JUNIPER NETWORKS | Engineering Simplicity

Junos OS

NextGen Port Extender User Guide

Published
2025-12-22

JUNOS

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

Junos OS NextGen Port Extender User Guide

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

CHAPTER 1

Overview

IN THIS CHAPTER

NextGen Port Extender Overview

IN THIS SECTION

Overview

The NextGen Port Extender (NGPE) is a solution for managing an aggregation network environment, in which a central aggregation device (AD) connects to multiple satellite devices (SD). The interfaces on each SD become extensions of the AD, allowing the AD to manage a large number of interfaces from a single device.

NGPE is the successor to Junos Fusion Provider Edge, and features key differences:

- NGPE uses the same Junos OS on all participating devices. Junos Fusion requires Junos OS on the AD, and a separate satellite OS for SDs.
- NGPE uses Ethernet VPN-Virtual Extensible Private LAN (EVPN-VXLAN) tunnels between the AD and SDs. Junos Fusion uses the IEEE 802.1BR standard.

We don't support NGPE and Junos Fusion configuration on the same devices since the two solutions aren't compatible.

Benefits

NGPE offers the following benefits:

- **Simplified Operations**: The AD is the single management point for easier control.
- **Simplified Initial Configuration**: NGPE utilizes Junos Node Unifier (JNU) and commit scripts to automate the initial setup. The user provides a few key parameters for JNU to configure the majority of the NGPE environment, including aggregate ethernet interfaces, underlay routing, and the EVPN-VXLAN overlay.

- **Cost Effective**: NGPE supports low speed ports, which can reduce hardware costs and improve ROI. Low speed ports refers to 1G and 10G interfaces, compared to 100G interfaces.
- **Deployment Flexibility**: Low speed connectivity options improve reachability to remote locations.
- **Satellte Provisioning**: Easy onboarding of SDs with minimal touch provisioning.

NextGen Port Extender Architecture

IN THIS SECTION

The following table gives an overview of the main NGPE components.

Table 1: NGPE Components Overview

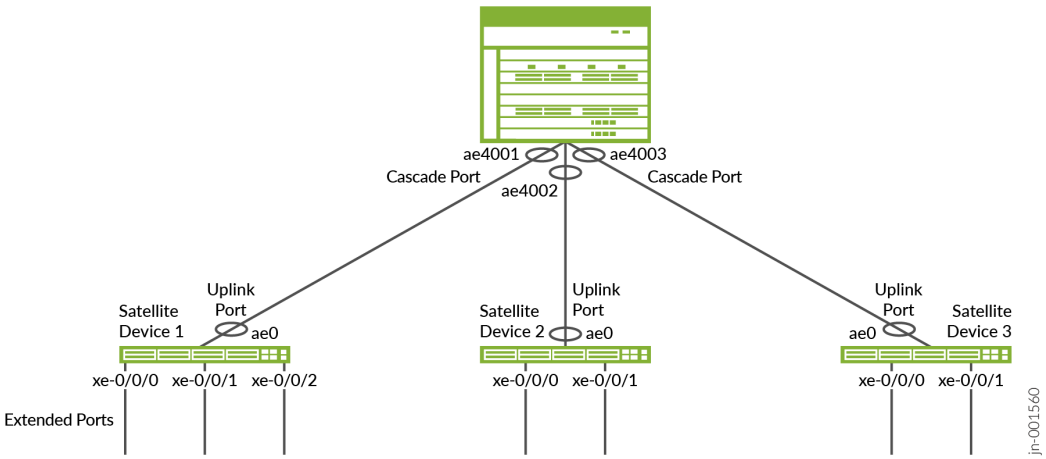
Component	Description
Aggregation Device	• Network core-facing device which manages all SDs. • Connects to every SD in the same aggregation network. • Central device for all SD configurations.
Satellite Device	• Customer-facing device which connects users with the rest of the network. • Occupies a virtual slot on the AD. • SDs are not seen independently of the AD.

