Junos® OS

Designing and Implementing a Junos Node Unifier Network

Release
1.3J3

Published: 2014-10-21

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Junos® OS Designing and Implementing Junos Node Unifier
Release 1.3J3
Copyright © 2014, Juniper Networks, Inc.
All rights reserved.

Revision History
October 2014—J3 Junos Node Unifier 1.3

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

END USER LICENSE AGREEMENT

Table of Contents

PART 1

Introduction to Junos Node Unifier

CHAPTER 1

Introduction to Junos Node Unifier

Audience for Junos Node Unifier

This guide is intended to assist service providers to design and plan an implementation for Junos Node Unifier (JNU). We intend the guide to be used by the following:

- Network architects—Responsible for creating the overall design and architecture of the dual-stack network.
- Network planners—Responsible for planning the implementation from a network perspective, including equipment.
- Network operations engineer—Responsible for creating the configuration that implements the overall design. Also responsible for deploying the implementation and actively monitoring the network.
- Sales engineers—Responsible for working with architects, planners, and operations engineers to design and implement the network solution.

Related Documentation

- Junos Node Unifier Overview on page 4
- Basic Architecture of a JNU Network on page 5
- Terms Used in the JNU Documentation on page 5
- JNU Management Plane Overview on page 7

Junos Node Unifier Overview

Junos Node Unifier (JNU) allows you to configure and manage many Juniper Networks platforms running Junos OS from one MX Series router. You can use JNU to manage thousands of 1-Gigabit and 10-Gigabit Ethernet ports in a single site or that are distributed across multiple sites from a single point. Starting with JNU Release 1.3, you can also configure an EX9000 Ethernet Switch as a controller.

JNU provides single-touch provisioning from one MX Series router or an EX9000 Ethernet Switch acting as a controller. It provides a single point of:

- Configuration and management
- Running operational mode commands
- SNMP polling and SNMP traps
- Collecting logging information

The JNU software answers the following needs:

- Ethernet port fanout or port multiplexer to control thousands of Ethernet ports from one MX Series router.
- Layer 2 switching on managed devices to meet Data Center needs, such as server port aggregation.
- Layer 3 MPLS routing on managed devices to provide business access and mobile backhaul applications.

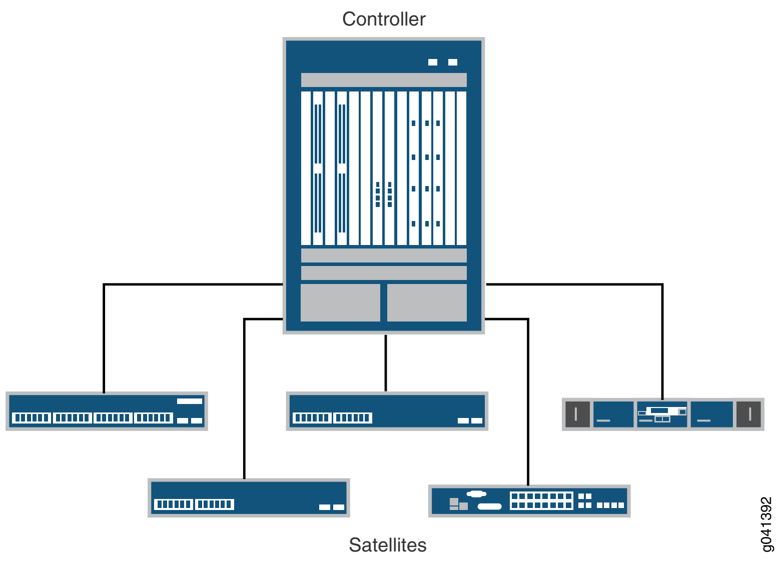
Related Documentation

- Basic Architecture of a JNU Network on page 5
- Terms Used in the JNU Documentation on page 5
- JNU Management Plane Overview on page 7

Basic Architecture of a JNU Network

The basic architecture of a JNU implementation is a star configuration with one MX Series router acting as a hub to the connected satellite devices. The satellite devices are devices running the Junos operating system (Junos OS), such as EX Series Ethernet switches and QFX Series devices. See "Platform Considerations for the JNU Controller" on page 15 and "Supported Platforms for JNU Satellite Devices" on page 15 for a list of devices that JNU Release 1.3J3 supports as controllers and satellites respectively.

Figure 1: Basic JNU Architecture



Related Documentation

- Junos Node Unifier Overview on page 4
- Terms Used in the JNU Documentation on page 5
- JNU Management Plane Overview on page 7

Terms Used in the JNU Documentation

Table 1 on page 5 defines terms used in the JNU documentation.

Table 1: JNU Terms

Term	Definition
Controller	An MX Series router that is used to manage and configure satellite devices. Starting with JNU Release 1.3, you can also configure an EX9000 Ethernet Switch as a controller.
JNU	Junos Node Unifier.
Satellite	Platforms that are managed by the controller.

Related Documentation

- Junos Node Unifier Overview on page 4
- Basic Architecture of a JNU Network on page 5
- JNU Management Plane Overview on page 7

CHAPTER 2

Understanding the JNU Architecture

JNU Management Plane Overview

The JNU software uses a private management plane on the MX Series controller to manage satellite devices as follows.

- Provision satellite devices
- Operate satellite devices
- Perform SNMP polling and trap collection
- Collect logs

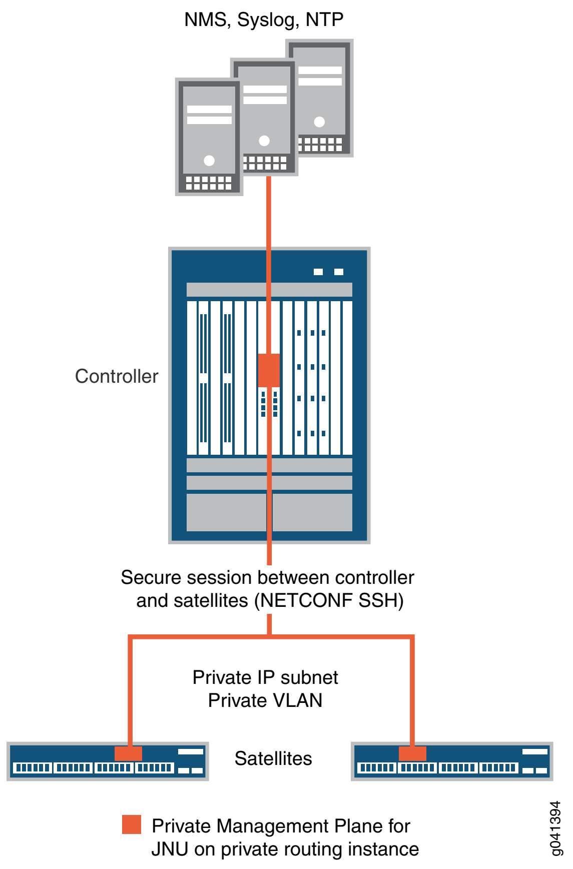
Related Documentation

- JNU Management Network on page 8
- JNU Data Plane Overview on page 10
- Junos Node Unifier Overview on page 4

JNU Management Network

The JNU architecture provides a private management plane for JNU that is separate from the control plane and the data plane. This design provides maximum performance and reliability and the ability to efficiently scale the JNU network. Figure 2 on page 8 shows a basic JNU management network. This network is created during the JNU initialization process.

Figure 2: JNU Management Network


NMS, Syslog, NTP
Controller
Secure session between controller and satellites (NETCONF SSH)
Private IP subnet
Private VLAN
Satellites
🟥 Private Management Plane for JNU on private routing instance
g041394

The JNU management network is created during the JNU initialization process. The process creates a private network for in-band management, and the configuration is placed in a configuration group on the controller and on each satellite device.

The management network has the following characteristics:

- Ethernet interfaces are used for the connection between the controller and the satellites and between the controller and network management systems (NMSs). A private VLAN between the controller and the satellite devices is used to separate JNU management traffic from data plane traffic.

  During the initialization process, you specify the physical interfaces, VLAN IDs, and IP addresses to be used in the management network for the downlink connection from the controller to the satellites, and for the uplink connection from the satellites to the controller.

  By default the software places the Ethernet interfaces into an aggregated Ethernet bundle. If you specify only one physical interface during initialization, you have the option to not use aggregated Ethernet.
- The following private routing instances are created on the controller during the initialization process. These routing instances are not visible outside of the JNU Network.
  - A private VPN routing and forwarding (VRF) routing instance to provide address for Layer 3 VPNs. The VRF routing instance makes it possible to reuse the management network IP addresses in the data plane.
  - A private virtual-switch routing instance is created on the controller for the Layer 2 VPN. The virtual-switch routing instances makes it possible to reuse the VLAN IDs in the data plane.

    An integrated routing and bridging (IRB) interface is created that is used to provide IP addresses to the virtual switch.
- If supported, a routing instance is created on the satellite device that contains the uplink configuration from the satellite to the controller.
- A secure NETCONF-over-SSH connection is created between the controller and the satellite device.

Network Management in the Management Plane

During the controller initialization process, you have the option of setting up SNMP, system logging, and NTP. If you choose to set up these features, the initialization process creates a network configuration over an Ethernet interface to the NMS servers. The configuration includes static routes to these servers in the VPN routing and forwarding (VRF) routing instance on the controller.

The initialization process creates a Network Address Translation (NAT) configuration that is used to translate the source address of the traffic sent to the SNMP or syslog server so that all network management traffic from the satellite devices originates from a source address on the controller.

You do not need a license to use NAT with the JNU management plane.

