Best Practices for SRX Series Cluster Management

Release

NCE0067

Modified: 2016-10-13

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

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

Best Practices for SRX Series Cluster Management
NCE0067

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 http://www.juniper.net/support/eula.html. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

CHAPTER 1

SRX Chassis Cluster Overview

About This Document

This document provides the best practices and methods for monitoring high-end SRX Series chassis clusters using instrumentation available in the Junos® operating system (Junos OS) such as SNMP, the NETCONF XML management protocol, and syslog. This document is applicable to all high-end SRX Series Services Gateways.

Chassis Cluster Overview

A chassis cluster provides high redundancy. Before you begin managing an SRX Series chassis cluster, you need to have a basic understanding of how the cluster is formed and how it works.

Chassis cluster functionality includes:

- A resilient system architecture, with a single active control plane for the entire cluster and multiple Packet Forwarding Engines. This architecture presents a single device view of the cluster.
- Synchronization of configuration and dynamic runtime states between nodes within a cluster.
- Monitoring of physical interfaces and failover if the failure parameters cross a configured threshold.

To form a chassis cluster, a pair of the same model of supported SRX Series devices are combined to act as a single system that enforces the same overall security. Chassis cluster formation depends on the model. For SRX3400 and SRX3600 chassis clusters, the location and type of Services Processing Cards (SPCs), I/O cards (IOCs), and Network Processing Cards (NPCs) must match in the two devices. For SRX5600 and SRX5800 chassis clusters, the placement and type of SPCs must match in the two clusters.

An SRX Series chassis cluster is created by physically connecting two identical cluster-supported SRX Series devices using a pair of the same type of Ethernet connections. The connection is made for both a control link and a fabric (data) link between the two devices.

Hardware Requirements

This following hardware components are required to support chassis clusters:

- SRX1400, SRX3400, SRX3600, SRX5600, or SRX5800 Services Gateways
- SRX100, SRX200, or SRX600 Series branch devices

Software Requirements

This following software components are required to support chassis clusters:

- Junos OS Release 9.5 or later
- Junos OS Release 10.1R2 or later for SRX Series branch device virtual chassis management and in-band management

For more information about how to form clusters, see the *Junos OS Security Configuration Guide*.

Control Plane and Data Plane

When creating a chassis cluster, the control ports on the respective nodes are connected to form a control plane that synchronizes configuration and kernel state to facilitate the chassis clustering of interfaces and services. The data planes on the respective nodes are connected over the fabric ports to form a unified data plane.

The control plane software operates in active/backup mode. When configured as a chassis cluster, the two nodes back up each other, with one node acting as the primary device and the other as the secondary device, ensuring stateful failover of processes and services in the event of a system or hardware failure. If the primary device fails, the secondary device takes over processing the traffic.

The data plane software operates in active/active mode. In a chassis cluster, session information is updated as traffic traverses either device, and this information is transmitted between the nodes over the fabric link to guarantee that established sessions are not dropped when a failover occurs. In active/active mode, it is possible for traffic to ingress the cluster on one node and egress the cluster from the other node.

When a device joins a cluster, it becomes a node of that cluster. With the exception of unique node settings and management IP addresses, nodes in a cluster share the same configuration.

You can deploy up to 15 chassis clusters in a Layer 2 domain. Clusters and nodes are identified in the following ways:

- A cluster is identified by a cluster ID (cluster-id) specified as a number from 1 through 15.
- A cluster node is identified by a node ID (node) specified as a number from 0 to 1.

Chassis clustering of interfaces and services is provided through redundancy groups and primacy within groups. A redundancy group is an abstract construct that includes and manages a collection of objects. A redundancy group contains objects on both nodes. A redundancy group is primary on one node and backup on the other at any time. When a redundancy group is said to be primary on a node, its objects on that node are active. See the *Junos OS Security Configuration Guide* for detailed information about redundancy groups. Redundancy groups are the concept in Junos OS Services Redundancy Protocol (JSRP) clustering that is similar to a virtual security interface (VSI) in Juniper Networks ScreenOS® Software. Basically, each node has an interface in the redundancy group, where only one interface is active at a time. A redundancy group is a concept similar to a virtual security device (VSD) in ScreenOS Software. Redundancy group 0 is always for the control plane, while redundancy group 1+ is always for the data plane ports.

Related Documentation

- Chassis Cluster Descriptions and Deployment Scenarios on page 9
- Best Practices for Managing a Chassis Cluster on page 25

Chassis Cluster Descriptions and Deployment Scenarios

- Various Deployments of a Chassis Cluster on page 9
- Connecting Primary and Secondary Nodes on page 12
- Managing Chassis Clusters on page 18
- Configuring Devices for In-Band Management and Administration on page 19
- Managing SRX Series Branch Chassis Clusters Through the Primary Node on page 20
- Communicating with a Chassis Cluster Device on page 23

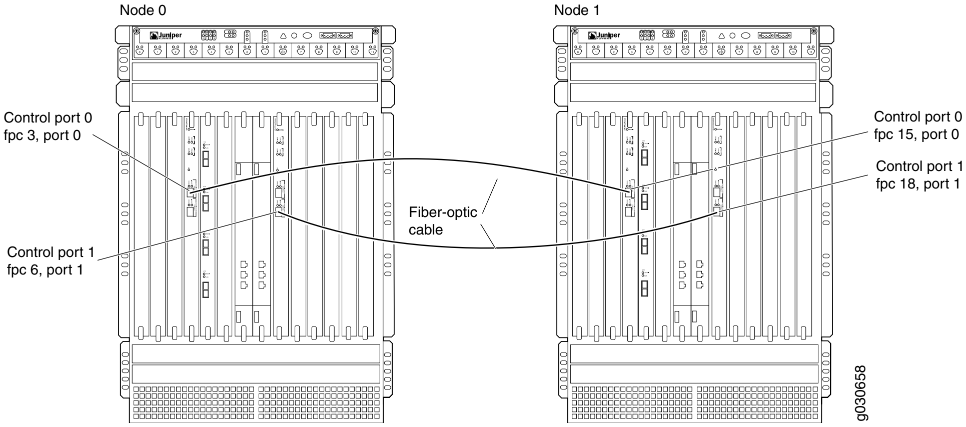
Various Deployments of a Chassis Cluster

Firewall deployments can be active/passive or active/active.

Active/passive chassis cluster mode is the most common type of chassis cluster firewall deployment and consists of two firewall members of a cluster. One actively provides routing, firewall, NAT, virtual private network (VPN), and security services, along with maintaining control of the chassis cluster. The other firewall passively maintains its state for cluster failover capabilities should the active firewall become inactive.

SRX Series devices support the active/active chassis cluster mode for environments in which you want to maintain traffic on both chassis cluster members whenever possible. In an SRX Series device active/active deployment, only the data plane is in active/active mode, while the control plane is actually in active/passive mode. This allows one control plane to control both chassis members as a single logical device, and in case of control plane failure, the control plane can fail over to the other unit. This also means that the data plane can fail over independently of the control plane. Active/active mode also allows for ingress interfaces to be on one cluster member, with the egress interface on the other. When this happens, the data traffic must pass through the data fabric to go to the other cluster member and out of the egress interface. This is known as Z mode. Active/active mode also allows the routers to have local interfaces on individual cluster members that are not shared among the cluster in failover, but rather only exist on a single chassis. These interfaces are often used in conjunction with dynamic routing protocols that fail traffic over to the other cluster member if needed. Figure 1 on page 10 shows two SRX5800 devices in a cluster.

Figure 1: SRX5800 Devices in a Cluster



To effectively manage the SRX clusters, network management applications must do the following:

- Identify and monitor primary and secondary nodes
- Monitor redundancy groups and interfaces
- Monitor control and data planes
- Monitor switchovers and failures

Figure 2 on page 11 shows the SRX Series high-end devices configuration for out-of-band management and administration.

Figure 2: SRX Series High-End Chassis Cluster Setup Connecting to a Management Station Through a Backup Router

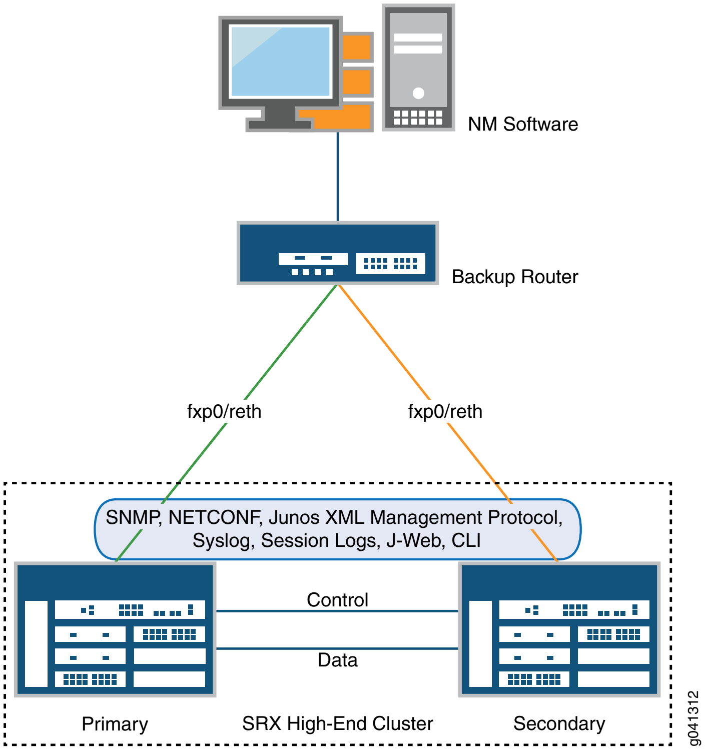
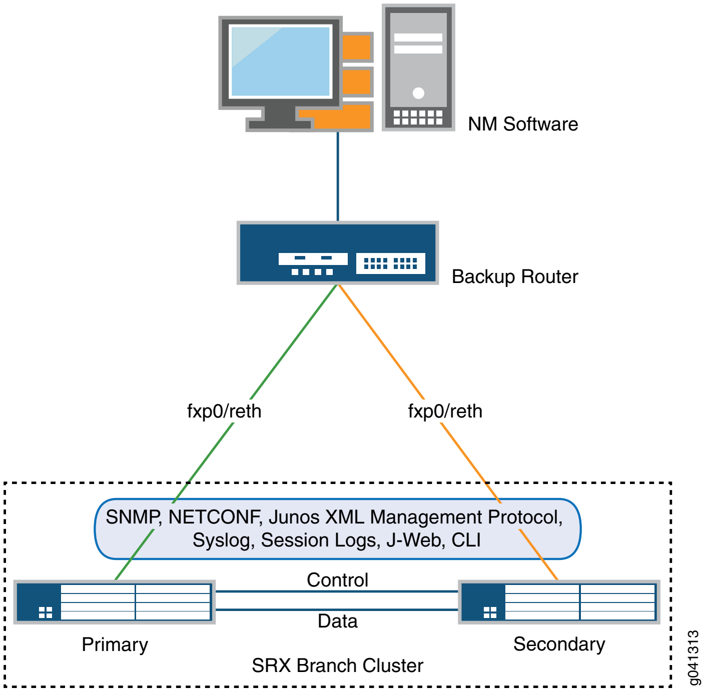


Figure 3 on page 12 shows the SRX Series branch devices configuration for out-of-band management and administration.

Figure 3: Branch SRX Series Cluster Setup Connecting to a Management Station Through a Backup Router



Connecting Primary and Secondary Nodes

The following is the best configuration to connect to the cluster from management systems. This configuration ensures that the management system is able to connect to both primary and secondary nodes.

```
user@host# show groups
node0 {
  system {
    host-name SRX3400-1;
    backup-router 172.19.100.1 destination 10.0.0.0/8;
    services {
      outbound-ssh {
        client nm-10.200.0.1 {
          device-id A9A2F7;
          secret "$9$T3Ct0BIEylIRs24JDjO1IRrevWLx-VeKoJUDkqtu0BhS"; ##
            SECRET-DATA
          services netconf;
          10.200.0.1 port 7804;
        }
      }
    }
    syslog {
      file messages {
        any notice;
        structured-data;
      }
    }
    }
    interfaces {
        fxp0 {
            unit 0 {
                family inet {
                    address 172.19.100.164/24;
                }
            }
        }
    }
}
node1 {
    system {
        host-name SRX3400-2;
        backup-router 172.19.100.1 destination 10.0.0.0/8;
        services {
            outbound-ssh {
                client nm-10.200.0.1 {
                    device-id F007CC;
                    secret "$9$kPFn9ApOIEAtvWXxdVfTQzCt0BlESrIR-VsYoa9At0Rh"; ##
                        SECRET-DATA
                    services netconf;
                    10.200.0.1 port 7804;
                }
            }
        }
    }
}

       # The following syslog configuration is not applicable for Branch SRX
       Series Services Gateways:
syslog {
    file default-log-messages {
        any notice;
        structured-data;
    }
}
interfaces {
    fxp0 {
        unit 0 {
            family inet {
                address 172.19.100.165/24;
            }
        }
    }
}
```

```
user@host# show apply-groups
apply-groups "${node}";
{primary:node0} [edit]
```

```
user@host# show interfaces
interfaces {
    fxp0 {
        unit 0 {
            family inet {
```

```
            filter {
               input protect;
            }
            address 172.19.100.166/24 {
               master-only;
            }
         }
      }
   }
}
{primary:node0}[edit]

user@host# show snmp
location "Systest lab";
contact "Lab Admin";
view all {
   oid .1 include;
}
community srxread {
   view all;
}
community srxwrite {
   authorization read-write;
}
trap-options {
   source-address 172.19.100.166;
   }
trap-group test {
   targets {
      10.200.0.1;
   }
}
v3 {
   vacm {
      security-to-group {
         security-model usm {
            security-name test123 {
               group test1;
            }
            security-name juniper {
               group test1;
            }
         }
      }
      access {
         group test1 {
            default-context-prefix {
               security-model any {
                  security-level authentication {
                     read-view all;
                  }
               }
            }
            context-prefix MGMT_10 {
               security-model any {
```

```
                        security-level authentication {
                            read-view all;
                        }
                    }
                }
            }
        }
    }
    target-address petserver {
        address 116.197.178.20;
        tag-list router1;
        routing-instance MGMT_10;
        target-parameters test;
    }
    target-parameters test {
        parameters {
            message-processing-model v3;
            security-model usm;
            security-level authentication;
            security-name juniper;
        }
        notify-filter filter1;
    }
    notify server {
        type trap;
        tag router1;
    }
    notify-filter filter1 {
        oid .1 include;
    }
}
{primary:node0}[edit]
```

```
user@host# show routing-options
static {
    route 10.200.0.1/32 next-hop 172.19.100.1;
    }
    primary:node0}[edit]
    root@SRX3400-1# show firewall
    term permit-ssh {
        from {
            source-address {
                10.200.0.0/24;
            }
            protocol tcp;
            destination-port [ ssh telnet ];
        }
        then accept;
    }
    term permit-udp {
        from {
            source-address {
                207.17.137.28/32;
            }
            protocol udp;
        }
```

```
            then accept;
        }
        term permit-icmp {
            from {
                protocol icmp;
                icmp-type [ echo-reply echo-request ];
            }
            then accept;
        }
        term permit-ntp {
            from {
                source-address {
                    149.20.68.16/32;
                }
                protocol udp;
                port ntp;
            }
            then accept;
        }
        term permit-ospf {
            from {
                protocol ospf;
            }
            then accept;
        }
        term permit-snmp {
            from {
                source-address {
                    10.200.0.0/24;
                }
                protocol udp;
                port [ snmp snmptrap ];
            }
            then accept;
        }
        term deny-and-count {
            from {
                source-address {
                    0.0.0.0/0;
                }
            }
            then {
                count denied;
                reject tcp-reset;
            }
        }
    }
```

Explanation of Configuration

- The best way to connect to an SRX Series chassis cluster through the fxp0 interface (a new type of interface) is to assign IP addresses to both management ports on the primary and secondary nodes using groups.
- Use a master-only IP address across the cluster. This way, you can query a single IP address and that IP address is always the master for redundancy group 0. If you are

not using a master-only IPv4 address, each node IP address must be added and monitored. Secondary node monitoring is limited, as detailed in this topic.

NOTE: **We recommend using a master-only IPv4 address for management, especially while using SNMP. This enables the device to be reachable even after failover.**

- With the fxp0 interface configuration previously shown, the management IPv4 address on the fxp0 interface of the secondary node in a chassis cluster is not reachable. The secondary node routing subsystem is not running. The fxp0 interface is reachable by hosts that are on the same subnet as the management IPv4 address. If the host is on a different subnet than the management IPv4 address, then communication fails. This is an expected behavior and works as designed. The secondary cluster member's Routing Engine is not operational until failover. The routing protocol process does not work in the secondary node when the primary node is active. When management access is needed, the **backup-router** configuration statement can be used.

  With the **backup-router** statement, the secondary node can be accessed from an external subnet for management purposes. Due to a system limitation, do not configure the destination address specified in the **backup-router** as '0.0.0.0/0' or '::/0'. The mask has to be a non-zero value. Multiple destinations can be included if your management IP address range is not contiguous. In this example, backup router 172.19.100.1 is reachable through the fxp0 interface, and the destination network management system IPv4 address is 10.200.0.1. The network management address is reachable through the backup router. For the backup router to reach the network management system, include the destination subnet in the backup router configuration.

- We recommend using the outbound SSH address to connect to the management systems by using the SSH protocol, NETCONF XML management protocol, or Junos OS XML Management Protocol. This ensures that the device connects back automatically even after a switchover.

- We recommend using same SNMP engine IDs for each node. This is because SNMPv3 uses the SNMP engine ID values for authentication of the protocol data units (PDUs), and if SNMP engine ID values are different for each node, then SNMPv3 might fail after a routing engine switchover.

- Keep other SNMP configurations, such as the SNMP communities, trap-groups, and so on, common between the nodes as shown in the sample configuration.

NOTE: **SNMP traps are sent only from the primary node. This includes events and failures detected on the secondary node. The secondary node never sends SNMP traps or alerts. Use the client-only configurable option to restrict SNMP access to the required clients only. Use SNMPv3 for encryption and authentication.**

- Syslog messages should be sent from both nodes separately as the log messages are node specific.

- If the management station is on a different subnet than the management IP addresses, specify the same subnet in the backup router configuration and add a static route under the **[edit routing-options]** hierarchy level if required. In the previous sample configuration, the network management address 10.200.0.1 is reachable through the backup router. Therefore, a static route is configured.
- You can restrict access to the device using firewall filters. The previous sample configuration shows that SSH, SNMP, and Telnet are restricted to the 10.0.0.0/8 network. This configuration allows UDP, ICMP, OSPF, and NTP traffic and denies other traffic. This filter is applied to the fxp0 interface.
- You can also use security zones to restrict the traffic. For more information, see the *Junos OS Security Configuration Guide*.

Additional Configuration for SRX Series Branch Devices

- The factory default configuration for the SRX100, SRX210, and SRX240 devices automatically enables Layer 2 Ethernet switching. Because Layer 2 Ethernet switching is not supported in chassis cluster mode, for these devices, if you use the factory default configuration, you must delete the Ethernet switching configuration before you enable chassis clustering.
- There is no dedicated fxp0 management interface. The fxp0 interface is repurposed from a built-in interface. For example, on SRX100 devices, the fe-0/0/06 interface is repurposed as the management interface and is automatically renamed fxp0. For more information about the management interface, see the *Junos OS Security Configuration Guide*.
- Syslog should be used with caution. It can cause cluster instability. Data plane logging should never be sent through syslogs for SRX Series branch devices.

Managing Chassis Clusters

- Managing chassis clusters through redundant Ethernet interfaces—SRX Series chassis clusters can be managed using the redundant Ethernet (reth) interfaces. Configuration of redundancy groups and reth interfaces differ based on deployments such as active/active mode and active/passive mode. See the *Junos OS Security Configuration Guide* for details of the configuration. Once the reth interfaces are configured and are reachable from the management station, secondary nodes can be accessed through the reth interfaces.

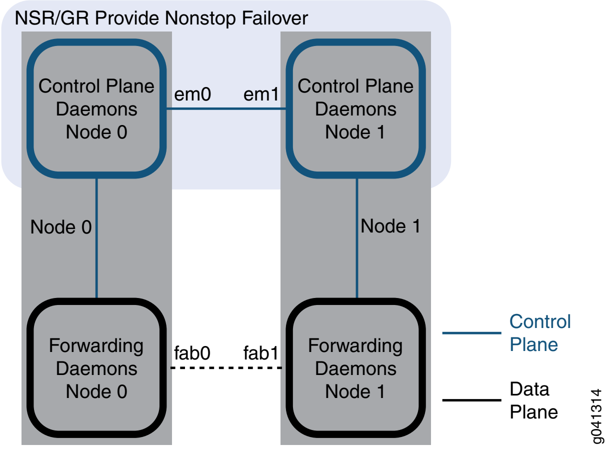
  If the reth interface belongs to redundancy group 1+, then the TCP connection to the management station is seamlessly transitioned to the new primary. But if redundancy group 0 failover occurs and the Routing Engine switches over to a new node, then connectivity is lost for all sessions for a couple of seconds.
- Managing clusters through the transit interfaces—Clustered devices can be managed using transit interfaces. A transit interface cannot be used directly to reach a secondary node.

Configuring Devices for In-Band Management and Administration

The chassis cluster feature available in Junos OS for SRX Series Services Gateways is modeled based on the redundancy features found in Junos OS-based devices. Designed with separate control and data planes, Junos OS-based devices provide redundancy in both planes. The control plane in Junos OS is managed by the Routing Engines, which perform all the routing and forwarding computations apart from other functions. Once the control plane converges, forwarding entries are pushed to all Packet Forwarding Engines in the system. Packet Forwarding Engines then perform route-based lookups to determine the appropriate destination for each packet without any intervention from the Routing Engines.

When enabling a chassis cluster in an SRX Series Services Gateway, the same model device is used to provide control plane redundancy as shown in Figure 4 on page 19.

Figure 4: SRX Series Clustering Model


NSR/GR Provide Nonstop Failover
Control Plane Daemons Node 0
em0
em1
Control Plane Daemons Node 1
Node 0
Node 1
Forwarding Daemons Node 0
fab0
fab1
Forwarding Daemons Node 1
Control Plane
Data Plane
g041314

Similar to a device with two Routing Engines, the control plane of an SRX Series cluster operates in an active/passive mode with only one node actively managing the control plane at any given time. Because of this, the forwarding plane always directs all traffic sent to the control plane (also referred to as host-inbound traffic) to the cluster's primary node. This traffic includes (but is not limited to):

- Traffic for the routing processes, such as BGP, OSPF, IS-IS, RIP, and PIM traffic
- IKE negotiation messages
- Traffic directed to management processes, such as SSH, Telnet, SNMP, and NETCONF
- Monitoring protocols, such as BFD or RPM

This behavior applies only to host-inbound traffic. Through traffic (that is, traffic forwarded by the cluster, but not destined to any of the cluster's interfaces) can be processed by either node, based on the cluster's configuration.

Because the forwarding plane always directs host-inbound traffic to the primary node, the fxp0 interface provides an independent connection to each node, regardless of the status of the control plane. Traffic sent to the fxp0 interface is not processed by the forwarding plane, but is sent to the Junos OS kernel, thus providing a way to connect to the control-plane of a node, even on the secondary node.

This topic explains how to manage a chassis cluster through the primary node without requiring the use of the fxp0 interfaces, that is, in-band management. This is particularly needed for SRX Series branch devices since the typical deployment for these devices is such that there is no management network available to monitor the remote branch office.