Table 1: NGPE Components Overview *(Continued)*

Component	Description
Cascade Port	• Aggregate Ethernet interface (aex) on the AD connecting to the SDs uplink port. • The AD discovers each SD through cascade ports.
Uplink Port	• Aex on the SD connecting to the ADs cascade port.
Extended Port	• Customer-facing access port on the SD. • Represented on the AD as a virtual port. • Automatically generated by commit-script, based on user configuration.
JNU	• Junos Node Unifier (JNU) • Used to create and manage the NGPE topology configurations

Figure 1: NGPE Topology

The following figure is a sample topology of a simple NGPE deployment. Cascade and uplink ports are bundled as aex. Extended ports on the satellites are represented on the AD as virtual ports. The cascade port aex numbering will start with 4001, by default. The uplink port aex numbering will always be ae0, by default.



EVPN-VXLAN tunnels transport upstream traffic toward the AD, and downstream traffic toward the SD. Cascade ports and uplink ports facilitate the traffic flow between devices.

Multiple cascade and uplink ports bundled into aex are deployed between the AD and connected SDs. Bundles may also contain only a single cascade port and uplink port. Aex is required for redundancy and bandwidth assurance. NGPE utilizes high-speed ports between the AD and SDs, and low-speed ports on the SDs toward the access nodes. SDs might connect to only one AD. SD multihoming is not supported in NGPE.

Each SD is represented as a remote VXLAN tunnel endpoint (VTEP) on the AD, and each extended port is assigned a virtual network identifier (VNI). VTEP's are the points at which VXLAN tunnel traffic is either encapsulated or de-encapsulated, depending on traffic direction. The VNI is a unique, 24-bit value per L2 segment. The VNI is assigned to each extended port on a given SD, with each SD hosting multiple extended ports. Upstream traffic from an extended port is encapsulated and sent through a VXLAN tunnel to the AD. The AD de-encapsulates the data and associates it with an extended port based on the source SDs IP and VNI. Downstream traffic is handled in a similar manner.

Hardware and Software Requirements

The following table shows the hardware platforms supported for AD and SD roles. MX Series routers in the AD role must use Advanced Forwarding Toolkit (AFT) FPCs. We don't support mixing AFT FPCs and non-AFT FPCs in the same chassis when NGPE is configured.

Table 2: Supported Hardware Platforms

Supported OS Release	Supported Aggregation Devices	Supported Satellite Devices
Junos OS 25.4R1	MX304	EX4400-48P
		EX4400-48T
		QFX5120-48T
		QFX5120-48Y
	MX10004 with MX10K-LC4800 or MX10K-LC9600 FPCs	EX4400-48P
		EX4400-48T
		QFX5120-48T
		QFX5120-48Y
	MX10008 with MX10K-LC4800 or MX10K-LC9600 FPCs	EX4400-48P
		EX4400-48T
		QFX5120-48T
		QFX5120-48Y

JNU

The AD is solely responsible for configuring and maintaining every SD. The AD is also responsible for distributing the initial configuration and future updates to the SDs. During the initial launch and reboot, individual SDs should automatically install their respective schema into the AD.

Table 1 introduces JNU's role in NGPE. NGPE runs JNU software on the AD to manage SD configurations, and JNU is included with Junos OS. NETCONF over SSH is used across the AD to SD connection.

NGPE Fabric

NGPE configuration is contained within a logical system on the AD. The fabric, or overlay network, provides the transport mechanism used to carry traffic between ADs and SDs. NGPE requires that EVPN-VXLAN is configured in MAC-VRF routing instances. A single MAC-VRF instance is provisioned, using vlan-aware-bundle service.

NGPE requires separate logical interfaces for carrying management traffic and fabric traffic. Both logical interfaces are configured on the same aex member link. NGPE supports the coexistence of the fabric traffic and the management traffic over this single connection. Per-unit scheduling functionality avoids any resource conflicts, and default COS settings guarantee management traffic bandwidth under periods of congestion.

Each extended port on an SD hosts an L2 logical interface. Traffic is sent to the ADs VTEP and VNI corresponding to the VLAN. ADs don't use EVPN Type 2 route advertisements to learn MACs or IPs, which helps to prevent SDs from creating tunnels between each other. VXLAN tunnels will only operate between the AD and connected SDs. Disabling MAC/IP learning means the traffic moving through the extended ports is flooded across the VLAN associated with the L2 interface. The AD floods a single copy of the traffic to minimize forwarding disruption.

