White Paper

Best Practices for SRX Series Cluster Management

This document provides best practices for managing Juniper Networks® SRX Series Services Gateways chassis clusters using network management applications and systems.

November 2011

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408.745.2000
1.888 JUNIPER

www.juniper.net
# Contents

White Paper ............................................................................................................................................................................ 1

Best Practices for SRX Series Cluster Management ................................................................................................................ 1

November 2011 ...................................................................................................................................................................... 1

Managing and Monitoring SRX Clusters ................................................................................................................................. 5

Introduction ............................................................................................................................................................................ 5

Scope....................................................................................................................................................................................... 5

Design Considerations............................................................................................................................................................. 6

Description and Deployment Scenarios.................................................................................................................................. 9

How to Communicate With a Device .................................................................................................................................... 20

High Availability Cluster Best Practices ................................................................................................................................. 21

Retrieving Chassis Inventory and Interfaces ......................................................................................................................... 23

Identifying Cluster and Primary and Secondary Nodes ........................................................................................................ 25

  Determining the IP Address of Nodes.................................................................................................................................. 27

Chassis Cluster Details .......................................................................................................................................................... 28

Performance and Fault Management of Clusters .................................................................................................................... 35

  Chassis Components and Environment ................................................................................................................................ 35

  Chassis Cluster Statistics ................................................................................................................................................... 37

Redundant Group Monitoring ........................................................................................................................................ 38

Interface Statistics ............................................................................................................................................................. 38

Services Processing Unit Monitoring .................................................................................................................................... 39

Security Features .............................................................................................................................................................. 42

Other Statistics and MIBS ................................................................................................................................................. 43

  RMON ................................................................................................................................................................................ 43

  Health Monitoring............................................................................................................................................................. 44

  Fault Management............................................................................................................................................................ 44

  SNMP Traps ........................................................................................................................................................................ 44

System Log Messages ....................................................................................................................................................... 66

Detecting a Failover or Switchover ....................................................................................................................................... 68

Other Indications for Failover ............................................................................................................................................... 72

Using Operational and Event Scripts for Management and Monitoring ........................................................................... 72

Utility MIB for Monitoring ................................................................................................................................................. 74
SRX Log Messages ................................................................................................................................................................. 74
Application Tracking .............................................................................................................................................................. 80
Summary ............................................................................................................................................................................... 80
APPENDIX A: XML RPC Sample Response ............................................................................................................................. 81
APPENDIX B: Sample SNMP Responses ................................................................................................................................ 87
APPENDIX C: Event Script for Generating jnxEventTrap SNMP Traps .................................................................................. 90
APPENDIX D: Utility MIB Examples ....................................................................................................................................... 92

List of Figures

Figure 1: SRX5800 Devices in a Cluster ................................................................................................................................... 8
Figure 2: High-End Cluster Setup Connecting to a Management Station Through a Backup Router ................................. 9
Figure 3: Branch SRX Cluster Setup Connecting to a Management Station Through a Backup Router .............................. 10
Figure 4: SRX Series Clustering Model ................................................................................................................................ 17
Figure 5: SRX Branch Deployment for In-Band Management Using a reth Interface ........................................................... 19
Figure 6: SRX Branch Deployment for In-Band Management Using a Layer 3 Interface .................................................... 19
Managing and Monitoring SRX Clusters

This document provides the configuration and methods to monitor and manage SRX Series chassis clusters using various management interfaces such as SNMP, CLI, syslog, the NETCONF XML management protocol, and the Junos OS XML Management Protocol (formerly known as the JUNOScript API).

Introduction

Chassis clustering provides network node redundancy by grouping a pair of the same kind of supported SRX Series devices into a cluster. Managing SRX clusters requires more involvement and analysis compared to a standalone SRX Series device. This document helps you efficiently configure, manage, and monitor SRX Series chassis clusters in an operational environment.

Scope

This document provides 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 devices. Any exceptions, limitations, and restrictions specific to any particular platform is mentioned whenever applicable. This document covers the most important aspects of cluster management. Management of specific features such as Network Address Translation (NAT), IP Security (IPsec), and protocols is covered only briefly in this document.
Design Considerations
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 kind 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 placement and type of 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 Services Processing Cards (SPCs) must match in the two clusters.

An SRX Series chassis cluster is created by physically connecting two identical cluster-supported SRX Series devices together 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

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

Software Requirements

- 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.

The control ports on the respective nodes are connected to form a control plane that synchronizes configuration and kernel state to facilitate the high availability of interfaces and services. The data plane on the respective nodes is 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 provides high availability of interfaces and services 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.

**Various Deployments of a Cluster**

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

Active/passive high availability mode is the most common type of high availability 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 high availability 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 dataplane 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 the egress interface (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 shows two 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

Each of these tasks are explained, with examples, in the following sections.
Description and Deployment Scenarios
This section describes two deployment scenarios.

Configuring Devices for Out-of-Band Management and Administration

Figure 2: High-End Cluster Setup Connecting to a Management Station Through a Backup Router
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.
Sample Configuration for SRX Cluster Management

root@SRX3400-1# 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 "$9$T3Ct0BIEylIRs24JDj01IRrevWLx-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 "$9$kFp9ApOIEAtvWXxdVfTQzCt0BIESrIR-VsYo9At0Rh"; ##SECRET-DATA
                    services netconf;
                    10.200.0.1 port 7804;
                }
            }
        }
    }
}
# The following syslog configuration is not applicable for Branch SRX services gateways

```bash
syslog {
    file default-log-messages {
        any notice;
        structured-data;
    }
}
}
```

```
interfaces {
    fxp0 {
        unit 0 {
            family inet {
                filter {
                    input protect;
                }
                address 172.19.100.165/24;
            }
        }
    }
}
```

```
root@SRX3400-1# show apply-groups
apply-groups "${node}";
{primary:node0}[edit]
```

```
root@SRX3400-1# show interfaces
interfaces {
    fxp0 {
        unit 0 {
            family inet {
                filter {
                    input protect;
                }
                address 172.19.100.166/24 {
                    master-only;
                }
            }
        }
    }
}
```

```
{primary:node0}[edit]
root@SRX3400-1# 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 {
Best Practices for SRX Cluster Management

```
type trap;
tag router1;
}
notify-filter filter1 {
  oid .1 include;
}
}

[primary:node0][edit]
root@SRX3400-1# 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 {
Best Practices for SRX Cluster Management

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 chassis cluster through fxp0 is to give IPs 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 document. Note: We recommend using a master-only IPv4 address for management, especially while using SNMP. This makes the device 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 same subnet as the management IPv4 address. If the host is on a different subnet than the management IPv4 address, it 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 (RPD) 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. We recommend that the destination address specified in the backup-router statement not be 0.0.0.0/0. The address must be a proper subnet range of /8 or higher. 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 using the SSH protocol, NETCONF XML management protocol, or Junos OS XML Management Protocol. This way the device connects back automatically even after a switchover.

- We recommend that the SNMP engine ID be different for each node. This is because SNMPv3 uses the engine boots value for authentication of the protocol data units (PDUs) and the SNMP engine boots value is different for each node. SNMPv3 might fail after a switchover when the engine boots value does not match the expected value. Most of the protocol stacks will resynchronize if the engine IDs are different.