Before Junos OS Release 10.1 R2, the management of an SRX Series branch chassis cluster required connectivity to the control plane of both members of the cluster, thereby requiring access to the fxp0 interface of each node. In Junos OS Release 10.1 R2 and later, SRX Series branch devices can be managed remotely using the reth interfaces or the Layer 3 interfaces.

Managing SRX Series Branch Chassis Clusters Through the Primary Node

Accessing the primary node of a cluster is as easy as establishing a connection to any of the node’s interfaces (other than the fxp0 interface). Layer 3 and reth interfaces always direct the traffic to the primary node, whichever node that is. Both deployment scenarios are common and are depicted in Figure 5 on page 21 and Figure 6 on page 22.

In both cases, establishing a connection to any of the local addresses connects to the primary node. To be precise, you are connected to the primary node of redundancy group 0. For example, you can connect to the primary node even when the reth interface, a member of the redundancy group 1, is active in a different node (the same applies to Layer 3 interfaces, even if they physically reside in the backup node). You can use SSH, Telnet, SNMP, or the NETCONF XML management protocol to monitor the SRX chassis cluster.

Figure 5 on page 21 shows an example of an SRX Series branch device being managed over a reth interface. This model can be used for SRX Series high-end devices as well, using Junos OS Release 10.4 or later.

Figure 5: SRX Series Branch Deployment for In-Band Management Using a reth Interface

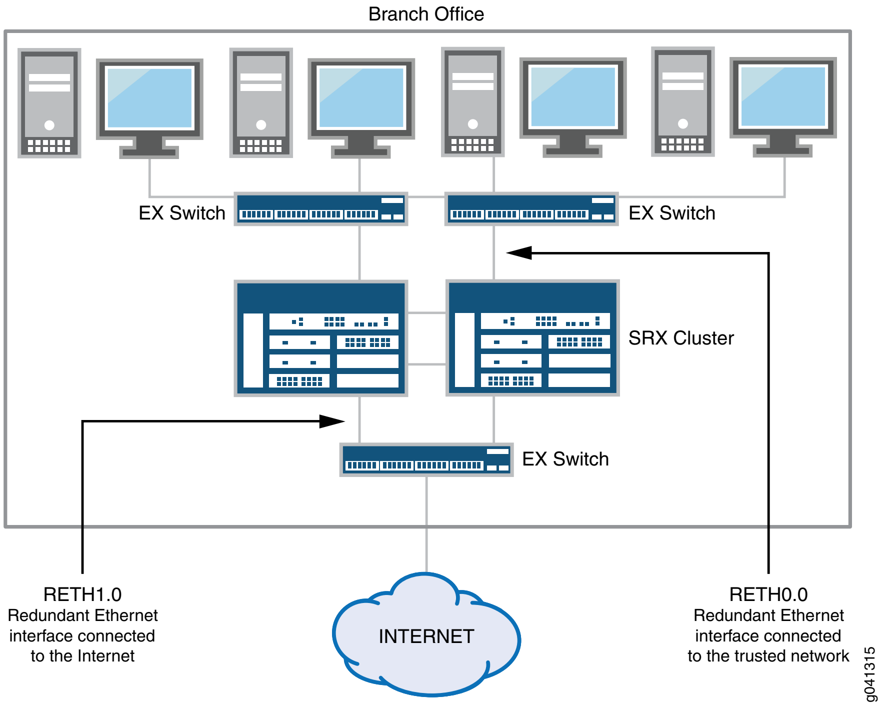
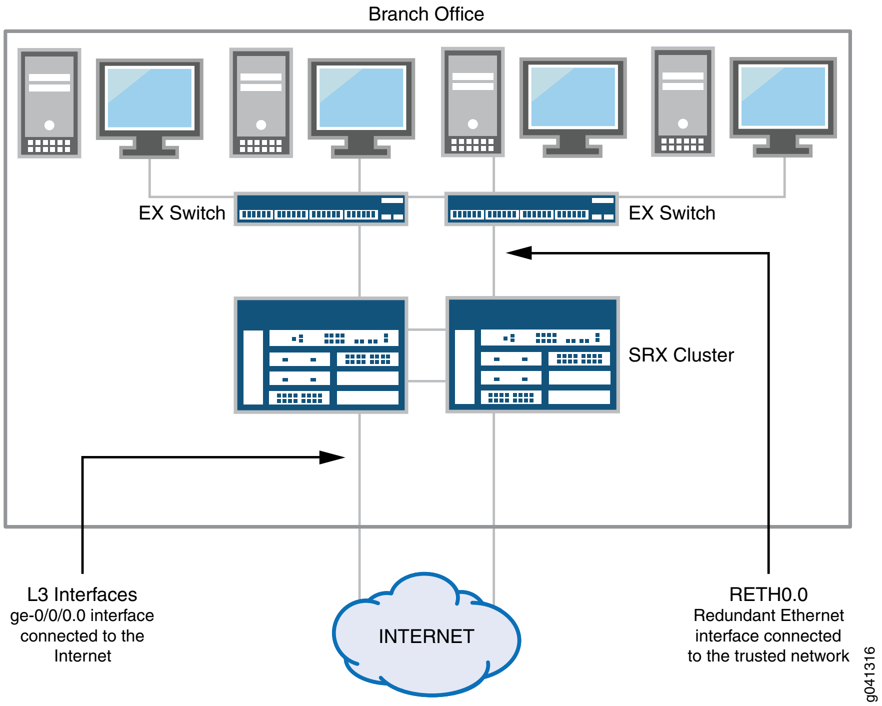

Branch Office
EX Switch
EX Switch
SRX Cluster
EX Switch
RETH1.0
Redundant Ethernet interface connected to the Internet
INTERNET
RETH0.0
Redundant Ethernet interface connected to the trusted network
g041315

Figure 6 on page 22 shows physical connections for in-band management using a Layer 3 interface.

Figure 6: SRX Series Branch Deployment for In-Band Management Using a Layer 3 Interface



NOTE: If there is a failover, only in-band connections need to be able to reach the new primary node through the reth or Layer 3 interfaces to maintain connectivity between the management station and the cluster.

Table 1 on page 22 lists the advantages and disadvantages of using different interfaces.

Table 1: Advantages and Disadvantages of Different Interfaces

fxp0 Interface	Reth and Transit Interfaces
Using the fxp0 interface with a master-only IP address allows access to all routing instances and virtual routers within the system. The fxp0 interface can only be part of the inet.0 routing table. Since the inet.0 routing table is part of the default routing instance, it can be used to access data for all routing instances and virtual routers.	A transit or reth interface has access only to the data of the routing instance or virtual router it belongs to. If it belongs to the default routing instance, it has access to all routing instances.
The fxp0 interface with a master-only IP address can be used for management of the device even after failover, and we highly recommend this.	Transit interfaces lose connectivity after a failover (or when the device hosting the interface goes down or is disabled), unless they are part of a reth group.

Table 1: Advantages and Disadvantages of Different Interfaces *(continued)*

fxp0 Interface	Reth and Transit Interfaces
Managing through the fxp0 interface requires two IP addresses, one per node. This also means that a switch needs to be present to connect to the cluster nodes using the fxp0 interface.	The reth interface does not need two IP addresses, and no switch is required to connect to the SRX Series chassis cluster. Transit interfaces on each node, if used for management, need two explicit IP addresses for each interface. But since this is a transit interface, the IP addresses are also used for traffic apart from management as well.
SRX Series branch device clusters with a non-Ethernet link (ADSL, T1\E1) cannot be managed using the fxp0 interface.	SRX Series branch devices with a non-Ethernet link can be managed using a reth or transit interface.

Communicating with a Chassis Cluster Device

Management stations can use the following methods to connect to the SRX Series chassis clusters. This is the same for any Junos OS devices and is not limited to SRX Series chassis clusters. We recommend using a master-only IP address for any of the following protocols on SRX Series chassis clusters. Reth interface IP addresses can be used to connect to the clusters using any of the following interfaces.

Table 2: Chassis Cluster Communication Methods

Method	Description
SSH or Telnet for CLI Access	This is only recommended for manual configuration and monitoring of a single cluster.
Junos OS XML Management Protocol	This is an XML-based interface that can run over Telnet, SSH, and SSL, and it is a precursor to the NETCONF XML management protocol. It provides access to Junos OS XML APIs for all configuration and operational commands that can be entered using the CLI. We recommend this method for accessing operational information. It can run over a NETCONF XML management protocol session as well.
NETCONF XML Management Protocol	This is the IETF-defined standard XML interface for configuration. We recommend using it to configure the device. This session can also be used to run Junos OS XML Management Protocol remote procedure calls (RPCs).
SNMP	From an SRX Series chassis cluster point of view, the SNMP system views the two nodes within the clusters as a single system. There is only one SNMP process running on the master Routing Engine. At initialization time, the protocol master indicates which SNMP process (snmpd) should be active based on the Routing Engine master configuration. The passive Routing Engine has no snmpd running. Therefore, only the primary node responds to SNMP queries and sends traps at any point of time. The secondary node can be directly queried, but it has limited MIB support, which is detailed in "Retrieving Chassis Inventory and Interfaces" on page 28. The secondary node does not send SNMP traps. SNMP requests to the secondary node can be sent using the fxp0 interface IP address on the secondary node or the reth interface IP address.
Syslogs	Standard system log messages can be sent to an external syslog server. Note that both the primary and secondary nodes can send syslog messages. We recommend that you configure both the primary and secondary nodes to send syslog messages separately.

Table 2: Chassis Cluster Communication Methods *(continued)*

Method	Description
Security Log Messages (SPU)	AppTrack, an application tracking tool, provides statistics for analyzing bandwidth usage of your network. When enabled, AppTrack collects byte, packet, and duration statistics for application flows in the specified zone. By default, when each session closes, AppTrack generates a message that provides the byte and packet counts and duration of the session, and sends the message to the host device. AppTrack messages are similar to session log messages and use syslog or structured syslog formats. The message also includes an application field for the session. If AppTrack identifies a custom-defined application and returns an appropriate name, the custom application name is included in the log message. Note that application identification has to be configured for this to occur. See the *Junos OS Security Configuration Guide* for details on configuring and using application identification and tracking.
J-Web	All Junos OS devices provide a graphical user interface for configuration and administration. This interface can be used for administering individual devices.

Related Documentation

- Chassis Cluster Overview on page 7
- Best Practices for Managing a Chassis Cluster on page 25

CHAPTER 2

Best Practices for SRX Series Cluster Management

Best Practices for Managing a Chassis Cluster

Following are some best practices for chassis clusters for SRX Series devices.

Using Dual Control Links

In dual control links, two pairs of control link interfaces are connected between each device in a cluster. Dual control links are supported on the SRX5000 and SRX3000 lines. Having two control links helps to avoid a possible single point of failure. For the SRX5000 line, this functionality requires a second Routing Engine, as well as a second Switch Control Board (SCB) to house the Routing Engine, to be installed on each device in the cluster. The purpose of the second Routing Engine is only to initialize the switch on the SCB. The second Routing Engine, to be installed on SRX5000 line devices only, does not provide backup functionality. For the SRX3000 line, this functionality requires an SRX Clustering Module (SCM) to be installed on each device in the cluster. Although the SCM fits in the Routing Engine slot, it is not a Routing Engine. SRX3000 line devices do not support a second Routing Engine. The purpose of the SCM is to initialize the second control link. SRX Series branch devices do not support dual control links.

Using Dual Data Links

You can connect two fabric links between each device in a cluster, which provides a redundant fabric link between the members of a cluster. Having two fabric links helps to avoid a possible single point of failure. When you use dual fabric links, the runtime objects (RTOs) and probes are sent on one link, and the fabric-forwarded and flow-forwarded packets are sent on the other link. If one fabric link fails, the other fabric link handles the RTOs and probes, as well as data forwarding. The system selects the physical interface with the lowest slot, PIC, or port number on each node for the RTOs and probes.

Using BFD

The Bidirectional Forwarding Detection (BFD) protocol is a simple hello mechanism that detects failures in a network. Hello packets are sent at a specified, regular interval. A neighbor failure is detected when the router stops receiving a reply after a specified interval. BFD works with a wide variety of network environments and topologies. BFD failure detection times are shorter than RIP detection times, providing faster reaction times to various kinds of failures in the network. These timers are also adaptive. For example, a timer can adapt to a higher value if the adjacency fails, or a neighbor can negotiate a higher value for a timer than the one configured. Therefore, BFD liveliness can be configured between the two nodes of an SRX Series chassis cluster using the local interfaces and not the fxp0 IP addresses on each node. This way BFD can keep monitoring the status between the two nodes of the cluster. When there is any network issue between the nodes, the BFD session-down SNMP traps are sent, which indicates an issue between the nodes.

Using IP Monitoring

IP monitoring is an automation script that enables you to use this critical feature on the SRX Series platforms. It allows for path and next-hop validation through the existing network infrastructure using the Internet Control Message Protocol (ICMP). Upon detection of a failure, the script executes a failover to the other node in an attempt to prevent downtime.

Using Interface Monitoring

The other SRX Series chassis cluster feature implemented is called interface monitoring. For a redundancy group to automatically fail over to another node, its interfaces must be monitored. When you configure a redundancy group, you can specify a set of interfaces that the redundancy group is to monitor for status or health to determine whether the interface is up or down. A monitored interface can be a child interface of any of its redundant Ethernet (reth) interfaces. When you configure an interface for a redundancy group to monitor, you give it a weight. Every redundancy group has a threshold tolerance value initially set to 255. When an interface monitored by a redundancy group becomes unavailable, its weight is subtracted from the redundancy group's threshold. When a redundancy group's threshold reaches 0, it fails over to the other node. For example, if redundancy group 1 was primary on node 0, on the threshold-crossing event, redundancy group 1 becomes primary on node 1. In this case, all the child interfaces of redundancy group 1's reth interfaces begin handling traffic. A redundancy group failover occurs because the cumulative weight of the redundancy group's monitored interfaces has brought its threshold value to 0. When the monitored interfaces of a redundancy group on both nodes reach their thresholds at the same time, the redundancy group is primary on the node with the lower node ID, in this case, node 0.

NOTE: Interface monitoring is not recommended for redundancy group 0.

```
chassis {
   cluster {
      reth-count 6;
      redundancy-group 0 {
         node 0 priority 129;
         node 1 priority 128;
      }
      redundancy-group 1 {
         node 0 priority 129;
         node 1 priority 128;
         interface-monitor {
            ge-0/0/0 weight 255;
            ge-8/0/0 weight 255;
         }
         ip-monitoring {
            global-weight 255;
            global-threshold 0;
            family {
               inet {
                  128.249.34.1 {
                     weight 10;
                     interface reth0.34 secondary-ip-address 128.249.34.202;
                  }
               }
            }
         }
      }
   }
}
```

Using Graceful Restart

With routing protocols, any service interruption requires that an affected router recalculate adjacencies with neighboring routers, restore routing table entries, and update other protocol-specific information. An unprotected restart of a router can result in forwarding delays, route flapping, wait times stemming from protocol reconvergence, and even dropped packets. The main benefits of graceful restart are uninterrupted packet forwarding and temporary suppression of all routing protocol updates. Graceful restart enables a router to pass through intermediate convergence states that are hidden from the rest of the network.

Three main types of graceful restart are available on Juniper Networks routing platforms:

- Graceful restart for aggregate and static routes and for routing protocols—Provides protection for aggregate and static routes and for BGP, End System-to-Intermediate

System (ES-IS), IS-IS, OSPF, RIP, next-generation RIP (RIPng), and Protocol Independent Multicast (PIM) sparse mode routing protocols.

- Graceful restart for MPLS-related protocols—Provides protection for LDP, RSVP, circuit cross-connect (CCC), and translational cross-connect (TCC).
- Graceful restart for virtual private networks (VPNs)—Provides protection for Layer 2 and Layer 3 VPNs.

Related Documentation

- Chassis Cluster Descriptions and Deployment Scenarios on page 9
- Retrieving Chassis Inventory and Interfaces on page 28

Retrieving Chassis Inventory and Interfaces

SRX Series chassis cluster inventory and interface information is gathered to monitor the hardware components and the interfaces on the cluster. The primary node contains information about the secondary node components and interfaces.

Using the Junos OS XML Management Protocol or NETCONF XML Management Protocol

- Use the **get-chassis-inventory** remote procedure call (RPC) to get the chassis inventory. This RPC reports components on both the primary and secondary nodes. For more information, see "Managing SRX Series Chassis Clusters Using RPCs" on page 92.
- Use the **get-interface-information** RPC to get the interfaces inventory. This RPC reports information about the interfaces on the secondary node except for the fxp0 interface.

See the *Junos XML API Operational Developer Reference* for details about using the RPCs and their responses.

Using SNMP

- Use the **jnx-chas-defines** MIB to understand the SRX Series chassis structure and modeling. This MIB is not for querying. It is only used to understand the chassis cluster modeling.

```
jnxProductLineSRX3600 OBJECT IDENTIFIER ::= { jnxProductLine    34 }
jnxProductNameSRX3600    OBJECT IDENTIFIER ::= { jnxProductName    34 }
jnxProductModelSRX3600    OBJECT IDENTIFIER ::= { jnxProductModel   34 }
jnxProductVariationSRX3600  OBJECT IDENTIFIER ::= { jnxProductVariation  34 }
jnxChassisSRX3600      OBJECT IDENTIFIER ::= { jnxChassis      34 }

jnxSlotSRX3600       OBJECT IDENTIFIER ::= { jnxSlot       34 }
 jnxSRX3600SlotFPC    OBJECT IDENTIFIER ::= { jnxSlotSRX3600 1 }
 jnxSRX3600SlotHM      OBJECT IDENTIFIER ::= { jnxSlotSRX3600 2 }
 jnxSRX3600SlotPower   OBJECT IDENTIFIER ::= { jnxSlotSRX3600 3 }
 jnxSRX3600SlotFan     OBJECT IDENTIFIER ::= { jnxSlotSRX3600 4 }
 jnxSRX3600SlotCB      OBJECT IDENTIFIER ::= { jnxSlotSRX3600 5 }
 jnxSRX3600SlotFPB     OBJECT IDENTIFIER ::= { jnxSlotSRX3600 6 }

jnxMediaCardSpaceSRX3600   OBJECT IDENTIFIER ::= { jnxMediaCardSpace    34 }
jnxSRX3600MediaCardSpacePIC   OBJECT IDENTIFIER ::= { jnxMediaCardSpaceSRX3600
 1 }

jnxMidplaneSRX3600      OBJECT IDENTIFIER ::= { jnxBackplane     34 }
```

- Use the following command to view the SNMP MIB.

```
user@host> show snmp mib ?
Possible completions:
  get                    Get SNMP object value
  get-next               Get next SNMP object value
  walk                   Walk SNMP object values
{secondary:node0}
```

```
user@host> show snmp mib get ?
Possible completions:
  <name>
  ascii                  Convert string indices to 'ascii-keys' representation
  decimal                Decimal format (default)
{secondary:node0}
```

```
user@host> show snmp mib walk ?
Possible completions:
  <name>                 Requested SNMP object names
  ascii                  Convert string indices to 'ascii-keys' representation
  decimal                Decimal format (default)
```

- Use the jnx-chassis MIB to get the chassis inventory.

Table 3: jnx-chassis MIB Information

MIB Item	Description
Top of MIB	Use the top-level objects to show chassis details such as the jnxBoxClass, jnxBoxDescr, jnxBoxSerialNo, jnxBoxRevision, and jnxBoxInstalled MIB objects.
jnxContainersTable	Use to show the containers that the device supports.
jnxContentsTable	Use to show the chassis contents.
jnxContentsChassisId	Use to show which components belong to which node.
jnxLedTable	Use to check the LED status of the components. This MIB only reports the LED status of the primary node.
jnxFilledTable	Use to show the empty/filled status of the container in the device containers table.
jnxOperatingTable	Use to show the operating status of Operating subjects in the box contents table.
jnxRedundancyTable	Use to show redundancy details on both nodes. Note that currently this MIB only reports on the Routing Engines. Both Routing Engines are reported as the master of the respective nodes. Do not use this to determine the active and backup status.
jnxFruTable	Use to show the field-replaceable unit (FRU) in the chassis. Note that even the empty slots are reported.

NOTE: The jnx-chassis MIB is not supported on SRX Series branch devices in cluster mode. It is supported on standalone SRX Series branch devices.

```
JUNIPER-MIB::jnxContentsDescr.1.1.0.0 = STRING: node0 midplane
JUNIPER-MIB::jnxContentsDescr.1.2.0.0 = STRING: node1 midplane
JUNIPER-MIB::jnxContentsDescr.2.1.0.0 = STRING: node0 PEM 0
JUNIPER-MIB::jnxContentsDescr.2.2.0.0 = STRING: node0 PEM 1
JUNIPER-MIB::jnxContentsDescr.2.5.0.0 = STRING: node1 PEM 0
JUNIPER-MIB::jnxContentsDescr.2.6.0.0 = STRING: node1 PEM 1
JUNIPER-MIB::jnxContentsDescr.4.1.0.0 = STRING: node0 Left Fan Tray
JUNIPER-MIB::jnxContentsDescr.4.1.1.0 = STRING: node0 Top Rear Fan
JUNIPER-MIB::jnxContentsDescr.4.1.2.0 = STRING: node0 Bottom Rear Fan
JUNIPER-MIB::jnxContentsDescr.4.1.3.0 = STRING: node0 Top Middle Fan
JUNIPER-MIB::jnxContentsDescr.4.1.4.0 = STRING: node0 Bottom Middle Fan
JUNIPER-MIB::jnxContentsDescr.4.1.5.0 = STRING: node0 Top Front Fan
JUNIPER-MIB::jnxContentsDescr.4.1.6.0 = STRING: node0 Bottom Front Fan
JUNIPER-MIB::jnxContentsDescr.4.2.0.0 = STRING: node1 Left Fan Tray
JUNIPER-MIB::jnxContentsDescr.4.2.1.0 = STRING: node1 Top Rear Fan
JUNIPER-MIB::jnxContentsDescr.1.1.0.0 = STRING: node0 midplane
JUNIPER-MIB::jnxContentsDescr.1.2.0.0 = STRING: node1 midplane
JUNIPER-MIB::jnxContentsDescr.2.1.0.0 = STRING: node0 PEM 0
JUNIPER-MIB::jnxContentsDescr.2.2.0.0 = STRING: node0 PEM 1
JUNIPER-MIB::jnxContentsDescr.2.5.0.0 = STRING: node1 PEM 0
JUNIPER-MIB::jnxContentsDescr.2.6.0.0 = STRING: node1 PEM 1
JUNIPER-MIB::jnxContentsDescr.4.1.0.0 = STRING: node0 Left Fan Tray
JUNIPER-MIB::jnxContentsDescr.4.1.1.0 = STRING: node0 Top Rear Fan
JUNIPER-MIB::jnxContentsDescr.4.1.2.0 = STRING: node0 Bottom Rear Fan
JUNIPER-MIB::jnxContentsDescr.4.1.3.0 = STRING: node0 Top Middle Fan
JUNIPER-MIB::jnxContentsDescr.4.1.4.0 = STRING: node0 Bottom Middle Fan
JUNIPER-MIB::jnxContentsDescr.4.1.5.0 = STRING: node0 Top Front Fan
JUNIPER-MIB::jnxContentsDescr.4.1.6.0 = STRING: node0 Bottom Front Fan
JUNIPER-MIB::jnxContentsDescr.4.2.0.0 = STRING: node1 Left Fan Tray
JUNIPER-MIB::jnxContentsDescr.4.2.1.0 = STRING: node1 Top Rear Fan
```

- ifTable—Use to show all the interfaces on the cluster. Note that except for the fxp0 interface on the secondary node, all interfaces of the secondary node are reported by the primary node.
- jnx-if-extensions/ifChassisTable—Use to show the interface mapping to the respective PIC and FPC.
- ifStackStatusTable—Use to show the sub-interfaces and respective parent interfaces.

