (ge) interfaces.

```
user@host# set vmhost virtualization-options interfaces ge-0/0/0 mapping interface heth-0-0
user@host# set vmhost virtualization-options interfaces ge-1/0/1 mapping interface heth-0-4
```

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 could be a trunk port if appropriate.

```
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 front panel port with vlan-tagging and 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 203.0.113.2/30
```

7. Configure the WAN side internal-facing 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 192.0.3.1/24
```

8. Commit the configuration.

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

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 192.0.3.1/24;
    }
}
```

```
[edit]
user@host# show interfaces ge-1/0/1
vlan-tagging;
unit 0 {
    vlan-id 1178;
    family inet {
          address 203.0.113.2/30;
    }
}
```

```
[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 vlans
default {
    vlan-id 1;
}
vlan1 {
    vlan-id 77;
}
```

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 the vmhost instance with either with LAN, WAN, or glue-vlan 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-vlan vlan-id 123
```

2. Bring up the VNF (vnf-name1) with one virtio interface mapped to VLAN, and another interface mapped to glue-vlan.

```
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-
vlan
```

3. Similarly bring up the second VNF (vnf-name2) with one interface mapped to VLAN2, and the second interface mapped to the same glue-vlan.

```
user@host# set virtual-network-functions vnf-name2 interfaces eth2 mapping vlan members glue-
vlan
user@host# set virtual-network-functions vnf-name2 interfaces eth3 mapping vlan members vlan2
```

4. Finally, configure the IP addresses and static routes for each interface of the VNFs as shown in Figure 13 on page 170.

RELATED DOCUMENTATION

Understanding Service Chaining on Disaggregated Junos OS Platforms

Disaggregated Junos OS VMs

Virtio and SR-IOV Usage

Example: Configuring Service Chaining Using SR-IOV on NFX150 Network Services Platform

IN THIS SECTION

- Requirements | 175
- Overview | 175
- Configuration | 177

This example shows how to configure service chaining using SR-IOV on NFX150 platforms.

Requirements

This example uses the following hardware and software components:

- NFX150 running Junos OS Release 18.1R1

Before you configure service chaining, be sure you have:

- Installed and launched the relevant VNFs

Overview

IN THIS SECTION

- Topology | 176

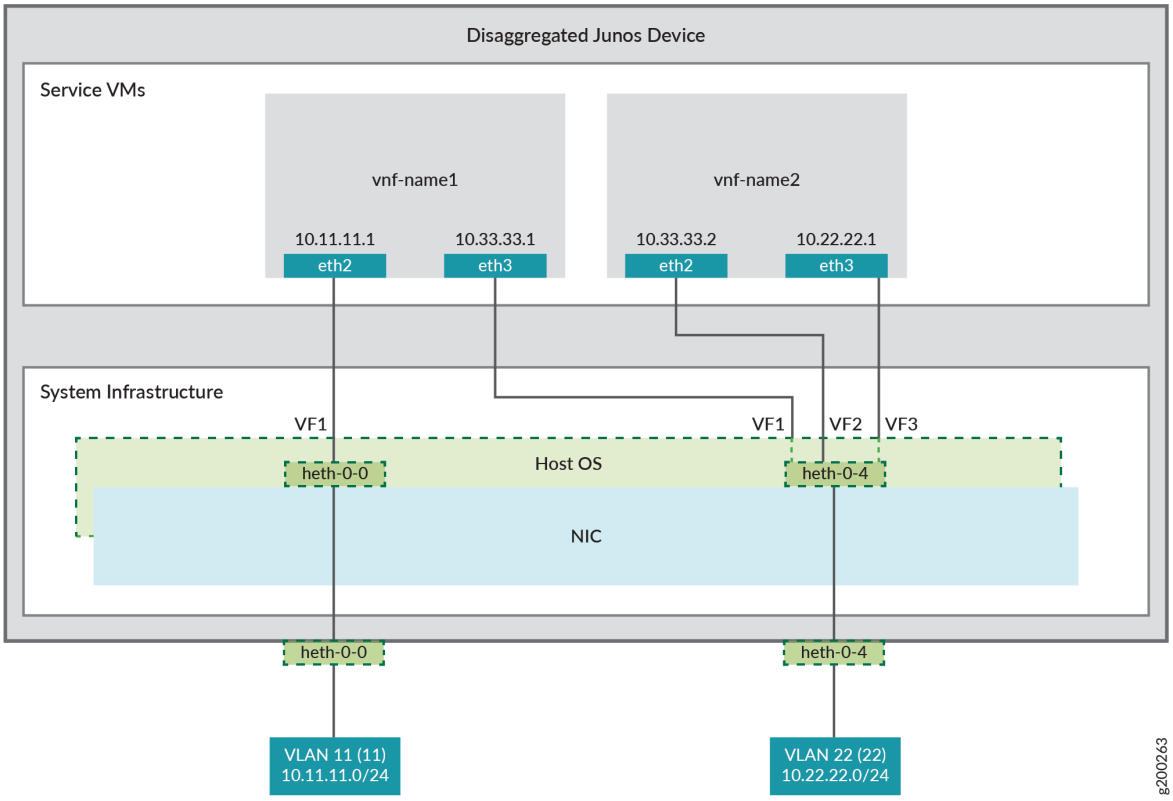
Service chaining on a device running the disaggregated Junos OS allows multiple services, or virtual network functions (VNFs), to be applied to traffic as it flows through the device. This example explains

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 device running the disaggregated Junos OS, as shown in Figure 14 on page 176.

Figure 14: Service Chaining Using SR-IOV—Device Infrastructure



This example uses the front panel ports heth-0-0 and heth-0-4. The VMs use two interfaces each, eth2 and eth3.

These elements are generally separated into two parts: a *LAN side* and a *WAN side*.

As this example uses SR-IOV, the NIC ports' virtual functions (VFs) are used to bypass the host OS and provide direct NIC-to-VM connectivity.

The key configuration elements include:

- The front panel ports, heth-0-0 and heth-0-4.

- The VNF interfaces. VNF interfaces must use the format eth#, where # is from 0 through 9.
- The virtual function setting, to indicate SR-IOV is being used to provide direct access between the sxe and VNF interfaces.

Configuration

IN THIS SECTION

- Creating the Service Chain | 177

Creating the Service Chain

Step-by-Step Procedure

To configure the VNF interfaces and create the service chain:

1. Configure *vnf-name1'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. VNF must use the interfaces from eth0 through eth9.

   The heth interface is the configurable representation of the related NIC interface.

   ```
   user@host> configure
   [edit]
   user@host# set virtual-network-functions vnf-name1 interfaces eth2 mapping heth-0-0 virtual-
   function
   ```

2. Configure the *vnf-name1's* WAN-side interface from the eth3 VNF interface as shown in Figure 14 on page 176.

   ```
   user@host# set virtual-network-functions vnf-name1 interfaces eth3 mapping heth-0-4 virtual-
   function
   ```

3. Similarly bring up *vnf-name2* with both interfaces eth2 and eth3 on heth-0-4.