Related Documentation

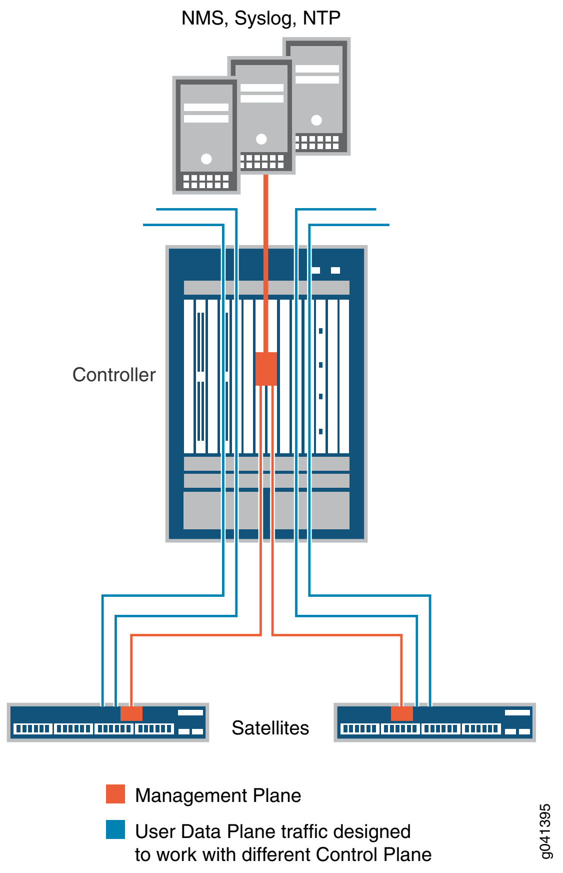
- JNU Management Plane Overview on page 7
- JNU Data Plane Overview on page 10
- Initializing JNU Mode on the MX Series Controller on page 31
- Initializing JNU Mode on the Satellite Device on page 36
- Centralized Collection of SNMP Statistics and Log Messages Overview on page 67

JNU Data Plane Overview

The data plane of the controller and satellite devices, which is responsible for forwarding user data, is separate from the management plane. If your satellite device supports routing instances, the management configuration is placed in a routing instance, and you can reuse IP addresses that were used for JNU management.

For load balancing and fast recovery, you can use link aggregation of both the management interfaces and data plane instances.

Figure 3: JNU Architecture with Data Plane


NMS, Syslog, NTP
Controller
Satellites
Management Plane
User Data Plane traffic designed to work with different Control Plane
g041395

Related Documentation

- JNU Management Plane Overview on page 7
- JNU Management Network on page 8

PART 2

Planning a JNU Implementation

CHAPTER 3

Planning Overview

- Platform Considerations for the JNU Controller on page 15
- Supported Platforms for JNU Satellite Devices on page 15

Platform Considerations for the JNU Controller

You can use the MX240, MX480, and MX960 MX2010, and MX2020 3D Universal Edge Router as the JNU controller. The MX Series router uses Modular Port Concentrators (MPCs) to connect to the satellites.

You must use an MX Series router as the controller, and you must use MPCs (not DPCs). We recommend that you allocate more than one interface for interconnect between MX Series routers and satellites. These interfaces will be placed into link aggregation (LAG) configuration for fast recovery, with traffic spreading across member links.

An MX Series router can manage one satellite on each of its Ethernet ports. For example, the MX480 router supports up to 176 10-Gigabit Ethernet interfaces. It can therefore manage up to 176 satellite devices on the 10-Gigabit Ethernet interfaces.

Related Documentation

- Supported Platforms for JNU Satellite Devices on page 15
- Junos Node Unifier Overview on page 4

Supported Platforms for JNU Satellite Devices

JNU 1.3J3 supports the following satellite devices:

- EX3200 Ethernet Switch
- EX3300 Ethernet Switch
- EX4200 Ethernet Switch
- EX4300 Ethernet Switch
- EX4500 Ethernet Switch
- EX4550 Ethernet Switch
- QFX 3500 devices

Related Documentation

- Platform Considerations for the JNU Controller on page 15
- Junos Node Unifier Overview on page 4

CHAPTER 4

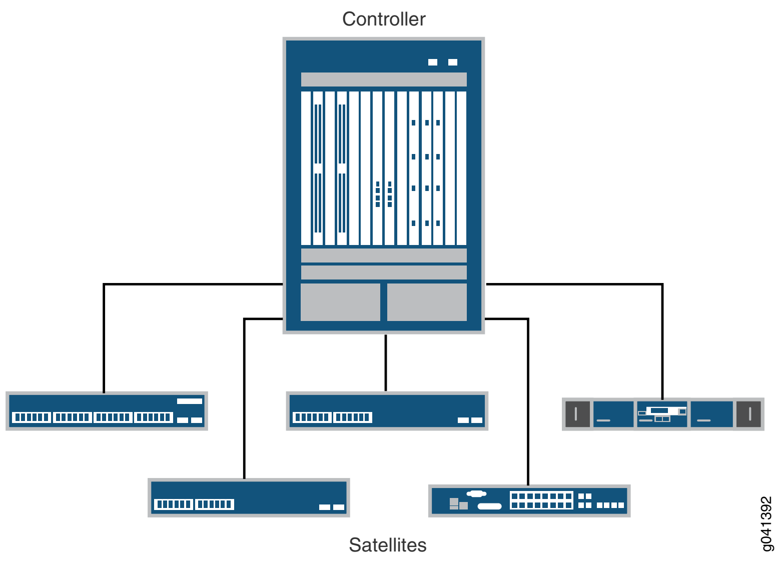
Designs for JNU Implementations

You can configure the satellite and controller devices in a JNU group in different topologies or designs, depending on your network needs and deployment requirements.

Design 1: One Controller with Satellites in a Star Configuration

This topology is a simple star topology with one JNU controller connected to a group of satellite devices.

Figure 4: Controller with Satellites in a Star Configuration



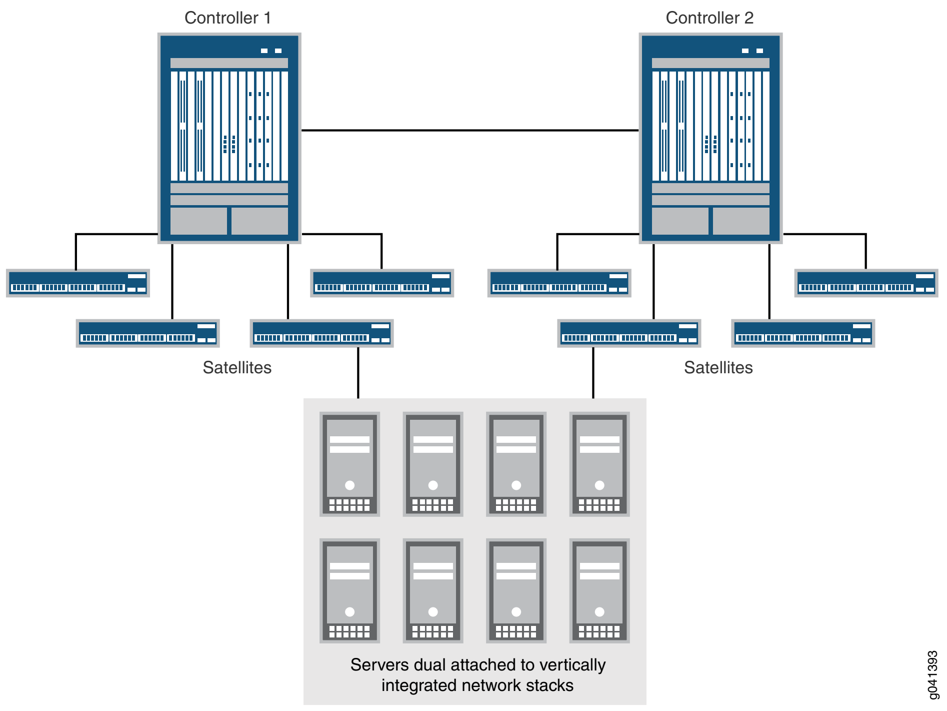
Related Documentation

- Design 2: Two Controllers Each With Satellites in a Star Configuration on page 18
- Design 3 V topology: Distributed Layer 2 and Layer 3 Mode on page 19

Design 2: Two Controllers Each With Satellites in a Star Configuration

This star topology is characterized by two groups of satellite devices each connected to their own hub device, or controller. There is no connection between the satellites in one hub to the group of satellites in the other hub. One controller can control all of the associated satellite devices.

Figure 5: Two Controllers Each With Satellites in a Star Configuration


Controller 1
Controller 2
Satellites
Satellites
Servers dual attached to vertically integrated network stacks
g041393

In this design, you deploy one JNU controller in active forwarding mode and the second JNU controller in standby forwarding mode. If the active JNU controller or its satellites fail, there is a switchover to the second controller.

The servers choose the active JNU controller by activating a set of interfaces connected to that controller. In case of a failure, the JNU takes down the server connection port(s) and switches to the second JNU controller by activating the interface set configured on that controller. The controllers can use the MX Series multichassis link aggregation group (MC-LAG) active-active functionality to provide a smooth switchover from one JNU controller to the other.

This model provides two independent JNU silos for resiliency, and use only one at a time through the control of the servers. This model has the advantage of simplicity, but only 50 percent of equipment is likely to be in use at any time.

Related Documentation

- Design 1: One Controller with Satellites in a Star Configuration on page 17
- Design 3 V topology: Distributed Layer 2 and Layer 3 Mode on page 19

Design 3 V topology: Distributed Layer 2 and Layer 3 Mode

This topology comprises MX Series routers in a Virtual Chassis configuration, that function as controllers, connected to each of the satellites in a JNU group. If the one of the JNU controllers in the Virtual Chassis cluster fails, there is a switchover to another controller to manage the satellites. JNU is designed to work with the control plane option of VLAN tagging in this type of deployment. A Virtual Chassis configuration enables a collection of member routers to function as a single virtual router, and extends the features available on a single router to the member routers in the Virtual Chassis. The interconnected member routers in a Virtual Chassis are managed as a single network element that appears to the network administrator as a single chassis with additional line card slots, and to the access network as a single system.

Related Documentation

- Design 1: One Controller with Satellites in a Star Configuration on page 17
- Design 2: Two Controllers Each With Satellites in a Star Configuration on page 18

CHAPTER 5

Use Cases

JNU enables the deployment of rich services on the satellite devices by maintaining their full individual device feature set while in the cluster mode, including Layer 2 switching on satellite for server port aggregation, and Layer 3 or MPLS routing on satellites for business access and mobile backhaul.