Related Documentation

- Best Practices for Managing a Chassis Cluster on page 25
- Identifying Nodes in a Chassis Cluster on page 30

Identifying Nodes in a Chassis Cluster

To determine if the SRX Series device is configured in a cluster, use the following methods. We recommend using the master-only IP address from the management station to perform the operations suggested.

Identifying the Chassis Cluster Primary and Secondary Nodes

- Using the Junos OS XML Management Protocol or NETCONF XML Management Protocol on page 31
- Using the get-chassis-inventory RPC Tag on page 31
- Using SNMP on page 32

Using the Junos OS XML Management Protocol or NETCONF XML Management Protocol

Use the <get-chassis-cluster-status> remote procedure call (RPC) to determine if the chassis is configured in a cluster.

RPC for Chassis Inventory

```
RPC :<rpc>
    <get-chassis-cluster-status>
    </get-chassis-cluster-status>
  </rpc>

Response: See “Managing SRX Series Chassis Clusters Using RPCs” on page 92.
```

Using the get-chassis-inventory RPC Tag

Use the get-chassis-inventory remote procedure call (RPC) to get the inventory of the chassis for both the primary and secondary nodes. This identifies two nodes as part of a multi-routing-engine-item. See “Managing SRX Series Chassis Clusters Using RPCs” on page 92 for sample output of the RPC. The following output shows only the relevant tags.

Sample Chassis Inventory Tags

```
RPC:<rpc><get-chassis-inventory/></rpc>

RELEVANT RESPONSE TAGS:
</multi-routing-engine-item>

  <multi-routing-engine-item>

      <re-name>node0</re-name>

          #Node 0 Items
  </multi-routing-engine-item>

  <multi-routing-engine-item>

  <re-name>node1></re-name>
  #Node 0 Items
  </multi-routing-engine-item>

</multi-routing-engine-item>
```

Using SNMP

- jnx-chassis-jnxRedundancyTable/jnxContentsTable – Use to show if two Routing Engines are in service.
- jnxContentsChassisId – Use to show which Routing Engine belongs to which node.

We recommend that you use the master-only IP address to do SNMP polling. After a switchover, the management system continues to use the master-only IP address to manage the cluster. If a master-only IP address is not used, only the primary node responds to the jnx-chassis MIB queries. The primary node includes components from the secondary node as well. The secondary node does not respond to the jnx-chassis MIB queries.

NOTE: **There are no MIBS to identify the primary and secondary nodes. The only method to identify the primary and secondary nodes using SNMP is to send queries to retrieve the jnx-chassis MIB objects on both IP addresses. Only the primary responds. If you use a master-only IP address, the active primary responds. Another option is to SNMP MIB walk the jnxLedTable MIB. This only returns data for the primary node.**

The following sample shows two Routing Engines and two nodes, node 0 and node 1, present on the device.

Sample SNMP Output

```
JUNIPER-MIB::jnxContentsDescr.9.1.0.0 = STRING: node0 Routing Engine 0

JUNIPER-MIB::jnxContentsDescr.9.3.0.0 = STRING: node1 Routing Engine 0

JUNIPER-MIB::jnxRedundancyDescr.9.1.0.0 = STRING: node0 Routing Engine 0

JUNIPER-MIB::jnxRedundancyDescr.9.3.0.0 = STRING: node1 Routing Engine 0

JUNIPER-MIB::jnxContentsChassisId.9.1.0.0 = INTEGER: node0(12)

JUNIPER-MIB::jnxContentsChassisId.9.3.0.0 = INTEGER: node1(13)
```

The jnx-chassis MIB is not supported on SRX Series branch devices in cluster mode. It is supported on standalone SRX Series branch devices.

Determining the IP Address of Nodes

We recommend that the management systems have options to provide additional IP addresses to communicate with the device, such as the secondary IP address and the primary IP address. The following are additional options for gathering IP addresses used on the cluster.

- Using the Junos OS XML Management Protocol or NETCONF XML Management Protocol on page 33
- Using SNMP MIBs on page 33

Using the Junos OS XML Management Protocol or NETCONF XML Management Protocol

- **get-config** – Use to show the node0 and node1 fxp0 interface and the reth interface configuration to identify the IP addresses used by the primary and secondary nodes.
- **get-interface-information** – Use to show the interfaces and basic details. Use the **interface-address** tag to identify the IP addresses for the fxp0 and reth interfaces. Using this remote procedure call (RPC), all interfaces are reported, including the addresses on the secondary node, except for the fxp0 interface on the secondary node. The following sample shows the fxp0 interface on the primary node:

```
 <physical-interface>
<name>
fxp0
</name>
<admin-status>
up
</admin-status>
<oper-status>
up
</oper-status>
<logical-interface>
<name>
fxp0.0
</name>
<admin-status>
up
</admin-status>
<oper-status>
up
</oper-status>
<filter-information>
</filter-information>
<address-family>
<address-family-name>
inet
</address-family-name>
<interface-address>
<ifa-local junos:emit="emit">
10.204.131.37/18
</ifa-local>
</interface-address>
</address-family>
</logical-interface>
</physical-interface>
</interface-information>
```

Using SNMP MIBs

Use the ifTable MIB table to get the ifIndex MIB object of the fxp0 interface and the reth interface on the primary node. Use the ipAddrTable MIB table to determine the IP address of the interfaces. The following is a sample showing the fxp0 interface on the active primary node. Note that the ifTable MIB table reports all interfaces on the secondary node, except for the fxp0 interface on the secondary node.

Sample SNMP MIB Walk of the ifTable MIB Table

```
{primary:node0}
user@host> show snmp mib walk ifTable | grep fxp0
ifDescr.1    = fxp0
ifDescr.13   = fxp0.0

user@host> show snmp mib walk ipAddrTable | grep 13
ipAdEntAddr.10.204.131.37 = 10.204.131.37
ipAdEntIfIndex.10.204.131.37 = 13
ipAdEntNetMask.10.255.131.37 = 255.255.255.255
```

For SNMP communication directly with the secondary node, the IP address of the secondary node should be predetermined and preconfigured on the management system. Querying the ifTable MIB table directly on the secondary node returns only the fxp0 interface and a few private interface details on the secondary node, and no other interfaces are reported. All other interfaces are reported by the primary node itself. Use the ifTable MIB table and the ipAddrTable MIB table as previously shown to directly query the secondary node to find the fxp0 interface details such as the ifAdminStatus and ifOperStatus MIB objects on the secondary node.

Related Documentation

- Retrieving Chassis Inventory and Interfaces on page 28
- Monitoring Nodes in a Chassis Cluster on page 34

Monitoring Nodes in a Chassis Cluster

To monitor the cluster, you need to discover the redundancy groups. When you initialize a device in chassis cluster mode, the system creates a redundancy group referred to in this topic as redundancy group 0. Redundancy group 0 manages the primacy and failover between the Routing Engines on each node of the cluster. As is the case for all redundancy groups, redundancy group 0 can be primary on only one node at a time. The node on which redundancy group 0 is primary determines which Routing Engine is active in the cluster. A node is considered the primary node of the cluster if its Routing Engine is the active one. You can configure one or more redundancy groups numbered 1 through 128, referred to in this section as redundancy group x. The maximum number of redundancy groups is equal to the number of redundant Ethernet interfaces +1 that you configure. Each redundancy group x acts as an independent unit of failover and is primary on only one node at a time. There are no MIBS available to retrieve this information.

Using the Junos OS XML Management Protocol or NETCONF XML Management Protocol

Use the **get-configuration** remote procedure call (RPC) to get the redundancy configuration and the redundancy groups present on the device. This provides the redundancy groups configured.

XML RPC for Configuration Retrieval

```
<rpc>
   <get-configuration inherit="inherit" database="committed">
     <configuration>
      <chassis>
```

```
    <cluster>
    </cluster>
      </chassis>
     </configuration>
   </get-configuration>
</rpc>
```

Response:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:junos="http://xml.juniper.net/junos/10.4I0/junos">
<configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm"
junos:commit-seconds="1277806450" junos:commit-localtime="2010-06-29 03:14:10
PDT" junos:commit-user="regress">
  <chassis>
   <cluster>
    <reth-count>10</reth-count>
    <control-ports>
     <fpc>4</fpc>
     <port>0</port>
    </control-ports>
    <control-ports>
     <fpc>10</fpc>
     <port>0</port>
    </control-ports>
    <redundancy-group>
     <name>0</name>
     <node>
      <name>0</name>
      <priority>254</priority>
     </node>
     <node>
      <name>1</name>
      <priority>1</priority>
     </node>
    </redundancy-group>
    <redundancy-group>
     <name>1</name>
     <node>
      <name>0</name>
      <priority>100</priority>
     </node>
     <node>
      <name>1</name>
      <priority>1</priority>
     </node>
    </redundancy-group>
   </cluster>
  </chassis>
</configuration>
</rpc-reply>
]]>]]>
```

Chassis Cluster Redundant Ethernet Interfaces

A redundant Ethernet interface is a pseudointerface that includes at minimum one physical interface from each node of the cluster. A redundant Ethernet interface is referred to as a reth in configuration commands. The following sample output shows two redundancy groups present and configured.

Using the Junos OS XML Management Protocol or NETCONF XML Management Protocol

- Use the **get-chassis-cluster-interfaces** remote procedure call (RPC) to obtain the reth interface details. The following sample output shows four reth interfaces configured:

```
user@host> show chassis cluster interfaces |display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/11.4R5/junos">
    <chassis-cluster-interface-statistics>
        <control-interface-status>Up</control-interface-status>
        <control-link-interfaces>
            <control-information>
                <control-link-interface-index>0</control-link-interface-index>

                <control-link-interface-name>em0</control-link-interface-name>

              <control-link-interface-status>Up</control-link-interface-status>

            </control-information>
            <control-information>
                <control-link-interface-index>1</control-link-interface-index>

                <control-link-interface-name>em1</control-link-interface-name>


<control-link-interface-status>Down</control-link-interface-status>
            </control-information>
        </control-link-interfaces>
        <dataplane-interface-status>Up</dataplane-interface-status>
        <dataplane-interfaces>
            <fabric-information>
                <fabric-interface-index>0</fabric-interface-index>

<fabric-child-interface-name>ge-6/0/15</fabric-child-interface-name>
              <fabric-child-interface-status>Up</fabric-child-interface-status>

                <fabric-interface-index>0</fabric-interface-index>
            </fabric-information>
            <fabric-information>
                <fabric-interface-index>1</fabric-interface-index>

<fabric-child-interface-name>ge-19/0/15</fabric-child-interface-name>
              <fabric-child-interface-status>Up</fabric-child-interface-status>

                <fabric-interface-index>1</fabric-interface-index>
            </fabric-information>
        </dataplane-interfaces>
        <reth>
            <reth-name>reth0</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth1</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>Not
configured</redundancy-group-id-for-reth>
            <reth-name>reth2</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth3</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>Not
configured</redundancy-group-id-for-reth>
            <reth-name>reth4</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth5</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth6</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth7</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth8</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth9</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth10</reth-name>
            <reth-status>Up</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth11</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth12</reth-name>
            <reth-status>Down</reth-status>
            <redundancy-group-id-for-reth>Not
configured</redundancy-group-id-for-reth>
            <reth-name>reth13</reth-name>
            <reth-status>Up</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth14</reth-name>
            <reth-status>Up</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth15</reth-name>
            <reth-status>Up</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
            <reth-name>reth16</reth-name>
            <reth-status>Up</reth-status>
            <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
        </reth>
        <interface-monitoring>
        </interface-monitoring>
    </chassis-cluster-interface-statistics>
    <cli>
        <banner>{secondary:node0}</banner>
    </cli>
</rpc-reply>
```

```
user@host> show chassis cluster interfaces
Control link status: Up

Control interfaces:
    Index   Interface        Status
    0       em0              Up
    1       em1              Down

Fabric link status: Up

Fabric interfaces:
    Name    Child-interface    Status
    fab0    ge-6/0/15          Up
    fab0
    fab1    ge-19/0/15         Up
    fab1

Redundant-ethernet Information:
    Name         Status      Redundancy-group
    reth0        Down        1
    reth1        Down        Not configured
    reth2        Down        1
    reth3        Down        Not configured
    reth4        Down        1
    reth5        Down        1
    reth6        Down        1
    reth7        Down        1
    reth8        Down        1
    reth9        Down        1
    reth10       Up          1
    reth11       Down        1
    reth12       Down        Not configured
    reth13       Up          1
    reth14       Up          1
    reth15       Up          1
    reth16       Up          1

{secondary:node0}
```

- Use the **get-interface-information** remote procedure call (RPC) to show reth interface details and to identify the reth interfaces on the device. This RPC also shows which Gigabit Ethernet or Fast Ethernet interfaces belong to which reth interface as shown in the following sample output:

```
<rpc>
   <get-interface-information>
    <terse/>
     <interface-name>reth0</interface-name>
   </get-interface-information>
  </rpc><rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:junos="http://xml.juniper.net/junos/10.4I0/junos">
<interface-information
xmlns="http://xml.juniper.net/junos/10.4I0/junos-interface" junos:style="terse">
<physical-interface>
<name>
reth0
</name>
<admin-status>
up
</admin-status>
<oper-status>
up
</oper-status>
<logical-interface>
<name>
reth0.0
</name>
<admin-status>
up
</admin-status>
<oper-status>
```

```
up
</oper-status>
<filter-information>
</filter-information>
<address-family>
<address-family-name>
inet
</address-family-name>
<interface-address>
<ifa-local junos:emit="emit">
192.168.29.2/24
</ifa-local>
</interface-address>
</address-family>
<address-family>
<address-family-name>
multiservice
</address-family-name>
</address-family>
</logical-interface>
</physical-interface>
</interface-information>

Now, the interface that belongs to this. Extracting only the relevant information
<rpc>
   <get-interface-information>
    <terse/>
   </get-interface-information>
   </rpc><rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:junos="http://xml.juniper.net/junos/10.4I0/junos">
<interface-information
xmlns="http://xml.juniper.net/junos/10.4I0/junos-interface" junos:style="terse">
<physical-interface>
<name>
ge-5/1/1
</name>
<admin-status>
up
</admin-status>
<oper-status>
up
</oper-status>
<logical-interface>
<name>
ge-5/1/1.0
</name>
<admin-status>
up
</admin-status>
<oper-status>
up
</oper-status>
<filter-information>
</filter-information>
<address-family>
<address-family-name>
aenet
</address-family-name>
<ae-bundle-name>
reth0.0
</ae-bundle-name>
```

```
</address-family>
</logical-interface>
</physical-interface>
</interface-information>
```

In the sample output, the **ae-bundle-name** tag identifies the reth interface it belongs to.

Using SNMP

- The ifTable MIB table reports all the reth interfaces.
- Use the ifStackStatus MIB table to map the reth interface to the underlying interfaces on the primary and secondary nodes. The reth interface is the high layer, and the individual interfaces from both nodes show up as lower layer indexes.

  In the following sample, ge-5/1/1 and ge-11/1/1 belong to reth0:

  ```
  {primary:node0}
  user@host> show interfaces terse | grep reth0

  ge-5/1/1.0 up up aenet --> reth0.0
  ge-11/1/1.0 up up aenet --> reth0.0
  reth0 up up
  reth0.0 up up inet 192.168.29.2/24
  ```

  Find the index of all interfaces from the ifTable. The following information shows indexes of interfaces required in this example:

  ```
  {primary:node0}
  user@host> show snmp mib walk ifDescr | grep reth0

  ifDescr.503 = reth0.0
  ifDescr.528 = reth0
  ```

  Now, search for the index for reth0 in the ifStackStatus table. In the following sample output, reth0 index 503 is the higher layer index, and index 522 and 552 are the lower layer indexes. Index 522 and 552 represent interfaces ge-5/1/1.0 and ge-11/1/1.0, respectively.

  ```
  {primary:node0}
  user@host> show snmp mib walk ifStackStatus | grep 503

  ifStackStatus.0.503 = 1
  ifStackStatus.503.522 = 1
  ifStackStatus.503.552 = 1
  ```

  ```
  {primary:node0}
  user@host> show snmp mib walk ifDescr | grep 522

  ifDescr.522 = ge-5/1/1.0
  ```

  ```
  {primary:node0}
  user@host> show snmp mib walk ifDescr | grep 552

  ifDescr.552 = ge-11/1/1.0
  ```

Using the Control Plane

The control plane software, which operates in active/backup mode, is an integral part of Junos OS that is active on the primary node of a cluster. It achieves redundancy by communicating state, configuration, and other information to the inactive Routing Engine on the secondary node. If the master Routing Engine fails, the secondary one is ready to assume control. The following methods can be used to discover control port information.

Using the Junos OS XML Management Protocol or NETCONF XML Management Protocol

Use the **get-configuration** remote procedure call (RPC) to get the control port configuration as shown in the following sample output.

XML RPC for Redundant Group Configuration

```
<rpc>
 <get-configuration inherit="inherit" database="committed">
   <configuration>
    <chassis>
  <cluster>
  </cluster>
    </chassis>
   </configuration>
 </get-configuration>
</rpc>
```

Using the Data Plane

The data plane software, which operates in active/active mode, manages flow processing and session state redundancy and processes transit traffic. All packets belonging to a particular session are processed on the same node to ensure that the same security treatment is applied to them. The system identifies the node on which a session is active and forwards its packets to that node for processing. The data link is referred to as the fabric interface. It is used by the cluster's Packet Forwarding Engines to transmit transit traffic and to synchronize the data plane software’s dynamic runtime state. When the system creates the fabric interface, the software assigns it an internally derived IP address to be used for packet transmission. The fabric is a physical connection between two nodes of a cluster and is formed by connecting a pair of Ethernet interfaces back-to-back (one from each node). The following methods can be used to determine the data plane interfaces.

Using the Junos OS XML Management Protocol or NETCONF XML Management Protocol

Use the **get-chassis-cluster-data-plane-interfaces** remote procedure call (RPC) to get the data plane interfaces as shown in the following sample output.

XML RPC for Cluster Dataplane Interface Details

```
<rpc>
<get-chassis-cluster-data-plane-interfaces>
</get-chassis-cluster-data-plane-interfaces>
</rpc><rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
```

```
xmlns:junos="http://xml.juniper.net/junos/10.4I0/junos">
<chassis-cluster-dataplane-interfaces>
<fabric-interface-index>0</fabric-interface-index>
<fabric-information>
<child-interface-name>xe-5/0/0</child-interface-name>
<child-interface-status>up</child-interface-status>
</fabric-information>
<fabric-interface-index>1</fabric-interface-index>
<fabric-information>
<child-interface-name>xe-11/0/0</child-interface-name>
<child-interface-status>up</child-interface-status>
</fabric-information>
</chassis-cluster-dataplane-interfaces>
```

Using SNMP

The ifTable MIB table reports fabric (fab) interfaces and the link interfaces. However, the relationship between the underlying interfaces and fabric interfaces cannot be determined using SNMP.

Provisioning Chassis Cluster Nodes

Use the NETCONF XML management protocol for configuration and provisioning of SRX Series devices and Junos OS devices in general. We recommend using groups to configure SRX Series chassis clusters. Use global groups for all configurations that are common between the nodes.

Junos OS commit scripts can be used to customize the configuration as desired.

Junos OS commit scripts are:

- Run at commit time
- Inspect the incoming configuration
- Perform actions including:
  - Failing the commit (self-defense)
  - Modifying the configuration (self-correcting)

Commit scripts can:

- Generate custom error/warning/syslog messages
- Make changes or corrections to the configuration

Commit scripts give you better control over how your devices are configured to enforce:

- Your design rules
- Your implementation details
- 100 percent of your design standards

Related Documentation

- Identifying Nodes in a Chassis Cluster on page 30
- Monitoring Chassis Cluster Performance on page 43

Monitoring Chassis Cluster Performance

This topic provides information about the options available for monitoring chassis components such as FPCs, PICs, and Routing Engines for data such as operating state, CPU, and memory.

NOTE: The jnx-chassis MIB is not supported for SRX Series branch devices in cluster mode. However, it is supported for standalone SRX Series branch devices. Therefore, we recommend using options other than SNMP for chassis monitoring of SRX Series branch devices.

The instrumentation used for monitoring chassis components is provided in Table 4 on page 43.

Table 4: Instrumentation for Chassis Component Monitoring

Junos OS XML RPC	SNMP MIB

• For temperature of sensors, fan speed, and status of each component, use the **get-environment-information** remote procedure call (RPC).
• For temperature thresholds for the hardware components for each element, use the **get-temperature-threshold-information** RPC.
• For Routing Engine status, CPU, and memory, use the **get-route-engine-information** RPC. This RPC provides 1, 5, and 15 minute load averages.
• For FPC status, temperature, CPU, and memory, use the **get-fpc-information** RPC.
• Use the **get-pic-detail** RPC with the **fpc-slot** and **pic-slot** RPCs to get the PIC status.
	• Use the jnxOperatingTable MIB table for temperature, fan speed, and so on. The jnxOperatingState MIB should be used to get the status of the component. If the component is a FRU, then use the jnxFruState MIB also. Use the jnxOperatingTemp MIB for the temperature of sensors. Use the jnxFruState MIB to get the FRU status such as offline, online, empty, and so on.
	• Note the following about the objects available for monitoring in the jnxOperatingTable MIB table:
	◦ No MIB is available for temperature thresholds.
	◦ For the Routing Engine, use the jnxOperatingCPU, jnxOperatingTemp, jnxOperatingMemory, jnxOperatingISR, and jnxOperatingBuffer MIB objects under container Index 9.
	◦ Look at the jnxRedundancyTable for redundancy status monitoring. This only gives data for the last 5 seconds.
	◦ For the FPCs, look at the objects in the jnxOperatingTable and jnxFruTable MIB tables on container Index 7 for temperature, CPU, and memory utilization.
	◦ For the PICs (including SPU/SPC cards flows), look at the objects in the jnxOperatingTable and jnxFruTable MIB tables under container Index 8 in the following sample output for temperature, CPU, and memory utilization.
	`user@host>` **show snmp mib walk**
	`jnxOperatingDescr.8`
	`jnxOperatingDescr.8.5.1.0 = node0 PIC: SPU Cp-Flow @ 4/0/*`
	`jnxOperatingDescr.8.5.2.0 = node0 PIC: SPU Flow @ 4/1/*`
	`jnxOperatingDescr.8.6.1.0 = node0 PIC: 4x 10GE XFP @ 5/0/*`
	`jnxOperatingDescr.8.6.2.0 = node0 PIC: 16x 1GE TX @ 5/1/*`
	`jnxOperatingDescr.8.11.1.0 = node1 PIC: SPU Cp-Flow @ 4/0/*`
	`jnxOperatingDescr.8.11.2.0 = node1 PIC: SPU Flow @ 4/1/*`
	`jnxOperatingDescr.8.12.1.0 = node1 PIC: 4x 10GE XFP @ 5/0/*`
	`jnxOperatingDescr.8.12.2.0 = node1 PIC: 16x 1GE TX @ 5/1/*`

Accounting Profiles	

Table 4: Instrumentation for Chassis Component Monitoring *(continued)*

<table>
<tr><th>Junos OS XML RPC</th><th>SNMP MIB</th></tr>
<tr><td colspan="2">
<ul>
<li>Use a Routing Engine accounting profile to get the master Routing Engine statistics in comma separated value (CSV) format. Configure the routing-engine-profile under the <b>[edit accounting-options]</b> hierarchy level. The collection interval fields and filename can be configured per your requirements. We recommend transferring the file directly to a management system using the Junos OS transfer options provided under the <b>[edit accounting-options]</b> hierarchy level. Note that only the primary node master Routing Engine statistics are available.
<p>The Routing Engine accounting profile is stored in the <b>/var/log</b> directory by default. The following is a sample of an accounting profile:</p>
<pre>
#FILE CREATED 1246267286 2010-4-29-09:21:26
#hostname SRX3400-1
#profile-layout reprf,epoch-timestamp,hostname,date-yyyymmdd,timeofday-hhmmss,uptime,cpu1min,
cpu5min,cpu15min,memory-usage,total-cpu-usage
reprf,1246267691,SRX3400-1,20090629,092811,3044505,0.033203,0.030762,0.000488,523,6.10
reprf,1246268591,SRX3400-1,20090629,094311,3045405,0.000000,0.014160,0.000000,523,5.00
</pre></li>
<li>Use a MIB accounting profile for any other MIBs listed in the SNMP MIB column to get results in a CSV format. You can select the MIB objects, collection interval, and so on.</li>
</ul>
</td></tr>
</table>

Monitoring Chassis Cluster Performance

The information in Table 5 on page 44 describes how to measure and monitor the cluster health, including the control plane and data plane statistics.

Table 5: Instrumentation for Chassis Cluster Monitoring

Junos OS XML RPC	SNMP MIB
• Use the **get-chassis-cluster-statistics** remote procedure call (RPC) to get the cluster statistics, including the control plane, fabric, and dataplane statistics. • If you want to monitor dataplane and control plane statistics separately, you can use the **get-chassis-cluster-control-plane-statistics** and **get-chassis-cluster-data-plane-statistics** RPCs, respectively.	Not available. The utility MIB can be used to provide this data using Junos OS operation scripts. For more information about operation scripts.

Redundant Group Monitoring

Ensure that the redundancy groups are discovered prior to monitoring the group status. Table 6 on page 45 lists the methods used to obtain redundancy group monitoring information.

Table 6: Instrumentation for Redundancy Group Monitoring

Junos OS XML RPC	SNMP MIB

Use the **get-chassis-cluster-status** remote procedure call (RPC) to get chassis cluster information as shown.
`<rpc>`
`    <get-chassis-cluster-status>`
`        <redundancy-group>1</redundancy-group>`
`    </get-chassis-cluster-status>`
`</rpc>`
	Not available. The utility MIB can be used to provide this data using Junos OS operation scripts. For more information about operation scripts.

Interface Statistics

You can use the methods listed in Table 7 on page 45 to get interface statistics including the reth and fabric interfaces. Note that you can poll the reth interface statistics and then use the information to determine the redundancy group status because the non-active reth link shows 0 output packets per second (output-pps).

Table 7: Instrumentation for Interface Monitoring

Junos OS XML RPC	SNMP MIB
- Use the **get-interface-information** remote procedure call (RPC) with the **extensive** tag to get information such as interface statistics, COS statistics, and traffic statistics. This works for all interfaces including reth interfaces and fabric interfaces on the primary node and the secondary node, except the fxp0 interface on the secondary node. - Use the relationship between reth and underlying interfaces to determine the statistics between the physical interfaces. - The fxp0 interface on the secondary node can be directly queried using the IP address of the fxp0 interface on the secondary node.	- Use the following MIB tables for interface statistics: - ifTable – Standard MIB II interface stats - ifXTable – Standard MIB II high-capacity interface stats - JUNIPER-IF-MIB – A list of Juniper extensions to the interface entries - JUNIPER-JS-IF-EXT-MIB – Used to monitor the entries in the interfaces pertaining to the security management of the interface - For secondary node fxp0 interface details, directly query the secondary node (optional).
Accounting Profiles	
- Use Interface accounting profiles for interface statistics in CSV format collected at regular intervals. - Use MIB accounting profiles for any MIBs collected at regular intervals with output in CSV format. - Use Class usage profiles for source class and destination class usage.	

Services Processing Unit Monitoring

The SRX3000 line and SRX5000 line have one or more Services Processing Units (SPUs) that run on a Services Processing Card (SPC). All flow-based services run on the SPU. SPU monitoring tracks the health of the SPUs and of the central point. The central point (CP) in the architecture has two basic flow functionalities: load balancing and traffic identification (global session matching). The central point forwards a packet to its SPU upon session matching, or distributes traffic to an SPU for security processing if the packet does not match any existing session. The chassis manager on each SPC monitors the SPUs and the central point, and also maintains the heartbeat with the Routing Engine chassisd. In this hierarchical monitoring system, the chassis process (chassisd) is the center for hardware failure detection. SPU monitoring is enabled by default.

Use the methods listed in "Junos OS XML RPC Instrumentation for SPU Monitoring" on page 46 and "SNMP MIB Instrumentation for SPU Monitoring" on page 47 to get the SPU to monitor data.

NOTE: We recommend that the management systems set an alarm when SPU CPU utilization goes above 85 percent as this adds latency to the processing. Packets are dropped if the CPU utilization exceeds 95 percent.

Junos OS XML RPC Instrumentation for SPU Monitoring

- Use the **get-flow-session-information** remote procedure call (RPC) to get the SPU to monitor data such as total sessions, current sessions, and max sessions per node.