```
user@host# set virtual-network-functions vnf-name2 interfaces eth2 mapping heth-0-4 virtual-function
user@host# set virtual-network-functions vnf-name2 interfaces eth3 mapping heth-0-4 virtual-function
```

4. Finally, configure the IP addresses and static routes for each interface of the VNFs, and add routes to achieve the 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

IN THIS SECTION

- Requirements | 179
- Overview | 179
- Configuration | 180
- Verifying the Configuration | 183

This example shows how to configure service chaining using a custom bridge.

Requirements

This example uses the following hardware and software components:

- NFX150 running Junos OS Release 18.1R1

Overview

IN THIS SECTION

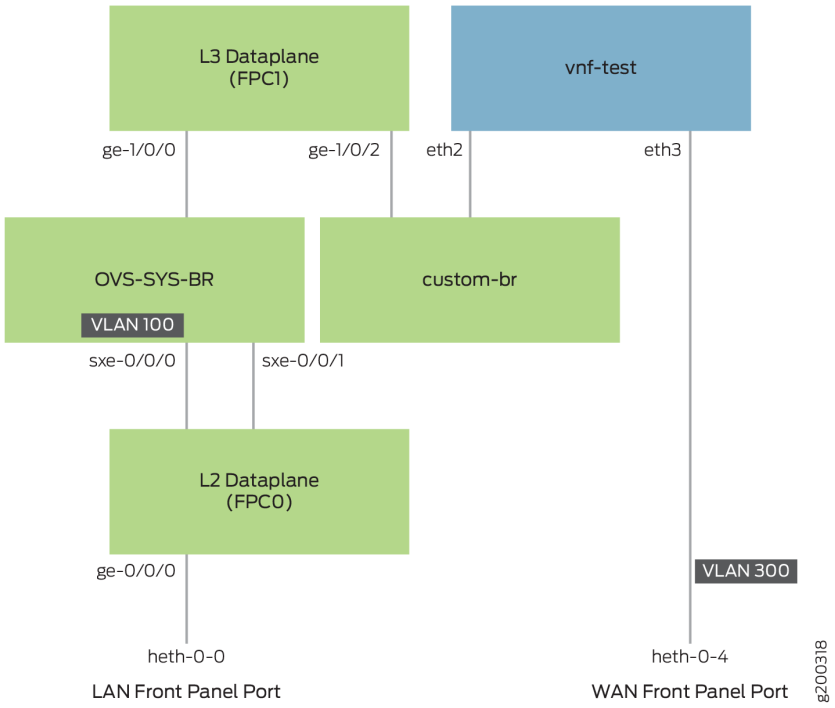
- Topology | 179

The default system bridge is OVS. The OVS bridge is a VLAN-aware system bridge, which acts as the 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 15 on page 180.

Figure 15: Service Chaining Using a Custom Bridge



Configuration

IN THIS SECTION

- Create VLANs and the Custom Bridge | **181**
- Map the Interfaces | **181**
- Configure the Layer 2 Datapath | **181**
- Configure the Layer 3 Datapath | **182**
- Configure the VNF | **182**

Create VLANs and the Custom Bridge

Step-by-Step Procedure

1. Configure VLANs for the LAN-side interfaces.

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

2. Create a custom bridge:

```
user@host# set vmhost vlans custom-br vlan-id none
```

Map the Interfaces

Step-by-Step Procedure

1. Map the heth-0-0 physical port to the FPC0 interface.

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

2. Map the FPC1 interface ge-1/0/2 to the custom bridge.

```
user@host# set vmhost virtualization-options interfaces ge-1/0/2 mapping vlan custom-br
```

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
```

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
```

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 192.168.2.1/24
```

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

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

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

3. Pin a virtual CPU to a physical CPU:

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

4. Create a VNF interface on the custom OVS bridge:

```
user@host# set virtual-network-functions vnf-test interfaces eth2 mapping vlan members custom-
br
```

5. Attach a VNF interface to a physical interface by using the SR-IOV virtual function:

```
user@host# set virtual-network-functions vnf-test interfaces eth3 mapping interface heth-0-4
virtual-function vlan-id 300
```

6. Specify the memory allocation for the VNF:

```
user@host# set virtual-network-functions vnf-test 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 | 184
- Verify the Data Plane Configuration | 185

Verify the Control Plane Configuration

Purpose

Verify the control plane configuration:

Action

To verify the control plane configuration:

- 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                  custom-br
                                                           ge-1/0/2.0
                                                           vnf-test_eth2.0
```

- Verify that the VNF is operational. View the status of the VNF to ensure that the VNF is up and running.

```
user@host> show virtual-network-functions vnf-test
ID       Name                                              State      Liveliness
--------------------------------------------------------------------------------
1        vjunos0                                           Running    alive
2        vnf-test                                          Running    alive
```

The `Liveliness` output field of the VNF indicates whether the IP address of the VNF is reachable or not reachable from Junos.

To view more details of the VNF:

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

Id:                 2
Name:               vnf-test
State:              Running
Liveliness:         alive
IP Address:         192.0.2.100
```

```
VCPUs:              1
Maximum Memory:     1048576 KiB
Used Memory:        1087795 KiB
Used 1G Hugepages:  0
Used 2M Hugepages:  0
Error:              None
```

Verify the Data Plane Configuration

Purpose

Verify the data plane configuration.

Action

To verify the data plane configuration:

- Verify the status of the physical ports.