Server Farm: Ethernet Port Fan-Out

In a topology in which an MX Series device that works as a controller and manages EX Series switches and QFX Series switches that are satellites in a JNU group, the satellites might be in turn connected to other external servers in the manner of a farm. In such a scenario, you can use JNU to enable thousands of Ethernet ports to be administered from one MX Series device. The network services are positioned in the most cost-effective location and a single touch-point from the controller can enable many services to be managed. The EX Series switches enable many Gigabit Ethernet ports to be used to connect to external legacy servers by deriving the advantages of optimal infrastructure costs and providing Layer 2, CoS, and filtering capabilities. The QFX Series switches enable many 10-Gigabit Ethernet ports to be used to connect to advanced, new servers by deriving the advantages of optimal infrastructure costs and providing Layer 2, CoS, and filtering capabilities.

Related Documentation

- Service Provider Edge: Fiber to the Building (FTTB) on page 21
- Video Streaming on page 22
- Mobile Backhaul on page 22

Service Provider Edge: Fiber to the Building (FTTB)

A typical service provider edge fiber to the home (FTTH) use case is a network that contains an MX Series router that functions as a controller governs and manages two satellite devices, such as an EX Series switch that is capable of Layer 2 and Layer 3 functionalities and an MX Series device that is used for subscriber services. In a network without the JNU solution, each network element has to be provisioned, configured, managed, and debugged separately. This proves to be a huge operational expense and nightmare for service providers. With the JNU solution, a single management and control plane is created on the MX Series controller device. Many services can be managed and provisioned from a single touch-point on this controller, thus reducing the time to deploy new services significantly. Moreover, the devices acting in satellite mode retain their rich Layer 2 and Layer 3 features, class of service (CoS), filtering, and also MPLS on the edge. Moreover, the solution uses standard protocol and allows packet replication and group membership, all of which are highly desirable for video streaming applications. With the JNU solution, the nuances of the satellite devices are hidden from the operator. A standard provisioning stanza is shared across the controller and the satellite devices, which allows the network to grow seamlessly, with each additional satellite device acting as a plug-and-play device.

Related Documentation

- Server Farm: Ethernet Port Fan-Out on page 21
- Video Streaming on page 22
- Mobile Backhaul on page 22

Video Streaming

With multicasting, servers can send a single stream to a group of recipients instead of sending multiple unicast streams. While the use of streaming video technology was previously limited to occasional company presentations, multicasting has provided a boost to the technology resulting in a constant stream of movies, real-time data, news clips, and amateur videos flowing nonstop to computers, TVs, tablets, and phones. However, all of these streams quickly overwhelmed the capacity of network hardware and increased bandwidth demands leading to unacceptable blips and stutters in transmission.

Consider a network environment in which the controller that manages satellites is used for streaming of video and multimedia to end-users or subscribers. Video stream leverages the functional capabilities of the satellite and enables snooping, group membership, and packet replication to be implemented. Each stream from the controller is multiplied by the satellites corresponding to the recipients. The advantage is minimized bandwidth between the controller and the satellites, thereby reducing administrative costs and compatibility with standard protocols for streaming.

Related Documentation

- Server Farm: Ethernet Port Fan-Out on page 21
- Service Provider Edge: Fiber to the Building (FTTB) on page 21
- Mobile Backhaul on page 22

Mobile Backhaul

A common mobile backhaul use case represents MX Series routers in a Virtual Chassis configuration that acts a controller and ACX Series routers that function as satellites. With JNU, connecting each port of the MX Series with the satellite and enabling the port to run in JNU mode can increase port density. These devices can be configured, managed, and provisioned from the controller. Moreover, the satellites also have a sophisticated feature set, which allows them to perform technologies such as MPLS on the satellites.

In the mobile backhaul scenario, the ACX Series router is primarily used in the access layer as the cell site router and the MX Series router is used as the edge and aggregation router. As the cell site router, the ACX Series router connects the base station (BS) to the packet network. Several cell site routers can be connected in a ring or hub-and-spoke fashion to the upstream preaggregation and aggregation routers (MX Series routers).

Related Documentation

- Server Farm: Ethernet Port Fan-Out on page 21
- Service Provider Edge: Fiber to the Building (FTTB) on page 21
- Video Streaming on page 22

PART 3

Implementing JNU

CHAPTER 6

Best Practices for Configuring JNU

Naming Your JNU Controller and Satellite Devices

It is important to plan the naming of controller and satellite devices in a JNU group so that you can easily identify the satellites that belong to the same group. The hostname of satellites is used in SNMP community strings and system log prefixes to identify the satellite associated with the SNMP message or system log message.

A naming structure like the following is recommended:

- jnu1-ctrl as the controller hostname
- jnu1-sat1, jun1-sat2, jun1-sat3, and so on as the satellite hostnames

Junos OS Releases on the JNU Controller and Satellites

We recommend that you run the same Junos OS release on the controller and on the satellite devices.

CHAPTER 7

Getting Started with the JNU Software

Installing the JNU Software on the Controller

To load the JNU package onto the controller:

- Enter the following command on the MX Series controller. For example:

```
user@jnu1-ctrlr> request system software add jnu-1.3J3.5-signed.tgz
Installing package '/var/tmp/jnu-1.3J3.5-signed.tgz' ...
Verified jnu-1.3J3.5.tgz signed by PackageProduction_11_4_0 Adding jnu...
Available space: 556676 require: 3220
NOTICE: uncommitted changes have been saved in
/var/db/config/juniper.conf.pre-install
Mounted jnu package on /dev/md10...
Restarting bslockd ...
mgd: commit complete
Saving package file in /var/sw/pkg/jnu-1.3J3.5-signed.tgz ...
Saving state for rollback ...
```

Related Documentation

- Initializing JNU Mode on the MX Series Controller on page 31
- Initializing JNU Mode on the Satellite Device on page 36
- Installing the JNU Software on Satellite Devices on page 30

Installing the JNU Software on Satellite Devices

To load the JNU package onto the satellite device:

- Enter the following command on the satellite device. For example:

```
user@jnu-satellite1> request system software add jnu-1.3J3.5-signed.tgz
Installing package '/var/tmp/jnu-1.3J3.5-signed.tgz' ...
Verified jnu-1.3J3.5.tgz signed by PackageProduction_11_4_0 Adding jnu...
Available space: 556676 require: 3220
NOTICE: uncommitted changes have been saved in
/var/db/config/juniper.conf.pre-install
Mounted jnu package on /dev/md10...
Restarting bslockd ...
mgd: commit complete
Saving package file in /var/sw/pkg/jnu-1.3J3.5-signed.tgz ...
Saving state for rollback ...
```

Related Documentation

- Installing the JNU Software on the Controller on page 29
- Initializing JNU Mode on the Satellite Device on page 36
- Initializing JNU Mode on the MX Series Controller on page 31

Initializing JNU Mode on the MX Series Controller

After you install the JNU software, you need to initially configure and initialize the MX Series controller. The initialization process creates a JNU management plane configuration on the controller and places it in a configuration group called jnu-controller-mgmt. The management plane configuration includes interfaces, internal routing-instances, virtual switch bridging, SNMP, system logs, and NTP in the main instance of the configuration. You can configure NAT when required.

As part of the initialization process, the JNU configuration is committed on the controller.

When you initialize the controller and the satellite devices, you must be logged in to the controller or satellite as the root user. The initialization process creates a user account called jnuadmin, which the controller uses to log in to the satellites. After the initialization process is complete, log in to the controller by using the jnuadmin user account.

To initially configure the controller:

1. Enter the **request jnu controller** command.

```
user@jnu1-ctrlr> request jnu controller
Initializing Controller...



JNU Controller configuration completed
```

JNU mode is enabled on the device, which starts to function as the controller. The following is an example of the configuration created on the controller as a result of running the controller initialization process.

```
groups {
    jnu-controller-mgmt {
        apply-macro pem;
        chassis {
            aggregated-devices {
                ethernet {
                    device-count 255;
                }
            }
            /* Slot of the Trio FPC */
            fpc 0{
                pic 0 {
                    inline-services {
                        bandwidth 10g;
                    }
                }
            }
        }
        services {
            nat {
                pool controller-management-ip {
                    address 192.168.164.164/32;
                }
            }
        }
        interfaces {
            si-0/0/0 {
                unit 0 {
                    family inet;
                    family inet6;
                }
                unit 1 {
                    family inet;
                    service-domain inside;
                }
                unit 2 {
                    family inet;
                    service-domain outside;
                }
            }
            irb {
                unit 16385 {
                    family inet {
                        address 192.168.0.1/24;
                    }
                }
                unit 0 {
                    family inet {
                        address 192.168.0.1/24;
                    }
                }
            }
            xe-0/0/0 {
                encapsulation ethernet-bridge;
                unit 0;
            }
            xe-0/0/1 {
                encapsulation ethernet-bridge;
                unit 0;
            }
            ...
            xe-0/2/0 {
                encapsulation ethernet-bridge;
                unit 0;
            }
            ge-1/0/0 {
                encapsulation ethernet-bridge;
                unit 0;
            }
            ge-1/0/1 {
                encapsulation ethernet-bridge;
                unit 0;
            }
            event-options {
                max-policies 20;
                generate-event {
                    event-script-timer time-interval 300;
                    lldp-script-timer time-interval 130;
                }
                policy jnu-satellite-connectivity {
                    events event-script-timer;
                    then {
                        event-script monitor-satellites.slax;
                    }
                }
                policy JNU-LLDP-Connectivity {
                    events lldp-script-timer;
                    then {
                        event-script monitor-lldp-script.slax;
                    }
                }
                event-script {
                    file monitor-satellites.slax;
                    file monitor-lldp-script.slax;
                }
            }
            protocols {
                lldp {
                    interface all;
                }
            }
            policy-options {
                policy-statement reject-all {
                    then reject;
                }
            }
            routing-instances {
                jnu-vrf {
                    instance-type vrf;
                    system {
                        services {
                            dhcp-local-server {
                                group dhcp {
                                    interface irb.0;
                                }
                            }
                        }
                    }
                    access {
                        address-assignment {
                            pool jnu_v4_pool {
                                family inet {
                                    network 192.169.0.0/24;
                                    range 1 {
                                        low 192.169.0.2;
                                        high 192.169.0.254;
                                    }
                                }
                            }
                        }
                    }
                    interface si-1/0/0.1;
                    interface irb.16385;
                    route-distinguisher 192.168.0.1:0;
```

```
                    vrf-import reject-all;
                    vrf-export reject-all;
                }
                jnu-vs {
                    instance-type virtual-switch;
                    bridge-domains {
                        jnu {
                            vlan-id 4094; interface ae479.16385;
                            routing-interface irb.16385;
                        }
                        jnu1 {
                            vlan-id none;
                            interface xe-0/0/0.0;
                            interface xe-0/0/1.0;
                            ...
                            interface xe-0/2/0.0
                            interface ge-1/0/0.0;
                            interface ge-1/0/1.0;
                            routing-interface irb.0;
                        }
                    }
                }
            }
        }
        jnu-module {
            system {
                scripts {
                    commit {
                        allow-transients;
                        file config-port-extender-commit.slax;
                    }
                    op {
                        file jnu-show-port-extender.slax;
                        file jnu-port-extender-command.slax;
                        file jnu-add-delete-downlink.slax;
                        file config-free-form-pem.slax {
                            command config-free-form;
                        }
                        file jnu-initialize-controller.slax;
                        file jnu-show-satellites.slax;
                        file jnu-show-configuration.slax;
                        file jnu-remote.slax;
                        file jnu-rollback.slax;
                        file jnu-order-satellites.slax;
                        file jnu-add-delete-satellites.slax;
                        file config-free-form-precommit.slax;
                        file jnu-delete-satellite.slax;
                    }
                    snmp jnu-snmp.slax {
                        oid .1.3.6.1.4.1.2636.3.1.13.1.89;
                    }
                }
            }
        }
    }
}
```