NGPE requires that all traffic is transparently transported between SDs and ADs. Layer 2 protocol tunneling (L2PT) must be configured on the SDs for this reason. Additionally, broadcast, unknown unicast, and multicast (BUM) traffic must not be throttled, as it must also be transparent.

Extended Ports Overview

Extended ports are customer-facing interfaces, and are created during satellite onboarding. The SD side of an extended port is an access port. The AD side is a virtual port. Extended ports are not core-facing. SDs are represented on the AD as a virtual slot, which allows extended port labeling to follow standard Junos nomenclature for interfaces. For example, an extended port on the AD would appear as xe-100/0/0, and on the SD as xe-0/0/0. Aex is supported for extended ports. Member links aren't required to reside on the same SD. All member links per aex must use the same interface speed.

EVPN Type 3 routes are used for tracking extended ports on the SD to reflect their status on the AD. Type 3 routes also track the status of available services on the extended ports.

Extended ports are created in the "Down" state. EVPN Type 3 advertisements signal the extended ports status to the AD, and the AD enables the extended port associated with the SDs VTEP and VNI. EVPN Type 3 routes also signal the AD when a working extended port stops transmitting. The AD updates the ports' status so alternate paths, if available, can be utilized. You configure extended ports on the AD.

Cascade and Uplink Ports Overview

Cascade ports and uplink ports are the physical, high-speed ports between the AD and SDs. Cascade ports on the AD side connect with uplink ports on the SD side. Multiple cascade ports and uplink ports are bundled into aex configuration to ensure bandwidth and redundancy. The user defines the ports during the initial configuration process, and then commit scripts provision the cascade and uplink ports as part of satellite onboarding, including the creation of the aex bundle.

Satellite Device Hardware

NGPE can support:

- Up to 10 SDs per AD.
- Each SD connects to its AD through 1 or more cascade ports.
- Each SD hosts up to 48 ports.

NGPE Commit Script and Configuration Management

NGPE uses a commit script-based mechanism for setting up the NGPE fabric. The commit script is triggered with the `port-extender` configuration statement at the `edit services` hierarchy. The port extender configuration stanza contains the fabric details and satellite connectivity information.

NGPE commit scripts are not user-configurable and should never be altered.

Configuration Management

When a commit is made at the AD, a commit check is issued to all SDs. The commit process will abort if the commit check fails. The configuration will revert to the previous state on all SDs. Conversely, if the commit check is successful, the AD sends a commit to all SDs. If the commit operation fails on any SD, the failure will be logged in the system, but the commit will still proceed successfully on the other SDs.

After a commit has been issued, the following message might appear. You can disregard it.

```
warning: Clear Sat change bits terminated abnormally
```

The AD will overwrite any existing configuration on an SD when the SD joins the NGPE network. Junos does maintain rollback versions, however you can't rollback an SD which is active in NGPE without the device being removed from the NGPE topology.

Direct configuration changes will be blocked on the SD once it has joined the AD. Only operational commands are allowed on the SD. The following example shows the error message when a user attempts to access configuration mode directly on an SD.

```
root@device:~# cli
root@device: > edit
error: unknown command: edit
Configuration changes cannot be made as the node is managed by a JNU controller. Please make the
configuration changes from the controller.
root@device: > quit
```

Operational Command Management

The AD serves as the central hub for command line interface (CLI) operations across all SDs. The ADs primary functions include distributing command requests to each SD and aggregating their responses. When a command is issued, it is forwarded to the specified SD(s), and all responses are collected sequentially in the output.

The AD should also manage different schema models running on the SDs. When an SD is added, the version and model details are provided. This process allows the schema to be installed and merged during the initial synchronization between the AD and SDs. The JNU management process (JNUD) is responsible for propagating commands from the AD to the SDs.

NextGen Port Extender Supported Features, Protocols, and Services

The following table shows the supported features, protocols, and services. Junos OS Release 25.4R1 or later is required, unless otherwise specified.