```
<rpc>
<get-flow-session-information>
<summary/>
</get-flow-session-information>
</rpc>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:junos="http://xml.juniper.net/junos/10.4D0/junos">
<multi-routing-engine-results>
<multi-routing-engine-item>
<re-name>node0</re-name>
<flow-session-information xmlns="http://xml.juniper.net/
junos/10.4D0/junos-flow">
<flow-fpc-pic-id> on FPC4 PIC0:</flow-fpc-pic-id>
</flow-session-information>
<flow-session-summary-information xmlns="http://
xml.juniper.net/junos/10.4D0/junos-flow">
<active-unicast-sessions>0</active-unicast-sessions>
<active-multicast-sessions>0</active-multicast-sessions>
<failed-sessions>0</failed-sessions>
<active-sessions>0</active-sessions>
<active-session-valid>0</active-session-valid>
<active-session-pending>0</active-session-pending>
<active-session-invalidated>0</active-session-invalidated>
<active-session-other>0</active-session-other>
<max-sessions>524288</max-sessions>
</flow-session-summary-information>
<flow-session-information xmlns="http://xml.juniper.net/
junos/10.4D0/junos-flow"></flow-session-information>
<flow-session-information xmlns="http://xml.juniper.net/
junos/10.4D0/junos-flow">
<flow-fpc-pic-id> on FPC4 PIC1:</flow-fpc-pic-id>
</flow-session-information>
<flow-session-summary-information xmlns="http://
xml.juniper.net/junos/10.4D0/junos-flow">
<active-unicast-sessions>0</active-unicast-sessions>
<active-multicast-sessions>0</active-multicast-sessions>
<failed-sessions>0</failed-sessions>
<active-sessions>0</active-sessions>
<active-session-valid>0</active-session-valid>
<active-session-pending>0</active-session-pending>
```

```
<active-session-invalidated>0</active-session-invalidated>
<active-session-other>0</active-session-other>
<max-sessions>1048576</max-sessions>
</flow-session-summary-information>
<flow-session-information xmlns="http://
xml.juniper.net/junos/10.4D0/junos-flow">
</flow-session-information>
</multi-routing-engine-item>
<multi-routing-engine-item>
<re-name>node1</re-name>
<flow-session-information xmlns="http://xml.juniper.net
/junos/10.4D0/junos-flow">
<flow-fpc-pic-id> on FPC4 PIC0:</flow-fpc-pic-id>
</flow-session-information>
<flow-session-summary-information xmlns="http://
xml.juniper.net/junos/10.4D0/junos-flow">
<active-unicast-sessions>0</active-unicast-sessions>
<active-multicast-sessions>0</active-multicast-sessions>
<failed-sessions>0</failed-sessions>
<active-sessions>0</active-sessions>
<active-session-valid>0</active-session-valid>
<active-session-pending>0</active-session-pending>
<active-session-invalidated>0</active-session-invalidated>
<active-session-other>0</active-session-other>
<max-sessions>524288</max-sessions>
</flow-session-summary-information>
<flow-session-information xmlns="http://
xml.juniper.net/junos/10.4D0/junos-flow">
</flow-session-information>
<flow-session-information xmlns="http://
xml.juniper.net/junos/10.4D0/junos-flow">
<flow-fpc-pic-id> on FPC4 PIC1:</flow-fpc-pic-id>
</flow-session-information>
<flow-session-summary-information xmlns="http://
xml.juniper.net/junos/10.4D0/junos-flow">
<active-unicast-sessions>0</active-unicast-sessions>
<active-multicast-sessions>0</active-multicast-sessions>
<failed-sessions>0</failed-sessions>
<active-sessions>0</active-sessions>
<active-session-valid>0</active-session-valid>
<active-session-pending>0</active-session-pending>
<active-session-invalidated>0</active-session-invalidated>
<active-session-other>0</active-session-other>
<max-sessions>1048576</max-sessions>
</flow-session-summary-information>
<flow-session-information xmlns="http://xml.juniper.
net/junos/10.4D0/junos-flow"></flow-session-information>
</multi-routing-engine-item>
</multi-routing-engine-results>
</rpc-reply>
```

- Use the **get-performance-session-information** RPC to obtain SPU session performance.
- Use the **get-spu-monitoring-information** RPC to monitor SPU CPU utilization, memory utilization, max flow sessions, and so on.

SNMP MIB Instrumentation for SPU Monitoring

- Use the jnxJsSPUMonitoring MIB to monitor the SPU data:

- jnxJsSPUMonitoringCurrentTotalSession – Returns the system-level current total sessions.
- jnxJsSPUMonitoringMaxTotalSession – Returns the system-level max sessions possible.
- jnxJsSPUMonitoringObjectsTable – Returns the SPU utilization statistics per node.

```
user@host> show snmp mib walk
```

```
jnxJsSPUMonitoringMIB
jnxJsSPUMonitoringFPCIndex.16 = 4
jnxJsSPUMonitoringFPCIndex.17 = 4
jnxJsSPUMonitoringFPCIndex.40 = 4
jnxJsSPUMonitoringFPCIndex.41 = 4
jnxJsSPUMonitoringSPUIndex.16 = 0
jnxJsSPUMonitoringSPUIndex.17 = 1
jnxJsSPUMonitoringSPUIndex.40 = 0
jnxJsSPUMonitoringSPUIndex.41 = 1
jnxJsSPUMonitoringCPUUsage.16 = 0
jnxJsSPUMonitoringCPUUsage.17 = 0
jnxJsSPUMonitoringCPUUsage.40 = 0
jnxJsSPUMonitoringCPUUsage.41 = 0
jnxJsSPUMonitoringMemoryUsage.16 = 70
jnxJsSPUMonitoringMemoryUsage.17 = 73
jnxJsSPUMonitoringMemoryUsage.40 = 70
jnxJsSPUMonitoringMemoryUsage.41 = 73
jnxJsSPUMonitoringCurrentFlowSession.16 = 0
jnxJsSPUMonitoringCurrentFlowSession.17 = 0
jnxJsSPUMonitoringCurrentFlowSession.40 = 0
jnxJsSPUMonitoringCurrentFlowSession.41 = 0
jnxJsSPUMonitoringMaxFlowSession.16 = 524288
jnxJsSPUMonitoringMaxFlowSession.17 = 1048576
jnxJsSPUMonitoringMaxFlowSession.40 = 524288
jnxJsSPUMonitoringMaxFlowSession.41 = 1048576
jnxJsSPUMonitoringCurrentCPSession.16 = 0
jnxJsSPUMonitoringCurrentCPSession.17 = 0
jnxJsSPUMonitoringCurrentCPSession.40 = 0
jnxJsSPUMonitoringCurrentCPSession.41 = 0
jnxJsSPUMonitoringMaxCPSession.16 = 2359296
jnxJsSPUMonitoringMaxCPSession.17 = 0
jnxJsSPUMonitoringMaxCPSession.40 = 2359296
jnxJsSPUMonitoringMaxCPSession.41 = 0
jnxJsSPUMonitoringNodeIndex.16 = 0
jnxJsSPUMonitoringNodeIndex.17 = 0
jnxJsSPUMonitoringNodeIndex.40 = 1
jnxJsSPUMonitoringNodeIndex.41 = 1
jnxJsSPUMonitoringNodeDescr.16 = node0
jnxJsSPUMonitoringNodeDescr.17 = node0
jnxJsSPUMonitoringNodeDescr.40 = node1
jnxJsSPUMonitoringNodeDescr.41 = node1
jnxJsSPUMonitoringCurrentTotalSession.0 =
jnxJsSPUMonitoringMaxTotalSession.0 = 1572864
```

NOTE:

- Junos OS versions prior to Junos OS Release 9.6 only return local node data for this MIB. To support a chassis cluster, Junos OS Release 9.6 and later support a jnxJsSPUMonitoringNodeIndex index and a jnxJsSPUMonitoringNodeDescr field in the table. Therefore, in chassis cluster mode, Junos OS Release 9.6 and later return SPU monitoring data of both the primary and secondary nodes.
- SRX Series branch devices have a virtualized dataplane across the cluster datacores. Therefore, they are reported as one SPU with an index of 0.

The jnxJsSPUMonitoringMaxFlowSession MIB object shows the maximum number of sessions per node.

Security Features

Following is a summary of Junos OS XML remote procedure calls (RPCs) and SNMP MIBs related to security features that are supported on SRX Series devices.

The RPCs and MIBs might not be directly comparable to each other. One might provide more or less information than the other. Use the following information to determine which instrumentation to use.

Table 8: Instrumentation for Security Monitoring

Feature and Functionality	Junos OS XML RPC	SNMP MIB
IPsec	<get-ipsec-tunnel-redundancy-information> <get-services-ipsec-statistics-information> <get-ike-security-associations>	JNX-IPSEC-MONITOR-MIB JUNIPER-JS-IPSEC-VPN JUNIPER-IPSEC-FLOW-MONITOR
NAT	<get-service-nat-mapping-information> <get-service-nat-pool-information>	JNX-JS-NAT-MIB
Screening	<get-ids-statistics>	JNX-JS-SCREENING-MIB
Firewall	<get-firewall-counter-information> <get-firewall-filter-information> <get-firewall-information> <get-firewall-log-information> <get-firewall-prefix-action-information> <get-flow-table-statistics-information>	JUNIPER-FIREWALL-MIB
Security Policies	<get-firewall-policies>	JUNIPER-JS-POLICY-MIB

Table 8: Instrumentation for Security Monitoring *(continued)*

Feature and Functionality	Junos OS XML RPC	SNMP MIB
AAA	<get-aaa-module-statistics> <get-aaa-subscriber-statistics> <get-aaa-subscriber-table>	JUNIPER-USER-AAA-MIB
IDP	<get-idp-addos-application-information> <get-idp-application-system-cache> <get-idp-counter-information> <get-idp-detail-status-information> <get-idp-memory-information> <get-idp-policy-template-information> <get-idp-predefined-attack-filters> <get-idp-predefined-attack-groups> <get-idp-predefined-attacks> <get-idp-recent-security-package-information> <get-idp-security-package-information> <get-idp-ssl-key-information> <get-idp-ssl-session-cache-information> <get-idp-status-information> <get-idp-subscriber-policy-list>	JUNIPER-JS-IDP-MIB

Other Statistics and MIBS

There are other MIBs such as the OSPF MIB and IP Forwarding MIB that are supported on SRX Series devices. See the *Network Management Administration Guide*, *MIB Reference for SRX1400, SRX3400, and SRX3600 Services Gateways*, and *MIB Reference for SRX5600 and SRX5800 Services Gateways* for details about other MIBs supported on SRX Series devices.

RMON

Junos OS supports the remote monitoring (RMON) MIB (RFC 2819). RMON can be used to send alerts for MIB variables when upper and lower thresholds are crossed. This can be used for various MIB variables. Some good examples are interface statistics monitoring and Routing Engine CPU monitoring.

The following configuration snippet shows RMON configuration for monitoring a Routing Engine on node 0 of a cluster and for monitoring octets out of interface index 2000:

```
rmon {
  alarm 100 {
    interval 5;
    variable jnxOperatingCPU.9.1.0.0;
    sample-type absolute-value;
    request-type get-request;
    rising-threshold 90;
    falling-threshold 80;
    rising-event-index 100;
    falling-event-index 100;
  }
  event 100 {
    type log-and-trap;
    community petblr;
  }
  alarm 10 {
    interval 60;
    variable ifHCInOctets.2000;
    sample-type delta-value;
    request-type get-request;
    startup-alarm rising-alarm;
    rising-threshold 100000;
    falling-threshold 0;
    rising-event-index 10;
    falling-event-index 10;
  }
  event 10 {
    type log-and-trap;
    community test;
  }
}
```

Chassis Cluster Device Health Monitoring

On Juniper Networks routers, RMON alarms and events provide much of the infrastructure needed to reduce the polling overhead from the network management system (NMS). However, with this approach, you must set up the NMS to configure specific MIB objects into RMON alarms. This often requires device-specific expertise and customization of the monitoring application. In addition, some MIB object instances that need monitoring are set only at initialization or change at runtime and cannot be configured in advance. To address these issues, the health monitor extends the RMON alarm infrastructure to provide predefined monitoring for a selected set of object instances (for file system usage, CPU usage, and memory usage) and includes support for unknown or dynamic object instances (such as Junos OS processes).

Related Documentation

- Monitoring Nodes in a Chassis Cluster on page 34
- Monitoring Chassis Cluster Faults on page 51

Monitoring Chassis Cluster Faults

You can use SNMP traps and system log messages for fault monitoring of SRX Series chassis clusters.

SNMP Traps

Table 9 on page 52 lists the SNMP traps supported on SRX Series devices. Note that only the primary node sends SNMP traps. For details of each trap, see the *Network Management Administration Guide*, *MIB Reference for SRX1400, SRX3400, and SRX3600 Services Gateways*, and *MIB Reference for SRX5600 and SRX5800 Services Gateways*.

Table 9: Supported SNMP Traps

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
authenticationFailure	1.3.6.1.6.3.1.1.5.5	Authentication	All Junos OS devices	None
linkDown	1.3.6.1.6.3.1.1.5.3	Link	All Junos OS devices	1. ifIndex -1.3.6.1.2.1.2.2.1.1 2. ifAdminStatus -1.3.6.1.2.1.2.2.1.7 3. ifOperStatus - 1.3.6.1.2.1.2.2.1.8 Additionally, ifName - 1.3.6.1.2.1.31.1.1.1.1 is also sent.
linkUp	1.3.6.1.6.3.1.1.5.4	Link	All Junos OS devices	1. ifIndex -1.3.6.1.2.1.2.2.1.1 2. ifAdminStatus -1.3.6.1.2.1.2.2.1.7 3. ifOperStatus - 1.3.6.1.2.1.2.2.1.8 Additionally, ifName - 1.3.6.1.2.1.31.1.1.1.1 is also sent.
pingProbeFailed	1.3.6.1.2.1.80.0.1	Remote operations	All Junos OS devices	1. pingCtlTargetAddressType - 1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress - 1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - 1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType -1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTargetAddress - 1.3.6.1.2.1.80.1.3.1.3 6. pingResultsMinRtt -1.3.6.1.2.1.80.1.3.1.4 7. pingResultsMaxRtt -1.3.6.1.2.1.80.1.3.1.5 8. pingResultsAverageRtt -1.3.6.1.2.1.80.1.3.1.6 9. pingResultsProbeResponses - 1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes -1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - 1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe - 1.3.6.1.2.1.80.1.3.1.10

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
pingTestFailed	1.3.6.1.2.1.80.0.2	Remote operations	All Junos OS devices	1. pingCtlTargetAddressType - 1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress - 1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - 1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType -1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTarget Address - 1.3.6.1.2.1.80.1.3.1.3 6. pingResultsMinRtt -1.3.6.1.2.1.80.1.3.1.4 7. pingResultsMaxRtt -1.3.6.1.2.1.80.1.3.1.5 8. pingResultsAverageRtt -1.3.6.1.2.1.80.1.3.1.6 9. pingResultsProbeResponses - 1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes -1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - 1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe - 1.3.6.1.2.1.80.1.3.1.10
pingTestCompleted	1.3.6.1.2.1.80.0.3	Remote operations	All Junos OS devices	1. pingCtlTargetAddressType - 1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress - 1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - 1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType -1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTargetAddress - 1.3.6.1.2.1.80.1.3.1.3 6. pingResultsMinRtt -1.3.6.1.2.1.80.1.3.1.4 7. pingResultsMaxRtt -1.3.6.1.2.1.80.1.3.1.5 8. pingResultsAverageRtt -1.3.6.1.2.1.80.1.3.1.6 9. pingResultsProbeResponses - 1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes -1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - 1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe - 1.3.6.1.2.1.80.1.3.1.10

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
traceRoutePathChange	1.3.6.1.2.1.81.0.1	Remote operations	All Junos OS devices	1. traceRouteCtlTargetAddressType - 1.3.6.1.2.1.81.1.2.1.3 2. traceRouteCtlTargetAddress - 1.3.6.1.2.1.81.1.2.1.4 3. traceRouteResultsIpTgtAddrType - 1.3.6.1.2.1.81.1.5.1.2 4. traceRouteResultsIpTgtAddr - 1.3.6.1.2.1.81.1.5.1.3
traceRouteTestFailed	1.3.6.1.2.1.81.0.2	Remote operations	All Junos OS devices	1. traceRouteCtlTargetAddressType-1.3.6.1.2.1.81.1.2.1.3 2. traceRouteCtlTargetAddress - 1.3.6.1.2.1.81.1.2.1.4 3. traceRouteResultsIpTgtAddrType -1.3.6.1.2.1.81.1.5.1.2 4. traceRouteResultsIpTgtAddr - 1.3.6.1.2.1.81.1.5.1.3
traceRouteTestCompleted	1.3.6.1.2.1.81.0.3	Remote operations	All Junos OS devices	1. traceRouteCtlTargetAddressType - 1.3.6.1.2.1.81.1.2.1.3 2. traceRouteCtlTargetAddress - 1.3.6.1.2.1.81.1.2.1.4 3. traceRouteResultsIpTgtAddrType - 1.3.6.1.2.1.81.1.5.1.2 4. traceRouteResultsIpTgtAddr - 1.3.6.1.2.1.81.1.5.1.3
fallingAlarm	1.3.6.1.2.1.16.0.1	RMON alarm	All Junos OS devices	1. alarmIndex - .1.3.6.1.2.1.16.3.1.1.1 2. alarmVariable - .1.3.6.1.2.1.16.3.1.1.3 3. alarmSampleType - .1.3.6.1.2.1.16.3.1.1.4 4. alarmValue - .1.3.6.1.2.1.16.3.1.1.5 5. alarmFallingThreshold - .1.3.6.1.2.1.16.3.1.1.8
risingAlarm	1.3.6.1.2.1.16.0.2	RMON alarm	All Junos OS devices	1. alarmIndex - .1.3.6.1.2.1.16.3.1.1.1 2. alarmVariable - .1.3.6.1.2.1.16.3.1.1.3 3. alarmSampleType - .1.3.6.1.2.1.16.3.1.1.4 4. alarmValue - .1.3.6.1.2.1.16.3.1.1.5 5. alarmRisingThreshold - .1.3.6.1.2.1.16.3.1.1.7
bgpEstablished	1.3.6.1.2.1.15.7.1	Routing	M, T, MX, J, EX, SRX Branch	1. bgpPeerLastError - 1.3.6.1.2.1.15.3.1.14 2. bgpPeerState - The BGP peer connection state.

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
bgpBackwardTransition	1.3.6.1.2.1.15.7.2	Routing	M, T, MX, J, EX, SRX Branch	1. bgpPeerLastError - 1.3.6.1.2.1.15.3.1.14 2. bgpPeerState - The BGP peer connection state.
ospfVirtIfStateChange	1.3.6.1.2.1.14.16.2.1	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId - 1.3.6.1.2.1.14.1.1 2. ospfVirtIfAreaId - 1.3.6.1.2.1.14.9.1.1 3. ospfVirtIfNeighbor - 1.3.6.1.2.1.14.9.1.2 4. ospfVirtIfState - 1.3.6.1.2.1.14.9.1.7
ospfNbrStateChange	1.3.6.1.2.1.14.16.2.2	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId -1.3.6.1.2.1.14.1.1 2. ospfNbrIpAddr -.1.3.6.1.2.1.14.10.1.1 3. ospfNbrAddressLessIndex - .1.3.6.1.2.1.14.10.1.2 4. ospfNbrRtrId - .1.3.6.1.2.1.14.10.1.3 5. ospfNbrState - .1.3.6.1.2.1.14.10.1.6
ospfVirtNbrStateChange	1.3.6.1.2.1.14.16.2.3	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId - .1.3.6.1.2.1.14.1.1 2. ospfVirtNbrArea - .1.3.6.1.2.1.14.11.1.1 3. ospfVirtNbrRtrId - .1.3.6.1.2.1.14.11.1.2 4. ospfVirtNbrState - .1.3.6.1.2.1.14.11.1.5
ospfIfConfigError	1.3.6.1.2.1.14.16.2.4	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId -.1.3.6.1.2.1.14.1.1.0 2. ospfIfIpAddress - .1.3.6.1.2.1.14.7.1.1 3. ospfAddressLessIf -.1.3.6.1.2.1.14.7.1.2 4. ospfPacketSrc - .1.3.6.1.2.1.14.16.1.4.0 5. ospfConfigErrorType - .1.3.6.1.2.1.14.16.1.2.0 6. ospfPacketType - .1.3.6.1.2.1.14.16.1.3.0
ospfVirtIfConfigError	1.3.6.1.2.1.14.16.2.5	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId - .1.3.6.1.2.1.14.1.1.0 2. ospfVirtIfAreaId - .1.3.6.1.2.1.14.9.1.1 3. ospfVirtIfNeighbor - .1.3.6.1.2.1.14.9.1.2 4. ospfConfigErrorType - 1.3.6.1.2.1.14.16.1.2.0 5. ospfPacketType - 1.3.6.1.2.1.14.16.1.3.0 1: 2: 3: 4: 5:

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
ospfIfAuthFailure	1.3.6.1.2.1.14.16.2.6	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId -.1.3.6.1.2.1.14.1.1.0 2. ospfIfIpAddress - .1.3.6.1.2.1.14.7.1.1 3. ospfAddressLessIf - .1.3.6.1.2.1.14.7.1.2 4. ospfPacketSrc - .1.3.6.1.2.1.14.16.1.4.0 5. ospfConfigErrorType - .1.3.6.1.2.1.14.16.1.2.0 6. ospfPacketType - .1.3.6.1.2.1.14.16.1.3.0
ospfVirtIfAuthFailure	1.3.6.1.2.1.14.16.2.7	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId - .1.3.6.1.2.1.14.1.1.0 2. ospfVirtIfAreaId - .1.3.6.1.2.1.14.9.1.1 3. ospfVirtIfNeighbor - .1.3.6.1.2.1.14.9.1.2 4. ospfConfigErrorType - 1.3.6.1.2.1.14.16.1.2.0 5. ospfPacketType - 1.3.6.1.2.1.14.16.1.3.0
ospfIfRxBadPacket	1.3.6.1.2.1.14.16.2.8	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId -.1.3.6.1.2.1.14.1.1.0 2. ospfIfIpAddress - .1.3.6.1.2.1.14.7.1.1 3. ospfAddressLessIf -.1.3.6.1.2.1.14.7.1.2 4. ospfPacketSrc - .1.3.6.1.2.1.14.16.1.4.0 5. ospfPacketType - .1.3.6.1.2.1.14.16.1.3.0
ospfVirtIfRxBadPacket	1.3.6.1.2.1.14.16.2.9	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId - .1.3.6.1.2.1.14.1.1.0 2. ospfVirtIfAreaId - .1.3.6.1.2.1.14.9.1.1 3. ospfVirtIfNeighbor - .1.3.6.1.2.1.14.9.1.2 4. ospfPacketType - 1.3.6.1.2.1.14.16.1.3.0

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
ospfTxRetransmit	1.3.6.1.2.1.14.16.2.10	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId -.1.3.6.1.2.1.14.1.1.0 2. ospfIfIpAddress - .1.3.6.1.2.1.14.7.1.1 3. ospfAddressLessIf -.1.3.6.1.2.1.14.7.1.2 4. ospfNbrRtrId - .1.3.6.1.2.1.14.11.1.2 5. ospfPacketType -.1.3.6.1.2.1.14.16.1.4.0 6. ospfLsdbType - .1.3.6.1.2.1.14.4.1.2 7. ospfLsdbLsid - .1.3.6.1.2.1.14.4.1.3 8. ospfLsdbRouterId - .1.3.6.1.2.1.14.4.1.4
ospfVirtIfTxRetransmit	1.3.6.1.2.1.14.16.2.11	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId - .1.3.6.1.2.1.14.1.1.0 2. ospfVirtIfAreaId - .1.3.6.1.2.1.14.9.1.1 3. ospfVirtIfNeighbor -.1.3.6.1.2.1.14.9.1.2 4. ospfPacketType -1.3.6.1.2.1.14.16.1.3.0 5. ospfLsdbType -.1.3.6.1.2.1.14.4.1.2 6. ospfLsdbLsid -.1.3.6.1.2.1.14.4.1.3 7. ospfLsdbRouterId -.1.3.6.1.2.1.14.4.1.4
ospfMaxAgeLsa	1.3.6.1.2.1.14.16.2.13	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId -.1.3.6.1.2.1.14.1.1.0 2. ospfLsdbAreaId -.1.3.6.1.2.1.14.4.1.1 3. ospfLsdbType-.1.3.6.1.2.1.14.4.1.2 4. ospfLsdbLsid-.1.3.6.1.2.1.14.4.1.3 5. ospfLsdbRouterId-.1.3.6.1.2.1.14.4.1.4
ospfIfStateChange	1.3.6.1.2.1.14.16.2.16	Routing	M, T, MX, J, EX, SRX Branch	1. ospfRouterId -.1.3.6.1.2.1.14.1.1.0 2. ospfIfIpAddress - .1.3.6.1.2.1.14.7.1.1 3. ospfAddressLessIf -.1.3.6.1.2.1.14.7.1.2 4. ospfIfState -.1.3.6.1.2.1.14.7.1.12
coldStart	1.3.6.1.6.3.1.1.5.1	Startup	All Junos OS devices	None
warmStart	1.3.6.1.6.3.1.1.5.2	Startup	All Junos OS devices	None
vrrpTrapNewMaster	1.3.6.1.2.1.68.0.1	VRRP	All Junos OS devices	vrrpOperMasterIpAddr - .1.3.6.1.2.1.68.1.3.1.7

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
vrrpTrapAuthFailure	1.3.6.1.2.1.68.0.2	VRRP	All Junos OS devices	1. vrrpTrapPacketSrc -.1.3.6.1.2.1.68.1.5.0 2. vrrpTrapAuthErrorType -.1.3.6.1.2.1.68.1.6.0
mplsTunnelUp	1.3.6.1.2.1.10.166.3.0.1	Routing	M, T, MX, J, EX, SRX Branch	1. mplsTunnelAdminStatus -.1.3.6.1.2.1.10.166.3.2.2.1.34 2. mplsTunnelOperStatus -.1.3.6.1.2.1.10.166.3.2.2.1.35
mplsTunnelDown	1.3.6.1.2.1.10.166.3.0.2	Routing	M, T, MX, J, EX, SRX Branch	1. mplsTunnelAdminStatus -.1.3.6.1.2.1.10.166.3.2.2.1.34 2. mplsTunnelOperStatus -.1.3.6.1.2.1.10.166.3.2.2.1.35
mplsTunnelRerouted	1.3.6.1.2.1.10.166.3.0.3	Routing	M, T, MX, J, EX, SRX Branch	1. mplsTunnelAdminStatus -.1.3.6.1.2.1.10.166.3.2.2.1.34 2. mplsTunnelOperStatus -.1.3.6.1.2.1.10.166.3.2.2.1.35
mplsTunnelReoptimized	1.3.6.1.2.1.10.166.3.0.4	Routing	M, T, MX, J, EX, SRX Branch	1. mplsTunnelAdminStatus -.1.3.6.1.2.1.10.166.3.2.2.1.34 2. mplsTunnelOperStatus -.1.3.6.1.2.1.10.166.3.2.2.1.35
jnxPowerSupplyFailure	1.3.6.1.4.1.2636.4.1.1	Chassis (alarm conditions)	All Junos OS devices	1. jnxContentsContainerIndex - .1.3.6.1.4.1.2636.3.1.8.1.1 2. jnxContentsL1Index -.1.3.6.1.4.1.2636.3.1.8.1.2 3. jnxContentsL2Index - .1.3.6.1.4.1.2636.3.1.8.1.3 4. jnxContentsL3Index -.1.3.6.1.4.1.2636.3.1.8.1.4 5. jnxContentsDescr -.1.3.6.1.4.1.2636.3.1.8.1.6 6. jnxOperatingState -.1.3.6.1.4.1.2636.3.1.13.1.6
jnxFanFailure	1.3.6.1.4.1.2636.4.1.2	Chassis (alarm conditions)	All Junos OS devices	1. jnxContentsContainerIndex - .1.3.6.1.4.1.2636.3.1.8.1.1 2. jnxContentsL1Index - .1.3.6.1.4.1.2636.3.1.8.1.2 3. jnxContentsL2Index - .1.3.6.1.4.1.2636.3.1.8.1.3 4. jnxContentsL3Index - .1.3.6.1.4.1.2636.3.1.8.1.4 5. jnxContentsDescr - .1.3.6.1.4.1.2636.3.1.8.1.6 6. jnxOperatingState - .1.3.6.1.4.1.2636.3.1.13.1.6

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxOverTemperature	1.3.6.1.4.1.2636.4.1.3	Chassis (alarm conditions)	All Junos OS devices	1. jnxContentsContainerIndex - .1.3.6.1.4.1.2636.3.1.8.1.1 2. jnxContentsL1Index - .1.3.6.1.4.1.2636.3.1.8.1.2 3. jnxContentsL2Index - .1.3.6.1.4.1.2636.3.1.8.1.3 4. jnxContentsL3Index - .1.3.6.1.4.1.2636.3.1.8.1.4 5. jnxContentsDescr - .1.3.6.1.4.1.2636.3.1.8.1.6 6. jnxOperatingTemp - 1.3.6.1.4.1.2636.3.1.13.1.7
jnxRedundancySwitchOver	1.3.6.1.4.1.2636.4.1.4	Chassis (alarm conditions)	All Junos OS devices	1. jnxRedundancyContentsIndex - 1.3.6.1.4.1.2636.3.1.14.1.1 2. jnxRedundancyL1Index - 1.3.6.1.4.1.2636.3.1.14.1.2 3. jnxRedundancyL2Index - 1.3.6.1.4.1.2636.3.1.14.1.3 4. jnxRedundancyL3Index - 1.3.6.1.4.1.2636.3.1.14.1.4 5. jnxRedundancyDescr - 1.3.6.1.4.1.2636.3.1.14.1.5 6. jnxRedundancyConfig - 1.3.6.1.4.1.2636.3.1.14.1.6 7. jnxRedundancyState - 1.3.6.1.4.1.2636.3.1.14.1.7 8. jnxRedundancySwitchoverCount - 1.3.6.1.4.1.2636.3.1.14.1.8 9. jnxRedundancySwitchoverTime - 1.3.6.1.4.1.2636.3.1.14.1.9 10. jnxRedundancySwitchoverReason - 1.3.6.1.4.1.2636.3.1.14.1.10
jnxFruRemoval	1.3.6.1.4.1.2636.4.1.5	Chassis (alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index - 1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxFruInsertion	1.3.6.1.4.1.2636.4.1.6	Chassis (alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index - 1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7
jnxFruPowerOff	1.3.6.1.4.1.2636.4.1.7	Chassis (alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index - 1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7 8. jnxFruOfflineReason - 1.3.6.1.4.1.2636.3.1.15.1.10 9. jnxFruLastPowerOff - 1.3.6.1.4.1.2636.3.1.15.1.11 10. jnxFruLastPowerOn - 1.3.6.1.4.1.2636.3.1.15.1.12

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxFruPowerOn	1.3.6.1.4.1.2636.4.1.8	Chassis (alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index - 1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7 8. jnxFruOfflineReason - 1.3.6.1.4.1.2636.3.1.15.1.10 9. jnxFruLastPowerOff - 1.3.6.1.4.1.2636.3.1.15.1.11 10. jnxFruLastPowerOn - 1.3.6.1.4.1.2636.3.1.15.1.12
jnxFruFailed	1.3.6.1.4.1.2636.4.1.9	Chassis (alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index -1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxFruOffline	1.3.6.1.4.1.2636.4.1.10	Chassis (alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index - 1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7 8. jnxFruOfflineReason - 1.3.6.1.4.1.2636.3.1.15.1.10 9. jnxFruLastPowerOff - 1.3.6.1.4.1.2636.3.1.15.1.11 10. jnxFruLastPowerOn - 1.3.6.1.4.1.2636.3.1.15.1.12
jnxFruOnline	1.3.6.1.4.1.2636.4.1.11	Chassis (alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index - 1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxFruCheck	1.3.6.1.4.1.2636.4.1.12	Chassis (alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index - 1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7
jnxFEBSwitchover	1.3.6.1.4.1.2636.4.1.13	Chassis (alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index - 1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7
jnxHardDiskFailed	1.3.6.1.4.1.2636.4.1.14	Chassis (alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index - 1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxHardDiskMissing	1.3.6.1.4.1.2636.4.1.15	Chassis (alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index - 1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7
jnxPowerSupplyOK	1.3.6.1.4.1.2636.4.2.1	Chassis (cleared alarm conditions)	All Junos OS devices	1. jnxContentsContainerIndex - .1.3.6.1.4.1.2636.3.1.8.1.1 2. jnxContentsL1Index - .1.3.6.1.4.1.2636.3.1.8.1.2 3. jnxContentsL2Index - .1.3.6.1.4.1.2636.3.1.8.1.3 4. jnxContentsL3Index - .1.3.6.1.4.1.2636.3.1.8.1.4 5. jnxContentsDescr - .1.3.6.1.4.1.2636.3.1.8.1.6 6. jnxOperatingState - .1.3.6.1.4.1.2636.3.1.13.1.6
jnxFanOK	1.3.6.1.4.1.2636.4.2.2	Chassis (cleared alarm conditions)	All Junos OS devices	1. jnxContentsContainerIndex - .1.3.6.1.4.1.2636.3.1.8.1.1 2. jnxContentsL1Index - .1.3.6.1.4.1.2636.3.1.8.1.2 3. jnxContentsL2Index - .1.3.6.1.4.1.2636.3.1.8.1.3 4. jnxContentsL3Index - .1.3.6.1.4.1.2636.3.1.8.1.4 5. jnxContentsDescr - .1.3.6.1.4.1.2636.3.1.8.1.6 6. jnxOperatingState - .1.3.6.1.4.1.2636.3.1.13.1.6

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxTemperatureOK		Chassis (cleared alarm conditions)	All Junos OS devices	1. jnxContentsContainerIndex - .1.3.6.1.4.1.2636.3.1.8.1.1 2. jnxContentsL1Index - .1.3.6.1.4.1.2636.3.1.8.1.2 3. jnxContentsL2Index - .1.3.6.1.4.1.2636.3.1.8.1.3 4. jnxContentsL3Index - .1.3.6.1.4.1.2636.3.1.8.1.4 5. jnxContentsDescr - .1.3.6.1.4.1.2636.3.1.8.1.6 6. jnxOperatingTemp - 1.3.6.1.4.1.2636.3.1.13.1.7
jnxFRUOK	1.3.6.1.4.1.2636.4.2.4	Chassis (cleared alarm conditions)	All Junos OS devices	1. jnxFruContentsIndex - 1.3.6.1.4.1.2636.3.1.15.1.1 2. jnxFruL1Index - 1.3.6.1.4.1.2636.3.1.15.1.2 3. jnxFruL2Index - 1.3.6.1.4.1.2636.3.1.15.1.3 4. jnxFruL3Index - 1.3.6.1.4.1.2636.3.1.15.1.4 5. jnxFruName - 1.3.6.1.4.1.2636.3.1.15.1.5 6. jnxFruType - 1.3.6.1.4.1.2636.3.1.15.1.6 7. jnxFruSlot - 1.3.6.1.4.1.2636.3.1.15.1.7
jnxCmCfgChange	1.3.6.1.4.1.2636.4.5.0.1	Configuration	All Junos OS devices	1. jnxCmCfgChgEventTime 2. jnxCmCfgChgEventDate 3. jnxCmCfgChgEventSource 4. jnxCmCfgChgEventUser 5. jnxCmCfgChgEventLog
jnxCmRescueChange	1.3.6.1.4.1.2636.4.5.0.2	Configuration	All Junos OS devices	1. jnxCmRescueChgTime 2. jnxCmRescueChgDate 3. jnxCmRescueChgSource 4. jnxCmRescueChgUser 5. jnxCmRescueChgState

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxPingRttThresholdExceeded	1.3.6.1.4.1.2636.4.9.0.1	Remote operations	All Junos OS devices except EX and high-end SRX Series devices	1. pingCtlTargetAddressType - .1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress - .1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - .1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType - .1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTargetAddress - .1.3.6.1.2.1.80.1.3.1.3 6. jnxPingResultsMinRttUs - .1.3.6.1.4.1.2636.3.7.1.3.1.3 7. jnxPingResultsMaxRttUs - .1.3.6.1.4.1.2636.3.7.1.3.1.4 8. jnxPingResultsAvgRttUs - .1.3.6.1.4.1.2636.3.7.1.3.1.5 9. pingResultsProbeResponses - .1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes - .1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - .1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe - .1.3.6.1.2.1.80.1.3.1.10 13. jnxPingCtlRttThreshold - .1.3.6.1.4.1.2636.3.7.1.2.1.7 14. jnxPingResultsRttUs - .1.3.6.1.4.1.2636.3.7.1.3.1.1

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxPingRttStdDevThresholdExceeded	1.3.6.1.4.1.2636.4.9.0.2	Remote operations	All Junos OS devices except EX and high-end SRX Series devices	1. pingCtlTargetAddressType - .1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress - .1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - .1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType - .1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTargetAddress - .1.3.6.1.2.1.80.1.3.1.3 6. jnxPingResultsMinRttUs - .1.3.6.1.4.1.2636.3.7.1.3.1.3 7. jnxPingResultsMaxRttUs - .1.3.6.1.4.1.2636.3.7.1.3.1.4 8. jnxPingResultsAvgRttUs -.1.3.6.1.4.1.2636.3.7.1.3.1.5 9. pingResultsProbeResponses - .1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes - .1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - .1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe -.1.3.6.1.2.1.80.1.3.1.10 13. jnxPingCtlRttStdDevThreshold - .1.3.6.1.4.1.2636.3.7.1.2.1.11 14. jnxPingResultsStdDevRttUs - .1.3.6.1.4.1.2636.3.7.1.3.1.6

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxPingRttJitterThresholdExceeded	1.3.6.1.4.1.2636.4.9.0.3	Remote operations	All Junos OS devices except EX and high-end SRX Series devices	1. pingCtlTargetAddressType - .1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress - .1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - .1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType - .1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTargetAddress - .1.3.6.1.2.1.80.1.3.1.3 6. jnxPingResultsMinRttUs - .1.3.6.1.4.1.2636.3.7.1.3.1.3 7. jnxPingResultsMaxRttUs - .1.3.6.1.4.1.2636.3.7.1.3.1.4 8. jnxPingResultsAvgRttUs - .1.3.6.1.4.1.2636.3.7.1.3.1.5 9. pingResultsProbeResponses - .1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes - .1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - .1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe - .1.3.6.1.2.1.80.1.3.1.10 13. jnxPingCtlRttJitterThreshold - .1.3.6.1.4.1.2636.3.7.1.2.1.9

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxPingEgressThresholdExceeded	1.3.6.1.4.1.2636.4.9.0.4	Remote operations	All Junos OS devices except EX and high-end SRX Series devices	1. pingCtlTargetAddressType - .1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress -.1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - .1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType - .1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTargetAddress - .1.3.6.1.2.1.80.1.3.1.3 6. jnxPingResultsMinEgressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.8 7. jnxPingResultsMaxEgressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.9 8. jnxPingResultsAvgEgressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.10 9. pingResultsProbeResponses - .1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes - .1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - .1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe - .1.3.6.1.2.1.80.1.3.1.10 13. jnxPingCtlEgressTimeThreshold - .1.3.6.1.4.1.2636.3.7.1.2.1.10 14. jnxPingResultsEgressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.7

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxPingEgressStdDevThresholdExceeded	1.3.6.1.4.1.2636.4.9.0.5	Remote operations	All Junos OS devices except EX and high-end SRX Series devices	1. pingCtlTargetAddressType - .1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress - .1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - .1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType - .1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTargetAddress - .1.3.6.1.2.1.80.1.3.1.3 6. jnxPingResultsMinEgressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.8 7. jnxPingResultsMaxEgressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.9 8. jnxPingResultsAvgEgressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.10 9. pingResultsProbeResponses - .1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes - .1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - .1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe - .1.3.6.1.2.1.80.1.3.1.10 13. jnxPingResultsStddevEgressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.11 14. jnxPingCtlEgressStdDevThreshold - .1.3.6.1.4.1.2636.3.7.1.2.1.11

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxPingEgressJitterThresholdExceeded	1.3.6.1.4.1.2636.4.9.0.6	Remote operations	All Junos OS devices except EX and high-end SRX Series devices	1. pingCtlTargetAddressType - .1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress - .1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - .1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType - .1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTargetAddress - .1.3.6.1.2.1.80.1.3.1.3 6. jnxPingResultsMinEgressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.8 7. jnxPingResultsMaxEgressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.9 8. jnxPingResultsAvgEgressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.10 9. pingResultsProbeResponses - .1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes - .1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - .1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe - .1.3.6.1.2.1.80.1.3.1.10 13. jnxPingCtlEgressJitterThreshold - .1.3.6.1.4.1.2636.3.7.1.2.1.12

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxPingIngressThresholdExceeded	1.3.6.1.4.1.2636.4.9.0.7	Remote operations	All Junos OS devices except EX and high-end SRX Series devices	1. pingCtlTargetAddressType - .1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress - .1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - .1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType - .1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTargetAddress - .1.3.6.1.2.1.80.1.3.1.3 6. jnxPingResultsMinIngressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.13 7. jnxPingResultsMaxIngressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.14 8. jnxPingResultsAvgIngressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.15 9. pingResultsProbeResponses - .1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes - .1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - .1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe - .1.3.6.1.2.1.80.1.3.1.10 13. jnxPingCtlIngressTimeThreshold - .1.3.6.1.4.1.2636.3.7.1.2.1.13 14. jnxPingResultsIngressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.12

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxPingIngressStddevThresholdExceeded	1.3.6.1.4.1.2636.4.9.0.8	Remote operations	All Junos OS devices except EX and high-end SRX Series devices	1. pingCtlTargetAddressType - .1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress - .1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - .1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType - .1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTargetAddress - .1.3.6.1.2.1.80.1.3.1.3 6. jnxPingResultsMinIngressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.13 7. jnxPingResultsMaxIngressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.14 8. jnxPingResultsAvgIngressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.15 9. pingResultsProbeResponses - .1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes - .1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - .1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe - .1.3.6.1.2.1.80.1.3.1.10 13. jnxPingResultsStddevIngressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.16 14. jnxPingCtlIngressStddevThreshold - .1.3.6.1.4.1.2636.3.7.1.2.1.14

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxPingIngressJitterThresholdExceeded	1.3.6.1.4.1.2636.4.9.0.9	Remote operations	All Junos OS devices except EX and high-end SRX Series devices	1. pingCtlTargetAddressType - .1.3.6.1.2.1.80.1.2.1.3 2. pingCtlTargetAddress - .1.3.6.1.2.1.80.1.2.1.4 3. pingResultsOperStatus - .1.3.6.1.2.1.80.1.3.1.1 4. pingResultsIpTargetAddressType - .1.3.6.1.2.1.80.1.3.1.2 5. pingResultsIpTargetAddress - .1.3.6.1.2.1.80.1.3.1.3 6. jnxPingResultsMinIngressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.13 7. jnxPingResultsMaxIngressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.14 8. jnxPingResultsAvgIngressUs - .1.3.6.1.4.1.2636.3.7.1.3.1.15 9. pingResultsProbeResponses - .1.3.6.1.2.1.80.1.3.1.7 10. pingResultsSentProbes - .1.3.6.1.2.1.80.1.3.1.8 11. pingResultsRttSumOfSquares - .1.3.6.1.2.1.80.1.3.1.9 12. pingResultsLastGoodProbe - .1.3.6.1.2.1.80.1.3.1.10 13. jnxPingCtlIngressJitterThreshold - .1.3.6.1.4.1.2636.3.7.1.2.1.15
jnxAccessAuthServiceUp	1.3.6.1.4.1.2636.3.51.1.0.1	Routing	J Series and SRX Series	None
jnxAccessAuthServiceDown	1.3.6.1.4.1.2636.3.51.1.0.2	Routing	J Series and SRX Series	None
jnxAccessAuthServerDisabled	1.3.6.1.4.1.2636.3.51.1.0.3	Routing	J Series and SRX Series	jnxUserAAAServerName - .1.3.6.1.4.1.2636.3.51.1.1.3.1.0
jnxAccessAuthServerEnabled	1.3.6.1.4.1.2636.3.51.1.0.4	Routing	J Series and SRX Series	jnxUserAAAServerName - .1.3.6.1.4.1.2636.3.51.1.1.3.1.0
jnxJsFwAuthFailure	1.3.6.1.4.1.2636.3.39.1.2.1.0.1	Routing	J Series and SRX Series	1. jnxJsFwAuthUserName - .1.3.6.1.4.1.2636.3.39.1.2.1.1.2.1.0 2. jnxJsFwAuthClientIpAddr - .1.3.6.1.4.1.2636.3.39.1.2.1.1.2.4.0 3. jnxJsFwAuthServiceDesc - .1.3.6.1.4.1.2636.3.39.1.2.1.1.2.2.0 4. jnxJsFwAuthReason - .1.3.6.1.4.1.2636.3.39.1.2.1.1.2.3.0

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxJsFwAuthServiceUp	1.3.6.1.4.1.2636.3.39.1.2.1.0.2	Routing	J Series and SRX Series	None
jnxJsFwAuthServiceDown	1.3.6.1.4.1.2636.3.39.1.2.1.0.3	Routing	J Series and SRX Series	None
jnxJsFwAuthCapacityExceeded	1.3.6.1.4.1.2636.3.39.1.2.1.0.4	Routing	J Series and SRX Series	None
jnxJsScreenAttack	1.3.6.1.4.1.2636.3.39.1.8.1.0.1	Routing	J Series and SRX Series	1. jnxJsScreenZoneName - .1.3.6.1.4.1.2636.3.39.1.8.1.1.1.1.1 2. ifName - 1.3.6.1.2.1.31.1.1.1.1 3. jnxJsScreenAttackType - .1.3.6.1.4.1.2636.3.39.1.8.1.2.1.0 4. jnxJsScreenAttackCounter - .1.3.6.1.4.1.2636.3.39.1.8.1.2.2.0 5. jnxJsScreenAttackDescr - .1.3.6.1.4.1.2636.3.39.1.8.1.2.3.0
jnxJsScreenCfgChange	1.3.6.1.4.1.2636.3.39.1.8.1.0.2	Routing	J Series and SRX Series	1. jnxJsScreenZoneName - .1.3.6.1.4.1.2636.3.39.1.8.1.1.1.1.1 2. jnxJsScreenAttackType - .1.3.6.1.4.1.2636.3.39.1.8.1.2.1.0 3. jnxJsScreenCfgStatus - .1.3.6.1.4.1.2636.3.39.1.8.1.2.4.0
jnxRmonAlarmGetFailure	1.3.6.1.4.1.2636.4.3.0.1	RMON alarm	All Junos OS devices	1. alarmIndex - .1.3.6.1.2.1.16.3.1.1.1 2. alarmVariable - .1.3.6.1.2.1.16.3.1.1.3 3. jnxRmonAlarmGetFailReason - .1.3.6.1.4.1.2636.3.13.1.1.3
jnxRmonGetOk	1.3.6.1.4.1.2636.4.3.0.2	RMON alarm	All Junos OS devices	1. alarmIndex - .1.3.6.1.2.1.16.3.1.1.1 2. alarmVariable - .1.3.6.1.2.1.16.3.1.1.3

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxSyslogTrap	.1.3.6.1.4.1.2636.4.12.0.1	Services	All Junos OS devices	1. jnxSyslogEventName - .1.3.6.1.4.1.2636.3.35.1.1.1.2 2. jnxSyslogTimestamp - .1.3.6.1.4.1.2636.3.35.1.1.1.3 3. jnxSyslogSeverity -.1.3.6.1.4.1.2636.3.35.1.1.1.4 4. jnxSyslogFacility - .1.3.6.1.4.1.2636.3.35.1.1.1.5 5. jnxSyslogProcessId - .1.3.6.1.4.1.2636.3.35.1.1.1.6 6. jnxSyslogProcessName - .1.3.6.1.4.1.2636.3.35.1.1.1.7 7. jnxSyslogHostName - .1.3.6.1.4.1.2636.3.35.1.1.1.8 8. jnxSyslogMessage - .1.3.6.1.4.1.2636.3.35.1.1.1.9 9. Apart from the jnxSyslogTrap objects, this notification can include one or more attribute-value pairs. The attribute-value pairs are identified by objects jnxSyslogAvAttribute and jnxSyslogAvValue.
jnxEventTrap	.1.3.6.1.4.1.2636.4.13.0.1	Services	All Junos OS devices	1. jnxEventTrapDescr - .1.3.6.1.4.1.2636.3.37.1.1.0 2. Apart from the jnxEventTrap objects,this notification can include one or more attribute-value pairs. The attribute-value pairs shall be identified by objects jnxEventAvAttribute and jnxEventAvValue.
jnxJsAvPatternUpdateTrap	.1.3.6.1.4.1.2636.3.39.1.13.1.0.1	Configuration	J Series and SRX Series	1. jnxAVPatternVersionString - .1.3.6.1.4.1.2636.3.39.1.13.1.3.1.0 2. jnxAVPatternTimestamp - .1.3.6.1.4.1.2636.3.39.1.13.1.3.2.0

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxJsChassisClusterSwitchover	.1.3.6.1.4.1.2636.3.39.1.14.1.0.1	Chassis (alarm conditions)	SRX5600, SRX5800, and J Series	1. jnxJsChClusterSwitchoverInfoRedundancyGroup - .1.3.6.1.4.1.2636.3.39.1.14.1.1.1.0 2. jnxJsChClusterSwitchoverInfoClusterId - .1.3.6.1.4.1.2636.3.39.1.14.1.1.2.0 3. jnxJsChClusterSwitchoverInfoNodeId - .1.3.6.1.4.1.2636.3.39.1.14.1.1.3.0 4. jnxJsChClusterSwitchoverInfoPreviousState - .1.3.6.1.4.1.2636.3.39.1.14.1.1.4.0 5. jnxJsChClusterSwitchoverInfoCurrentState - .1.3.6.1.4.1.2636.3.39.1.14.1.1.5.0 6. jnxJsChClusterSwitchoverInfoReason - .1.3.6.1.4.1.2636.3.39.1.14.1.1.6.0
bfdSessUp	.1.3.6.1.4.1.2636.5.3.1.0.1	Routing	All Junos OS devices	1. bfdSessDiag, -- low range value - .1.3.6.1.4.1.2636.5.3.1.1.2.1.8 2. bfdSessDiag -- high range value - .1.3.6.1.4.1.2636.5.3.1.1.2.1.8
bfdSessDown	.1.3.6.1.4.1.2636.5.3.1.0.2	Routing	All Junos OS devices	1. bfdSessDiag, -- low range value - .1.3.6.1.4.1.2636.5.3.1.1.2.1.8 2. bfdSessDiag -- high range value - .1.3.6.1.4.1.2636.5.3.1.1.2.1.8
jnxBfdSessTxIntervalHigh	.1.3.6.1.4.1.2636.3.45.1.0.1	Routing	All Junos OS devices	1. jnxBfdSessThreshTxInterval - .1.3.6.1.4.1.2636.3.45.1.1.1.1.1 2. jnxBfdSessCurrTxInterval - .1.3.6.1.4.1.2636.3.45.1.1.1.1.2
jnxBfdSessDetectionTimeHigh	.1.3.6.1.4.1.2636.3.45.1.0.2	Routing	All Junos OS devices	1. jnxBfdSessThreshDectTime - .1.3.6.1.4.1.2636.3.45.1.1.1.1.3 2. jnxBfdSessCurrDectTime - .1.3.6.1.4.1.2636.3.45.1.1.1.1.4
jnxBgpM2Established	.1.3.6.1.4.1.2636.5.1.1.1.0.1	Routing	All Junos OS devices	1. jnxBgpM2PeerLocalAddrType - .1.3.6.1.4.1.2636.5.1.1.2.1.1.1.6 2. jnxBgpM2PeerLocalAddr - .1.3.6.1.4.1.2636.5.1.1.2.1.1.1.7 3. jnxBgpM2PeerRemoteAddrType - .1.3.6.1.4.1.2636.5.1.1.2.1.1.1.10 4. jnxBgpM2PeerRemoteAddr - .1.3.6.1.4.1.2636.5.1.1.2.1.1.1.11 5. jnxBgpM2PeerLastErrorReceived - .1.3.6.1.4.1.2636.5.1.1.2.2.1.1.1 6. jnxBgpM2PeerState - .1.3.6.1.4.1.2636.5.1.1.2.1.1.1.2

Table 9: Supported SNMP Traps *(continued)*

Trap Name	SNMPv2 Trap OID	Category	Platforms Supported	Varbinds
jnxBgpM2BackwardTransition	.1.3.6.1.4.1.2636.5.1.1.1.0.2	Routing	All Junos OS devices	1. jnxBgpM2PeerLocalAddrType - .1.3.6.1.4.1.2636.5.1.1.2.1.1.1.6 2. jnxBgpM2PeerLocalAddr - .1.3.6.1.4.1.2636.5.1.1.2.1.1.1.7 3. jnxBgpM2PeerRemoteAddrType - .1.3.6.1.4.1.2636.5.1.1.2.1.1.1.10 4. jnxBgpM2PeerRemoteAddr - .1.3.6.1.4.1.2636.5.1.1.2.1.1.1.11 5. jnxBgpM2PeerLastErrorReceived - .1.3.6.1.4.1.2636.5.1.1.2.2.1.1.1 6. jnxBgpM2PeerLastErrorReceivedText - .1.3.6.1.4.1.2636.5.1.1.2.2.1.1.5 7. jnxBgpM2PeerState - .1.3.6.1.4.1.2636.5.1.1.2.1.1.1.2

NOTE: If the fxp0 interface fails on the backup Routing Engine, it does not send any traps. The system logging (syslog) feature can be used to monitor the secondary node fxp0 interface by logging a link down message.

System Log Messages

The system logging feature can be sent by both the primary and secondary nodes. You can configure the system to send specific syslog messages to the external syslog servers based on regular expressions or severity. For more information, see the System Log Explorer.

The following sample shows the jnxSyslog trap configuration for a **ui_commit_progress** (configuration commit in progress) event.

jnxSyslog Trap Configuration