Related Documentation

- **request-jnu-controller on page 64**
- Installing the JNU Software on the Controller on page 29
- Initializing JNU Mode on the Satellite Device on page 36

Initializing JNU Mode on the Satellite Device

When you initialize the satellite device, the software creates a management configuration on the device that allows the controller to configure and manage the satellite.

When you run the satellite initialization process, the controller connects to the satellite and copies JNU code elements that are based on scripting technology to the satellite.

Before you initialize the satellite device, you must configure a root (superuser) password by including the **root-authentication** statement at the **[edit system]** hierarchy level. When you initialize the satellite devices, you must be logged in to the satellite as the root user.

To initialize a satellite device:

1. Enter the following command on the satellite device.

```
user@jnu-satellite1> request jnu satellite
Initializing Satellite...

Commit Completed

JNU Satellite configuration completed
```

The following is an example of the configuration created on the satellite as a result of running the satellite initialization process.

```
groups {
  jnu-satellite-mgmt {
    system {
      ntp {
        server 192.168.0.1;
      }
    }
    chassis {
      aggregated-devices {
        ethernet {
          device-count 31;
        }
      }
    }
    security {
      ssh-known-hosts {
        host 192.169.0.1 {
          ecdsa-sha2-nistp256-key
"AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNn6OCxqurMlNxK3G08p
CfHEEYMK7k6IBO4csN0M58yo\nw06u+TBk3QtRkjKNzj18Py1Mx48Ml7GcfrC0mAjoTrc=";

        }
      }
    }
    interfaces {
      ae0 {
        aggregated-ether-options {
          lacp {
            active;
```

```
                    }
                }
                unit 0 {
                    family ethernet-switching {
                        port-mode trunk;
                        vlan {
                            members all;
                        }
                    }
                }
            }
        }
        protocols {
            lldp {
                port-id-subtype interface-name;
                interface all;
            }
        }
        ethernet-switching-options {
            dot1q-tunneling {
                ether-type 0x8100;
            }
        }
        vlans {
            jnu {
                vlan-id 4094;
            }
        }
    }
    jnu-module {
        system {
            scripts {
                commit {
                    allow-transients;
                }
                op {
                    file jnu-initialize-satellite.slax;
                    file config-free-form-precommit.slax;
                }
            }
        }
    }
}
apply-groups [ jnu-module jnu-satellite-mgmt ];
system {
    autoinstallation {
        traceoptions {
            file autod;
            level all;
            flag {
                all;
            }
        }
        interfaces {
            ge-* {
                bootp;
            }
            xe-* {
                bootp;
            }
        }
    }
    ports {
        console log-out-on-disconnect;
    }
    root-authentication {
        encrypted-password "$1$CSJA9A09$sdnsPhaXKxChqcgPB4HUT0"; ##
SECRET-DATA
    }
    login {
        user jnuadmin {
            uid 928;
            class super-user;
            shell csh;
            authentication {
                encrypted-password "$1$a7DYfEN0$huf8FDLIshFB5v6zb2WMz0"; ##
SECRET-DATA
            }
        }
    }
    services {
        ftp;
        ssh;
        telnet;
    }
}
interfaces {
    me0 { /* satellite management interface */
        unit 0 {
            family inet {
                address /* satellite management address */
            }
        }
    }
}
```

After satellite initialization, LLDP neighborship is established between the satellite device and the controller. The connected interfaces are automatically added to bring up an active aggregated Ethernet link across the satellite device and the controller.

Sample Initial Configuration on an MX Series Controller After Satellite Initialization

The following is an example of the configuration that is configured MX series controller during the satellite initialization process.

```
groups {
    jnu-controller-mgmt {
        apply-macro pem;
        apply-macro satellite-link-details {
            jnu-satellite1:xe-0/1/0 xe-0/1/0;
        }
    services {
      service-set ss-nat {
        nat-rules jnu-use-controller;
        next-hop-service {
          inside-service-interface si-0/0/0.1;
          outside-service-interface si-0/0/0.2;
        }
      }
      nat {
        pool jnu-jnu-satellite1 {
          address 192.168.164.164/32;
        }
        allow-overlapping-nat-pools;
        rule jnu-use-controller {
          match-direction input;
          term term-jnu-satellite1 {
            from {
              source-address {
                192.168.0.2/32;
              }
            }
            then {
              translated {
                source-pool jnu-jnu-satellite1;
                translation-type {
                  basic-nat44;
                }
              }
            }
          }
        }
      }
    }
    security {
      ssh-known-hosts {
        host 192.168.0.2 {
          ecdsa-sha2-nistp256-key
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNn6OCxqurMlNxK3G08p
CfHEEYMK7k6IBO4csN0M58yow06u+TBk3QtRkjKNzj18Py1Mx48Ml7GcfrC0mAjoTrc=;

        }
      }
    }
    interfaces {
      xe-0/1/0 {
        description "Connected to Satellite - jnu-satellite1";
        gigether-options {
          802.3ad ae0;
        }
      }
      ae0 {
        flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        aggregated-ether-options {
          lacp {
            active;
```

```
                    }
                }
                unit 16385 {
                    encapsulation vlan-bridge;
                    vlan-id 4094;
                }
            }
        }
        routing-instances {
            jnu-vs {
                bridge-domains {
                    jnu {
                        interface ae0.16385;
                    }
                }
            }
        }
    }
    jnu-satellites {
        system {
            static-host-mapping {
                jnu-satellite1 inet 192.168.0.2;
            }
        }
    }
    jnu-satellite1 {
        apply-macro pe-svlans {
            ssvlan 1;
            ssvlan-single 1;
        }
    }
}
security {
    ssh-known-hosts {
        host 192.169.0.2 {
            ecdsa-sha2-nistp256-key
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNn6OCxqurMlNxK3G08p
CfHEEYMK7k6IBO4csN0M58yow06u+TBk3QtRkjKNzj18Py1Mx48Ml7GcfrC0mAjoTrc=;

        }
    }
}
```

Sample Initial Configuration on an EX Series Ethernet Switch After Satellite Initialization

The following is an example of the configuration that is configured and committed an EX Series Ethernet Switch satellite device during the initialization process.

```
groups {
    jnu-satellite-mgmt {
        chassis {
            aggregated-devices {
                ethernet {
                    device-count 63;
                }
```

```
      }
    }
    security {
        host 192.168.0.1 {
          ecdsa-sha2-nistp256-key
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNn6OCxqurMlNx
K3G08pCfHEEYMK7k6IBO4csN0M58yow06u+TBk3QtRkjKNzj18Py1Mx48Ml7GcfrC0mAjoTrc=;

        }
      }
    }
    interfaces {
      xe-0/1/0 {
        ether-options {
          802.3ad ae0;
        }
      }
      vlan {
        unit 4094 {
          family inet {
            address 192.168.0.2/24;
          }
        }
      }
      lo0 {
        unit 0 {
          family inet {
            address 192.168.0.2/32;
          }
        }
      }
    }
    event-options {
      policy monitor-port-extenders {
        events [ snmp_trap_link_up snmp_trap_link_down ];
        then {
          event-script monitor-port-extenders-ex4200.slax {
            arguments {
              cntrlr-ip 192.168.0.1;
            }
          }
        }
      }
      event-script {
        file monitor-port-extenders-ex4200.slax;
      }
    }
    routing-options {
      static {
        rib-group ntp;
      }
      rib-groups {
        ntp {
          import-rib [ inet.0 jnu.inet.0 ];
          import-policy jnu-mgmt;
        }
```

```
                JNU {
                    import-rib [ jnu.inet.0 inet.0 ];
                    import-policy jnu-rib;
                }
            }
        }
        policy-options {
            policy-statement jnu-mgmt {
                from protocol static;
                then accept;
            }
            policy-statement reject-all {
                then reject;
            }
            policy-statement jnu-rib {
                term direct-local {
                    from protocol [ direct local ];
                    then accept;
                }
            }
        }
        routing-instances {
            jnu {
                instance-type vrf;
                interface vlan.4094;
                route-distinguisher 192.168.0.2:0;
                vrf-import reject-all;
                vrf-export reject-all;
                routing-options {
                    interface-routes {
                        rib-group inet JNU;
                    }
                    static {
                        rib-group JNU;
                        route 0.0.0.0/0 next-hop 192.168.0.1;
                    }
                }
            }
        }
        vlans {
            jnu {
                l3-interface vlan.4094;
            }
        }
    }
    jnu-module {
        system {
            scripts {
                commit {
                    file config-master-pem-commit-ex4200.slax;
                }
            }
        }
    }
}
system {
      autoinstallation {
        interfaces {
          xe-0/1/2 {
            bootp;
          }
          ge-0/0/0 {
            bootp;
          }
          ge-0/0/1 {
            bootp;
          }
          ge-0/0/2 {
            bootp;
          }
      ....
        }
      }
    }
```

Use the **show satellites** command to display the state of the satellite and to determine if the satellite is up and connected to the controller.

```
user@jnu1-ctrlr> show satellites
Satellite-System     State    Address         Model         Version    Downlinks
jnu-sat1              Up       192.168.0.2     ex4200-24t    12.3R7.7   xe-0/1/0
```

Deleting Configurations Related to a Specific Satellite Device from the JNU Controller

When you remove a specific satellite device from the JNU controller, manually deleting all configurations related to that satellite device from the JNU controller is a tedious process. You can use the **jnu-delete-satellite.slax** op script to delete all satellite device–specific configuration from the JNU controller. For example:

- This example shows how to delete a single configuration related to a satellite device from the JNU controller:

  ```
  user@jnu1-ctrlr> op jnu-delete-satellite satellite sat01
  ```
- This example shows how to delete multiple configurations related to a satellite device from the JNU controller:

  ```
  user@jnu1-ctrlr> op jnu-delete-satellite satellite sat01 downlink xe-0/1/0,xe-0/0/1
  ```

NOTE: Starting with JNU Release 1.3J3, you need to explicitly commit the configuration change after you delete all configurations related to a specific satellite device from the JNU controller.