- Keep other SNMP configurations, such as the SNMP communities, trap-groups, and so on, common between the nodes as shown in the previous 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 informs. Use the `client-only` configurable option to restrict SNMP access to 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. As in the following 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 and denies other traffic. This filter is applied to the `fxp0` interface.

- You can also use security zones to restrict the traffic. See the *Junos OS Security Configuration Guide* for details.

### 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-0/0/0/6` interface is repurposed as the management interface and is automatically renamed `fxp0`. See the *Junos OS Security Configuration Guide* for management interface details.

- Syslog should be used with caution. It can cause cluster instability. Data plane logging should never be sent through syslogs for Branch SRX devices.

**Note:** [PR/413719] – For Junos OS releases before Release 9.5, on SRX3400, SRX3600, SRX5600, and SRX5800 devices, no trap is generated for redundancy group 0 failover. You can check on redundancy group 0 states only when you log in to the device. 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.

### Managing Clusters Through the Redundant Ethernet Interfaces

SRX 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 high availability feature available in Junos OS for SRX Series Services Gateways is modeled after the redundancy features found in Junos OS-based routers. Designed with separate control and data planes, Junos OS-based routers 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 (among many other things). 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.

![Figure 4: SRX Series Clustering Model](image)

Just like in a router with two Routing Engines, the control plane of an SRX 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
Note that 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, a new type of interface, the fxp0 interface, has been added to provide 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 section explains how to manage a chassis cluster through the primary node without requiring the use of the fxp0 interfaces i.e. In-band management. This is particularly needed for SRX Series branch devices since the typical deployment for SRX Series for the branch devices is such that there is no management network available to monitor the remote branch office.

Note: Before Junos OS Release 10.1R2, 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 10.1R2 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 and Figure 6.

In both cases, establishing a connection to any of the local addresses connects to the primary node. To be precise, it connects 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 4 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 Branch Deployment for In-Band Management Using a reth Interface

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

Figure 6: SRX 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: Advantages and Disadvantages of Using Different Interfaces

<table>
<thead>
<tr>
<th>fxp Interface</th>
<th>Reth and Transit Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 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.</td>
<td>• 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.</td>
</tr>
<tr>
<td>• 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.</td>
<td>• 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.</td>
</tr>
<tr>
<td>• 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.</td>
<td>• The reth interface does not need two IP addresses, and no switch is required to connect to the SRX chassis cluster. Transit interfaces on each node, if used for management, needs two explicit IP addresses for each interface. But since this is a transit interface, the IP addresses are used for traffic other than management as well.</td>
</tr>
<tr>
<td>• SRX branch device clusters with a non-Ethernet link (ADSL, T1/E1) cannot be managed using the fxp0 interface.</td>
<td>• SRX Series branch devices with a non-Ethernet link can be managed using a reth or transit interface.</td>
</tr>
</tbody>
</table>

### How to Communicate With a Device

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

- **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 it for configuring the device. This session can also be used to run Junos OS XML Management Protocol remote procedure calls (RPCs).

- **SNMP** – From an SRX 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 further sections of this document. 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.

- **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 it 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. See the Junos OS Security Configuration Guide for details on configuring and using application identification and tracking. This document only provides the best method and configuration to get the application tracking log messages from the SRX chassis clusters.

- **J-Web** – All Junos OS devices provide a graphical user interface for configuration and administration. This interface can be used for administering individual devices. The J-Web interface is not described in this document.

**High Availability Cluster Best Practices**

Following are some best practices to provide high availability for SRX chassis clusters:

- **Dual Control Links** – This is where 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.

- **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.

- **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 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 raised, which indicates an issue between the nodes.
- **Track-IP** – The track IP feature was missing from the initial release of the SRX Series Services Gateways platform. In response to this, a Junos OS automation script was created to implement this feature. The Junos OS track IP automation script 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 the detection of a failure, the script executes a failover to the other node in an attempt to prevent downtime.

- **Interface Monitoring** – The other SRX Series high availability 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:** We do not recommend interface monitoring for redundancy group 0.

**Sample Interface Monitoring Configuration**

```markdown
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;
            }
          }
        }
      }
    }
  }
}
```


• **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.

The main benefits of graceful restart are uninterrupted packet forwarding and temporary suppression of all routing protocol updates. Graceful restart thus enables a router to pass through intermediate convergence states that are hidden from the rest of the network.

**Retrieving Chassis Inventory and Interfaces**

SRX Series chassis cluster inventory and interface information can be gathered to monitor the hardware components and the interfaces on the cluster. The primary node has the 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>` RPC to get the chassis inventory. This RPC reports components on both the primary and secondary nodes. See APPENDIX A: XML RPC Sample Response for samples.
- 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 OS Junos XML API Operational Reference* for details about using the RPCs and varying the details of the 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.

**Sample Chassis Definition MIB**

```
<table>
<thead>
<tr>
<th>MIB Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jnxProductLineSRX3600</td>
<td>OBJECT IDENTIFIER ::= { jnxProductLine 34 }</td>
</tr>
<tr>
<td>jnxProductNameSRX3600</td>
<td>OBJECT IDENTIFIER ::= { jnxProductName 34 }</td>
</tr>
<tr>
<td>jnxProductModelSRX3600</td>
<td>OBJECT IDENTIFIER ::= { jnxProductModel 34 }</td>
</tr>
<tr>
<td>jnxProductVariationSRX3600</td>
<td>OBJECT IDENTIFIER ::= { jnxProductVariation 34 }</td>
</tr>
<tr>
<td>jnxChassisSRX3600</td>
<td>OBJECT IDENTIFIER ::= { jnxChassis 34 }</td>
</tr>
</tbody>
</table>
```
Best Practices for SRX Cluster Management

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

**Chassis MIB Walk Example**

```
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.

Identifying Cluster and Primary and Secondary Nodes
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.

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

• Use the <get-chassis-cluster-status> 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 Appendix for sample output.

• Use the get-chassis-inventory 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 APPENDIX A: XML RPC Sample Response for sample output of the RPC. The following 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>

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 using 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) |
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.

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

- `<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 fxp0 and reth interface IP addresses. Using this 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.

**Sample Interface Details From the XML RPC Response**

```
<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>
```
Using SNMP MIBs

- Use the `ifTable` MIB table to get the `ifIndex` MIB object of the fxp0 interface and the reth interfaces 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**

```plaintext
{primary:node0}
regress@SRX5600-1> show snmp mib walk ifDescr | grep fxp0
ifDescr.1     = fxp0
ifDescr.13    = fxp0.0

regress@SRX5600-1> 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.

**Chassis Cluster Details**

To monitoring 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 document 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 that you configure. Each redundancy group x acts as an independent unit of failover and is primary on only one node at a time. Management systems can monitor the cluster using the following Junos OS XML API. 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>` RPC to get the redundancy configuration and the redundancy groups present on the device. This provides the redundancy groups configured. The following sample output shows two redundancy groups present and configured.
XML RPC for Configuration Retrieval

```xml
<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>
        </redundancy-group>
      </cluster>
    </chassis>
  </configuration>
</rpc-reply>
```
• **Chassis Cluster Redundant Ethernet Interfaces** — A redundant Ethernet interface is a pseudo interface that includes at minimum one physical interface from each node of the cluster as previously explained. A redundant Ethernet interface is referred to as a reth in configuration commands.