Table 3: Supported Features, Protocols, and Services

Feature/Service	Description
Class of Service	• Class of service (COS), hierarchical quality of service (HQOS), scheduling and queueing, classifiers, policers and rewrite rules are supported on the extended ports. Configuration of COS and HQOS is only supported on the AD. • Classification and rewrite support is implemented on the AD, including behavior aggregation and multifield classifers.
Connectivity Fault Management	• Support down maintenance endpoint (MEP) and up MEP continuity check messages (CCM) on extended ports and aex logical members. • Support loopback ping and link trace on non-aex and aex logical members. • MIP support for extended port non-aex and aex logical members. • ITU-T Y.1731 support for Ethernet frame delay measurement (ETH-DM) in MEF 35 mode only. • ITU-T Y.1731 support for Ethernet synthetic loss measurement (ETH-SLM) in MEF 35 mode only. • Re-anchor CFM sessions when anchor FPC goes down. • Support for CFM session distribution in enhanced CFM mode only.

Table 3: Supported Features, Protocols, and Services *(Continued)*

Feature/Service	Description
EVPN-MPLS	• Extended port as single-homed CE interface. • Extended port as CE interface in all-active or single-active ESI mode. • EVPN E-LAN over MPLS without IRB (L2 gateway). • EVPN E-LAN over MPLS with IRB (L3 gateway). • EVPN-Etree over MPLS without IRB (L2 gateway). • Extended ports don't support `family mpls`. MPLS is supported on the core-facing ports on the AD.
EVPN-VPWS	• Extended port as single-homed CE interface. • Extended port as CE interface in all-active or single-active ESI mode. • Local switching with and without single-active or all-active multihomed interface. • Egress link protection for CE-PE access link failure.
EVPN-VXLAN	Default configuration.
EVPN-VPWS Flexible Cross-Connect	• VLAN-unaware with extended port as single-homed CE interface. • VLAN-unaware with extended port as CE interface in all-active or single-active ESI mode. • VLAN-aware with extended port as single-homed CE interface. • VLAN-aware with extended port as CE interface in all-active or single-active ESI mode.

Table 3: Supported Features, Protocols, and Services *(Continued)*

Feature/Service	Description
Layer 2	• L2 bridging on extended ports. • L2VPN services on extended ports. • L2circuit services on extended ports.
Layer 3 Protocols and Services	• BFD (centralized mode only), BGP, OSPF, ISIS, and Multicast on extended ports. • Integrated routing and bridging (IRB) • L3VPN, SR-MPLS as transport, interface-based tunnels and next hop-based tunnels on extended ports as CEs. • Extended ports don't support `family mpls`. MPLS is supported on the cascade port/uplink port connection.
LACP	• Load balancing for cascade ports and extended ports. • Extended ports configured for slow or fast periodic intervals. • Centralized LACP. • EVPN core isolation on extended ports.
Link Fault Management	• IEEE 802.3ah LFM discovery on extended ports and aex extended ports.
LLDP	• AD and extended ports. • Aex members from the same or different SDs. • Telemetry support for LLDP openconfig model. • SNMP support for lldp.mib.

Table 3: Supported Features, Protocols, and Services *(Continued)*

Feature/Service	Description

STP	• RSTP, MSTP, and VSTP support:
	  • AD and extended ports
	  • Aex on the same SD or different SDs
	  • Non-stop bridging (NSB) for RSTP, MSTP, and VSTP on stand-alone and extended port aex.
	  • Telemetry support for STP openconfig model.
	  • SNMP support for rfc4188.mib.

VPLS	• MAC learning and MAC move on extended ports. • Flood route and next hop management on* extended ports.
VRRP	• Extended ports and aex. • VRRP v2 and v3. • IPv4 and IPv6. • Untagged, single tag, and dual tagged Ethernet. • VRRP session inheritance. • Track route and track interface. • SNMP Get. • Openconfig. • VRRP proxy for IPv4. • IRB support for CE interface in a bridge domain only.

2

CHAPTER

NGPE Configuration

IN THIS CHAPTER

Configuring NextGen Port Extender

IN THIS SECTION

NGPE Configuration Process

Use this manual process to create an NGPE network environment. Satellite devices participating in NGPE must have the Junos factory default configuration, or must be zeroized before satellite onboarding. You must have root access to all devices. All interfaces intended for NGPE must be cabled. We strongly recommend that you have console access to all devices during the initial setup process.

Generate the SSH Key on the SD

1. You'll need to generate and capture the SSH key from each SD. This key is used by the AD to communicate with the SD. Each SD will have a unique SSH key.

```
root@sd> request jnu role satellite
```

   This sets the device as a satellite node and also generates its unique SSH key.