Related Documentation

- Initializing JNU Mode on the MX Series Controller on page 31
- Initializing JNU Mode on the Satellite Device on page 36
- Installing the JNU Software on Satellite Devices on page 30

Adding Hardware to a Functioning JNU System

When you add hardware, such as an interface or an FPC, to a JNU system that is in operation, the JNU system does not recognize the newly added device by default. To enable the JNU system to recognize and use the new hardware, use the **jnu-add-delete-downlink.slax** op script.

- This example shows how to add a single interface to a JNU system:

  ```
  user@jnu1-ctrlr> op jnu-add-delete-downlink action set downlinks ge-0/0/0
  ```

- This example shows how to add multiple interfaces to a JNU system:

  ```
  user@jnu1-ctrlr> op jnu-add-delete-downlink action set downlinks
  ge-0/0/0,ge-0/0/1,ge-0/0/2
  ```

- This example shows how to add an FPC to a JNU system:

  ```
  user@jnu1-ctrlr> op jnu-add-delete-downlink action set fpc 1
  ```

CHAPTER 8

JNU Port Extender Mode With Junos OS CLI Interface

JNU Port Extender Mode Overview

To provide a robust, cohesive method of managing satellites, the JNU controller and satellite devices operate in *port-extender mode*. In this mode, a JNU group that consists of the controller and a number of satellites is regarded as a single, unified network entity, with the controller owning all the interface resources, including those residing on the satellites (remote line-cards) as extended ports. A satellite interface functions as the satellite port that is extended to the controller. In this mode, when an interface that resides on the satellite is enabled within the JNU group, a service VLAN (S-VLAN) ID or tag is used to transmit the data traffic from the remote interface of the satellite to the controller. Multiple VLANs can traverse the same satellite interface. All the customer VLANs (C-VLANs) are encapsulated using the same S-VLAN ID and are extended to the controller. Each C-VLAN is assigned a logical interface on the controller.

The satellite interface that is extended on to a controller behaves as a logical interface and appears native to the controller. This capability enables the controller to manage the satellite interfaces as though they were the controller's own, in-built interfaces. Each extended satellite port is a shared medium to the controller, and you can configure multiple VLANs on each port. Each VLAN is hosted on the controller on an individual interface.

Configuring JNU Port-Extender By Using the Junos OS CLI Settings

Starting with JNU Release 1.3J1, you can use the embedded, in-built Junos OS CLI interface as the mechanism to enable the JNU application on controller and satellites, and also to activate port-extender mode on these devices. By using the Junos OS configuration statements at the corresponding hierarchy levels, you can configure port-extender features and interfaces in the same manner in which you configure other types of interfaces or applications on a device running Junos OS. In this release, only the port-extender mode of operation is supported and the feature-rich or non-port-extender mode is not supported.

Related Documentation

- Configuring the Port Extender Interface Settings from the Junos OS CLI on page 46

Configuring the Port Extender Interface Settings from the Junos OS CLI

You can configure a satellite interface to function as the extended interface and host it on a controller by using the **port-extender** statement at the [**edit interfaces**] hierarchy level. You can also define the attributes of the satellite extended interface by using the options available at the [**edit interfaces port-extender interface-name**] hierarchy level.

The following configuration illustrates the attributes and settings that you can configure at the [**edit interfaces**] hierarchy level for port-extender interfaces:

```
interfaces {
  port-extender {
    sat1:ge-0/0/10 {
      description "JNU port extender";
      disable;
      traps;
      hold-time up 1000 down 1000;
      no-gratuitous-arp-reply;
      no-gratuitous-arp-request;
      optics-options {
        alarm low-light-alarm {
          syslog;
        }
      }
      ether-options {
     loopback;
     auto-negotiation;
     flow-control;
     link-mode automatic;
        speed {
          1g;
        }
      }
      mtu 4000;
      vlan-tagging;
      unit 0 {
        vlan-id 101; /* vlan-id 0 means untagged */
        family inet {
          address 10.10.0.2/30;
        }
      }
    }
  }
}
```

The following configuration illustrates the attributes and settings that you can configure at the [**edit interfaces**] hierarchy level for aggregated Ethernet port-extender interfaces:

```
interfaces {
  port-extender {
    sat1:ge-0/0/0 {
      ether-options {
        802.1ad ae1;
    }
  }
    sat1:ge-0/0/1 {
      ether-options {
        802.1ad ae1;
    }
  }
  sat1:ae1 {
    aggregated-ether-options {
  lacp {
    active;
  }
    }
    vlan-tagging;
    unit 0 {
  vlan-id 2001;
  family inet {
    address 137.0.0.1/30;
  }
    }
  }
   }
}
```

For JNU Release 1.3J3, only the following physical interface statements or options are supported for the satellite interfaces that are extended on the controller:

- **disable**
- **ether-options** (for EX Series switches and QFX Series devices)
- **aggregated-ether-options** (for EX Series Switches and QFX series devices)
- **mtu <mtu>**
- **description**
- **gratuitous-arp-reply**
- **hold-time**
- **no-gratuitous-arp-reply**
- **no-gratuitous-arp-request**

- **no-traps optics-options**
- **traps**

Related Documentation

- JNU Port Extender Mode Overview on page 45

Generating the Configuration on the Controller and Satellites by Using the Interface Alias Name

The JNU application expands the port-extender configuration on both the controller and satellite devices. The JNU application identifies the interface-alias configuration and creates the effective configuration on the controller and satellite, which involves the following operations:

- Create the 802.1q-tunneling configuration for the satellite by assigning the service VLAN for port-extension.
- Create the port-extender name in the satellite-name:physical-interface-name.logical-unit-number format as the interface-alias name.
- Configure the logical interface on the controller downlink aggregated Ethernet interface to contain the port-extender interface.
- Transfer the port-extender configurations to the logical interface on the controller that hosts the satellite extended interface.
- The port-extender physical interface configurations are forwarded from the controller to the physical interface of the satellite.

The following is an example of the configuration generated on a controller:

```
interfaces {
  ae479 {
    flexible-vlan-tagging;
    unit 21 {
      interface-alias "sat1:ge-0/0/0.88";
      vlan-tags outer 1001 inner 100;
      family inet {
        address 1.1.1.1/30;
      }
    }
  }
}
```

The following is an example of the configuration generated on a satellite:

```
ethernet-switching-options {
  dot1q-tunneling {
    ether-type 0x8100;
  }
}
interfaces {
  ge-0/0/0 {
        unit 0 {
            family ethernet-switching;
        }
    }
    vlans {
        jnu-port-extender-vlan1001 {
            vlan-id 1001;
            interface {
                ge-0/0/0.0;
            }
            dot1q-tunneling {
                customer-vlans 100;
            }
        }
    }
}
```

Restrictions While Configuring a Satellite Interface to be Extended on a Controller

The following limitations and conditions apply, when you configure a satellite interface to be extended and residing on a controller:

- Because flexible VLAN tagging and service VLAN ID parameters are generated by the JNU process, you cannot modify those configurations on the downlink interface to the satellite.
- You cannot configure **vlan-tagging** or **stacked-vlan-tagging** statements on the physical interface.
- You cannot modify the VLAN tags for the port extender logical interface on the controller.
- The outer tags used to perform tunneling for the satellite ports are unique within a JNU group (a number in the range 1 - 4093) regardless of the satellites.
- The outer S-VLAN tags used for tunneling traffic for the satellite ports can be reused in other bridge-domains on the controller for interfaces that are not connected or related to the extended satellite ports.

Committing Configurations with Port-Extender Interfaces

The JNU application performs a validation of the defined settings when you attempt to commit a configuration with port-extender attributes. For example, the system verifies whether the interface-alias name follows the format that is defined for the port-extender interface name syntax, such as **<satellite-name>:<physical-interface>.<vlan-id>**. The JNU process on the controller generates controller and satellite configurations. The JNU process propagates the satellite-specific configuration to the corresponding satellite device after performing a commit check operation.

If all the satellites approve the new configuration, the JNU process examines the result of the commit-check operation on the controller. If the controller and all satellites mark the new configuration as valid, the new configuration is activated. Otherwise, the commit process is canceled. The controller forwards only the portion of the configuration that has been modified from the previously committed configuration to the satellites.

Diagnosing Connectivity Failure Between the Controller and Satellites

On the satellite, when at least one port is extended from the satellite to the controller, **protocols uplink-failure-detection** is automatically configured on the JNU device to monitor the JNU link between the controller and the satellite. If the link fails, all the extended ports are disabled by **uplink-failure-detection**.