```
event-options {
  policy syslogtrap {
    events [ ui_commit ui_commit_progress ];
    then {
      raise-trap;
    }
  }
}
```

jnxSyslog Trap

```
Jul 6 13:31:21 snmpd[0] <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Jul 6 13:31:21 snmpd[0] <<< V2 Trap
Jul 6 13:31:21 snmpd[0] <<< Source: 116.197.179.6
Jul 6 13:31:21 snmpd[0] <<< Destination: 116.197.179.5
Jul 6 13:31:21 snmpd[0] <<< Version: SNMPv2
Jul 6 13:31:21 snmpd[0] <<< Community: petblr
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : sysUpTime.0
Jul 6 13:31:21 snmpd[0] <<< type : TimeTicks
Jul 6 13:31:21 snmpd[0] <<< value: (284292835) 789:42:08.35
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : snmpTrapOID.0
Jul 6 13:31:21 snmpd[0] <<< type : Object
Jul 6 13:31:21 snmpd[0] <<< value: jnxSyslogTrap
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : jnxSyslogEventName.83
Jul 6 13:31:21 snmpd[0] <<< type : OctetString
Jul 6 13:31:21 snmpd[0] <<< value: "UI_COMMIT_PROGRESS"
Jul 6 13:31:21 snmpd[0] <<< HEX : 55 49 5f 43 4f 4d 4d 49
Jul 6 13:31:21 snmpd[0] <<< 54 5f 50 52 4f 47 52 45
Jul 6 13:31:21 snmpd[0] <<< 53 53
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : jnxSyslogTimestamp.83
Jul 6 13:31:21 snmpd[0] <<< type : OctetString
Jul 6 13:31:21 snmpd[0] <<< HEX : 07 da 07 06 0d 1f 11 00
Jul 6 13:31:21 snmpd[0] <<< 2b 00 00
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : jnxSyslogSeverity.83
Jul 6 13:31:21 snmpd[0] <<< type : Number
Jul 6 13:31:21 snmpd[0] <<< value: 7
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : jnxSyslogFacility.83
Jul 6 13:31:21 snmpd[0] <<< type : Number
Jul 6 13:31:21 snmpd[0] <<< value: 24
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : jnxSyslogProcessId.83
Jul 6 13:31:21 snmpd[0] <<< type : Gauge
Jul 6 13:31:21 snmpd[0] <<< value: 84003
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : jnxSyslogProcessName.83
Jul 6 13:31:21 snmpd[0] <<< type : OctetString
Jul 6 13:31:21 snmpd[0] <<< value: "mgd"
Jul 6 13:31:21 snmpd[0] <<< HEX : 6d 67 64
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : jnxSyslogHostName.83
Jul 6 13:31:21 snmpd[0] <<< type : OctetString
Jul 6 13:31:21 snmpd[0] <<< HEX :
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : jnxSyslogMessage.83
Jul 6 13:31:21 snmpd[0] <<< type : OctetString
Jul 6 13:31:21 snmpd[0] <<< value: "UI_COMMIT_PROGRESS: Commit opera
Jul 6 13:31:21 snmpd[0] <<< tion in progress: notifying mib
Jul 6 13:31:21 snmpd[0] <<< 2d(15)"
Jul 6 13:31:21 snmpd[0] <<< HEX : 55 49 5f 43 4f 4d 4d 49
Jul 6 13:31:21 snmpd[0] <<< 54 5f 50 52 4f 47 52 45
Jul 6 13:31:21 snmpd[0] <<< 53 53 3a 20 43 6f 6d 6d
Jul 6 13:31:21 snmpd[0] <<< 69 74 20 6f 70 65 72 61
Jul 6 13:31:21 snmpd[0] <<< 74 69 6f 6e 20 69 6e 20
Jul 6 13:31:21 snmpd[0] <<< 70 72 6f 67 72 65 73 73
Jul 6 13:31:21 snmpd[0] <<< 3a 20 20 6e 6f 74 69 66
Jul 6 13:31:21 snmpd[0] <<< 79 69 6e 67 20 6d 69 62
Jul 6 13:31:21 snmpd[0] <<< 32 64 28 31 35 29
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : jnxSyslogAvAttribute.83.1
Jul 6 13:31:21 snmpd[0] <<< type : OctetString
Jul 6 13:31:21 snmpd[0] <<< value: "message"
Jul 6 13:31:21 snmpd[0] <<< HEX : 6d 65 73 73 61 67 65
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : jnxSyslogAvValue.83.1
Jul 6 13:31:21 snmpd[0] <<< type : OctetString
Jul 6 13:31:21 snmpd[0] <<< value: " notifying mib2d(15)"
Jul 6 13:31:21 snmpd[0] <<< HEX : 20 6e 6f 74 69 66 79 69
Jul 6 13:31:21 snmpd[0] <<< 6e 67 20 6d 69 62 32 64
Jul 6 13:31:21 snmpd[0] <<< 28 31 35 29
Jul 6 13:31:21 snmpd[0] <<<
Jul 6 13:31:21 snmpd[0] <<< OID : snmpTrapEnterprise.0
Jul 6 13:31:21 snmpd[0] <<< type : Object
Jul 6 13:31:21 snmpd[0] <<< value: jnxProductNameSRX240
Jul 6 13:31:21 snmpd[0] <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

A switchover can be detected using a failover trap, the chassis cluster status, or an automatic failover trap.

Failover Trap

The trap message can help you troubleshoot failovers. It contains the following information:

- Cluster ID and node ID
- Reason for the failover
- Redundancy group that is involved in the failover
- Redundancy group's previous state and current state

The cluster can be in any of the different states at any given instant: hold, primary, secondary-hold, secondary, ineligible, and disabled. Traps are generated for the following state transitions (only a transition from a hold state does not trigger a trap):

- primary <–> secondary
- primary –> secondary-hold
- secondary-hold –> secondary
- secondary –> ineligible
- ineligible –> disabled
- ineligible –> primary
- secondary –> disabled

A transition can be triggered due to events such as interface monitoring, SPU monitoring, failures, and manual failovers.

Event triggering is applicable for all redundancy groups including RG0, RG1, and so on. All redundancy group failover events trigger the same trap, and the actual group can be identified by examining the jnxJsChClusterSwitchoverInfoRedundancyGroup parameter in the trap varbind.

The trap is forwarded over the control link if the outgoing interface is on a node different from the node of the Routing Engine that generates the trap. The following are sample traps for manual and automatic failovers. Note that the traps are generated by the current primary devices before the failover occurs.

NOTE: A failover in any redundancy group (RG) other than redundancy group 0 does not make the other node the primary node.

In the following example, node 0 is the primary node in RG0, while it is the secondary node in RG1. Node 0 remains the primary node for the cluster. Only when the failover happens on node 1 in RG0 does node 1 become the primary node for the cluster. So even if a switchover happens on other groups, the primary node should be queried for all statistics and data as previously mentioned.

NOTE: Junos OS can be configured to send a desirable IP address as the source IP address of SNMP trap PDUs. Otherwise, SNMP traps always contain the outgoing interface IP address.

Chassis Cluster Status

```
user@host> show chassis cluster status

Cluster ID: 12
Node  Priority Status     Preempt  Manual failover Redundancy group: 0 , Failover
 count: 3
node  255    primary     no    yes
node1 1     secondary-hold  no    yes   Redundancy group: 1 , Failover count: 4
node0 100    secondary    no    yes
node1 255    primary     no    yes
```

Manual Failover Trap

```
Jul 6 05:14:57 snmpd[0] <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Jul 6 05:14:57 snmpd[0] <<< V2 Trap
Jul 6 05:14:57 snmpd[0] <<< Source: 192.168.29.2
Jul 6 05:14:57 snmpd[0] <<< Destination: 10.204.132.188
Jul 6 05:14:57 snmpd[0] <<< Version: SNMPv2
Jul 6 05:14:57 snmpd[0] <<< Community: test
Jul 6 05:14:57 snmpd[0] <<<
Jul 6 05:14:57 snmpd[0] <<< OID : sysUpTime.0
Jul 6 05:14:57 snmpd[0] <<< type : TimeTicks
Jul 6 05:14:57 snmpd[0] <<< value: (754507) 2:05:45.07
Jul 6 05:14:57 snmpd[0] <<<
Jul 6 05:14:57 snmpd[0] <<< OID : snmpTrapOID.0
Jul 6 05:14:57 snmpd[0] <<< type : Object
Jul 6 05:14:57 snmpd[0] <<< value: jnxJsChassisClusterSwitchover
Jul 6 05:14:57 snmpd[0] <<<
Jul 6 05:14:57 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoRedundancyGroup.0
Jul 6 05:14:57 snmpd[0] <<< type : OctetString
Jul 6 05:14:57 snmpd[0] <<< value: "1"
Jul 6 05:14:57 snmpd[0] <<< HEX : 31
Jul 6 05:14:57 snmpd[0] <<<
Jul 6 05:14:57 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoClusterId.0
Jul 6 05:14:57 snmpd[0] <<< type : OctetString
Jul 6 05:14:57 snmpd[0] <<< value: "12"
Jul 6 05:14:57 snmpd[0] <<< HEX : 31 32
Jul 6 05:14:57 snmpd[0] <<<
Jul 6 05:14:57 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoNodeId.0
Jul 6 05:14:57 snmpd[0] <<< type : OctetString
Jul 6 05:14:57 snmpd[0] <<< value: "0"
Jul 6 05:14:57 snmpd[0] <<< HEX : 30
Jul 6 05:14:57 snmpd[0] <<<
Jul 6 05:14:57 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoPreviousState.0
Jul 6 05:14:57 snmpd[0] <<< type : OctetString
Jul 6 05:14:57 snmpd[0] <<< value: "primary"
Jul 6 05:14:57 snmpd[0] <<< HEX : 70 72 69 6d 61 72 79
Jul 6 05:14:57 snmpd[0] <<<
Jul 6 05:14:57 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoCurrentState.0
Jul 6 05:14:57 snmpd[0] <<< type : OctetString
Jul 6 05:14:57 snmpd[0] <<< value: "secondary-hold"
Jul 6 05:14:57 snmpd[0] <<< HEX : 73 65 63 6f 6e 64 61 72
Jul 6 05:14:57 snmpd[0] <<< 79 2d 68 6f 6c 64
Jul 6 05:14:57 snmpd[0] <<<
Jul 6 05:14:57 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoReason.0
Jul 6 05:14:57 snmpd[0] <<< type : OctetString
Jul 6 05:14:57 snmpd[0] <<< value: "manual failover"
Jul 6 05:14:57 snmpd[0] <<< HEX : 6d 61 6e 75 61 6c 20 66
Jul 6 05:14:57 snmpd[0] <<< 61 69 6c 6f 76 65 72
Jul 6 05:14:57 snmpd[0] <<<
Jul 6 05:14:57 snmpd[0] <<< OID : snmpTrapEnterprise.0
Jul 6 05:14:57 snmpd[0] <<< type : Object
Jul 6 05:14:57 snmpd[0] <<< value: jnxProductNameSRX5600
Jul 6 05:14:57 snmpd[0] <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Automatic Failover Trap

```
Jul 6 03:12:20 snmpd[0] <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Jul 6 03:12:20 snmpd[0] <<< V1 Trap
Jul 6 03:12:20 snmpd[0] <<< Source: 192.168.29.2
Jul 6 03:12:20 snmpd[0] <<< Destination: 10.204.132.188
Jul 6 03:12:20 snmpd[0] <<< Version: SNMPv1
Jul 6 03:12:20 snmpd[0] <<< Community: test
Jul 6 03:12:20 snmpd[0] <<< Agent addr: 10.255.131.37
Jul 6 03:12:20 snmpd[0] <<< sysUpTime: (18763) 0:03:07.63
Jul 6 03:12:20 snmpd[0] <<< Enterprise: jnxJsChassisClusterMIB
Jul 6 03:12:20 snmpd[0] <<< Generic trap: 6, Specific trap: 1
Jul 6 03:12:20 snmpd[0] <<<
Jul 6 03:12:20 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoRedundancyGroup.0
Jul 6 03:12:20 snmpd[0] <<< type : OctetString
Jul 6 03:12:20 snmpd[0] <<< value: "1"
Jul 6 03:12:20 snmpd[0] <<< HEX : 31
Jul 6 03:12:20 snmpd[0] <<<
Jul 6 03:12:20 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoClusterId.0
Jul 6 03:12:20 snmpd[0] <<< type : OctetString
Jul 6 03:12:20 snmpd[0] <<< value: "12"
Jul 6 03:12:20 snmpd[0] <<< HEX : 31 32
Jul 6 03:12:20 snmpd[0] <<<
Jul 6 03:12:20 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoNodeId.0
Jul 6 03:12:20 snmpd[0] <<< type : OctetString
Jul 6 03:12:20 snmpd[0] <<< value: "0"
Jul 6 03:12:20 snmpd[0] <<< HEX : 30
Jul 6 03:12:20 snmpd[0] <<<
Jul 6 03:12:20 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoPreviousState.0
Jul 6 03:12:20 snmpd[0] <<< type : OctetString
Jul 6 03:12:20 snmpd[0] <<< value: "secondary-hold"
Jul 6 03:12:20 snmpd[0] <<< HEX : 73 65 63 6f 6e 64 61 72
Jul 6 03:12:20 snmpd[0] <<< 79 2d 68 6f 6c 64
Jul 6 03:12:20 snmpd[0] <<<
Jul 6 03:12:20 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoCurrentState.0
Jul 6 03:12:20 snmpd[0] <<< type : OctetString
Jul 6 03:12:20 snmpd[0] <<< value: "secondary"
Jul 6 03:12:20 snmpd[0] <<< HEX : 73 65 63 6f 6e 64 61 72
Jul 6 03:12:20 snmpd[0] <<< 79
Jul 6 03:12:20 snmpd[0] <<<
Jul 6 03:12:20 snmpd[0] <<< OID : jnxJsChClusterSwitchoverInfoReason.0
Jul 6 03:12:20 snmpd[0] <<< type : OctetString
Jul 6 03:12:20 snmpd[0] <<< value: "back to back failover interval expired"
Jul 6 03:12:20 snmpd[0] <<< HEX : 62 61 63 6b 20 74 6f 20
Jul 6 03:12:20 snmpd[0] <<< 62 61 63 6b 20 66 61 69
Jul 6 03:12:20 snmpd[0] <<< 6c 6f 76 65 72 20 69 6e
Jul 6 03:12:20 snmpd[0] <<< 74 65 72 76 61 6c 20 65
Jul 6 03:12:20 snmpd[0] <<< 78 70 69 72 65 64
Jul 6 03:12:20 snmpd[0] <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Other Indications for Failover

When a failover occurs in the RG0 redundancy group:

- An SNMP warm start trap is sent by the new primary node.
- After a failover, LinkUp traps are sent for all the interfaces that come up on the new primary node.
- Syslog messages are sent from the new primary node.