```
user@host> show interfaces heth-0-0 statistics
Physical interface: heth-0-0, 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:00:53:8d, Hardware address: 00:00:5e:00:53:8d
    Input packets : 1522
    Output packets: 1466
  MAC statistics:
        Input bytes: 161164, Input packets: 1522, Output bytes: 155312, Output packets: 1466
  VF statistics:
    VF Number: 0, PCI Address: 0000:06:10:1, Mapped to: ge-0/0/0
      Input bytes: 161164, Input packets: 1522, Output bytes: 155312, Output packets: 1466,
Multicast packets: 4
    VF Number: 1, PCI Address: 0000:06:10:5, Mapped to: ge-0/0/0
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
    VF Number: 2, PCI Address: 0000:06:11:1, Mapped to: ge-0/0/0
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
    VF Number: 3, PCI Address: 0000:06:11:5, Mapped to: ge-0/0/0
```

```
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
```

- 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: 169, SNMP ifIndex: 521
  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   : 2020-07-08 09:22:06 UTC (20:46:40 ago)
  Statistics last cleared: Never
  Input rate     : 792 bps (0 pps)
  Output rate    : 792 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 : 1608
    Output packets: 1552
    Protocol eth-switch, MTU: 9192
      Flags: Is-Primary, Trunk-Mode
```

Example: Configuring Service Chaining for LAN-WAN Routing

IN THIS SECTION

- Requirements | 187
- Overview | 188
- Configuration | 189
- Verification | 190

This example shows how to configure service chaining for LAN-WAN routing.

Requirements

This example uses the following hardware and software components:

- NFX150 running Junos OS Release 18.1R1

Overview

IN THIS SECTION

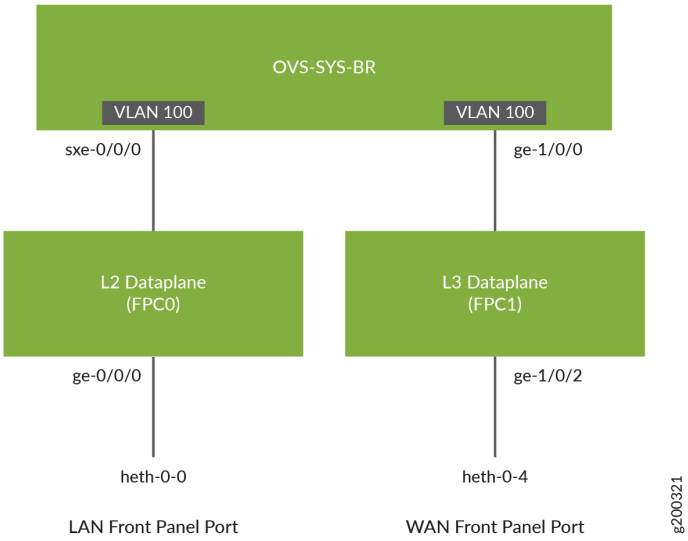
- Topology | 188

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 188.

Figure 16: Service Chaining Using a Custom Bridge



Configuration

IN THIS SECTION

- Map the Interfaces | 189
- Configure the Layer 2 Datapath | 189
- Configure the Layer 3 Datapath | 190

Map the Interfaces

Step-by-Step Procedure

1. Map the heth-0-0 physical port to the FPC0 interface.

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

2. Map the FPC1 interface ge-1/0/2 to the physical port heth-0-4.

```
user@host# set vmhost virtualization-options interfaces ge-1/0/2 mapping interface heth-0-4
```

Configure the Layer 2 Datapath

Step-by-Step Procedure

1. Configure VLANs for the LAN-side interfaces.

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

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
```

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
```

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 192.0.3.1/24
```

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 203.0.113.2/30
```

Verification

IN THIS SECTION

- Verifying the Status of the Interfaces | 191

Verifying the Status of the Interfaces

Purpose

Verify the status of the Layer 2 and Layer 3 interfaces.

Action

To verify the status of the interfaces:

- Verify the status of the physical ports.

```
user@host> show interfaces heth-0-0 statistics
Physical interface: heth-0-0, 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:00:53:8d, Hardware address: 00:00:5e:00:53:8d
    Input packets : 272469
    Output packets: 674
  MAC statistics:
    Input bytes: 17438016, Input packets: 272469, Output bytes: 48658, Output packets: 674
  VF statistics:
    VF Number: 0, PCI Address: 0000:02:10:1, Mapped to: ge-0/0/0
      Input bytes: 17433984, Input packets: 272406, Output bytes: 48658, Output packets: 674,
Multicast packets: 272406
    VF Number: 1, PCI Address: 0000:02:10:5, Mapped to: ge-0/0/0
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
    VF Number: 2, PCI Address: 0000:02:11:1, Mapped to: ge-0/0/0
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
    VF Number: 3, PCI Address: 0000:02:11:5, Mapped to: ge-0/0/0
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
```

- 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 Cross Connect on NFX150 Devices

IN THIS SECTION

- Requirements | 193
- Overview | 193
- Configuration | 195
- Verifying the Configuration | 198

This example shows how to configure cross-connect on NFX150 devices.

Requirements

This example uses the following hardware and software components:

- NFX150 running Junos OS Release 18.1R1

Overview

IN THIS SECTION

- Topology | 194

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.

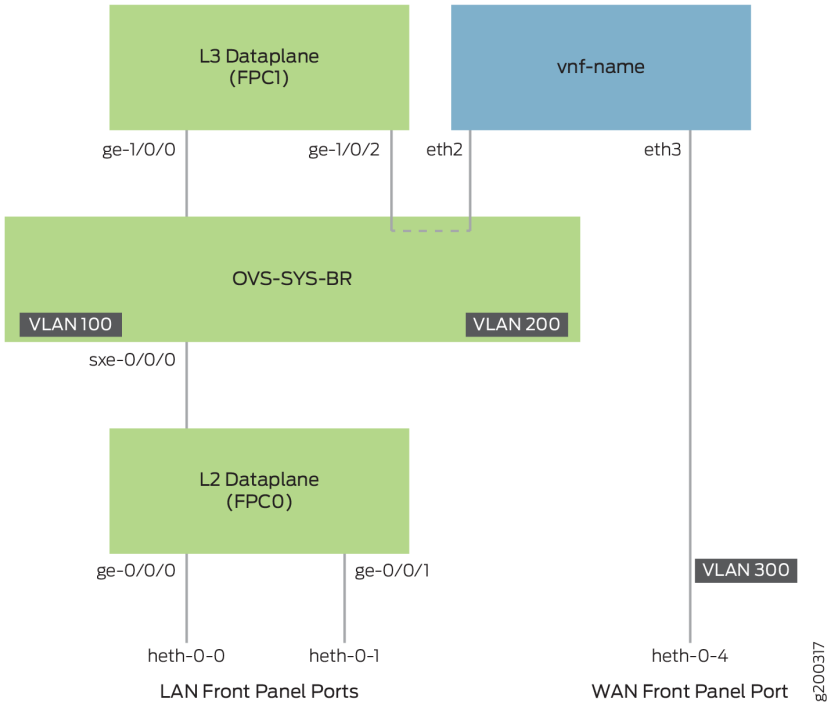
The **Cross-connect** feature supports the following:

- 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 network traffic flow from trunk to access port through POP operation.
  - Supports network traffic flow from access to trunk port through PUSH operation.
  - Supports VLAN PUSH, POP, and SWAP operations.