For example, if ports **ge-0/0/0** and **ge-0/0/2** are extended from the satellite to the controller, and the uplink on the satellite is **ae0**, the **uplink-failure-detection** effective on the satellite:

```
protocols {
    uplink-failure-detection {
        group jnu-port-extender {
            link-to-monitor {
                ae0;
            }
            link-to-disable {
                ge-0/0/9;
            }
        }
    }
}
```

Using SNMP and System Logs for Satellite Extended Interfaces

After the satellite interfaces are extended from the satellites to the controller, the network management system (NMS) can query the interface statistics of the satellite port-extender by querying the controller, because the port-extender interface is now anchored on the controller and becomes a first-class logical interface on the controller. The SNMP functionality on the controller is augmented to store information about the satellites. This is achieved by implementing JNU SNMP OIDs in the Juniper Networks Enterprise MIB (1.3.6.1.4.1.2636). The additional information available in the new MIBs:

```
satellite: chassis environment information
    physical interface statistics

NMS> snmpwalk -v2 -community public -host controller-fxp0 -oid enterprise.jnu
1.3.6.1.4.1.2636.3.1.13.1.89.1.1.192.168.0.2 = 192.168.0.2
1.3.6.1.4.1.2636.3.1.13.1.89.1.1.192.168.0.3 = 192.168.0.3
1.3.6.1.4.1.2636.3.1.13.1.89.1.2.192.168.0.2 = sat-ex4200-1
1.3.6.1.4.1.2636.3.1.13.1.89.1.2.192.168.0.3 = sat-qfx3500-1
1.3.6.1.4.1.2636.3.1.13.1.89.1.3.192.168.0.2 = ex4200-24f
1.3.6.1.4.1.2636.3.1.13.1.89.1.3.192.168.0.3 = qfx3500
....
```

Assigning Aggregated Ethernet Interfaces Dynamically to Be Extended from Satellites

You can configure an aggregated Ethernet interface to be used as the extended satellite interface to the controller. To extend an aggregated Ethernet interface from the satellite, you need to specify the physical interfaces to be link members in an aggregated Ethernet interface.

CoS Features on the Logical Interface Extension

Class-of-service (CoS) settings on logical interfaces that are of the port-extender type are not propagated to the satellites and they are enabled directly on the controller. These attributes include the following:

- rewrite-rules
- classifiers

The following example shows a CoS logical interface configuration:

```
class-of-service {
   interfaces {
      <ifd> {
         /* Existing configurations */
      }
      port-extender {
         <satellite>:<ifd> {
            /*Port-extender configurations */
            /* All satellite ifd class-of-service configurations */
            unit <n> {
               /* All port-extender ifl class-of-service configurations */
            }
         }
      }
   }
}
```

VLAN-ID List Support on Port-Extender Interfaces

You can configure a list of VLAN IDs on port-extender interfaces by including the **vlan-id-list** statement and specifying the VLAN IDs. You can configure a list of VLAN IDs on port-extender interfaces by using the following statement:

```
vlan-id-list [number number-number];
```

You can specify individual VLAN IDs with a space separating the ID numbers, a range of VLAN IDs with a dash separating the ID numbers, or a combination of individual VLAN IDs and a range of VLAN IDs.

RSTP and VSTP Support on Port-Extender Interfaces

Port-extender interfaces provide Layer 2 loop prevention through Rapid Spanning Tree Protocol (RSTP) and VLAN Spanning Tree Protocol (VSTP). While configuring RSTP or VSTP, you should configure the controller as the root bridge or configure the root bridge northbound to the controller. This configuration prevents the controller–satellite device link from going into blocking state.

The following example shows the RSTP and VSTP configuration on a port-extender interface:

```
protocols {
  rstp {
    interface <interface-name> {
      /* STP interface configurations */
    }
    port-extender {
      interface <port-extender-interface-name> {
        /* STP interface configurations */
      }
    }
  }
  vstp {
    interface <interface-name> {
      /* STP interface configurations */
    }
    port-extender {
      interface <port-extender-interface-name> {
        /* STP interface configurations */
      }
    }
  }
}
```

RSTP

When RSTP is configured on a port-extender interface, the entire RSTP configuration is configured on the controller. Only the interfaces are configured on the corresponding satellite devices.

For example, consider the following configuration:

```
protocols {
  rstp {
    interface ge-0/0/0;
    interface ge-0/0/1;
    port-extender {
      interface sat1:ge-0/0/0;
      interface sat2:xe-0/1/0;
    }
    max-age 40;
  }
}
```

The effective configuration on the controller is:

```
protocols {
  rstp {
    interface ge-0/0/0;
    interface ge-0/0/1;
   interface ae0;       /* downlink to satellite sat1 */
   interface ae1;       /* downlink to satellite sat2 */
    max-age 40;
  }
```

The effective configuration on the sat1 satellite device is:

```
protocols {
    rstp {
       bridge-priority 60k
       interface ge-0/0/0 {
         no-root-port;
        }
       interface ae0{
        bpdu-timeout-action {
           alarm
           }
        }
         /* uplink to controller */
       max-age 40;
    }
}
```

The effective configuration on the sat2 satellite device is:

```
protocols {
    rstp {
       bridge-priority 60k
       interface Xe-0/1/0 {
        no-root-port;
        }
       interface ae0{
        bpdu-timeout-action {
          alarm
           }
       }
         /* uplink to controller */
       max-age 40;
    }
}
```

VSTP

When a port-extender interface is configured for VSTP, JNU the different access ports that use the same VLAN ID in the same VLAN on the satellite device and extends the VLAN to the MX Series controller. The MX Series controller uses one logical interface to host the bridged VLAN. VSTP, on both the controller and the satellite device, runs on the customer VLAN (C-VLAN) directly. For example, consider the following configuration:

```
protocols {
     vstp {
        interface ge-0/0/0;
        interface ge-0/0/1;
        port-extender {
           interface sat1:ge-0/0/10;
           interface sat1:ge-0/0/11;
           interface sat2:xe-0/1/0;
        }
        vlan 100;
    }
}
```

The effective configuration on the controller is:

```
protocols {
  vstp {
    interface ge-0/0/0;
    interface ge-0/0/1;
   interface ae0;        /* downlink to satellite sat1 */
   interface ae1;        /* downlink to satellite sat2 */
    vlan 100;
  }
```

The effective configuration on the sat1 satellite device is:

```
protocols {
  vstp {
   bridge-priority 60k
   interface ge-0/0/10 {
    no-root-port;
    }
   interface ge-0/0/11 0 {
    no-root-port;
    }
   interface ae0 {    /* uplink to controller */
    bpdu-timeout-action {
      alarm
       }
   van 100;
  }
}
vlans {
  jnu-port-extender-vlan100 {
    vlan-id 100;
    interface ae0.0;
    interface ge-0/0/10.0;
    interface ge-0/0/11.0;
  }
}
```

NOTE: VSTP cannot be configured on provider edge logical interfaces that have a vlan-id-list defined under them. If VSTP is configured, you receive a commit error.

CHAPTER 9

JNU Operational Mode Commands

Operational mode commands used with JNU are at the user@host> prompt. You run operational mode commands on the JNU controller.

- show chassis hardware jnu device all
- show interfaces
- show port-extenders
- show satellites
- show version jnu device all
- request-jnu-controller

show chassis hardware jnu device all

Syntax show chassis hardware jnu device all

Release Information Command introduced in JNU 1.3.

Description Display a list of all devices in the JNU group including the hardware version level and serial number.

Required Privilege Level view

Sample Output

show chassis hardware jnu device all

```
user@jnu1-ctrlr> show chassis hardware jnu device all
Device: jnu-sat1
----------------------------------
Hardware inventory:
Item             Version  Part number  Serial number     Description
Chassis                                JN122E469AFC      MX240
Midplane         REV 07   760-021404   ACRB4299          MX240 Backplane
FPM Board        REV 05   760-021392   CABZ0480          Front Panel Display
PEM 0            Rev 07   740-017343   QCS1128A03C       DC Power Entry Module
Routing Engine 0 REV 18   740-015113   9009163234        RE-S-1300
Routing Engine 1 REV 07   740-015113   1000741877        RE-S-1300
CB 0             REV 15   710-021523   CABZ7189          MX SCB
CB 1             REV 09   710-021523   ABBH2384          MX SCB
FPC 1            REV 10   750-044444   CACB3249          MPCE Type 2 3D P
  CPU            REV 06   711-038484   CACH7125          MPCE PMB 2G
  MIC 0          REV 27   750-028392   CABZ8382          3D 20x 1GE(LAN) SFP
    PIC 0                 BUILTIN      BUILTIN           10x 1GE(LAN) SFP
      Xcvr 0     REV 01   740-038291   AN1310YLV3        SFP-T
      Xcvr 1     REV 01   740-038291   AN1310YLVQ        SFP-T
      Xcvr 2     REV 01   740-031851   PND6XTW           SFP-SX
      Xcvr 4     REV 01   740-038291   AN1310YKXU        SFP-T
    PIC 1                 BUILTIN      BUILTIN           10x 1GE(LAN) SFP
      Xcvr 0     REV 02   740-011613   PQN1056           SFP-SX
      Xcvr 1     REV 02   740-011613   PQN10CP           SFP-SX
      Xcvr 2     REV 02   740-011613   PQN10BD           SFP-SX
      Xcvr 3     REV 02   740-011613   PQN10GN           SFP-SX
      Xcvr 4     REV 02   740-011613   PQN0MSL           SFP-SX
      Xcvr 5     REV 02   740-011613   PQN11BU           SFP-SX
      Xcvr 6     REV 02   740-011613   PQN108H           SFP-SX
      Xcvr 7     REV 02   740-011613   PQN0MSM           SFP-SX
      Xcvr 8     REV 02   740-011613   PQN0MST           SFP-SX
      Xcvr 9     REV 02   740-011613   PQN0MQE           SFP-SX
  MIC 1          REV 26   750-028392   CAAP9050          3D 20x 1GE(LAN) SFP
    PIC 2                 BUILTIN      BUILTIN           10x 1GE(LAN) SFP
      Xcvr 0     REV 02   740-011613   PQN0MYD           SFP-SX
      Xcvr 1     REV 02   740-011613   PQN11B4           SFP-SX
      Xcvr 2     REV 02   740-011613   PQN0MYQ           SFP-SX
      Xcvr 3     REV 02   740-011613   PQN10N8           SFP-SX
      Xcvr 4     REV 02   740-011613   PQN0MY9           SFP-SX
      Xcvr 5     REV 02   740-011613   PQN11BK           SFP-SX
      Xcvr 6     REV 02   740-011613   PQN10HF           SFP-SX
      Xcvr 7     REV 02   740-011613   PQN0MZ2           SFP-SX
    PIC 3                BUILTIN      BUILTIN           10x 1GE(LAN) SFP
      Xcvr 0     REV 01   740-031851   PLB25FK           SFP-SX
      Xcvr 1     REV 01   740-031851   PM20BBX           SFP-SX
      Xcvr 2     REV 01   740-031851   AM1122SUP7T       SFP-SX
      Xcvr 3     REV 02   740-011613   PP72VBE           SFP-SX
      Xcvr 9     REV 02   740-011613   PH25HCQ           SFP-SX
  QXM 0          REV 06   711-028408   CACH0613          MPC QXM
  QXM 1          REV 06   711-028408   CACH0321          MPC QXM
Fan Tray 0       REV 01   710-030216   CABV1747          Enhanced Fan Tray

Device: jnu-sat2
----------------------------------
Hardware inventory:
Item             Version  Part number  Serial number     Description
Chassis                                GG0211086275      EX4500-40F
Routing Engine 0 REV 10   750-035700   GG0211086275      EX4500-40F
FPC 0            REV 10   750-035700   GG0211086275      EX4500-40F
  CPU                     BUILTIN      BUILTIN           FPC CPU
  PIC 0                   BUILTIN      BUILTIN           40x 1/10GE
    Xcvr 0       REV 02   740-011613   PH25HCP           SFP-SX
    Xcvr 1       REV 01   740-038291   AN1310YLWR        SFP-T
    Xcvr 2       REV 02   740-011613   PQN0MZ9           SFP-SX
    Xcvr 3       REV 02   740-011613   PQN0MQD           SFP-SX
    Xcvr 4       REV 02   740-011613   PHE2JB0           SFP-SX
    Xcvr 5       REV 01   740-031851   PNB539S           SFP-SX
    Xcvr 6       REV 01   740-031851   PM307TN           SFP-SX
    Xcvr 7       REV 01   740-011783   PAG40S1           SFP-LX10
    Xcvr 10      REV 01   740-011613   PDK32NH           SFP-SX
    Xcvr 11      REV 01   740-031851   PM15CHX           SFP-SX
  PIC 3          REV 01   711-031050   EC0210118389      2x 32GE Virtual Chassis
 Module
Power Supply 1   REV 02   740-029654   EK0711033453      PS 1000W AC FORWARD AIR
 FLOW CONTROL
Fan Tray                                                 Fan Tray, Front to back
 Airflow
```