2. You will need to copy the SSH key and apply it to the AD configuration in a later step. Use this command to display the key. Ensure you copy the entire key output.

```
root@sd> file show /var/db/jnu/.ssh/id_rsa.pub
```

Configure the AD

3. Configure the required parameters on the AD, under the port-extender stanza.

Configure the management loopback address used by the AD. This loopback is for management traffic only and is a required parameter..

```
root@ad# set services port-extender jnu loopback-ip-address-mgmt-ad <address>
```

```
Example:
set services port-extender jnu loopback-ip-address-mgmt-ad 10.100.100.0
```

4. Configure the data loopback address used by the AD. This is a required parameter.

   This address is for data traffic only.

```
root@ad# set services port-extender fabric loopback-ip-address-data-ad <address>
```

```
Example:
set services port-extender jnu loopback-ip-address-data-ad 10.101.100.0
```

5. Enable the EVPN-VXLAN fabric.

```
root@ad# set services port-extender fabric evpn-vxlan
```

6. Define the satellite model for this SD instance. You can use any name you prefer for <satellite-name>.

```
root@ad# set services port-extender satellite <satellite-name> device-model <sd-model-name>
```

```
Example:
set services port-extender satellite sd1 device-model qfx5120-48y
```

7. Add the interfaces which will become the cascade and uplink ports.

   Note the interfaces are differentiated in this example by the "ad" or "sd" identifier. This example includes 4 interfaces per device. You may configure between 1 and 4 interfaces, although we recommend at least 2 per device. Configure the necessary number of interfaces for your deployment.

```
root@ad# set services port-extender satellite <satellite-name> connectivity-interface-ad
<ad-interface-name>
```

```
set services port-extender satellite <satellite-name> connectivity-interface-ad <ad-
interface-name>
set services port-extender satellite <satellite-name> connectivity-interface-ad <ad-
interface-name>
set services port-extender satellite <satellite-name> connectivity-interface-ad <ad-
interface-name>

set services port-extender satellite <satellite-name> connectivity-interface-sd <sd-
interface-name>
set services port-extender satellite <satellite-name> connectivity-interface-sd <sd-
interface-name>
set services port-extender satellite <satellite-name> connectivity-interface-sd <sd-
interface-name>
set services port-extender satellite <satellite-name> connectivity-interface-sd <sd-
interface-name>
```

```
Example:
set services port-extender satellite sd1 connectivity-interface-ad et-0/0/0
set services port-extender satellite sd1 connectivity-interface-ad et-0/1/0
set services port-extender satellite sd1 connectivity-interface-ad et-0/0/1
set services port-extender satellite sd1 connectivity-interface-ad et-0/1/1

set services port-extender satellite sd1 connectivity-interface-sd et-0/0/48
set services port-extender satellite sd1 connectivity-interface-sd et-0/0/49
set services port-extender satellite sd1 connectivity-interface-sd et-0/0/50
set services port-extender satellite sd1 connectivity-interface-sd et-0/0/51
```

8. Add the SSH key generated during step 2.

```
root@ad# set services port-extender satellite <satellite-name> ssh-key <ssh key from SD>
```

9. Enable the commit scripts used for auto-configuration.

```
root@sd> edit
root@sd# set system scripts language python3 commit file ngpe_commit.py
```

10. Commit the configuration from the AD.

```
root@ad# commit synchronize
```

You can use "commit" if your device has a single routing engine.

Apply the SD Minimal Configuration

11. The AD creates a file which has the required configuration to bring the SD online. You'll need to view and copy the configuration in this file, then apply it on the SD.

    Enter shell mode and grep the contents of the minimal_config.txt. <satellite_name> will be the actual satellite device name.

```
root@ad# run start shell
```

```
# cd /var/log
```

```
# ls | grep minimal
<satellite-name>_minimal_config_txt
#The <satellite-name> will be identical to the name provided in step 6.
```

```
Example: ls | grep minimal
sd1_minimal_config.txt
```

```
# cat <satellite-name>_minimal_config.txt
Step 1 : On satellite <satellite-name> :
Add below configuration :
<SD device config will be displayed; copy all of the "set" commands.>

Step 2 : Reboot the SD to activate the shared-tunnels
```