Topology

This example uses the topology shown in Figure 17 on page 194.

Figure 17: Configuring Cross-Connect



Configuration

IN THIS SECTION

- Create VLANs | 195
- Map the Interfaces | 195
- Configure the Layer 2 Datapath | 196
- Configure the Layer 3 Datapath | 196
- Configure the VNF | 197
- Configure Cross-Connect | 198

Create VLANs

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
```

Map the Interfaces

Step-by-Step Procedure

1. Map the heth-0-0 physical port to the FPC0 interface.

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

2. Map the heth-0-1 physical port to the FPC0 interface.

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

3. Map the FPC1 interface ge-1/0/2 to the system bridge OVS.

```
user@host# set vmhost virtualization-options interfaces ge-1/0/2
```

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
```

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
```

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 192.0.3.1/24
```

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-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. Create a host VLAN:

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

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
```

6. Attach a VNF interface to a physical interface by using the SR-IOV virtual function:

```
user@host# set virtual-network-functions vnf-name interfaces eth3 mapping interface heth-0-4
virtual-function vlan-id 300
```

7. Specify the memory allocation for the VNF:

```
user@host# set virtual-network-functions 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
```

Verifying the Configuration

IN THIS SECTION

- Verify the Control Plane Configuration | 198
- Verify the Data Plane Configuration | 199

Verify the Control Plane Configuration

Purpose

Verify the control plane configuration:

Action

To verify the control plane configuration:

- 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                  custom-br
                                                           ge-1/0/2.0
```

```
                                                                vnf-name_eth2.0
vmhost                  vlan200               200
```

- Verify that the VNF is operational. View the status of the VNF to ensure that the VNF is up and running.

```
user@host# show virtual-network-functions vnf-name
ID       Name                                                  State      Liveliness
--------------------------------------------------------------------------------
2        vnf-name                                               Running    alive
```

The `Liveliness` output field of the VNF indicates whether the IP address of the VNF is reachable or not reachable from Junos.

To view more details of the VNF:

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

Id:                 2
Name:               vnf-name
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
```

Verify the Data Plane Configuration

Purpose

Verify the data plane configuration.

Action

To verify the data plane configuration:

- Verify the status of the physical ports.

```
user@host> show interfaces heth-0-0 statistics
Physical interface: heth-0-0, 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:00:53:8d, Hardware address: 00:00:5e:00:53:8d
    Input packets : 311143
    Output packets: 674
  MAC statistics:
    Input bytes: 19913152, Input packets: 311143, Output bytes: 48658, Output packets: 674
  VF statistics:
    VF Number: 0, PCI Address: 0000:02:10:1, Mapped to: ge-0/0/0
      Input bytes: 19909120, Input packets: 311080, Output bytes: 48658, Output packets: 674,
Multicast packets: 311080
    VF Number: 1, PCI Address: 0000:02:10:5, Mapped to: ge-0/0/0
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
    VF Number: 2, PCI Address: 0000:02:11:1, Mapped to: ge-0/0/0
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
    VF Number: 3, PCI Address: 0000:02:11:5, Mapped to: ge-0/0/0
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
```

- 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 (6d 00:28 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 : 311115
    Output packets: 22
    Protocol eth-switch, MTU: 9192
      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: 158, SNMP ifIndex: 536
  Link-level type: Ethernet, MTU: 1518, LAN-PHY mode, Link-mode: Full-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: 00:00:5e:00:53:5d, Hardware address: 00:00:5e:00:53:5d
  Last flapped   : 2018-04-23 06:03:29 UTC (1d 00:04 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-1/0/2.0 (Index 342) (SNMP ifIndex 538)
    Flags: Up SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.200 ]  Encapsulation: ENET2
    Input packets : 0
    Output packets: 0
    Security: Zone: untrust
    Allowed host-inbound traffic : dns dhcp tftp https
    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
    Protocol multiservice, MTU: Unlimited

  Logical interface ge-1/0/2.32767 (Index 343) (SNMP ifIndex 545)
    Flags: Up SNMP-Traps 0x4004000 VLAN-Tag [ 0x0000.0 ]  Encapsulation: ENET2
    Input packets : 0
    Output packets: 0
    Security: Zone: Null
```

```
    Protocol multiservice, MTU: Unlimited
      Flags: None
```

RELATED DOCUMENTATION

Example: Configuring Cross-Connect Using a Custom Bridge on NFX150 Devices | 207

Example: Configuring Service Chaining for LAN Routing

IN THIS SECTION

- Requirements | 203
- Overview | 204
- Configuration | 205

This example shows how to configure service chaining for LAN routing.

Requirements

This example uses the following hardware and software components:

- NFX150 running Junos OS Release 18.1R1

Overview

IN THIS SECTION

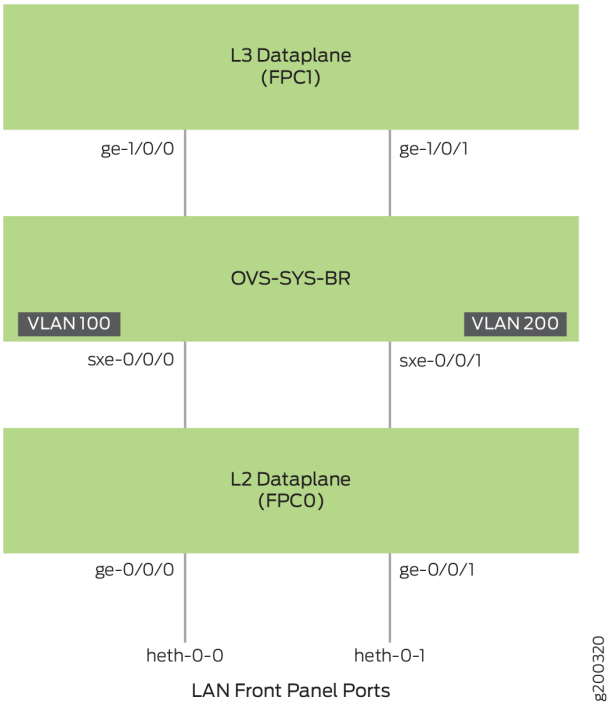
- Topology | 204

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 18 on page 204.