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

• Use the `<get-chassis-cluster-interfaces>` RPC to find out the reth interface details. The following sample output shows four reth interfaces configured.

**XML RPC for Chassis Cluster Interfaces**

```xml
<rpc>
  <get-chassis-cluster-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-interface-statistics>
    <control-interface-index>0</control-interface-index>
    <control-interface-name>em0</control-interface-name>
    <reth>
      <reth-name>reth0</reth-name>
      <reth-status>Up</reth-status>
      <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
      <reth-name>reth1</reth-name>
      <reth-status>Up</reth-status>
      <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
      <reth-name>reth2</reth-name>
      <reth-status>Up</reth-status>
      <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
      <reth-name>reth3</reth-name>
      <reth-status>Up</reth-status>
      <redundancy-group-id-for-reth>1</redundancy-group-id-for-reth>
  </reth>
</chassis-cluster-interface-statistics>
</rpc-reply>
```
Use the `<get-interface-information>` 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.

**XML RPC for Interface Information**

```xml
<rpc>
  <get-interface-information>
    <terse/>
    <interface-name>reth0</interface-name>
  </get-interface-information>
</rpc>
```
<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>
        </logical-interface>
      </physical-interface>
    </interface-information>
  </rpc-reply>
<address-family>
  <address-family-name>aenet</address-family-name>
</address-family>
<ae-bundle-name>reth0.0</ae-bundle-name>
</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 indices. See the following sample output.

Sample SNMP Data for the Reth Interface Details

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

{primary:node0}
regress@SRX5600-1> 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 Index of all interfaces from ifTable. Following shows indexes of interfaces we need in this example
{primary:node0}
regress@SRX5600-1> show snmp mib walk ifDescr | grep reth0
ifDescr.503 = reth0.0
ifDescr.528 = reth0

Next 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 index. Index 522 and 552 represent interfaces ge-5/1/1.0 and ge-11/1/1.0 respectively.
{primary:node0}
regress@SRX5600-1> show snmp mib walk ifStackStatus | grep 503
ifStackStatus.0.503 = 1
ifStackStatus.503.522 = 1
ifStackStatus.503.552 = 1
{primary:node0}
regress@SRX5600-1> show snmp mib walk ifDescr | grep 522
ifDescr.522 = ge-5/1/1.0
Best Practices for SRX Cluster Management

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>` 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>
```

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 find out the data plane interfaces.

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

- Use the `<get-chassis-cluster-data-plane-interfaces>` 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>
```
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 Cluster Nodes

Use the NETCONF XML management protocol for configuration and provisioning of SRX Series devices and Junos OS devices in general. See the NETCONF XML Management Protocol Guide for usage. We recommend using groups to configure SRX Series chassis clusters. Use global groups for all configurations that are common between the nodes and keep the node-specific configuration to a minimum as mentioned in previous sections of this document.

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 allow you better control over how your devices are configured:
- Your design rules
- Your implementation details
- 100 percent of your design standards

Performance and Fault Management of Clusters

Note that the primary node returns the status of the secondary nodes as well. Exceptions to this rule, if there are any, are mentioned in specific sections. Sample responses and outputs are presented where it is feasible and necessary.

Chassis Components and Environment

This section shows the options available for monitoring chassis components such as FPCs, PICs, and Routing Engines for measurements 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.
Table 2: Instrumentation for Chassis Component Monitoring

<table>
<thead>
<tr>
<th>Junos OS XML RPC</th>
<th>SNMP MIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>• For temperature of sensors, fan speed and status of each component use the <code>&lt;get-environment-information&gt;</code> RPC.</td>
<td>• Use the <code>jnxOperatingTable</code> MIB table for temperature, fan speed, and so on. The <code>jnxOperatingState</code> MIB should be used to get the status of the component. If the component is a FRU, then use the <code>jnxFruState</code> MIB also. Use the <code>jnxOperatingTemp</code> MIB for the temperature of sensors. Use the <code>jnxFruState</code> MIB to get the FRU status such as offline, online, empty, and so on.</td>
</tr>
<tr>
<td>• For temperature thresholds for the hardware components for each element, use the <code>&lt;get-temperature-threshold-information&gt;</code> RPC.</td>
<td>• Note the following about the objects available for monitoring in the <code>jnxOperatingTable</code> MIB table.</td>
</tr>
<tr>
<td>• For Routing Engine status, CPU, and memory, use the <code>&lt;get-route-engine-information&gt;</code> RPC.</td>
<td>• No MIB is available for temperature thresholds.</td>
</tr>
<tr>
<td>This RPC provides 1, 5, and 15 minute load averages.</td>
<td>• For the Routing Engine, use the <code>jnxOperatingCPU</code>, <code>jnxOperatingTemp</code>, <code>jnxOperatingMemory</code>, <code>jnxOperatingISR</code>, and <code>jnxOperatingBuffer</code> MIB objects under container Index 9.</td>
</tr>
<tr>
<td>• For FPC status, temperature, CPU, and memory, use the <code>&lt;get-fpc-information&gt;</code> RPC.</td>
<td>• Look at the <code>jnxRedundancyTable</code> for redundancy status monitoring. This only gives data for the last 5 seconds.</td>
</tr>
<tr>
<td>• Use the <code>&lt;get-pic-detail&gt;</code> RPC with the <code>&lt;fpc-slot&gt;</code> and <code>&lt;pic-slot&gt;</code> RPCs to get the PIC status.</td>
<td>• For the FPCs, look at the objects in the <code>jnxOperatingTable</code> and <code>jnxFruTable</code> MIB tables on container Index 7 for temperature, CPU, and memory utilization.</td>
</tr>
<tr>
<td></td>
<td>• For the PICs (including SPU/SPC cards flows), look at the objects in the <code>jnxOperatingTable</code> and <code>jnxFruTable</code> MIB tables under container Index 8 in the following sample output for temperature, CPU, and memory utilization.</td>
</tr>
</tbody>
</table>

```
regress@SRX5600-1> 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/*
```
Accounting Profiles

- 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 [edit accounting-options] 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 [edit accounting-options] hierarchy level. Note that only the primary node master Routing Engine statistics are available.

The Routing Engine accounting profile is stored in the /var/log directory by default. The following sample shows what the accounting profile looks like.  

```
#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
```

- Use a MIB accounting profile for any other MIBs listed in the SNMP MIB column to get results in a CSV format. You can choose which MIB objects, collection interval, and so on. See the Junos OS Network Management Configuration Guide for details on accounting profiles.

Chassis Cluster Statistics

This section describes how to measure and monitor the cluster health including the control plane and data plane statistics.

Instrumentation for Chassis Cluster Monitoring

<table>
<thead>
<tr>
<th>Junos OS XML RPC</th>
<th>SNMP MIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Use the <code>&lt;get-chassis-cluster-statistics&gt;</code> RPC to get the cluster statistics, including the control plane, fabric, and dataplane statistics.</td>
<td>Not available. The utility MIB can be used to provide this data using Junos OS operation scripts. For more information about operation scripts, see the Junos OS Configuration and Operations Automation Guide.</td>
</tr>
<tr>
<td>- If you want to monitor, dataplane and control plane statistics separately, you can use the <code>&lt;get-chassis-cluster-control-plane-statistics&gt;</code> and <code>&lt;get-chassis-cluster-data-plane-statistics&gt;</code> RPCs respectively.</td>
<td></td>
</tr>
</tbody>
</table>
Redundant Group Monitoring
Note that the redundancy groups should be discovered prior to monitoring the group status.