    Exit the shell and return to configuration mode.

```
# cli
> configure
```

12. Add the minimal configuration to the SD.

```
root@sd# <paste the copied configuration into the SD>
#The following is a sanitized configuration. Do not copy this into your device.
set chassis aggregated-devices ethernet device-count 16
set system login user jnuadmin uid <uid>
set system login user jnuadmin class super-user
set system login user jnuadmin authentication ssh-rsa "<ssh-key previously added will be
displayed here"
set chassis jnu-management mode feature-rich
set chassis jnu-management satellite-name <satellite-name>
set chassis jnu-management user jnuadmin
set chassis jnu-management controller <ip address>
set system static-host-mapping samosa-a inet <ip address>
set system host-name <satellite-name>
set groups ngpe-satellite-connect system login user ftpuser uid <uid>
set groups ngpe-satellite-connect system login user ftpuser class super-user
set groups ngpe-satellite-connect system login user ftpuser authentication encrypted-
password "<value>"
set groups ngpe-satellite-connect interfaces lo0 unit 0 family inet address <ip address>/32
set groups ngpe-satellite-connect interfaces ae0 vlan-tagging
set groups ngpe-satellite-connect interfaces ae0 mtu 9000
set groups ngpe-satellite-connect interfaces ae0 aggregated-ether-options minimum-links 1
set groups ngpe-satellite-connect interfaces ae0 aggregated-ether-options lacp active
set groups ngpe-satellite-connect interfaces ae0 unit 0 vlan-id 100
set groups ngpe-satellite-connect interfaces ae0 unit 0 family inet address <ip address>/31
set groups ngpe-satellite-connect interfaces <interface-name> gigether-options 802.3ad ae0
set groups ngpe-satellite-connect forwarding-options evpn-vxlan shared-tunnels
set groups ngpe-satellite-connect forwarding-options vxlan-disable-copy-tos-decap
set groups ngpe-satellite-connect routing-options static route <ip address>/32 next-hop <ip
address>
set apply-groups ngpe-satellite-connect
set system services netconf ssh
set system services ftp
set system services ssh
```

```
# commit
# exit
```

13. Monitor the SD configuration progress from the console. You can use the "tail" command from the SDs console to view the jnud log.

```
Example:
root@sd1:RE:0% tail -f /var/log/jnud
Nov 25 05:18:39 jnud_open_netconf_session : Host name : 10.100.100.0 - User name : jnuadmin
Nov 25 05:18:39 jnud_send_request_to_node - Node : 10.100.100.0 RPC :  <mgd-jnu-get-lock/>
Nov 25 05:18:39 jnud_send_request_to_node - Node : 10.100.100.0 RPC :  <mgd-jnu-get-lock/>
successfull
Nov 25 05:18:39 jnud_receive_response_from_node : 10.100.100.0
Nov 25 05:18:40 jnud_get_release_lock: rpc  <mgd-jnu-get-lock/> passed
Nov 25 05:18:40 jnud_get_release_lock: rpc  <mgd-jnu-get-lock/> passed
Nov 25 05:18:40 jnud_send_file_remote_scp: Copying the files with scp -O -o
StrictHostKeychecking=no -i /var/db/jnu/.ssh/id_rsa /var/tmp/jnu_initial_sync
jnuadmin@10.100.100.0:/var/tmp/jnu_initial_sync
Nov 25 05:18:41 jnud_send_request_to_node - Node : 10.100.100.0 RPC : <mgd-jnu-schema-add>
<model>qfx5120-48y-8c</model>              <version>25.4R1</version> <filename>/var/tmp/sd1-
schema-258.tar.gz</filename> <model-id>169</model-id>              </mgd-jnu-schema-add>
Nov 25 05:18:41 jnud_send_request_to_node - Node : 10.100.100.0 RPC : <mgd-jnu-schema-add>
<model>qfx5120-48y-8c</model>              <version>25.4R1</version> <filename>/var/tmp/sd1-
schema-258.tar.gz</filename> <model-id>169</model-id>              </mgd-jnu-schema-add>
successfull
Nov 25 05:18:41 jnud_receive_response_from_node : 10.100.100.0
Nov 25 05:18:42 jnud_sync_dual_controller: schema create rpc passed
```

Verification

IN THIS SECTION

Verify the NGPE Configuration

Once the commit script has been started, the full configuration process may take up to 180 seconds to complete. You may seem some warnings about required licences, and you can disregard those warnings.