Figure 18: Service Chaining for LAN Routing



Configuration

IN THIS SECTION

- Map the Interfaces | 205
- Configure the Layer 2 Datapath | 205
- Configure the Layer 3 Datapath | 206

Map the Interfaces

Step-by-Step Procedure

1. Map the heth-0-0 physical port to the FPC0 interface.

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

2. Map the heth-0-1 physical port to the FPC0 interface.

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

Configure 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@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
```

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
```

Configure the Layer 3 Datapath

Step-by-Step Procedure

1. Map the ge-1/0/1 interface to the NFV backplane:

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

2. 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 192.0.3.1/24
```

3. 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 203.0.113.2/30
```

RELATED DOCUMENTATION

Example: Configuring Service Chaining for LAN-WAN Routing | 187

Example: Configuring Cross-Connect Using a Custom Bridge on NFX150 Devices

IN THIS SECTION

- Requirements | 207
- Overview | 207
- Configuration | 209
- Verifying the Configuration | 212

This example shows how to configure cross-connect using a custom bridge on NFX150 devices.

Requirements

This example uses the following hardware and software components:

- NFX150 running Junos OS Release 18.1R1

Overview

IN THIS SECTION

- Topology | 208

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.

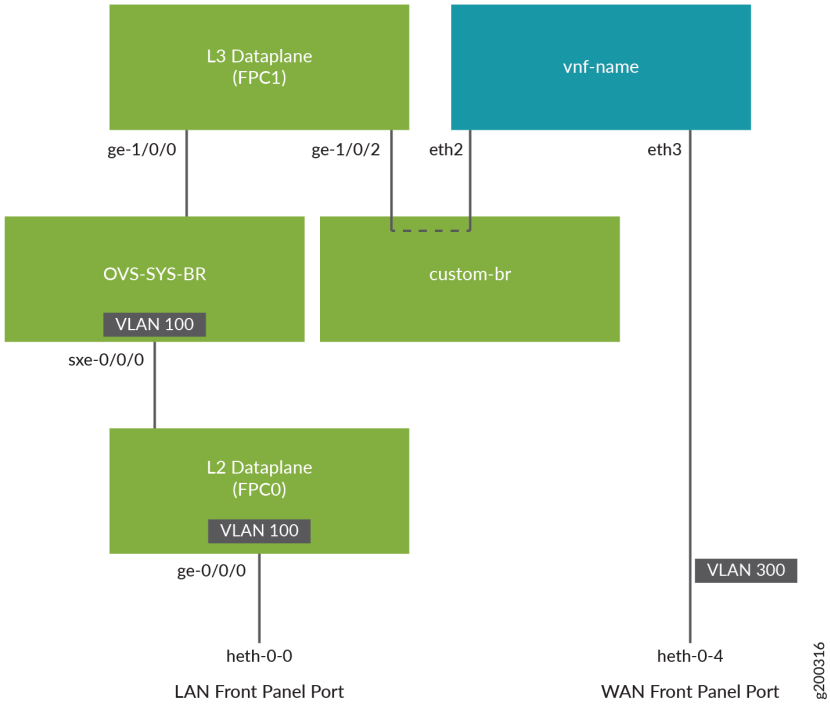
The **Cross-connect** feature supports the following:

- 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 network traffic flow from trunk to access port through POP operation.
  - Supports network traffic flow from access to trunk port through PUSH operation.
  - Supports VLAN PUSH, POP, and SWAP operations.

Topology

This example uses the topology shown in Figure 19 on page 208.

Figure 19: Configuring Cross-Connect



Configuration

IN THIS SECTION

- Create VLANs | 209
- Map the Interfaces | 209
- Configure the Layer 2 Datapath | 210
- Configure the Layer 3 Datapath | 210
- Configure the VNF | 211
- Configure Cross-Connect | 211

Create VLANs

Step-by-Step Procedure

1. Configure VLANs for the LAN-side interfaces.

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

Map the Interfaces

Step-by-Step Procedure

1. Map the heth-0-0 physical port to the FPC0 interface.

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

2. Create the custom bridge.

```
user@host# set vmhost vlans custom-br vlan-id none
```

3. Map the FPC1 interface ge-1/0/2 to the custom bridge.

```
user@host# set vmhost virtualization-options interfaces ge-1/0/2 mapping vlan custom-br
```

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
```

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
```

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 192.0.3.1/24
```

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-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. Create a VNF interface on the custom OVS bridge:

```
user@host# set virtual-network-functions vnf-name interfaces eth2 mapping vlan members custom-
br
```

5. Attach a VNF interface to a physical interface by using the SR-IOV virtual function:

```
user@host# set virtual-network-functions vnf-name interfaces eth3 mapping interface heth-0-4
virtual-function vlan-id 300
```

6. Specify the memory allocation for the VNF:

```
user@host# set virtual-network-functions 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
```

Verifying the Configuration

IN THIS SECTION

- Verify the Control Plane Configuration | 212
- Verify the Data Plane Configuration | 213

Verify the Control Plane Configuration

Purpose

Verify the control plane configuration:

Action

To verify the control plane configuration:

- 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                  custom-br
                                                           ge-1/0/2.0
```

- Verify that the VNF is operational. View the status of the VNF to ensure that the VNF is up and running.

```
user@host# show virtual-network-functions vnf-name
ID       Name                                          State      Liveliness
--------------------------------------------------------------------------------
2        vnf-name                                       Running    alive
```

The `Liveliness` output field of the VNF indicates whether the IP address of the VNF is reachable or not reachable from Junos.

To view more details of the VNF:

```
user@host# show virtual-network-functions VNF detail
Virtual Network Function Information
------------------------------------

Id:                 2
Name:               vnf-name
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
```

Verify the Data Plane Configuration

Purpose

Verify the data plane configuration.