Instrumentation for Redundancy Group Monitoring

<table>
<thead>
<tr>
<th>Junos OS XML RPC</th>
<th>SNMP MIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the <code>&lt;get-chassis-cluster-status&gt;</code> RPC to get chassis cluster information as shown. RPC: <code>&lt;get-chassis-cluster-status&gt;</code> Sample: <code>&lt;rpc&gt;</code> &lt;get-chassis-cluster-status&gt; &lt;redundancy-group&gt;1&lt;/redundancy-group&gt; &lt;/get-chassis-cluster-status&gt; &lt;/rpc&gt;</td>
<td>Not available. The utility MIB can be used to provide this data using Junos OS operation scripts. For more information about operation scripts, see the Junos OS Configuration and Operations Automation Guide.</td>
</tr>
</tbody>
</table>

Interface Statistics
You can use the following methods 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).

Instrumentation for Interface Monitoring

<table>
<thead>
<tr>
<th>Junos OS XML RPC</th>
<th>SNMP MIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Use the <code>&lt;get-interface-information&gt;</code> RPC with the <code>&lt;extensive&gt;</code> 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.</td>
<td>• Use the following MIB tables for interface statistics: o <code>ifTable</code> – Standard MIB II interface stats o <code>ifXTable</code> – Standard MIB II high-capacity interface stats o <code>JUNIPER-IF-MIB</code> – A list of Juniper extensions to the interface entries o <code>JUNIPER-JS-IF-EXT-MIB</code> – 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).</td>
</tr>
</tbody>
</table>

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 and SRX5000 lines 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 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, chassisd is the center for hardware failure detection. SPU monitoring is enabled by default.

Use the following methods 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, with greater than 95 percent utilization leading to packet drops.

### Instrumentation for SPU Monitoring

<table>
<thead>
<tr>
<th>Junos OS XML RPC</th>
<th>SNMP MIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the <code>&lt;get-flow-session-information&gt;</code> RPC to get the SPU to monitor data such as total sessions, current sessions, and max sessions per node.</td>
<td>Use the <code>jnxJsSPU</code> MIB for monitoring the SPU data:</td>
</tr>
<tr>
<td><code>&lt;rpc&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>o <code>jnxJsSPUCurrentTotalSession</code> – Returns the system-level current total sessions.</td>
</tr>
<tr>
<td><code>&lt;get-flow-session-information&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>o <code>jnxJsSPUMaxTotalSession</code> – Returns the system-level max sessions possible.</td>
</tr>
<tr>
<td><code>&lt;summary/&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>o <code>jnxJsSPUObjectsTable</code> – Returns the SPU utilization statistics per node.</td>
</tr>
<tr>
<td><code>&lt;/get-flow-session-information&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;/rpc&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>

**Sample Walk**

```bash
regress@SRX5600-1> show snmp mib walk
jnxJsSPU
jnxJsSPUCurrentTotalSession.16 = 4
jnxJsSPUCurrentTotalSession.17 = 4
jnxJsSPUMaxTotalSession.16 = 524288
jnxJsSPUMaxTotalSession.17 = 1048576
```
Note: Junos OS versions prior to Junos OS Release 9.6, only return local node data for this MIB. To support high availability, Junos OS Release 9.6 and later support a jnxJsSPUMonitoringNodeIndex index and a jnxJsSPUMonitoringNodeDescr field in the table. Therefore, in high-availability mode, Junos OS Release 9.6 and later return SPU monitoring data of both the primary and secondary nodes.

Note that 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.
• Use the `<get-performance-session-information>` RPC for SPU session performance.
• Use the `<get-spu-monitoring-information>` RPC for monitoring SPU CPU utilization, memory utilization, max flow sessions, and so on.
Security Features
Following is a summary of Junos OS XML RPCs and SNMP MIBs related to security features and 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 help you decide on which instrumentation to use.

Instrumentation for Security Monitoring

<table>
<thead>
<tr>
<th>Feature and Functionality</th>
<th>Junos OS XML RPC</th>
<th>SNMP MIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPsec</td>
<td>• &lt;get-ipsec-tunnel-redundancy-information&gt;</td>
<td>JNX-IPSEC-MONITOR-MIB</td>
</tr>
<tr>
<td></td>
<td>• &lt;get-services-ipsec-statistics-information&gt;</td>
<td>JUNIPER-JS-IPSEC-VPN</td>
</tr>
<tr>
<td></td>
<td>• &lt;get-ike-security-associations&gt;</td>
<td>Juniper-IPSEC-FLOW-MONITOR</td>
</tr>
<tr>
<td>NAT</td>
<td>• &lt;get-service-nat-mapping-information&gt;</td>
<td>JNX-JS-NAT-MIB</td>
</tr>
<tr>
<td></td>
<td>• &lt;get-service-nat-pool-information&gt;</td>
<td></td>
</tr>
<tr>
<td>Screening</td>
<td>• &lt;get-ids-statistics&gt;</td>
<td>JNX-JS-SCREENING-MIB</td>
</tr>
<tr>
<td>Firewall</td>
<td>• &lt;get-firewall-counter-information&gt;</td>
<td>JUNIPER-FIREWALL-MIB</td>
</tr>
<tr>
<td></td>
<td>• &lt;get-firewall-filter-information&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-firewall-information&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-firewall-log-information&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-firewall-prefix-action-information&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-flow-table-statistics-information&gt;</td>
<td></td>
</tr>
<tr>
<td>Security Policies</td>
<td>• &lt;get-firewall-policies&gt;</td>
<td>JUNIPER-JS-POLICY-MIB</td>
</tr>
<tr>
<td>AAA</td>
<td>• &lt;get-aaa-module-statistics&gt;</td>
<td>JUNIPER-USER-AAA-MIB</td>
</tr>
<tr>
<td></td>
<td>• &lt;get-aaa-subscriber-statistics&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-aaa-subscriber-table&gt;</td>
<td></td>
</tr>
<tr>
<td>IDP</td>
<td>• &lt;get-idp-addos-application-information&gt;</td>
<td>JUNIPER-JS-IDP-MIB</td>
</tr>
<tr>
<td></td>
<td>• &lt;get-idp-application-system-cache&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-idp-counter-information&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-idp-detail-status-information&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-idp-memory-information&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-idp-policy-template-information&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-idp-predefined-attack-filters&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-idp-predefined-attack-groups&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-idp-predefined-attacks&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-idp-recent-security-package-information&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• &lt;get-idp-security-package-information&gt;</td>
<td></td>
</tr>
</tbody>
</table>
Best Practices for SRX Cluster Management