show interfaces

Syntax **show interfaces jnu [device *device-name*] [*interface-name*] (brief | extensive | terse | detail)]**

Release Information Command introduced in JNU 1.3.

Description Displays the specified level of interface information for one or all interfaces on the JNU controller and satellite devices.

Options **device *device-name***—Name of the controller or satellite device for which you want to display the information.

interface-name—Name of the interface you want to view, such as **sat1:ge-0/0/0.100**. To view information for all interfaces in the JNU group, omit the interface name.

brief—(Optional) Display the specified level of output.

extensive—(Optional) Display the specified level of output.

terse—(Optional) Display the specified level of output.

detail—(Optional) Display the specified level of output.

Required Privilege Level view

Sample Output

show interfaces jnu

```
user@jnu1-ctrlr> show interfaces jnu
Device: controller
   ----------------------------------
   Interface              Admin Link Proto  Local            Remote
   lc-0/0/0               up    up
   lc-0/0/0.32769         up    up   vpls
   pfe-0/0/0              up    up
   pfe-0/0/0.16383        up    up   inet
                                     inet6
   pfh-0/0/0              up    up
   pfh-0/0/0.16383        up    up   inet
   si-0/0/0               up    up
   si-0/0/0.0             up    up   inet
                                     inet6  fe80::2a0:a500:7b:e74f/64
   si-0/0/0.1             up    up   inet
   si-0/0/0.2             up    up   inet
   xe-0/0/0               up    down
   xe-0/0/1               up    down
   xe-0/0/2               up    down
   xe-0/0/3               up    down
          .....
   Device: sat-ex4200-1
   ----------------------------------
   Interface              Admin Link Proto  Local            Remote
   ge-0/0/1               up    up
   ge-0/0/2               up    down
```

```
ge-0/0/3                up    down
ge-0/0/4                up    down
ge-0/0/5                up    down
       .....
Device: sat-ex4200-2
----------------------------------
Interface               Admin Link Proto  Local            Remote
ge-0/0/1                up    up
       .....
```

show port-extenders

Syntax **show port-extenders**

Release Information Command introduced in JNU 1.3.

Description Display the mapping between a satellite interface that is extended to a controller and the logical interface on the controller that hosts or anchors the extended satellite interface.

Required Privilege Level view

Sample Output

show port-extenders

```
user@jnu1-ctrlr> show port-extenders

Port Extender Interface        Controller IFL     S-VLAN     C-VLAN
jnu-sat1:ge-0/0/10.10            ae0.16384          11         10
jnu-sat1:ge-0/0/7.10             ae0.16383          8          10
```

show satellites

Syntax **show satellites**

Release Information Command introduced in JNU 1.3.

Description Displays the state of each satellite to determine if the satellite is up and connected to the controller.

Required Privilege Level view

Sample Output

show satellites

```
user@jnu1-ctrlr> show satellites
Satellite-System            State    Address          Model         Version
 Downlinks

jnu-sat1                      Up       192.168.0.2      ex4500-40f    12.3R5.7
   ge-1/0/1,ge-1/2/2
```

show version jnu device all

Syntax **show version jnu device all**

Release Information Command introduced in JNU 1.3.

Description Displays the current software release along with other version details about the JNU device.

Required Privilege Level view

Sample Output

show version jnu device all

```
user@jnu1-ctrlr> show version jnu device all
Device: jnu-sat1
----------------------------------
Hostname: jnu-sat1
Model: mx240
Junos: 13.3-20140603.0
JUNOS Base OS boot [13.3-20140603.0]
JUNOS Base OS Software Suite [13.3-20140603.0]
JUNOS Kernel Software Suite [13.3-20140603.0]
JUNOS Crypto Software Suite [13.3-20140603.0]
JUNOS Packet Forwarding Engine Support (M/T/EX Common) [13.3-20140603.0]
JUNOS Packet Forwarding Engine Support (MX Common) [13.3-20140603.0]
JUNOS Online Documentation [13.3-20140603.0]
JUNOS Services AACL Container package [13.3-20140603.0]
JUNOS Services Application Level Gateways [13.3-20140603.0]
JUNOS AppId Services [13.3-20140603.0]
JUNOS Border Gateway Function package [13.3-20140603.0]
JUNOS Services Captive Portal and Content Delivery Container package
[13.3-20140603.0]
JUNOS Services HTTP Content Management package [13.3-20140603.0]
JUNOS IDP Services [13.3-20140603.0]
JUNOS Services Jflow Container package [13.3-20140603.0]
JUNOS Services LL-PDF Container package [13.3-20140603.0]
JUNOS Services MobileNext Software package [13.3-20140603.0]
JUNOS Services Mobile Subscriber Service Container package [13.3-20140603.0]
JUNOS Services NAT [13.3-20140603.0]
JUNOS Services PTSP Container package [13.3-20140603.0]
JUNOS Services RPM [13.3-20140603.0]
JUNOS Services Stateful Firewall [13.3-20140603.0]
JUNOS Voice Services Container package [13.3-20140603.0]
JUNOS Services Crypto [13.3-20140603.0]
JUNOS Services SSL [13.3-20140603.0]
JUNOS Services IPSec [13.3-20140603.0]
JUNOS platform Software Suite [13.3-20140603.0]
JUNOS Routing Software Suite [13.3-20140603.0]
JUNOS Runtime Software Suite [13.3-20140603.0]
JUNOS py-base-i386 [13.3-20140603.0]
JUNOS Node Unifier Package [1.3J2.4-mx_13.3]

Device: jnu-sat2
----------------------------------
  fpc0:
```

```
--------------------------------------------------------------------------
Hostname: jnu-sat2
Model: ex4500-40f
JUNOS Base OS boot [12.3R5.7]
JUNOS Base OS Software Suite [12.3R5.7]
JUNOS Kernel Software Suite [12.3R5.7]
JUNOS Crypto Software Suite [12.3R5.7]
JUNOS Online Documentation [12.3R5.7]
JUNOS Enterprise Software Suite [12.3R5.7]
JUNOS Packet Forwarding Engine Enterprise Software Suite [12.3R5.7]
JUNOS Routing Software Suite [12.3R5.7]
JUNOS Web Management [12.3R5.7]
JUNOS FIPS mode utilities [12.3R5.7]
JUNOS Node Unifier Package [1.3J2.3-ex_12.3]
```

request-jnu-controller

Syntax **request jnu controller**

Release Information Command introduced in JNU 1.3J1.

Description The first time you initialize JNU on the controller, you must run the **request jnu controller** command.

Related Documentation
- Initializing JNU Mode on the MX Series Controller on page 31

Sample Output

```
user@jnu1-ctrlr> request jnu controller
Initializing Controller...



JNU Controller configuration completed
```

PART 4

Monitoring and Troubleshooting in the JNU Network

CHAPTER 10

Monitoring in the JNU Network

Centralized Collection of SNMP Statistics and Log Messages Overview

The JNU software provides a single point of collecting SNMP statistics and sending SNMP traps to the SNMP server. You use the Junos OS SNMP proxy feature to set up the controller as a proxy SNMP agent through which the network management system (NMS) can query satellite devices.

The NMS polls the controller for SNMP statistics on both the controller and the satellite devices. SNMP statistics from the satellite devices are routed through the controller. A Network Address Translation (NAT) configuration set up by the controller initialization process is used to translate the source address of the satellite devices to the source address of the controller for traffic sent to the SNMP server. This process means that all SNMP traffic originates from one source address on the controller.

SNMP Community Strings

Each satellite in the JNU group uses a different community string that is based on the hostname assigned to the satellite during the initialization process. The format of the string is *controller-community-string.satellite-host-name*. For example, if you configure the read-only community string to be public, and the satellite hostname to be sat1, the community string for the satellite is public:sat1.