Action

To verify the data plane configuration:

- Verify the status of the physical ports.

```
user@host> show interfaces heth-0-0 statistics
Physical interface: heth-0-0, 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:00:53:8d, Hardware address: 00:00:5e:00:53:8d
    Input packets : 311143
    Output packets: 674
  MAC statistics:
    Input bytes: 19913152, Input packets: 311143, Output bytes: 48658, Output packets: 674
  VF statistics:
    VF Number: 0, PCI Address: 0000:02:10:1, Mapped to: ge-0/0/0
      Input bytes: 19909120, Input packets: 311080, Output bytes: 48658, Output packets: 674,
Multicast packets: 311080
    VF Number: 1, PCI Address: 0000:02:10:5, Mapped to: ge-0/0/0
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
    VF Number: 2, PCI Address: 0000:02:11:1, Mapped to: ge-0/0/0
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
    VF Number: 3, PCI Address: 0000:02:11:5, Mapped to: ge-0/0/0
      Input bytes: 0, Input packets: 0, Output bytes: 0, Output packets: 0, Multicast
packets: 0
```

- 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 (6d 00:28 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 : 311115
    Output packets: 22
    Protocol eth-switch, MTU: 9192
      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: 158, SNMP ifIndex: 536
  Link-level type: Ethernet, MTU: 1518, LAN-PHY mode, Link-mode: Full-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: 00:00:5e:00:53:5d, Hardware address: 00:00:5e:00:53:5d
  Last flapped   : 2018-04-23 06:03:29 UTC (1d 00:04 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-1/0/2.0 (Index 342) (SNMP ifIndex 538)
    Flags: Up SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.200 ]  Encapsulation: ENET2
    Input packets : 0
    Output packets: 0
    Security: Zone: untrust
    Allowed host-inbound traffic : dns dhcp tftp https
    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
    Protocol multiservice, MTU: Unlimited

  Logical interface ge-1/0/2.32767 (Index 343) (SNMP ifIndex 545)
    Flags: Up SNMP-Traps 0x4004000 VLAN-Tag [ 0x0000.0 ]  Encapsulation: ENET2
    Input packets : 0
    Output packets: 0
    Security: Zone: Null
    Protocol multiservice, MTU: Unlimited
      Flags: None
```

RELATED DOCUMENTATION

Example: Configuring Cross Connect on NFX150 Devices | **193**

11

CHAPTER

Monitoring and Troubleshooting

Configuring SNMP on NFX150, NFX250 NextGen, and NFX350 Devices | 219

Monitor NFX150 Device Health | 228

Recovering the Root Password for NFX150, NFX250 NextGen, and NFX350 Devices | 240

Troubleshooting Interfaces on NFX Devices | 244

Configuring SNMP on NFX150, NFX250 NextGen, and NFX350 Devices

IN THIS SECTION

- How to Configure SNMPv2c to Access Libvirt MIB Data | 219
- How to Configure SNMPv3 to Access Libvirt MIB Data | 221
- How to Query Libvirt MIB Data | 223
- Supported Chassis MIBs and Traps | 225
- Supported libvirt MIB Traps | 226

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.

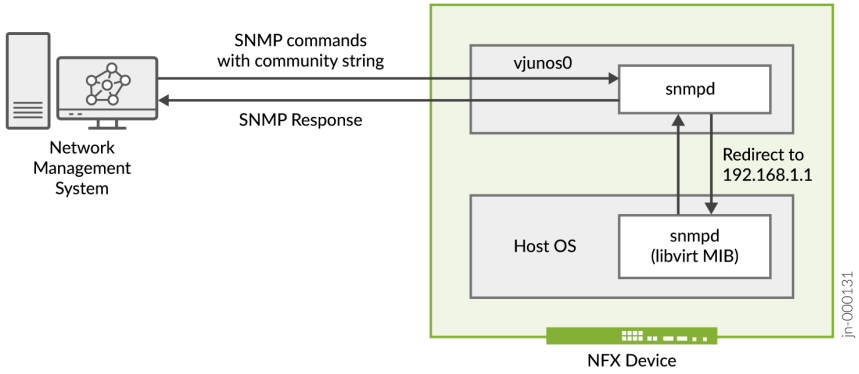
Starting in Junos OS Release 21.4R1, NFX Series devices support LM-SENSORS-MIB, ENTITY-SENSORS-MIB, and libvirt MIB. The LM-SENSORS-MIB and ENTITY-SENSORS-MIB data is available on vjunos0 whereas the libvirt MIB data is available on the Host OS. You can use the libvirt MIB to monitor virtual machines. 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 20 on page 220 illustrates the communication flow for SNMPv2c on NFX Series devices.

Figure 20: 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
```

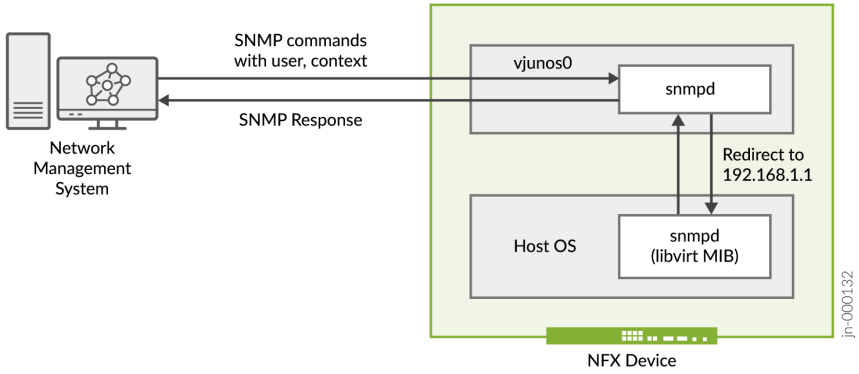
To enable traps, see "How to Enable libvirt SNMPv2c Trap Support" on page 227.

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 21 on page 221 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 21: 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 228.

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

How to Enable libvirt SNMPv2c Trap Support | 227
How to Enable libvirt SNMPv3 Trap Support | 228

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
```

Monitor NFX150 Device Health

IN THIS SECTION

Sample Output—show system visibility host | 230

- Sample Output—show system visibility jdm | 233
- Sample Output—show system visibility cpu | 234
- Sample Output—show system visibility memory | 235
- Sample Output—show vmhost mode | 236
- Sample Output—show chassis fpc pic-status | 237
- Sample Output—show chassis fpc detail | 237
- Sample Output—show chassis fpc errors | 238

You can monitor the health of an NFX150 device by using the following commands:

Command	Description
`show system visibility host`	Displays status of the host CPU, interfaces, and port statistics. "Sample Output—show system visibility host" on page 230
`show system visibility jdm`	Displays the status of the JDM container. "Sample Output—show system visibility jdm" on page 233
`show system visibility cpu`	Displays the CPU usage and allocation information. "Sample Output—show system visibility cpu" on page 234
`show system visibility memory`	Displays details of memory allocation and hugepages. "Sample Output—show system visibility memory" on page 235
`show vmhost mode`	Displays the current operating mode, and CPU and memory allocations for the mode. "Sample Output—show vmhost mode" on page 236
`show chassis fpc pic-status` *`fpc-slot`*	Displays the status of the FPCs. "Sample Output—show chassis fpc pic-status" on page 237

(Continued)

Command	Description
`show chassis fpc detail`	Displays the details, such as CPU DRAM, uptime, for each FPC. "Sample Output—show chassis fpc detail" on page 237
`show chassis fpc errors`	Displays the errors for the FPC components. "Sample Output—show chassis fpc errors" on page 238

Sample Output—show system visibility host