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 Junos OS Network Management Configuration Guide and the SRX5x00 and SRX3x00 Services Gateways MIB References 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 they cross upper and lower thresholds. 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;
  }
}
```
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). See the Junos OS Network Management Configuration Guide for details about configuration and health monitoring.

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

SNMP Traps
The following SNMP traps are supported on SRX Series devices. Note that only the primary node sends SNMP traps. For details of each trap, see the Junos OS Network Management Configuration Guide and the SRX5600 and SRX5800 Services Gateways MIB Reference.

Traps Supported by SRX Series Chassis Clusters

<table>
<thead>
<tr>
<th>Trap Name</th>
<th>SNMPv2 Trap OID</th>
<th>Category</th>
<th>Platforms Support the trap</th>
<th>Varbinds : See the note in the Overview and Configuration sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>authenticationFailure</td>
<td>1.3.6.1.6.3.1.1.5.5</td>
<td>Authentication</td>
<td>All Junos OS devices</td>
<td>None</td>
</tr>
<tr>
<td>linkDown</td>
<td>1.3.6.1.6.3.1.1.5.3</td>
<td>Link</td>
<td>All Junos OS devices</td>
<td>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.</td>
</tr>
<tr>
<td>linkUp</td>
<td>1.3.6.1.6.3.1.1.5.4</td>
<td>Link</td>
<td>All Junos OS devices</td>
<td>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.</td>
</tr>
<tr>
<td>pingProbeFailed</td>
<td>1.3.6.1.2.1.80.0.1</td>
<td>Remote operations</td>
<td>All Junos OS devices</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pingTestFailed</td>
<td>1.3.6.1.2.1.80.0.2</td>
<td>Remote operations</td>
<td>All Junos OS devices</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>OID</td>
<td>Remote Operations</td>
<td>All Junos OS Devices</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td>pingResultsRttSumOfSquares</td>
<td>1.3.6.1.2.1.80.1.3.1.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pingResultsLastGoodProbe</td>
<td>1.3.6.1.2.1.80.1.3.1.10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pingTestCompleted</td>
<td>1.3.6.1.2.1.80.0.3</td>
<td>Remote operations</td>
<td>All Junos OS devices</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: pingCtlTargetAddressType - 1.3.6.1.2.1.80.1.2.1.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2: pingCtlTargetAddress - 1.3.6.1.2.1.80.1.2.1.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3: pingResultsOperStatus - 1.3.6.1.2.1.80.1.3.1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4: pingResultsIpTargetAddressType - 1.3.6.1.2.1.80.1.3.1.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5: pingResultsIpTargetAddress - 1.3.6.1.2.1.80.1.3.1.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6: pingResultsMinRtt - 1.3.6.1.2.1.80.1.3.1.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7: pingResultsMaxRtt - 1.3.6.1.2.1.80.1.3.1.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8: pingResultsAverageRtt - 1.3.6.1.2.1.80.1.3.1.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9: pingResultsProbeResponses - 1.3.6.1.2.1.80.1.3.1.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10: pingResultsSentProbes - 1.3.6.1.2.1.80.1.3.1.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11: pingResultsRttSumOfSquares - 1.3.6.1.2.1.80.1.3.1.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12: pingResultsLastGoodProbe - 1.3.6.1.2.1.80.1.3.1.10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>traceRoutePathChange</td>
<td>1.3.6.1.2.1.81.0.1</td>
<td>Remote operations</td>
<td>All Junos OS devices</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: traceRouteCtlTargetAddressType - 1.3.6.1.2.1.81.1.2.1.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2: traceRouteCtlTargetAddress - 1.3.6.1.2.1.81.1.2.1.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3: traceRouteResultsIpTgtAddrType - 1.3.6.1.2.1.81.1.5.1.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4: traceRouteResultsIpTgtAddr - 1.3.6.1.2.1.81.1.5.1.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>traceRouteTestFailed</td>
<td>1.3.6.1.2.1.81.0.2</td>
<td>Remote operations</td>
<td>All Junos OS devices</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: traceRouteCtlTargetAddressType - 1.3.6.1.2.1.81.1.2.1.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2: traceRouteCtlTargetAddress - 1.3.6.1.2.1.81.1.2.1.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3: traceRouteResultsIpTgtAddrType - 1.3.6.1.2.1.81.1.5.1.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4: traceRouteResultsIpTgtAddr - 1.3.6.1.2.1.81.1.5.1.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metric</td>
<td>OID</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>----------------------------</td>
<td>----------------</td>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>traceRouteTestCompleted</td>
<td>1.3.6.1.2.1.81.0.3</td>
<td>Remote operations</td>
<td>All Junos OS devices</td>
<td></td>
</tr>
<tr>
<td>fallingAlarm</td>
<td>1.3.6.1.2.1.16.0.1</td>
<td>RMON alarm</td>
<td>All Junos OS devices</td>
<td></td>
</tr>
<tr>
<td>risingAlarm</td>
<td>1.3.6.1.2.1.16.0.2</td>
<td>RMON alarm</td>
<td>All Junos OS devices</td>
<td></td>
</tr>
<tr>
<td>bgpEstablished</td>
<td>1.3.6.1.2.1.15.7.1</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
<td></td>
</tr>
<tr>
<td>bgpBackwardTransition</td>
<td>1.3.6.1.2.1.15.7.2</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
<td></td>
</tr>
<tr>
<td>ospfVirtIfStateChange</td>
<td>1.3.6.1.2.1.14.16.2.1</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
<td></td>
</tr>
</tbody>
</table>

**Best Practices for SRX Cluster Management**

JUNIPER NETWORKS

© Juniper Networks, Inc.
<table>
<thead>
<tr>
<th>MIB Object</th>
<th>OID Path</th>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ospfNbrStateChange</td>
<td>1.3.6.1.2.1.14.16.2.2</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
</tr>
<tr>
<td>ospfVirtNbrStateChange</td>
<td>1.3.6.1.2.1.14.16.2.3</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
</tr>
<tr>
<td>ospfIfConfigError</td>
<td>1.3.6.1.2.1.14.16.2.4</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
</tr>
<tr>
<td>ospfVirtIfConfigError</td>
<td>1.3.6.1.2.1.14.16.2.5</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
</tr>
<tr>
<td>ospfIfAuthFailure</td>
<td>1.3.6.1.2.1.14.16.2.6</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
</tr>
</tbody>
</table>

- 4) ospfVirtIfState-1.3.6.1.2.1.14.9.1.7
- 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