Managing and Monitoring a Chassis Cluster Using Operational and Event Scripts

Junos OS operation (op) scripts automate network and router management and troubleshooting. Op scripts can perform any function available through the remote procedure calls (RPCs) supported by either of the two application programming interfaces (APIs): the Junos OS Extensible Markup Language (XML) API and the Junos OS XML Management Protocol API. Scripts are written in the Extensible Stylesheet Language Transformations (XSLT) or Stylesheet Language Alternative Syntax (SLAX) scripting languages.

Op scripts allow you to:

- Monitor the overall status of a routing platform.
- Customize the output of operational mode commands.
- Reconfigure the routing platform to avoid or work around known problems in the Junos OS software.
- Change the router’s configuration in response to a problem.

Junos OS event scripts automate network and router management and troubleshooting. These are operational scripts triggered by event policies.

An example of a jnx event trap follows. In the example, the **ev-syslog-trap** event script raises a jnxEvent trap whenever an alarm is triggered on the device.

jnx event trap

```
{
events SYSTEM;
  attributes-match {
  SYSTEM.message matches "Alarm set";
```

```
    }
    then {
      event-script ev-syslog-trap.slax {
        arguments {
          event SYSTEM;
          message "{$$.message}";
        }
      }
    }
  }
```

The following trap is sent to bring a link down on the device to set an alarm.

```
Apr 16 05:09:43 snmpd[0] <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Apr 16 05:09:43 snmpd[0] <<< V2 Trap
Apr 16 05:09:43 snmpd[0] <<<  Source: 116.197.178.12
Apr 16 05:09:43 snmpd[0] <<<  Destination: 66.129.237.197
Apr 16 05:09:43 snmpd[0] <<<  Version: SNMPv2
Apr 16 05:09:43 snmpd[0] <<<  Community: test
Apr 16 05:09:43 snmpd[0] <<<
Apr 16 05:09:43 snmpd[0] <<<   OID  : sysUpTime.0
Apr 16 05:09:43 snmpd[0] <<<   type : TimeTicks
Apr 16 05:09:43 snmpd[0] <<<   value: (246317536) 684:12:55.36
Apr 16 05:09:43 snmpd[0] <<<
Apr 16 05:09:43 snmpd[0] <<<   OID  : snmpTrapOID.0
Apr 16 05:09:43 snmpd[0] <<<   type : Object
Apr 16 05:09:43 snmpd[0] <<<   value: jnxEventTrap
Apr 16 05:09:43 snmpd[0] <<<
Apr 16 05:09:43 snmpd[0] <<<   OID  : jnxEventTrapDescr.0
Apr 16 05:09:43 snmpd[0] <<<   type : OctetString
Apr 16 05:09:43 snmpd[0] <<<   value: "'Event-Trap'"
Apr 16 05:09:43 snmpd[0] <<<   HEX  :   27 45 76 65 6e 74 2d 54
Apr 16 05:09:43 snmpd[0] <<<       72 61 70 27
Apr 16 05:09:43 snmpd[0] <<<
Apr 16 05:09:43 snmpd[0] <<<   OID  : jnxEventAvAttribute.1
Apr 16 05:09:43 snmpd[0] <<<   type : OctetString
Apr 16 05:09:43 snmpd[0] <<<   value: "'event'"
Apr 16 05:09:43 snmpd[0] <<<   HEX  : 27 65 76 65 6e 74 27
Apr 16 05:09:43 snmpd[0] <<<
Apr 16 05:09:43 snmpd[0] <<<   OID  : jnxEventAvValue.1
Apr 16 05:09:43 snmpd[0] <<<   type : OctetString
Apr 16 05:09:43 snmpd[0] <<<   value: "'SYSTEM'"
Apr 16 05:09:43 snmpd[0] <<<   HEX  : 27 53 59 53 54 45 4d 27
Apr 16 05:09:43 snmpd[0] <<<
Apr 16 05:09:43 snmpd[0] <<<   OID  : jnxEventAvAttribute.2
Apr 16 05:09:43 snmpd[0] <<<   type : OctetString
Apr 16 05:09:43 snmpd[0] <<<   value: "'message'"
Apr 16 05:09:43 snmpd[0] <<<   HEX  : 27 6d 65 73 73 61 67 65
Apr 16 05:09:43 snmpd[0] <<<      27
Apr 16 05:09:43 snmpd[0] <<<
Apr 16 05:09:43 snmpd[0] <<<   OID  : jnxEventAvValue.1
Apr 16 05:09:43 snmpd[0] <<<   type : OctetString
Apr 16 05:09:43 snmpd[0] <<<   value: "' Minor alarm set, ge-1/0/0: Link down'"
Apr 16 05:09:43 snmpd[0] <<<   HEX  : 27 20 4d 69 6e 6f 72 20
Apr 16 05:09:43 snmpd[0] <<<      61 6c 61 72 6d 20 73 65
Apr 16 05:09:43 snmpd[0] <<<      74 2c 20 67 65 2d 31 2f
Apr 16 05:09:43 snmpd[0] <<<      30 2f 30 3a 20 4c 69 6e
Apr 16 05:09:43 snmpd[0] <<<      6b 20 64 6f 77 6e 27
Apr 16 05:09:43 snmpd[0] <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Using the Utility MIB for Monitoring a Chassis Cluster

The Juniper Networks utility MIB (jnxUtil) is a powerful tool to expose Junos OS data using SNMP. A generic utility MIB is defined to hold data populated by op scripts or event scripts. There are five separate tables in this MIB, one for each of the following data types: 32-bit counters, 64-bit counters, signed integers, unsigned integers, and octet strings. Each data instance is identified by an arbitrary ASCII name defined when the data is populated. Each data instance also has a corresponding timestamp identifying when it was last updated.

The data in these MIB tables can be populated using hidden CLI commands, which are also accessible from an op script using the jcs:invoke remote procedure call (RPC) API.

One of the examples we use for reading power on the device, which is not available using SNMP, is the jnxUtil MIB. With a simple event script, you can read the power output every minute and populate the jnxUtil MIB. Similarly, you can write op scripts or event scripts that can populate a variety of data of different types. For more information about utility MIB examples for sample scripts and usage of the utility MIB, see *Utility MIB Examples*.

Related Documentation

- *Utility MIB Examples*
- *Monitoring Chassis Cluster Performance*
- *Log Messages for SRX Series Chassis Clusters*

Log Messages for SRX Series Chassis Clusters

Sessions and Packet Flows Overview

You can obtain information about the sessions and packet flows active on your device, including detailed information about specific sessions. (The SRX Series device also displays information about failed sessions.) You can display this information to observe activity and for debugging purposes.

For example, use the **show security flow session** command to:

- Display a list of incoming and outgoing IP flows, including services.

- Show the security attributes associated with a flow, for example, the policies that apply to traffic belonging to that flow.
- Display the session timeout value, when the session became active, how long it has been active, and if there is active traffic on the session.

For detailed information about this command, see the *Junos OS CLI Reference*.

Session information can also be logged if a related policy configuration includes the logging option. Session logging infrastructure logs the session log messages when a session is created, closed, denied, or rejected. In the SRX3000 and SRX5000 lines, the log messages are streamed directly to an external syslog server/repository, bypassing the Routing Engine. The SRX Series devices support both traditional and structured syslog. The SRX3000 and SRX5000 lines support 1000 log messages per second, and the management station must be equipped to handle this volume. See the *Junos OS Security Configuration Guide* for configuration examples and details about these logs. The logs are available through the management interface of both the primary and secondary nodes. Ensure that the external server receiving these log messages is reachable by both nodes.

The high-end SRX Series devices have a distributed processing architecture that processes traffic as well as generates log messages. In the SRX Series devices, the firewall processes the traffic sessions on each of the SPUs in the chassis. After each session is created, it is processed by the same SPU in the chassis, which is also the SPU that generates the log message.

The standard method of generating log messages is to have each SPU generate the message as a UDP syslog message and send it directly out the data plane to the syslog server. The SRX Series devices can log extremely high rates of traffic. They can log up to 750 MB per second of log messages, which surpasses the limits of the control plane. Therefore, we do not recommended logging messages to the control plane, except under certain circumstances.

For SRX Series branch devices running Junos OS Release 9.6 and later and high-end SRX Series devices running Junos OS Release 10.0 and later, the devices can log messages to the control plane at a limited maximum rate (1000 log messages per second) rather than logging to the data plane. If the log messages are sent through the data plane using syslog, a syslog collector—such as the Juniper Security Threat Response Manager (STRM)—must be used to collect the logs for viewing, reporting, and alerting. In SRX Series branch devices running Junos OS Release 9.6 and later and high-end SRX Series devices running Junos OS Release 10.0 and later, the devices can only send log messages to the data plane or the control plane, but not to both at the same time.

Configuring High-End SRX Series Device Logging

1. Configure the logging format.

   There are two supported formats for system log messages: structured and standard. Structured syslog is generally preferred because it prepends the fields with a title. For instance, the source-IP address field is source-address="10.102.110.52" rather than just the IP address 10.102.110.52. In the following command, the **format sd-syslog** option configures structured syslog, whereas the **format syslog** option configures standard syslog.

```
user@host# set security log format sd-syslog
```

2. Configure the syslog source address.

   The syslog source address can be any arbitrary IP address. It does not have to be an IP address that is assigned to the device. Rather, this IP address is used on the syslog collector to identify the syslog source. The best practice is to configure the source address as the IP address of the interface that the traffic is sent out on.

   ```
   user@host# set security log source-address ip-address
   ```

3. Configure the system log stream.

   The system log stream identifies the destination IP address that the syslog messages are sent to. On the high-end SRX Series devices running Junos OS Release 9.5 and later, up to two syslog streams can be defined (all messages are sent to the syslog streams). Note that you must give a name to the stream. This name is arbitrary, but it is a best practice to use the name of the syslog collector for easy identification in the configuration.

   You can also define the UDP port to which the log messages are sent. By default, log messages are sent to UDP port 1514.

   To configure the system log server IP address:

   ```
   user@host# set security log stream name host ip-address
   ```

   To configure the system log server IP address and specify the UDP port number:

   ```
   user@host# set security log stream name host ip-address port port
   ```

Configuring High-End SRX Series Device Data Plane Logging to the Control Plane

If the management station cannot receive log messages from the data plane, then configure it to send messages through the management connection. If you log to the control plane, the SRX Series devices can also send these syslog messages out the fxp0 interface. If event logging is configured, all log messages from the data plane go to the control plane.

1. Configure event logging.

   ```
   user@host# set security log mode event
   ```

2. Rate-limit the event log messages.

   It might be necessary to rate-limit the event log messages from the data plane to the control plane due to limited resources on the control plane to process high volumes of log messages. This is especially applicable if the control plane is busy processing dynamic routing protocols such as BGP or large-scale routing implementations. The following command rate-limits the log messages so that they do not overwhelm the control plane. Log messages that are rate-limited are discarded. A best practice for high-end SRX Series devices is to log no more than 1000 log messages per second to the control plane.

user@host# **set security log mode event event-rate *logs per second***

Configuring SRX Series Branch Devices to Send Traffic Log Messages Through the Data Plane

The SRX Series branch device traffic log messages can be sent through the data plane security logs in stream mode. Note that this is possible only using stream mode. The following is a sample configuration and log output.

Configuration

set security log mode stream
set security log format sd-syslog
set security log source-address 10.204.225.164
set security log stream vmware-server severity debug
set security log stream vmware-server host 10.204.225.218

Sample Log Message Output

```
Sep 06 16:54:22 10.204.225.164 1 2010-09-06T04:24:22.094 nsm-vidar-a RT_FLOW -
RT_FLOW_SESSION_CLOSE [junos@2636.1.1.1.2.39 reason="TCP FIN"
source-address="1.1.1.2" source-port="62736" destination-address="2.1.1.1"
destination-port="23" service-name="junos-telnet" nat-source-address="1.1.1.2"
nat-source-port="62736" nat-destination-address="2.1.1.1" nat-destination-port="23"
 src-nat-rule-name="None" dst-nat-rule-name="None" protocol-id="6"
policy-name="trust-untrust" source-zone-name="trust"
destination-zone-name="untrust" session-id-32="206" packets-from-client="64"
bytes-from-client="3525" packets-from-server="55" bytes-from-server="3146"
elapsed-time="21"]
```

```
Sep 06 16:54:26 10.204.225.164 1 2010-09-06T04:24:26.095 nsm-vidar-a RT_FLOW -
RT_FLOW_SESSION_CREATE [junos@2636.1.1.1.2.39 source-address="1.1.1.2"
source-port="49780" destination-address="2.1.1.1" destination-port="23"
service-name="junos-telnet" nat-source-address="1.1.1.2" nat-source-port="49780"
 nat-destination-address="2.1.1.1" nat-destination-port="23"
src-nat-rule-name="None" dst-nat-rule-name="None" protocol-id="6"
policy-name="trust-untrust" source-zone-name="trust"
destination-zone-name="untrust" session-id-32="208"]
```

```
Sep 06 16:54:34 10.204.225.164 1 2010-09-06T04:24:34.098 nsm-vidar-a RT_FLOW -
RT_FLOW_SESSION_CLOSE [junos@2636.1.1.1.2.39 reason="TCP FIN"
source-address="1.1.1.2" source-port="49780" destination-address="2.1.1.1"
destination-port="23" service-name="junos-telnet" nat-source-address="1.1.1.2"
nat-source-port="49780" nat-destination-address="2.1.1.1" nat-destination-port="23"
 src-nat-rule-name="None" dst-nat-rule-name="None" protocol-id="6"
policy-name="trust-untrust" source-zone-name="trust"
destination-zone-name="untrust" session-id-32="208" packets-from-client="37"
bytes-from-client="2094" packets-from-server="30" bytes-from-server="1822"
elapsed-time="6"]
```

In this case, the SRX Series device traffic log messages are sent to an external syslog server through the dataplane. This ensures that the Routing Engine is not a bottleneck for logging. It also ensures that the Routing Engine does not get impacted during excessive logging. In addition to traffic log messages, the control plane and the log messages sent to the Routing Engine are written to a file in flash memory. The following is a sample configuration to enable this type of logging.

Configuration

Syslog (self logs)—This configuration can be customized as per required **self** logging.

set system syslog file messages any notice

```
set system syslog file messages authorization info
set system syslog file messages kernel info
```

Traffic logs (using dataplane)

```
set security log mode stream
set security log format sd-syslog
set security log source-address 10.204.225.164
set security log stream vmware-server severity debug
set security log stream vmware-server host 10.204.225.218
```

In this case both the traffic log messages and log messages sent to the Routing Engine are sent to a syslog server. The following is a sample configuration to enable this type of logging.

Configuration

Syslog (syslog server)

```
set system syslog host 10.204.225.218 any notice
set system syslog host 10.204.225.218 authorization info
set system syslog host 10.204.225.218 kernel info
```

Traffic logs

```
set security log mode stream
set security log format sd-syslog
set security log source-address 10.204.225.164
set security log stream vmware-server severity debug
set security log stream vmware-server host 10.204.225.218
```

Configuring Control Plane Logs

The SRX Series device control plane is responsible for overall control of the SRX Series platform, along with running a number of software processes to perform tasks such as routing protocol operations, routing table calculations, managing administrators, managing SNMP, authentication, and many other mission-critical functions. There are a wide range of log messages that are generated on the control plane, and the control plane offers granular support for defining what log messages should be written to both files as well as sent to syslog servers. This topic provides an overview of how to configure various syslog options on the control plane. Only sending log messages through syslog services is covered in this section.

1. Configure the syslog server and selected log messages.

   To configure the syslog server to receive log messages from the SRX Series device, define which syslog hosts receive the streams along with which facilities and severities to send. Note that multiple facilities and priorities can be configured to send multiple log message types. To send all message types, specify the **any** option for the facility and severity.

   user@host# **set system syslog host *syslog server facility severity***

2. Configure the syslog source IP address.

   The source IP address of the syslog stream is needed because the SRX Series device can send the syslog message with any address. The same IP address should be used regardless of which interface is selected.

user@host# **set system syslog host *syslog server* source-address *source-address***

3. (Optional) Configure regular expression matching.

   Sometimes an administrator might want to filter the log messages that are sent to the syslog server. Log filtering can be specified with the **match** statement. In this example, only logs defined in the **match** statement regular expression (IDP) are sent to the syslog server.

   user@host# **set system syslog host *syslog server facility severity* match IDP**

Configuring SRX Series Branch Devices for Logging

You can configure the SRX Series device to send only traffic logs to the syslog server using the control plane.

In this configuration:

- No security logs are configured.
- No control plane logs are received.

Use the **match** statement regular expression to send traffic log messages only. These log messages are sent directly to the syslog server without writing them to flash memory. This configuration does not send log messages normally sent to the Routing Engine to the syslog server. However, it is possible to create a separate file and write control plane log messages to a file on the Routing Engine as shown.

Configuration

```
set system syslog host 10.204.225.218 any any
set system syslog host 10.204.225.218 match RT_FLOW_SESSION
set system syslog file messages any any
```

Sample log messages:

```
Sep 06 15:22:29 10.204.225.164 Sep 6 02:52:30 RT_FLOW: RT_FLOW_SESSION_CREATE:
session created 1.1.1.2/54164->2.1.1.1/23 junos-telnet 1.1.1.2/54164->2.1.1.1/23
 None None 6 trust-untrust trust untrust 192
Sep 06 15:22:43 10.204.225.220 Sep 6 02:52:30 last message repeated 10 times
Sep 06 15:23:49 10.204.225.164 Sep 6 02:53:49 RT_FLOW: RT_FLOW_SESSION_CLOSE:
session closed TCP FIN: 1.1.1.2/54164->2.1.1.1/23 junos-telnet
1.1.1.2/54164->2.1.1.1/23 None None 6 trust-untrust trust untrust 192 60(3307)
46(2784) 79
```

The following configuration sends both traffic and control log messages to the syslog server, but might overwhelm the syslog server and cause cluster instability. We do not recommend using this configuration.

Configuration

```
set system syslog host 10.204.225.218 any any
set system syslog file messages any any
```

Security log event mode is the default mode on SRX Series branch devices, and it is not advisable for these devices. We recommend changing the default behavior.

NOTE: Extensive logging on local flash can have an undesired impact on the device such as instability on the control plane.

Sending Data Plane Log Messages with an IP Address in the Same Subnet as the fxp0 Interface

You might want to deploy fault management and performance management applications and systems such as Juniper Networks Security Threat Response Manager (STRM). STRM collects log messages through the management network and is connected through the fxp0 interface. The fault management and performance management applications manage the SRX Series device through the fxp0 interface, but the SRX Series device also needs to send the data plane log messages to STRM on the same network. For instance, if the rate of log messages is going to be greater than 1000 log messages per second, then logging to the control plane is not supported. The issue is that two interfaces in the same virtual router cannot be in the same subnet, and the fxp0 interface cannot be moved to any virtual router other than inet.0.

To work around these issues, place a data plane interface in a virtual router other than the default virtual router inet.0, and place a route in the inet.0 routing table to route traffic to STRM through that virtual router. The following configuration example shows how to do this.

In this example:

- fxp0 has an IP address of 172.19.200.164/24.
- Application A (AppA) has an IP address of 172.19.200.175.
- STRM has an IP address of 172.19.200.176.
- The ge-0/0/7 interface is a data plane interface, with an IP address of 172.19.200.177/24 (which is in the same subnet as the fxp0 interface).

To configure this example, include the following statements:

```
set interfaces fxp0 unit 0 family inet address 172.19.200.164/24
set system syslog host 172.19.200.176 any any
set system syslog host 172.19.200.176 source-address 172.19.200.177
set interface ge-0/0/7 unit 0 family inet address 172.19.200.177/24
set security log format sd-syslog
set security log source-address 172.19.200.177
set security log stream Log host 172.19.200.176
set routing-instances Logging instance-type virtual-router
set routing-instances Logging interface ge-0/0/7.0
set routing-options static route 172.19.200.176/32 next-table Logging.inet.0
```

NOTE: AppA is now able to manage the ge-0/0/7 interface since AppA manages the device using the fxp0 interface in the default routing instance. To do this, AppA must use the *Logging@<snmp-community-string-name>* message format to access the ge-0/0/7 interface data using SNMP.

Related Documentation

- Monitoring Chassis Cluster Faults on page 51
- Tracking Applications on an SRX Series Chassis Cluster on page 92

Tracking Applications on an SRX Series Chassis Cluster

AppTrack, an application tracking tool, provides statistics for analyzing bandwidth usage of your network. When enabled, AppTrack collects byte, packet, and duration statistics for application flows in the specified zone. By default, when each session closes, AppTrack generates a message that provides the byte and packet counts and the duration of the session, and sends it to the host device. An AppTrack message is similar to session log messages and uses syslog or structured syslog formats. The message also includes an application field for the session. If AppTrack identifies a custom-defined application and returns an appropriate name, the custom application name is included in the log message.

Management stations can subscribe to receive log messages for application tracking. An SRX Series device can support a high volume of these log messages (minimum 1000 log messages per second). Management stations should be able to handle this volume. The log messages are available through the management interface of both the primary and secondary nodes. Additional care should be taken that the external server receiving these log messages is reachable by both nodes. See the *Junos OS Security Configuration Guide* for information about configuring application identification and tracking.

Related Documentation

- Log Messages for SRX Series Chassis Clusters on page 85
- Managing SRX Series Chassis Clusters Using RPCs on page 92

Managing SRX Series Chassis Clusters Using RPCs

This topic provides details related to managing SRX Series chassis clusters using remote procedure calls (RPCs).

Chassis Inventory Sample RPC Response

```
<rpc><get-chassis-inventory/></rpc>
<rpc><get-chassis-inventory/></rpc>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:junos="http://xml.juniper.net/junos/10.4I0/junos">
<multi-routing-engine-results>
<multi-routing-engine-item>
<re-name>node0</re-name>
<chassis-inventory xmlns="http://xml.juniper.net/junos/10.4I0/junos-chassis">
<chassis junos:style="inventory">
<name>Chassis</name>
<serial-number>JN1146479AGB</serial-number>
<description>SRX 5600</description>
<chassis-module>
<name>Midplane</name>
<version>REV 01</version>
<part-number>710-024804</part-number>
<serial-number>ABAA1033</serial-number>
<description>SRX 5600 Midplane</description>
<model-number>SRX5600-MP-A</model-number>
```

```
</chassis-module>
<chassis-module>
<name>FPM Board</name>
<version>REV 01</version>
<part-number>710-024631</part-number>
<serial-number>XD8881</serial-number>
<description>Front Panel Display</description>
<model-number>SRX5600-CRAFT-A</model-number>
</chassis-module>
<chassis-module>
<name>PEM 0</name>
<version>Rev 03</version>
<part-number>740-023485</part-number>
<serial-number>QCS0852H02U</serial-number>
<description>PS 1.2-1.7kW; 100-240V AC in</description>
<model-number>SRX5600-PWR-AC-A</model-number>
</chassis-module>
<chassis-module>
<name>PEM 1</name>
<version>Rev 03</version>
<part-number>740-023485</part-number>
<serial-number>QCS0852H01P</serial-number>
<description>PS 1.2-1.7kW; 100-240V AC in</description>
<model-number>SRX5600-PWR-AC-A</model-number>
</chassis-module>
<chassis-module>
<name>Routing Engine 0</name>
<version>REV 03</version>
<part-number>740-023530</part-number>
<serial-number>9009008946</serial-number>
<description>RE-S-1300</description>
<model-number>SRX5K-RE-13-20-A</model-number>
</chassis-module>
<chassis-module>
<name>CB 0</name>
<version>REV 03</version>
<part-number>710-024802</part-number>
<serial-number>WX5291</serial-number>
<description>SRX5k SCB</description>
<model-number>SRX5K-SCB-A</model-number>
</chassis-module>
<chassis-module>
<name>FPC 4</name>
<version>REV 01</version>
<part-number>750-023996</part-number>
<serial-number>WW0754</serial-number>
<description>SRX5k SPC</description>
<model-number>SRX5K-SPC-2-10-40</model-number>
<chassis-sub-module>
<name>CPU</name>
<version>REV 02</version>
<part-number>710-024633</part-number>
<serial-number>WY0854</serial-number>
<description>SRX5k DPC PMB</description>
</chassis-sub-module>
<chassis-sub-module>
<name>PIC 0</name>
<part-number>BUILTIN</part-number>
<serial-number>BUILTIN</serial-number>
<description>SPU Cp-Flow</description>
</chassis-sub-module>
```

```
<chassis-sub-module>
<name>PIC 1</name>
<part-number>BUILTIN</part-number>
<serial-number>BUILTIN</serial-number>
<description>SPU Flow</description>
</chassis-sub-module>
</chassis-module>
<chassis-module>
<name>FPC 5</name>
<version>REV 07</version>
<part-number>750-027945</part-number>
<serial-number>XH9092</serial-number>
<description>SRX5k FIOC</description>
<chassis-sub-module>
<name>CPU</name>
<version>REV 03</version>
<part-number>710-024633</part-number>
<serial-number>XH8755</serial-number>
<description>SRX5k DPC PMB</description>
</chassis-sub-module>
<chassis-sub-module>
<name>PIC 0</name>
<version>REV 05</version>
<part-number>750-021378</part-number>
<serial-number>XH3698</serial-number>
<description>4x 10GE XFP</description>
<model-number>SRX-IOC-4XGE-XFP</model-number>
<chassis-sub-sub-module>
<name>Xcvr 0</name>
<version>REV 02</version>
<part-number>740-011571</part-number>
<serial-number>C850XJ02G</serial-number>
<description>XFP-10G-SR</description>
</chassis-sub-sub-module>
<chassis-sub-sub-module>
<name>Xcvr 1</name>
<version>REV 02</version>
<part-number>740-011571</part-number>
<serial-number>C850XJ01T</serial-number>
<description>XFP-10G-SR</description>
</chassis-sub-sub-module>
</chassis-sub-module>
<chassis-sub-module>
<name>PIC 1</name>
<version>REV 03</version>
<part-number>750-027491</part-number>
<serial-number>XH8525</serial-number>
<description>16x 1GE TX</description>
</chassis-sub-module>
</chassis-module>
<chassis-module>
<name>Fan Tray</name>
<description>Left Fan Tray</description>
<model-number>SRX5600-FAN</model-number>
</chassis-module>
</chassis>
</chassis-inventory>
</multi-routing-engine-item>
<multi-routing-engine-item>
<re-name>node1</re-name>
<chassis-inventory xmlns="http://xml.juniper.net/junos/10.4I0/junos-chassis">
```

```
<chassis junos:style="inventory">
<name>Chassis</name>
<serial-number>JN11471C4AGB</serial-number>
<description>SRX 5600</description>
<chassis-module>
<name>Midplane</name>
<version>REV 01</version>
<part-number>710-024804</part-number>
<serial-number>ABAA4537</serial-number>
<description>SRX 5600 Midplane</description>
<model-number>SRX5600-MP-A</model-number>
</chassis-module>
<chassis-module>
<name>FPM Board</name>
<version>REV 01</version>
<part-number>710-024631</part-number>
<serial-number>XD8876</serial-number>
<description>Front Panel Display</description>
<model-number>SRX5600-CRAFT-A</model-number>
</chassis-module>
<chassis-module>
<name>PEM 0</name>
<version>Rev 03</version>
<part-number>740-023485</part-number>
<serial-number>QCS0901H015</serial-number>
<description>PS 1.2-1.7kW; 100-240V AC in</description>
<model-number>SRX5600-PWR-AC-A</model-number>
</chassis-module>
<chassis-module>
<name>PEM 1</name>
<version>Rev 03</version>
<part-number>740-023485</part-number>
<serial-number>QCS0901H011</serial-number>
<description>PS 1.2-1.7kW; 100-240V AC in</description>
<model-number>SRX5600-PWR-AC-A</model-number>
</chassis-module>
<chassis-module>
<name>Routing Engine 0</name>
<version>REV 03</version>
<part-number>740-023530</part-number>
<serial-number>9009020065</serial-number>
<description>RE-S-1300</description>
<model-number>SRX5K-RE-13-20-A</model-number>
</chassis-module>
<chassis-module>
<name>CB 0</name>
<version>REV 03</version>
<part-number>710-024802</part-number>
<serial-number>XH7224</serial-number>
<description>SRX5k SCB</description>
<model-number>SRX5K-SCB-A</model-number>
</chassis-module>
<chassis-module>
<name>FPC 4</name>
<version>REV 01</version>
<part-number>750-023996</part-number>
<serial-number>WY2679</serial-number>
<description>SRX5k SPC</description>
<model-number>SRX5K-SPC-2-10-40</model-number>
<chassis-sub-module>
<name>CPU</name>
```

```
<version>REV 02</version>
<part-number>710-024633</part-number>
<serial-number>WY3712</serial-number>
<description>SRX5k DPC PMB</description>
</chassis-sub-module>
<chassis-sub-module>
<name>PIC 0</name>
<part-number>BUILTIN</part-number>
<serial-number>BUILTIN</serial-number>
<description>SPU Cp-Flow</description>
</chassis-sub-module>
<chassis-sub-module>
<name>PIC 1</name>
<part-number>BUILTIN</part-number>
<serial-number>BUILTIN</serial-number>
<description>SPU Flow</description>
</chassis-sub-module>
</chassis-module>
<chassis-module>
<name>FPC 5</name>
<version>REV 07</version>
<part-number>750-027945</part-number>
<serial-number>XH9087</serial-number>
<description>SRX5k FIOC</description>
<chassis-sub-module>
<name>CPU</name>
<version>REV 03</version>
<part-number>710-024633</part-number>
<serial-number>XH8765</serial-number>
<description>SRX5k DPC PMB</description>
</chassis-sub-module>
<chassis-sub-module>
<name>PIC 0</name>
<version>REV 05</version>
<part-number>750-021378</part-number>
<serial-number>XH3692</serial-number>
<description>4x 10GE XFP</description>
<model-number>SRX-IOC-4XGE-XFP</model-number>
<chassis-sub-sub-module>
<name>Xcvr 0</name>
<version>REV 02</version>
<part-number>740-011571</part-number>
<serial-number>C850XJ05M</serial-number>
<description>XFP-10G-SR</description>
</chassis-sub-sub-module>
<chassis-sub-sub-module>
<name>Xcvr 1</name>
<version>REV 02</version>
<part-number>740-011571</part-number>
<serial-number>C850XJ05F</serial-number>
<description>XFP-10G-SR</description>
</chassis-sub-sub-module>
</chassis-sub-module>
<chassis-sub-module>
<name>PIC 1</name>
<version>REV 03</version>
<part-number>750-027491</part-number>
<serial-number>XH8521</serial-number>
<description>16x 1GE TX</description>
</chassis-sub-module>
</chassis-module>
```

```
<chassis-module>
<name>Fan Tray</name>
<description>Left Fan Tray</description>
<model-number>SRX5600-FAN</model-number>
</chassis-module>
</chassis>
</chassis-inventory>
</multi-routing-engine-item>
</multi-routing-engine-results>
</rpc-reply>
```

Related Documentation

- Tracking Applications on an SRX Series Chassis Cluster on page 92
- Managing SRX Series Chassis Clusters Using SNMP on page 97

Managing SRX Series Chassis Clusters Using SNMP

This topic provides details related to managing SRX Series chassis clusters using SNMP.

SNMP MIB Walk of the jnxOperating MIB Table with Secondary Node Details

```
user@host> show snmp mib walk jnxOperatingDescr |grep node1
jnxOperatingDescr.1.2.0.0 = node1 midplane
jnxOperatingDescr.2.4.0.0 = node1 PEM 1
jnxOperatingDescr.4.2.0.0 = node1 Fan Tray
jnxOperatingDescr.4.2.1.0 = node1 Fan 1
jnxOperatingDescr.4.2.2.0 = node1 Fan 2
jnxOperatingDescr.4.2.3.0 = node1 Fan 3
jnxOperatingDescr.4.2.4.0 = node1 Fan 4
jnxOperatingDescr.7.9.0.0 = node1 FPC: SRX3k SFB 12GE @ 0/*/*
jnxOperatingDescr.7.10.0.0 = node1 FPC: SRX3k 16xGE TX @ 1/*/*
jnxOperatingDescr.7.11.0.0 = node1 FPC: SRX3k 2x10GE XFP @ 2/*/*
jnxOperatingDescr.7.14.0.0 = node1 FPC: SRX3k SPC @ 5/*/*
jnxOperatingDescr.7.15.0.0 = node1 FPC: SRX3k NPC @ 6/*/*
jnxOperatingDescr.8.9.1.0 = node1 PIC: 8x 1GE-TX 4x 1GE-SFP @ 0/0/*
jnxOperatingDescr.8.10.1.0 = node1 PIC: 16x 1GE-TX @ 1/0/*
jnxOperatingDescr.8.11.1.0 = node1 PIC: 2x 10GE-XFP @ 2/0/*
jnxOperatingDescr.8.14.1.0 = node1 PIC: SPU Cp-Flow @ 5/0/*
jnxOperatingDescr.8.15.1.0 = node1 PIC: NPC PIC @ 6/0/*
jnxOperatingDescr.9.3.0.0 = node1 Routing Engine 0
jnxOperatingDescr.10.2.1.0 = node1 FPM Board
jnxOperatingDescr.12.3.0.0 = node1 CB 0
root@SRX3400-1> show snmp mib walk jnxOperatingState
jnxOperatingState.1.1.0.0 = 2
jnxOperatingState.1.2.0.0 = 2
jnxOperatingState.2.2.0.0 = 2
jnxOperatingState.2.4.0.0 = 2
jnxOperatingState.4.1.0.0 = 2
jnxOperatingState.4.1.1.0 = 2
jnxOperatingState.4.1.2.0 = 2
jnxOperatingState.4.1.3.0 = 2
jnxOperatingState.4.1.4.0 = 2
jnxOperatingState.4.2.0.0 = 2
jnxOperatingState.4.2.1.0 = 2
jnxOperatingState.4.2.2.0 = 2
jnxOperatingState.4.2.3.0 = 2
jnxOperatingState.4.2.4.0 = 2
jnxOperatingState.7.1.0.0 = 2
jnxOperatingState.7.2.0.0 = 2
jnxOperatingState.7.3.0.0 = 2
jnxOperatingState.7.6.0.0 = 2
jnxOperatingState.7.7.0.0 = 2
jnxOperatingState.7.9.0.0 = 2
jnxOperatingState.7.10.0.0 = 2
jnxOperatingState.7.11.0.0 = 2
jnxOperatingState.7.14.0.0 = 2
jnxOperatingState.7.15.0.0 = 2
jnxOperatingState.8.1.1.0 = 2
jnxOperatingState.8.2.1.0 = 2
jnxOperatingState.8.3.1.0 = 2
jnxOperatingState.8.6.1.0 = 2
jnxOperatingState.8.7.1.0 = 2
jnxOperatingState.8.9.1.0 = 2
jnxOperatingState.8.10.1.0 = 2
jnxOperatingState.8.11.1.0 = 2
jnxOperatingState.8.14.1.0 = 2
jnxOperatingState.8.15.1.0 = 2
jnxOperatingState.9.1.0.0 = 2
jnxOperatingState.9.3.0.0 = 2
jnxOperatingState.10.1.1.0 = 2
jnxOperatingState.10.2.1.0 = 2
jnxOperatingState.12.1.0.0 = 2
jnxOperatingState.12.3.0.0 = 2
```

Redundant Ethernet Interface Statistics in MIB II Tables

```
user@host> show snmp mib walk ifName |grep reth
ifName.194 = reth0
ifName.195 = reth1
ifName.236 = reth2
ifName.537 = reth0.0
ifName.544 = reth1.0
ifName.546 = reth2.0
```

{primary:node0}

```
user@host> show snmp mib walk ifTable|grep 537
ifIndex.537   = 537
ifDescr.537   = reth0.0
ifType.537    = 161
ifMtu.537    = 1500
ifSpeed.537   = 4294967295
ifPhysAddress.537 = 00 10 db ff 10 00
ifAdminStatus.537 = 1
ifOperStatus.537 = 1
ifLastChange.537 = 67326279
ifInOctets.537 = 487194518
ifInUcastPkts.537 = 3822455
ifInNUcastPkts.537 = 0
ifInDiscards.537 = 0
ifInErrors.537 = 0
ifInUnknownProtos.537 = 0
ifOutOctets.537 = 9790018
ifOutUcastPkts.537 = 227792
ifOutNUcastPkts.537 = 0
ifOutDiscards.537 = 0
ifOutErrors.537 = 0
ifOutQLen.537 = 0
ifSpecific.537 = 0.0
```

```
user@host> show snmp mib walk ifXTable|grep 537
```

```
ifName.537 = reth0.0
ifInMulticastPkts.537 = 0
ifInBroadcastPkts.537 = 0
ifOutMulticastPkts.537 = 0
ifOutBroadcastPkts.537 = 0
ifHCInOctets.537 = 17667077627
ifHCInUcastPkts.537 = 17183691748
ifHCInMulticastPkts.537 = 0
ifHCInBroadcastPkts.537 = 0
ifHCOutOctets.537 = 9790270
ifHCOutUcastPkts.537 = 227798
ifHCOutMulticastPkts.537 = 0
ifHCOutBroadcastPkts.537 = 0
ifLinkUpDownTrapEnable.537 = 1
ifHighSpeed.537 = 10000
ifPromiscuousMode.537 = 2
ifConnectorPresent.537 = 2
ifAlias.537
ifCounterDiscontinuityTime.537 = 0
```

Related Documentation

- Managing SRX Series Chassis Clusters Using RPCs on page 92
- Event Script for Generating Chassis Cluster SNMP Traps on page 99

Event Script for Generating Chassis Cluster SNMP Traps

The following information shows the output for generating the jnxEventTrap SNMP trap event script.

```
version 1.0;

ns junos = "http://xml.juniper.net/junos/*/junos";
ns xnm = "http://xml.juniper.net/xnm/1.1/xnm";
ns jcs = "http://xml.juniper.net/junos/commit-scripts/1.0";

param $event;
param $message;

match  / {
    /*
   * trapm utilty wants the following characters in the value to be escaped
   * '[', ']', ' ', '=', and ','
   */
   var $event-escaped = {
     call escape-string($text = $event, $vec = '[] =,');
   }

   var $message-escaped = {
     call escape-string($text = $message, $vec = '[] =,');
   }

   <op-script-results> {
    var $rpc = <request-snmp-spoof-trap> {
      <trap> "jnxEventTrap";
      <variable-bindings> "jnxEventTrapDescr[0]='Event-Trap' , "
      _ "jnxEventAvAttribute[1]='event' , "
      _ "jnxEventAvValue[1]='" _ $event-escaped _ "' , "
      _ "jnxEventAvAttribute[2]='message' , "
      _ "jnxEventAvValue[1]='" _ $message-escaped _ "'";
  }

  var $res = jcs:invoke($rpc);
 }
}
template escape-string ($text, $vec) {

  if (jcs:empty($vec)) {
    expr $text;
  } else {
    var $index = 1;
    var $from = substring($vec, $index, 1);
    var $changed-value = {
      call replace-string($text, $from) {
        with $to = {
          expr "\\";
          expr $from;
        }
      }
    }
    call escape-string($text = $changed-value, $vec = substring($vec, $index+
1));
  }
}
template replace-string ($text, $from, $to) {
  if (contains($text, $from)) {
    var $before = substring-before($text, $from);
    var $after = substring-after($text, $from);
    var $prefix = $before _ $to;

    expr $before;
    expr $to;
    call replace-string($text = $after, $from, $to);

  } else {
    expr $text;
  }
}
```

Related Documentation

- Managing SRX Series Chassis Clusters Using SNMP on page 97
- Utility MIB Examples on page 100

Utility MIB Examples

This topic presents examples of **show** command output and SNMP MIB walk results using the utility MIB for power readings on a Junos OS device.

Show Command Output for Power Readings

```
user@host> show chassis environment pem

PEM 0 status:
  State              Online
  Temperature           OK
  AC Input:            OK
  DC Output      Voltage  Current    Power   Load
                 50      12        600    35
```

```
PEM 1 status:
  State Online
  Temperature OK
  AC Input: OK
  DC Output     Voltage  Current    Power   Load
                50     13       650    38

PEM 2 status:
  State             Present
PEM 3 status:
  State             Present
```

In the following example, the index is PEM *pem number type of reading*.

SNMP MIB Walk Results for the jnx-utility MIB Populated with Power Readings

```
user@host> show snmp mib walk jnxUtil ascii
jnxUtilStringValue."PEM0dc-current" = 12
jnxUtilStringValue."PEM0dc-load" = 35
jnxUtilStringValue."PEM0dc-power" = 600
jnxUtilStringValue."PEM0dc-voltage" = 50
jnxUtilStringValue."PEM1dc-current" = 13
jnxUtilStringValue."PEM1dc-load" = 38
jnxUtilStringValue."PEM1dc-power" = 650
jnxUtilStringValue."PEM1dc-voltage" = 50
jnxUtilStringTime."PEM0dc-current" = 07 d9 09 15 0a 10 2d 00 2b 00 00
jnxUtilStringTime."PEM0dc-load" = 07 d9 09 15 0a 10 2d 00 2b 00 00
jnxUtilStringTime."PEM0dc-power" = 07 d9 09 15 0a 10 2d 00 2b 00 00
jnxUtilStringTime."PEM0dc-voltage" = 07 d9 09 15 0a 10 2d 00 2b 00 00
jnxUtilStringTime."PEM1dc-current" = 07 d9 09 15 0a 10 2d 00 2b 00 00
jnxUtilStringTime."PEM1dc-load" = 07 d9 09 15 0a 10 2d 00 2b 00 00
jnxUtilStringTime."PEM1dc-power" = 07 d9 09 15 0a 10 2d 00 2b 00 00
jnxUtilStringTime."PEM1dc-voltage" = 07 d9 09 15 0a 10 2d 00 2b 00 00
```

Sample Script

```
version 1.0;
ns junos = "http://xml.juniper.net/junos/*/junos";
ns xnm = "http://xml.juniper.net/xnm/1.1/xnm";
ns jcs = "http://xml.juniper.net/junos/commit-scripts/1.0";
ns ext = "http://xmlsoft.org/XSLT/namespace";

import "../import/junos.xsl";
match /  {
   <op-script-results> {
     var $command="get-environment-pem-information";
     var $pem = jcs:invoke($command);

      var $chassis= "get-chassis-inventory";
      var $getchassis = jcs:invoke($chassis);


if ( contains( $getchassis , "EX") or contains( $getchassis , "M10") or contains(
 $getchassis , "M7")) {
<xsl:message terminate="yes"> "Power readings not supported";
      }

     /* if PEM is empty then exit and terminate*/
        if( jcs:empty( $pem) ) {
              <xsl:message terminate="yes"> "Power readings not reported";
      }
```

```
    for-each ($pem/environment-component-item)
    {
      var $pemslot = substring-after(name, " ");

     for-each (./dc-information/dc-detail/*)
     {
    var $info=name();
    var $valueofinfo = .;
   call snmp_set($instance = "PEM" _ $pemslot _ $info, $value = $valueofinfo);
     }
    }
      }
}
template snmp_set($instance, $value = "0", $type = "string" ) {

   var $set_rpc = <request-snmp-utility-mib-set> {
      <object-type> $type;
    <object-value> $value;
      <instance> $instance;
  }
   var $out = jcs:invoke($set_rpc);
}
```

Configuration on the Device

```
user@host> show configuration event-options

  generate-event {
    1-min time-interval 60;
  }
    policy powerUtil {
      events 1-min;
      then {
        event-script power.slax;
      }
    }
```

Related Documentation

- Monitoring Chassis Cluster Faults on page 51
- Event Script for Generating Chassis Cluster SNMP Traps on page 99