```
root@host> show system visibility host
Host Uptime
-----------
Uptime: 4 days 5:36:09.17000

Host Tasks
----------
Total:    185
Running:  2
Sleeping: 180
Stopped:  0
Zombie:   3

Host CPU Information (Time in sec)
----------------------------------
User Time:              114556
System Time:            74061
Idle Time:              1196820
I/O Wait Time:          596
Nice Time:              0
Interrupt Service Time: 0

Host Disk Partitions
-----------------------------------------------------------------------------------------------
---------
```

```
Device        Mount Point                                   File System  Options
------------- ------------------------------------------- -----------
----------------------------------
/dev/sda1     /boot/efi                                     vfat
rw,noatime,fmask=0022,dmask=0022,codepage=437,iocharset=iso8859-1,shortname=mixed,errors=remount-
ro
/dev/sda7     /config                                       ext4         rw,noatime,data=ordered
/dev/sda8     /var/log                                      ext4         rw,noatime,data=ordered
/dev/sda9     /mnt/.share                                   ext4
rw,noatime,discard,data=ordered
/dev/loop1    /var/tmp                                      ext4         rw,relatime,data=ordered

Host Disk Usage Information
---------------------------
Total (MiB):     189
Used  (MiB):     30
Free  (MiB):     145
Percentage Used: 16.1

Host Disk I/O Information
-------------------------
Read Count: 293751
Write Count: 914147
Read Bytes: 3777713664
Write Bytes: 12500229120
  Read Time: 59732
Write Time: 1150226

Host Interfaces
----------------------------------------------
Interface          State     MAC
------------------ --------- -----------------
heth-0-1           inactive  00:00:5e:00:53:89
heth-0-0           inactive  00:00:5e:00:53:88
heth-0-3           inactive  00:00:5e:00:53:8b
heth-0-2           inactive  00:00:5e:00:53:8a
heth-0-5           inactive  00:00:5e:00:53:8d
heth-0-4           active    00:00:5e:00:53:8c
ctrlbr0            active    00:00:5e:00:53:10
docker0            inactive  00:00:5e:00:53:3d
eth0br             active    00:00:5e:00:53:00
l3_h_ge_1_0_0      active    00:00:5e:00:53:68
lo                 inactive  00:00:5e:00:53:00
```

```
ovs-sys-br         inactive  00:00:5e:00:53:4f
ovs-system         inactive  00:00:5e:00:53:6c
sit0               inactive  00:00:00:00
veth00             active    00:00:5e:00:53:b9
veth01             active    00:00:5e:00:53:cc
veth10             active    00:00:5e:00:53:27
veth11             active    00:00:5e:00:53:4b
virbr0             active    00:00:5e:00:53:83
virbr1             active    00:00:5e:00:53:41

Host Port Statistics
-----------------------------------------------------------------------------------------------
----
Interface Bytes Sent   Bytes Rcvd   Packets Sent Packets Rcvd Errors In Errors Out Drops In
Drops Out
--------- ------------ ------------ ------------ ------------ --------- ---------- --------
---------
l3_h_ge_1_0_0 0        9281595      0            175121       0         0          0        0
veth10    0            908          0            14           0         0          0        0
veth11    908          0            14           0            0         0          0        0
ovs-system 0           0            0            0            0         0          0        0
ovs-sys-br 0           0            0            0            0         0          6        0
vnet0     113734993    40060637     492686       444304       0         0          0        0
vnet1     1746218019   1666641076   19129819     22429555     0         0          0        0
vnet2     5760102921   231049019    37804203     1495998      0         0          0
35688791
vnet3     9525642      13090        183190       305          0         0          0        0
vnet4     564779       4916         10838        57           0         0          0        0
vnet5     1113225127   4920163      8061220      30370        0         0          0        328
eth0      237823455    9779714118   1543719      74854926     0         0          13177    0
lo        3163525      3163525      56215        56215        0         0          0        0
virbr0-nic 0           0            0            0            0         0          0        0
irb       0            0            0            0            0         0          0        0
docker0   0            0            0            0            0         0          0        0
veth01    908          0            14           0            0         0          0        0
veth00    0            908          0            14           0         0          0        0
dcapi-tap 0            0            0            0            0         0          0        0
sit0      0            0            0            0            0         0          0        0
flowd_h_mgmt 775774835 715134308    10087893     8985215      0         0          0        0
virbr1    110674581    32411759     466777       429883       0         0          0        0
virbr0    341638       633559       3351         3732         0         0          0        0
jdm-hbme1 9725715      544247       185105       2276         0         0          0        0
jdm-hbme2 1682446      3061900      15650        25937        0         0          0        0
```

```
eth0br    0            1175685054   0            17912122     0         0          0        0
ctrlbr0   1028692265   718118709    10108369     12342918     0         0          0        0
heth-0-1  0            0            0            0            0         0          0        0
heth-0-0  0            0            0            0            0         0          0        0
heth-0-3  0            0            0            0            0         0          0        0
heth-0-2  0            0            0            0            0         0          0        0
heth-0-5  0            0            0            0            0         0          0        0
heth-0-4  0            9762         0            110          0         0          0        0
```

Sample Output—show system visibility jdm

```
root@host> show system visibility jdm
JDM CPU Statistics (Time in sec)
-----------------------------
User Time:              114569
System Time:            74066
Idle Time:              1196860
I/O Wait Time:          596
Nice Time:              0
Interrupt Service Time: 0

JDM Disk Usage Information
--------------------------
Total (MiB):      15829
Used  (MiB):      4882
Free  (MiB):      10121
Percentage Used:  30.8

JDM Disk I/O Information
------------------------
Read Count: 293759
Write Count: 914196
Read Bytes: 3777840640
Write Bytes: 12500643840
  Read Time: 59733
Write Time: 1150254

JDM Memory Usage
-------------------
```

```
Maximum Memory (KiB): 1048576
Used Memory (KiB):    118096

JDM Uptime
----------
Uptime: 4 days 5:35:2

JDM Tasks
---------
Total:    11
Running:  1
Sleeping: 9
Stopped:  0
Zombie:   1

JDM Internal IP Addresses
-------------------------
Interface: bme1
Address  : 192.0.2.254
Interface: bme2
Address  : 192.168.1.254
```

Sample Output—show system visibility cpu