Use the following commands to check the state of the NGPE deployment.

- **show chassis jnu role**: displays whether a device is a controller or satellite.

```
If run from the AD: show chassis jnu role
controller
If run from a satellite: show chassis jnu role
satellite
```

- **show chassis jnu satellites**: displays the status of a single satellite or all satellites.

```
Satellite            Alive      Model      Version
-----------------------------------------------------
sd1                           up         qfx5120-48y-8c 25.4R1
```

- **show chassis port-extender**: displays each virtual SD slot number, along with its IP address, MAC address, and cascade port number.

```
                                                Target  MAC-address        MAC
Slot  Description                 IP-address      mode    Base               count
Cascade-ports
100   sd1                         10.100.100.1      N       80:63:7c:0e:af:39  1280    ae4001
```

- **show interfaces terse**: displays the up/down status of interfaces; useful for confirming the NGPE fabric is up.

```
Interface               Admin Link Proto    Local                 Remote
et-0/0/0                up    up
et-0/0/0.0              up    up   aenet    --> ae4001.0
et-0/0/0.16384          up    up   aenet    --> ae4001.16384
et-0/0/0.32767          up    up   aenet    --> ae4001.32767
```

- **show interfaces vtep**: check the status of VTEP interfaces.

```
Physical interface: vtep, Enabled, Physical link is Up
  Interface index: 136, SNMP ifIndex: 521
```

```
  Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: Unlimited, Speed:
Unlimited
  Device flags   : Present Running
  Interface Specific flags: Internal: 0x200
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32768 (Index 387) (SNMP ifIndex 546)
    Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
    Ethernet segment value: 00:00:00:00:00:00:00:00:00:00, Mode: single-homed, Multi-homed
status: Forwarding
    VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 10.101.100.0, L2 Routing Instance:
ngpe/ngpe-ad, L3 Routing Instance: ngpe/default
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32769 (Index 432) (SNMP ifIndex 726)
    Flags: Up SNMP-Traps Encapsulation: ENET2
    VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.101.100.1, L2 Routing Instance:
ngpe/ngpe-ad, L3 Routing Instance: ngpe/default
    Input packets : 0
    Output packets: 0
    Protocol bridge, MTU: Unlimited
      Flags: Is-Primary, Trunk-Mode, 0xc000000
```

- **request jnu satellite sync**: needed if the SD failed to initially sync

```
admin@sd1> request jnu satellite sync
Junos node unifier process started, pid 99442
```

- **show configuration | display set**: displays the active configuration on the device
- **show log jnud**: use this command to either monitor the progress of a satellite onboarding, or use after onboarding to check for any errors.

```
Nov 25 05:18:39 jnud_open_netconf_session : Host name : 10.100.100.0 - User name : jnuadmin
Nov 25 05:18:39 jnud_send_request_to_node - Node : 10.100.100.0 RPC :  <mgd-jnu-get-lock/>
```

```
Nov 25 05:18:39 jnud_send_request_to_node - Node : 10.100.100.0 RPC :  <mgd-jnu-get-lock/>
successfull
Nov 25 05:18:39 jnud_receive_response_from_node : 10.100.100.0
Nov 25 05:18:40 jnud_get_release_lock: rpc  <mgd-jnu-get-lock/> passed
Nov 25 05:18:40 jnud_get_release_lock: rpc  <mgd-jnu-get-lock/> passed
Nov 25 05:18:40 jnud_send_file_remote_scp: Copying the files with scp -O -o
StrictHostKeychecking=no -i /var/db/jnu/.ssh/id_rsa /var/tmp/jnu_initial_sync
jnuadmin@10.100.100.0:/var/tmp/jnu_initial_sync
Nov 25 05:18:41 jnud_send_request_to_node - Node : 10.100.100.0 RPC : <mgd-jnu-schema-add>
<model>qfx5120-48y-8c</model>              <version>25.4R1</version> <filename>/var/tmp/sd1-
schema-258.tar.gz</filename> <model-id>169</model-id>              </mgd-jnu-schema-add>
Nov 25 05:18:41 jnud_send_request_to_node - Node : 10.100.100.0 RPC : <mgd-jnu-schema-add>
<model>qfx5120-48y-8c</model>              <version>25.4R1</version> <filename>/var/tmp/sd1-
schema-258.tar.gz</filename> <model-id>169</model-id>              </mgd-jnu-schema-add>
successfull
Nov 25 05:18:41 jnud_receive_response_from_node : 10.100.100.0
Nov 25 05:18:42 jnud_sync_dual_controller: schema create rpc passed
```