- 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
- 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
- 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) 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
<table>
<thead>
<tr>
<th>Metric</th>
<th>OID</th>
<th>Community</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ospfVirtIfAuthFailure</td>
<td>1.3.6.1.2.1.14.16.2.7</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
</tr>
<tr>
<td>ospfIfRxBadPacket</td>
<td>1.3.6.1.2.1.14.16.2.8</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
</tr>
<tr>
<td>ospfVirtIfRxBadPacket</td>
<td>1.3.6.1.2.1.14.16.2.9</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
</tr>
<tr>
<td>ospfTxRetransmit</td>
<td>1.3.6.1.2.1.14.16.2.10</td>
<td>Routing</td>
<td>M, T, MX, J, EX, SRX Branch</td>
</tr>
</tbody>
</table>

- ospfRouterId - 1.3.6.1.2.1.14.1.1.0
- ospfVirtIfAreaId - 1.3.6.1.2.1.14.9.1.1
- ospfVirtIfNeighbor - 1.3.6.1.2.1.14.9.1.2
- ospfConfigErrorType - 1.3.6.1.2.1.14.16.1.2.0
- ospfPacketType - 1.3.6.1.2.1.14.16.1.3.0
- ospfIfRxBadPacket - 1.3.6.1.2.1.14.16.2.8
- ospfVirtIfRxBadPacket - 1.3.6.1.2.1.14.16.2.9
- ospfTxRetransmit - 1.3.6.1.2.1.14.16.2.10
- ospfRouterId - 1.3.6.1.2.1.14.1.1.0
- ospfVirtIfAreaId - 1.3.6.1.2.1.14.9.1.1
- ospfVirtIfNeighbor - 1.3.6.1.2.1.14.9.1.2
- ospfPacketType - 1.3.6.1.2.1.14.16.1.3.0
- ospfInterfaceIpAddress - 1.3.6.1.2.1.14.7.1.1
- ospfAddressLessIf - 1.3.6.1.2.1.14.7.1.2
- ospfIfRxBadPacket - 1.3.6.1.2.1.14.16.2.8
- ospfVirtIfRxBadPacket - 1.3.6.1.2.1.14.16.2.9
- ospfTxRetransmit - 1.3.6.1.2.1.14.16.2.10
- ospfNbrRtrId - 1.3.6.1.2.1.14.11.1.2
- ospfLsdbType - 1.3.6.1.2.1.14.4.1.2
- ospfLsdbLsid - 1.3.6.1.2.1.14.4.1.3
- ospfLsdbRouterId - 1.3.6.1.2.1.14.4.1.4
## Best Practices for SRX Cluster Management

**ospfvirtIfTxRetransmit**  
`1.3.6.1.2.1.14.16.2.1 1`  
**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**  
`1.3.6.1.2.1.14.16.2.1 3`  
**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**  
`1.3.6.1.2.1.14.16.2.1 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) `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  
1) `vrrpOperMasterIpAddr` - `1.3.6.1.2.1.68.1.3.1.7`  

**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` - `1.3.6.1.2.1.10.166.3.2.2.1.3`  
2) `mplsTunnelOperStatus` - `1.3.6.1.2.1.10.166.3.2.2.1.3`
| mplsTunnelDown          | 1.3.6.1.2.1.10.166.3.0.2 | Routing | M, T, MX, J, EX, SRX Branch | 1) mplsTunnelAdminStatus-1.3.6.1.2.1.10.166.3.2.2.1.3 4  
2) mplsTunnelOperStatus-1.3.6.1.2.1.10.166.3.2.2.1.3 5 |
|-------------------------|--------------------------|---------|-----------------------------|----------------------------------------------------------|
| mplsTunnelRerouted      | 1.3.6.1.2.1.10.166.3.0.3 | Routing | M, T, MX, J, EX, SRX Branch | 1) mplsTunnelAdminStatus-1.3.6.1.2.1.10.166.3.2.2.1.3 4  
2) mplsTunnelOperStatus-1.3.6.1.2.1.10.166.3.2.2.1.3 5 |
| mplsTunnelReoptimized   | 1.3.6.1.2.1.10.166.3.0.4 | Routing | M, T, MX, J, EX, SRX Branch | 1) mplsTunnelAdminStatus-1.3.6.1.2.1.10.166.3.2.2.1.3 4  
2) mplsTunnelOperStatus-1.3.6.1.2.1.10.166.3.2.2.1.3 5 |
| jnxPowerSupplyFailure   | 1.3.6.1.4.1.2636.4.1.1  | Chassis | 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 | 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 |
| jnxOverTemperature      | 1.3.6.1.4.1.2636.4.1.3  | Chassis | 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-     |
<table>
<thead>
<tr>
<th>Table 1: JUNOS SNMP MIBs for Chassis Status and Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MIB</strong></td>
</tr>
<tr>
<td>jnxRedundancySwitchOver</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>jnxFruRemoval</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>jnxFruInsertion</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>jnxFruPowerOff</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>jnxFruPowerOn</td>
</tr>
<tr>
<td>jnxFruFailed</td>
</tr>
<tr>
<td>jnxFruOffline</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>jnxFruOnline</th>
<th>1.3.6.1.4.1.2636.4.1.11</th>
<th>Chassis (alarm conditions)</th>
<th>All Junos OS devices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1) jnxFruContentsIndex- 1.3.6.1.4.1.2636.3.1.15.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) jnxFruL1Index-       1.3.6.1.4.1.2636.3.1.15.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3) jnxFruL2Index-       1.3.6.1.4.1.2636.3.1.15.1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4) jnxFruL3Index-       1.3.6.1.4.1.2636.3.1.15.1.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5) jnxFruName-          1.3.6.1.4.1.2636.3.1.15.1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6) jnxFruType-          1.3.6.1.4.1.2636.3.1.15.1.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7) jnxFruSlot-          1.3.6.1.4.1.2636.3.1.15.1.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>jnxFruCheck</th>
<th>1.3.6.1.4.1.2636.4.1.12</th>
<th>Chassis (alarm conditions)</th>
<th>All Junos OS devices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1) jnxFruContentsIndex- 1.3.6.1.4.1.2636.3.1.15.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) jnxFruL1Index-       1.3.6.1.4.1.2636.3.1.15.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3) jnxFruL2Index-       1.3.6.1.4.1.2636.3.1.15.1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4) jnxFruL3Index-       1.3.6.1.4.1.2636.3.1.15.1.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5) jnxFruName-          1.3.6.1.4.1.2636.3.1.15.1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6) jnxFruType-          1.3.6.1.4.1.2636.3.1.15.1.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7) jnxFruSlot-          1.3.6.1.4.1.2636.3.1.15.1.7</td>
</tr>
<tr>
<td>jnxFEBSwitchover</td>
<td>1.3.6.1.4.1.2636.4.1.13</td>
<td>Chassis (alarm conditions)</td>
<td>All Junos OS devices</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------</td>
<td>-----------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>jnxHardDiskFailed</th>
<th>1.3.6.1.4.1.2636.4.1.14</th>
<th>Chassis (alarm conditions)</th>
<th>All Junos OS devices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>jnxHardDiskMissing</th>
<th>1.3.6.1.4.1.2636.4.1.15</th>
<th>Chassis (alarm conditions)</th>
<th>All Junos OS devices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>jnxPowerSupplyOK</th>
<th>1.3.6.1.4.1.2636.4.2.1</th>
<th>Chassis (cleared alarm conditions)</th>
<th>All Junos OS devices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trap Name</td>
<td>Object Identifier</td>
<td>Description</td>
<td>Managed Objects</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>jnxFanOK</td>
<td>1.3.6.1.4.1.2636.4.2.2</td>
<td>Chassis (cleared alarm conditions)</td>