Collecting Log Messages

The JNU software provides a single point of collecting logging messages and sending them to the system log server. The controller sends all system log messages for the JNU group of satellites and controller to the server. The NAT configuration set up by the controller initialization process is used to translate the source address of the satellite devices to the source address of the controller for traffic sent to the system log server.

This process means that all logging traffic originates from one source address on the controller.

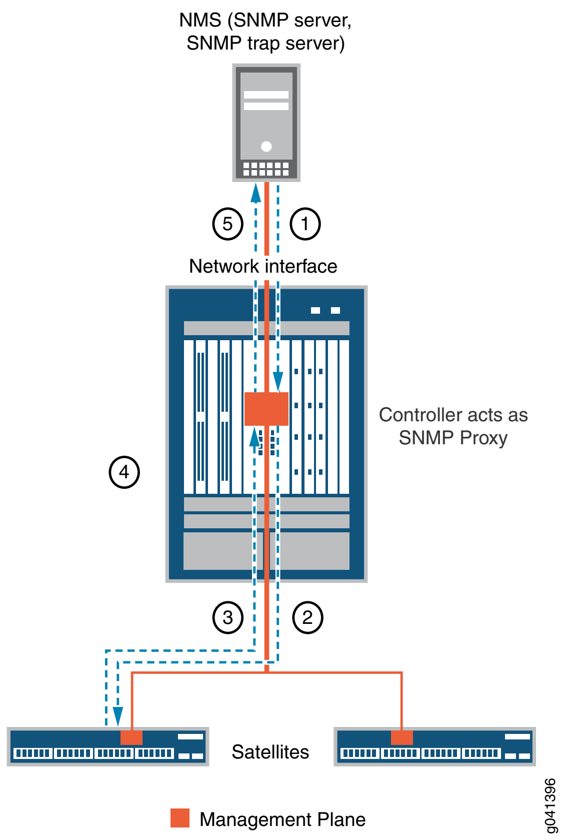
Related Documentation

- Configuring the JNU Controller as an SNMP Proxy Agent on page 73
- SNMP Get Process in the JNU Network on page 69
- SNMP Trap Process in the JNU Network on page 70
- System Logging in the JNU Network on page 71

SNMP Get Process in the JNU Network

Figure 6 on page 69 shows the SNMP Get process in the JNU network.

Figure 6: SNMP Get Process



1—SNMP server sends a Get request message to the controller with the community string for a satellite.	4—The controller receives the Get reply and processes it as an SNMP client.
2—The controller, acting as SNMP proxy, terminates the Get request from the SNMP server. It then acts as an SNMP client, and sends the request to the appropriate satellite.	5—Acting as an SNMP server, the controller then sends the Get reply message to the SNMP server.
3—The satellite sends a Get reply message to the controller.	

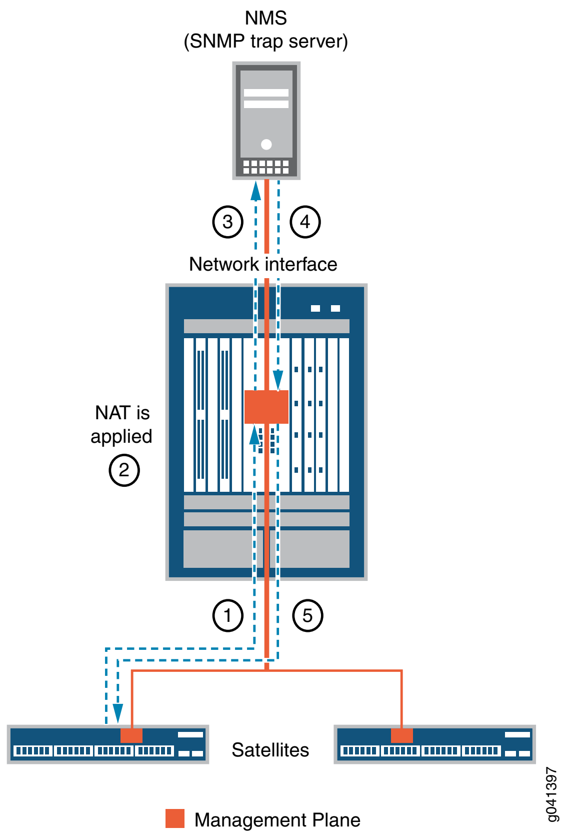
Related Documentation

- Centralized Collection of SNMP Statistics and Log Messages Overview on page 67
- Configuring the JNU Controller as an SNMP Proxy Agent on page 73
- SNMP Trap Process in the JNU Network on page 70

SNMP Trap Process in the JNU Network

Figure 7 on page 70 shows the SNMP trap process in the JNU network.

Figure 7: SNMP Trap Process



1—The satellite sends an SNMP trap notification or inform request to the controller.	4—For SNMPv3, the server responds to the inform request with an acknowledgment that it sends to the controller.
2—The controller applies NAT to the trap message so that the message source is the management IP address of the controller.	5—The controller sends the acknowledgment to the satellite.
3—The controller sends the trap message to the SNMP server.	

Related Documentation

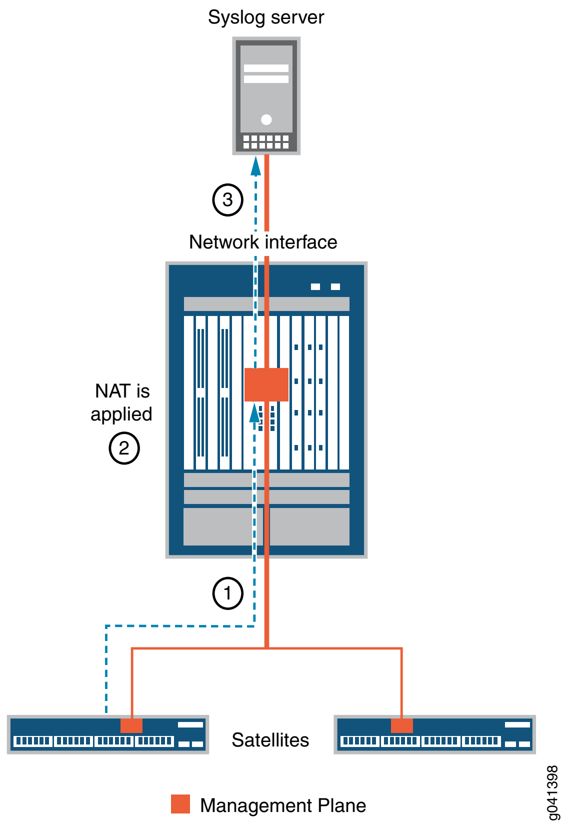
- Centralized Collection of SNMP Statistics and Log Messages Overview on page 67
- Configuring the JNU Controller as an SNMP Proxy Agent on page 73
- SNMP Get Process in the JNU Network on page 69
- System Logging in the JNU Network on page 71

System Logging in the JNU Network

Each device in the JNU group, including the controller, originates its own system log messages (called syslog messages). The hostname in messages sent from satellites and the controller is the hostname configured during the JNU initialization process. The syslog server uses the hostname to identify the JNU device that originated the message.

Figure 8 on page 71 shows how log messages are processed for JNU satellites.

Figure 8: System Logging Collection Process



1—The satellite sends a syslog message to the controller.	3—The controller sends the log message to the syslog server.
2—The controller applies NAT to the syslog message so that the message source is the management IP address of the controller.	

Related Documentation

- Centralized Collection of SNMP Statistics and Log Messages Overview on page 67
- SNMP Get Process in the JNU Network on page 69
- SNMP Trap Process in the JNU Network on page 70

Network Time Protocol (NTP) in the JNU Network

When you initialize the controller, you can specify the address of an NTP server. If you do so, the controller synchronizes its time to the external NTP server, and the controller then acts as the NTP server for all the satellites in the JNU group. During satellite initialization, JNU creates an NTP configuration that sets the NTP server address to the controller downlink IP address.

Configuring the JNU Controller as an SNMP Proxy Agent

You use the Junos OS SNMP proxy feature to set up the controller as a proxy SNMP agent through which the network management system (NMS) can query satellite devices.

When the JNU controller acts as the proxy SNMP agent for the satellite devices, the NMS specifies the community name (for SNMPv1 and SNMPv2) or the context and security name (for SNMPv3) of the satellite from which it requires the information. If you have configured authentication and privacy methods and passwords for SNMPv3, those parameters are also specified in the query for SNMPv3 information.

The community and security configuration for the proxy should match the corresponding configuration on the satellite device that is to be managed.

If you configure SNMP when you run the controller initialization process, the JNU software creates an SNMP proxy configuration with SNMPv2 support. For example:

```
proxy jnu-sat1 {
   device-name 192.168.0.2;
   version-v2c {
      snmp-community public:jnu-sat1;
   }
   routing-instance jnu-vrf;
}
```

Use this procedure if you need to add SNMPv3 support.

To configure the controller as an SNMP proxy agent:

1. Create a proxy configuration, and assign a name to the configuration.

   ```
   user@jnu1-ctrlr# edit snmp proxy proxy-ctrlr
   ```

2. Assign the proxy configuration to a satellite device.

   ```
   [edit snmp proxy proxy-ctrlr]
   user@jnu1-ctrlr# set device-name jnu-sat-1
   ```

3. (Optional) Configure SNMP version 3. Specify a security name to be used for messaging security and user access control. Specify the ID of the SNMP context that is accessible to the SNMP proxy.

   ```
   [edit snmp proxy proxy-ctrlr]
   user@jnu1-ctrlr# edit version-v3
   [edit snmp proxy proxy-ctrlr version-v3]
   user@jnu1-ctrlr# set security-name jnu-user
   user@jnu1-ctrlr# set context jnu
   ```

   NOTE: This security name must match the security name configured at the [edit snmp v3 target-parameters *target-parameters-name* parameters] hierarchy level when you configure traps.

You can use the **show snmp proxy** operational mode command to view proxy details on a device. The **show snmp proxy** command returns the proxy names, device names, SNMP version, community and security, and context information.

Related Documentation

- Centralized Collection of SNMP Statistics and Log Messages Overview on page 67
- SNMP Get Process in the JNU Network on page 69
- SNMP Trap Process in the JNU Network on page 70