Offboard and Re-onboard a Satellite Device

IN THIS SECTION

Offboard a Satellite Device

You can offboard an existing SD from your NGPE topology by following this process:

1. Disable the fabric for the target SD.

```
admin@ad1# set services port-extender satellite <satelle-name> fabric-disable
```

2. Commit the change.

```
admin@ad1# commit synchronize
admin@ad1# commit synchronize
re0:
configuration check succeeds
re1:
commit complete
re0:
warning: Clear Sat change bits terminated abnormally
commit complete
```

3. The satellite will be in the down state, however it will still be part of the NGPE topology. Remove the SD from the topology. You may either `deactivate` or `delete` the SD.

```
admin@ad1# deactivate services port-extender satellite <satellite-name>
```

4. Commit the change.

```
admin@ad1# commit synchronize
#You may see output similar to this. It is normal.
admin@ad1# commit synchronize
re0:
warning: Port extender configuration is not detected in the candidate configuration..
warning: Checking for any pending satellites to deboard..
warning: Attempting to delete sd1
warning: Commit script clean up is successful...
warning: No configuration to be done with script as port-extender is disabled
[edit routing-options static]
  'route 10.100.100.1/32'
    warning: requires 'L3 Static' license
re0:
configuration check succeeds
re1:
[edit chassis]
  'satellite sd1'
    warning: statement does not exist
commit complete
```

```
re0:
commit complete
```

5. The device should now be reverted to its pre-SD state.

Re-onboard a Satellite Device

You can re-onboard an SD to your NGPE topology by following this process:

1. Activate the satellite. If you previously deleted the `port-extender satellite` configuration, then use `set` instead of `activate`.

```
admin@ad1# activate service port-extender satellite <satellite-name>
```

2. Commit the change.

```
admin@ad1# commit
```

3. Synchronize the satellites.

```
admin@ad1# run request jnu satellite synchronize
```

4. This will bring the SD back into the topology. Ensure that the `fabric-disable` statement is not in effect. If it is, remove it.

```
admin@ad1# delete services port-extender satellite <satellite-name> fabric-disable
```

5. Commit the change.

```
admin@ad1# commit
```

Saving the Configuration

IN THIS SECTION

Save NGPE User-Generated Configuration

Follow this process to save only the user-generated configuration. This process should be performed when NGPE is in a maintanence window since the process requires disabling the NGPE fabric. We don't support loading configurations with NGPE elements into an active NGPE environment since this can have unexpected effects.

1. Disable the NGPE fabric.

```
admin@ad1# deactivate services port-extender
admin@ad# commit synchronize
```

2. Save the configuration.

```
admin@ad1# show | display set | save /var/tmp/device.conf
Wrote 2000 lines of output to '/var/tmp/device.conf'
```

3. Restore the NGPE fabric.

```
admin@ad1# activate services port-extender
admin@ad# commit synchronize
```

Reload Existing Saved NGPE Configuration

As previously stated, we don't support loading an NGPE configuration onto an active NGPE environment. You may however load a saved NGPE configuration to a non-NGPE device which will act as the AD. The target AD must have either a baseline Junos configuration, or you must `zeroize` the device prior to following this process. Ensure that all necessary cabling has been completed in advance, and

that any intended SDs are also running a base configuration. Please ensure you have console access to the intended AD.

1. Load the saved configuration.

```
admin@ad1# load override /location/<filename>
```

```
Example:
admin@ad1# load override /var/tmp/device.conf
...config loads...
admin@ad1# commit synchronize
```

2. Once the configuration has been loaded and committed, you'll need to activate the port-extender stanza.

```
admin@ad1# activate services port-extender
admin@ad1# commit synchronize
```

3. Enable an SD.

```
admin@sd1# request jnu role satellite
```

4. Synchronize the satellite.

```
admin@sd1# request jnu satellite sync
```

5. Repeat steps 3 and 4 for each SD.