```
root@host> show system visibility cpu | no-more

CPU Statistics (Time in sec)
--------------------------------------------------------------------------------
CPU Id User Time System Time Idle Time Nice Time IOWait Time Intr. Service Time
------ --------- ----------- --------- --------- ----------- ------------------
0      51370     54035       236329    0         128         0
1      55214     18066       243032    0         0           0
2      3864      1223        357665    0         214         0
3      3985      688         359420    0         253         0

CPU Usages
----------------
CPU Id CPU Usage
------ ---------
```

```
0      100.0
1      27.600000000000001
2      0.0
3      2.0

CPU Pinning Information
-----------------------------------
Virtual Machine             vCPU CPU
--------------------------- ---- ---
testagent                   0    2
testagent                   1    3
vjunos0                     0    0


System Component                CPUs
------------------------------- --------
ovs-vswitchd                    0
riot                            0, 1
srxpfe                          0, 1
jdm                             0
```

Sample Output—show system visibility memory

```
root@host> show system visibility memory | no-more
Memory Information
------------------

Virtual Memory:
---------------
Total       (KiB): 7949028
Used        (KiB): 7555808
Available   (KiB): 1177716
Free        (KiB): 393220
Percent Used     : 85.2

Huge Pages:
------------
Total 1GiB Huge Pages:      2
Free 1GiB Huge Pages:       1
```

```
Configured 1GiB Huge Pages: 1
Total 2MiB Huge Pages:      529
Free 2MiB Huge Pages:       1
Configured 2MiB Huge Pages: 0

Hugepages Usage:
-----------------------------------------------------------------------------------------------
---------
Name                              Type                              Used 1G Hugepages  Used 2M
Hugepages
--------------------------------- ---------------------------------- ------------------
------------------
srxpfe                            other process                     1                  400
riot                              other process                     0                  128
```

Sample Output—show vmhost mode

This sample output shows the details for a device operating in compute mode. For information on the supported modes, see Performance Modes

```
root@host> show vmhost mode | no-more
Mode:
--------
Current Mode: compute

CPU Allocations:
Name                                Configured                              Used
-----------------------------------------------------------------------------------------------
---------------------
Junos Control Plane                 0                                       0,2
Juniper Device Manager              0                                       0
LTE                                 0                                       -
NFV Backplane Control Path          0                                       0
NFV Backplane Data Path             -                                       -
Layer 2 Control Path                0                                       0
Layer 2 Data Path                   1                                       0,1
Layer 3 Control Path                0                                       0
Layer 3 Data Path                   1                                       1
CPUs available for VNFs             2,3                                     2,3
```

```
CPUs turned off                     -                                       -

Memory Allocations:
Name                                Configured                              Used
-----------------------------------------------------------------------------------------------
---------------------
Junos Control Plane (mB)            2048                                    1995
NFV Backplane 1G hugepages          -                                       0
NFV Backplane 2M hugepages          -                                       0
Layer 2 1G hugepages                -                                       0
Layer 2 2M hugepages                128                                     128
Layer 3 1G hugepages                1                                       1
Layer 3 2M hugepages                401                                     400
```

Sample Output—show chassis fpc pic-status

```
root@host> show chassis fpc pic-status 1
Slot 1   Online       FPC
  PIC 0  Online       VSRX DPDK GE
```

```
root@host> show chassis fpc pic-status 0
Slot 0   Online       Virtual FPC
  PIC 0  Online       Virtual
```

Sample Output—show chassis fpc detail

```
root@porter3c-p1b-ft-04> show chassis fpc detail
Slot 0 information:
  State                                 Online
  Total CPU DRAM                  256 MB
  Total SRAM                        0 MB
  Total SDRAM                       0 MB
  Start time                            2011-07-26 00:23:36 UTC
  Uptime                                1 hour, 43 minutes, 47 seconds
```

```
Slot 1 information:
  State                                 Online
  Total CPU DRAM                    800 MB
  Total SRAM                          0 MB
  Total SDRAM                         0 MB
  Start time                            2011-07-26 00:26:16 UTC
  Uptime                                1 hour, 41 minutes, 7 seconds
```

Sample Output—show chassis fpc errors

```
root@host> show chassis fpc errors
FPC Scope    Category   Level Occurred Cleared Threshold Action-Taken Action
  0 board    functional Minor      0        0       0         0
                        Major      0        0       0         0
                        Fatal      0        0       0         0
             memory     Minor      0        0       0         0
                        Major      0        0       0         0
                        Fatal      0        0       0         0
             io         Minor      0        0       0         0
                        Major      0        0       0         0
                        Fatal      0        0       0         0
             storage    Minor      0        0       0         0
                        Major      0        0       0         0
                        Fatal      0        0       0         0
             switch     Minor      0        0       0         0
                        Major      0        0       0         0
                        Fatal      0        0       0         0
             processing Minor      0        0       0         0
                        Major      0        0       0         0
                        Fatal      0        0       0         0
             internal   Minor      0        0       0         0
                        Major      0        0       0         0
                        Fatal      0        0       0         0
    pfe      functional Minor      0        0       0         0
                        Major      0        0       0         0
                        Fatal      0        0       0         0
             memory     Minor      0        0       0         0
                        Major      0        0       0         0
                        Fatal      0        0       0         0
```

```
            io         Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            storage    Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            switch     Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            processing Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            internal   Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
  1 board   functional Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            memory     Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            io         Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            storage    Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            switch     Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            processing Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            internal   Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
    pfe     functional Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            memory     Minor      0        0       0         0
                       Major      0        0       0         0
                       Fatal      0        0       0         0
            io         Minor      0        0       0         0
```

```
                    Major     0        0       0         0
                    Fatal     0        0       0         0
          storage   Minor     0        0       0         0
                    Major     0        0       0         0
                    Fatal     0        0       0         0
          switch    Minor     0        0       0         0
                    Major     0        0       0         0
                    Fatal     0        0       0         0
          processing Minor    0        0       0         0
                    Major     0        0       0         0
                    Fatal     0        0       0         0
          internal  Minor     0        0       0         0
                    Major     0        0       0         0
                    Fatal     0        0       0         0
```

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 | 244

Monitoring Interface Status and Traffic on NFX Series Devices

IN THIS SECTION

- Purpose | 244
- Action | 244

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
```

12

CHAPTER

Configuration Statements and Operational Commands

Junos CLI Reference Overview | 250

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Learn about the syntax and options that make up the statements and commands and 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