<td>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</td>
</tr>
<tr>
<td>jnxTemperatureOK</td>
<td>1.3.6.1.4.1.2636.4.2.3</td>
<td>Chassis (cleared alarm conditions)</td>
<td>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</td>
</tr>
<tr>
<td>jnxFRUOK</td>
<td>1.3.6.1.4.1.2636.4.2.4</td>
<td>Chassis (cleared alarm conditions)</td>
<td>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</td>
</tr>
<tr>
<td>jnxCmCfgChange</td>
<td>1.3.6.1.4.1.2636.4.5.0.1</td>
<td>Configuration</td>
<td>All Junos OS devices 1) jnxCmCfgChgEventTime 2) jnxCmCfgChgEventDate 3) jnxCmCfgChgEventSource 4) jnxCmCfgChgEventUser 5) jnxCmCfgChgEventLog</td>
</tr>
<tr>
<td>jnxCmRescueChange</td>
<td>1.3.6.1.4.1.2636.4.5.0.2</td>
<td>Configuration</td>
<td>All Junos OS devices 1) jnxCmRescueChgTime 2) jnxCmRescueChgDate 3) jnxCmRescueChgSource 4) jnxCmRescueChgUser 5) jnxCmRescueChgState</td>
</tr>
<tr>
<td>jnxPingRttThresholdExceeded</td>
<td>1.3.6.1.4.1.2636.4.9.0.1</td>
<td>Remote operations</td>
<td>All Junos OS devices except EX and high-end SRX Series devices</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------------------------</td>
<td>-------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>jnxPingRttThresholdExceeded</td>
<td>1.3.6.1.4.1.2636.4.9.0.2</td>
<td>Remote operations</td>
<td>All Junos OS devices except EX and high-end SRX Series devices</td>
</tr>
<tr>
<td>Metric</td>
<td>OID</td>
<td>Remote Operations</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>------------------------------</td>
<td>-------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>jnxPingRttJitterThresholdExceeded</strong></td>
<td>1.3.6.1.4.1.2636.4.9.0.3</td>
<td>Remote operations</td>
<td>All Junos OS devices except EX and high-end SRX Series devices</td>
</tr>
<tr>
<td><strong>jnxPingEgressThresholdExceeded</strong></td>
<td>1.3.6.1.4.1.2636.4.9.0.4</td>
<td>Remote operations</td>
<td>All Junos OS devices except EX and high-end SRX Series devices</td>
</tr>
<tr>
<td>1) pingCtlTargetAddressType</td>
<td>.1.3.6.1.2.1.80.1.2.1.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2) pingCtlTargetAddress</td>
<td>.1.3.6.1.2.1.80.1.2.1.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3) pingResultsOperStatus</td>
<td>.1.3.6.1.2.1.80.1.3.1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4) pingResultsIpTargetAddressType</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5) pingResultsIpTargetAddress</td>
<td>.1.3.6.1.2.1.80.1.3.1.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6) jnxPingResultsMinRttUs</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7) jnxPingResultsMaxRttUs</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8) jnxPingResultsAvgRttUs</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9) pingResultsProbeResponses</td>
<td>.1.3.6.1.2.1.80.1.3.1.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10) pingResultsSentProbes</td>
<td>.1.3.6.1.2.1.80.1.3.1.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11) pingResultsRttSumOfSquares</td>
<td>.1.3.6.1.2.1.80.1.3.1.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12) pingResultsLastGoodProbe</td>
<td>.1.3.6.1.2.1.80.1.3.1.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13) jnxPingCtlRttJitterThreshold</td>
<td>.1.3.6.1.4.1.2636.3.7.1.2.1.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14) jnxPingResultsStdDevRttUs</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15) jnxPingResultsMinRttUs</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16) jnxPingResultsMaxRttUs</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17) jnxPingResultsAvgRttUs</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18) pingResultsProbeResponses</td>
<td>.1.3.6.1.2.1.80.1.3.1.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19) pingResultsSentProbes</td>
<td>.1.3.6.1.2.1.80.1.3.1.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jnxPingEgressStdDevThresholdExceed</td>
<td>1.3.6.1.4.1.2636.4.9.0.5</td>
<td>Remote operations</td>
<td>All Junos OS devices except EX and high-end SRX Series devices</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------------------</td>
<td>-------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>1) pingCtlTargetAddressType</td>
<td>.1.3.6.1.2.1.80.1.2.1.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2) pingCtlTargetAddress</td>
<td>.1.3.6.1.2.1.80.1.2.1.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3) pingResultsOperStatus</td>
<td>.1.3.6.1.2.1.80.1.3.1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4) pingResultsIpTargetAddressType</td>
<td>.1.3.6.1.2.1.80.1.3.1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5) pingResultsIpTargetAddress</td>
<td>.1.3.6.1.2.1.80.1.3.1.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6) jnxPingResultsMinEgressUs</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7) jnxPingResultsMaxEgressUs</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8) jnxPingResultsAvgEgressUs</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9) pingResultsProbeResponses</td>
<td>.1.3.6.1.2.1.80.1.3.1.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10) pingResultsSentProbes</td>
<td>.1.3.6.1.2.1.80.1.3.1.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11) pingResultsRttSumOfSquares</td>
<td>.1.3.6.1.2.1.80.1.3.1.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12) pingResultsLastGoodProbe</td>
<td>.1.3.6.1.2.1.80.1.3.1.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13) jnxPingResultsStddevEgressUs</td>
<td>.1.3.6.1.4.1.2636.3.7.1.3.1.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14) jnxPingCtlEgressStdDevThreshold</td>
<td>.1.3.6.1.4.1.2636.3.7.1.2.1.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Best Practices for SRX Cluster Management

#### 1) pingCtlTargetAddressType
- Specifies the type of the target address.

#### 2) pingCtlTargetAddress
- Specifies the target address for ping operations.

#### 3) pingResultsOperStatus
- Indicates the operational status of the ping results.

#### 4) pingResultsIpTargetAddressType
- Specifies the type of the IP target address.

#### 5) pingResultsIpTargetAddress
- Specifies the IP target address for ping operations.

#### 6) jnxPingResultsMinEgressUs
- Represents the minimum egress utilization.

#### 7) jnxPingResultsMaxEgressUs
- Represents the maximum egress utilization.

#### 8) jnxPingResultsAvgEgressUs
- Represents the average egress utilization.

#### 9) pingResultsProbeResponses
- Represents the number of probe responses received.

#### 10) pingResultsSentProbes
- Represents the number of sent probes.

#### 11) pingResultsRttSumOfSquares
- Represents the sum of squares of round-trip times.

#### 12) pingResultsLastGoodProbe
- Represents the last good probe in the series.

#### 13) jnxPingResultsStddevEgressUs
- Represents the standard deviation of the egress utilization.

#### 14) jnxPingCtlEgressStdDevThreshold
- Represents the threshold for egress utilization.
<table>
<thead>
<tr>
<th>jnxPingEgressJitterThreshold Exceeded</th>
<th>1.3.6.1.4.1.2636.4.9.0.6</th>
<th>Remote operations</th>
<th>All Junos OS devices except EX and high-end SRX Series devices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1) pingCtlTargetAddressType-1.3.6.1.2.1.80.1.2.1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) pingCtlTargetAddress-1.3.6.1.2.1.80.1.2.1.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3) pingResultsOperStatus-1.3.6.1.2.1.80.1.3.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4) pingResultsIpTargetAddressType-1.3.6.1.2.1.80.1.3.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5) jnxPingResultsMinEgressUs-1.3.6.1.4.1.2636.3.7.1.3.1.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6) jnxPingResultsMaxEgressUs-1.3.6.1.4.1.2636.3.7.1.3.1.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7) jnxPingResultsAvgEgressUs-1.3.6.1.4.1.2636.3.7.1.3.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8) jnxPingResultsProbeResponses-1.3.6.1.2.1.80.1.3.1.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9) jnxPingResultsSentProbes-1.3.6.1.2.1.80.1.3.1.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10) jnxPingResultsRttSumOfSquares-1.3.6.1.2.1.80.1.3.1.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11) jnxPingCtlEgressJitterThreshold-1.3.6.1.4.1.2636.3.7.1.2.1.2</td>
</tr>
<tr>
<td>jnxPingIngressThreshold Exceeded</td>
<td>1.3.6.1.4.1.2636.4.9.0.7</td>
<td>Remote operations</td>
<td>All Junos OS devices except EX and high-end SRX Series devices</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1) pingCtlTargetAddressType-1.3.6.1.2.1.80.1.2.1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) pingCtlTargetAddress-1.3.6.1.2.1.80.1.2.1.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3) pingResultsOperStatus-1.3.6.1.2.1.80.1.3.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4) pingResultsIpTargetAddressType-1.3.6.1.2.1.80.1.3.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5) pingResultsIpTargetAddress-1.3.6.1.2.1.80.1.3.1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6) jnxPingResultsMinIngressUs-1.3.6.1.4.1.2636.3.7.1.3.1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7) jnxPingResultsMaxIngressUs-1.3.6.1.4.1.2636.3.7.1.3.1.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8) jnxPingResultsAvgIngressUs-1.3.6.1.4.1.2636.3.7.1.3.1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9) pingResultsProbeResponses-1.3.6.1.2.1.80.1.3.1.7</td>
</tr>
<tr>
<td>jnxPingIngressStddevThresholdExceeded</td>
<td>Remote operations</td>
<td>All Junos OS devices except EX and high-end SRX Series devices</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| jnxPingIngressStddevThresholdExceeded       | 1.3.6.1.4.1.2636.4.9.0.8 | 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 - 1.3.6.1.4.1.2636.3.7.1.3.1.1  
7) jnxPingResultsMaxIngressUs - 1.3.6.1.4.1.2636.3.7.1.3.1.1  
8) jnxPingResultsAvgIngressUs - 1.3.6.1.4.1.2636.3.7.1.3.1.1  
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 - 1.3.6.1.4.1.2636.3.7.1.2.1.1  
14) jnxPingResultsIngressUs - 1.3.6.1.4.1.2636.3.7.1.3.1.1  
15) jnxPingResultsStddevIngressUs - 1.3.6.1.4.1.2636.3.7.1.3.1.1  
16) jnxPingCtlIngressStddevThreshold - 1.3.6.1.4.1.2636.3.7.1.2.1.1  
17) jnxPingResultsSentProbes - 1.3.6.1.2.1.80.1.3.1.8  
18) pingResultsRttSumOfSquares - 1.3.6.1.2.1.80.1.3.1.9  
19) pingResultsLastGoodProbe - 1.3.6.1.2.1.80.1.3.1.10  |
<table>
<thead>
<tr>
<th>MIB Object</th>
<th>OIDs</th>
<th>Remote Operations</th>
<th>All Junos OS Devices except EX and high-end SRX Series Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>jnxPingIngressJitterThresholdExceeded</td>
<td>1.3.6.1.4.1.2636.4.9.0.9</td>
<td>Remote operations</td>
<td>1) pingCtlTargetAddressType - 1.3.6.1.2.1.80.1.2.1.1.3 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 - 1.3.6.1.4.1.2636.3.7.1.3.1.1 3 7) jnxPingResultsMaxIngressUs - 1.3.6.1.4.1.2636.3.7.1.3.1.1 4 8) jnxPingResultsAvgIngressUs - 1.3.6.1.4.1.2636.3.7.1.3.1.1 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) jnxPingCtlIngressJitterThreshold - 1.3.6.1.4.1.2636.3.7.1.2.1.1.5</td>
</tr>
<tr>
<td>jnxAccessAuthServiceUp</td>
<td>1.3.6.1.4.1.2636.3.5.1.1.0.1</td>
<td>Routing</td>
<td>J Series and SRX Series</td>
</tr>
<tr>
<td>jnxAccessAuthServiceDown</td>
<td>1.3.6.1.4.1.2636.3.5.1.1.0.2</td>
<td>Routing</td>
<td>J Series and SRX Series</td>
</tr>
<tr>
<td>jnxAccessAuthServerDisabled</td>
<td>1.3.6.1.4.1.2636.3.5.1.1.0.3</td>
<td>Routing</td>
<td>J Series and SRX Series</td>
</tr>
<tr>
<td>jnxAccessAuthServerEnabled</td>
<td>1.3.6.1.4.1.2636.3.5.1.1.0.4</td>
<td>Routing</td>
<td>J Series and SRX Series</td>
</tr>
<tr>
<td>jnxJsFwAuthTokenFailure</td>
<td>1.3.6.1.4.1.2636.3.3.9.1.2.1.0.1</td>
<td>Routing</td>
<td>J Series and SRX Series</td>
</tr>
</tbody>
</table>

**Note:** The table above lists MIB objects and their associated OIDs, along with the remote operations and all Junos OS devices except EX and high-end SRX Series devices.
<table>
<thead>
<tr>
<th>jnxJsFwAuthServiceUp</th>
<th>1.3.6.1.4.1.2636.3.3 9.1.2.1.0.2</th>
<th>Routing</th>
<th>J Series and SRX Series</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>jnxJsFwAuthServiceDown</td>
<td>1.3.6.1.4.1.2636.3.3 9.1.2.1.0.3</td>
<td>Routing</td>
<td>J Series and SRX Series</td>
<td>None</td>
</tr>
<tr>
<td>jnxJsFwAuthCapacityExceeded</td>
<td>1.3.6.1.4.1.2636.3.3 9.1.2.1.0.4</td>
<td>Routing</td>
<td>J Series and SRX Series</td>
<td>None</td>
</tr>
<tr>
<td>jnxJsScreenAttack</td>
<td>1.3.6.1.4.1.2636.3.3 9.1.8.1.0.1</td>
<td>Routing</td>
<td>J Series and SRX Series</td>
<td>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</td>
</tr>
<tr>
<td>jnxJsScreenCfgChange</td>
<td>1.3.6.1.4.1.2636.3.3 9.1.8.1.0.2</td>
<td>Routing</td>
<td>J Series and SRX Series</td>
<td>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</td>
</tr>
<tr>
<td>jnxRmonAlarmGetFailure</td>
<td>1.3.6.1.4.1.2636.4.3. 0.1</td>
<td>RMON alarm</td>
<td>All Junos OS devices</td>
<td>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</td>
</tr>
<tr>
<td>jnxRmonGetOk</td>
<td>1.3.6.1.4.1.2636.4.3. 0.2</td>
<td>RMON alarm</td>
<td>All Junos OS devices</td>
<td>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</td>
</tr>
<tr>
<td>jnxSyslogTrap</td>
<td>.1.3.6.1.4.1.2636.4.1.2.0.1</td>
<td>Services</td>
<td>All Junos OS devices</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------</td>
<td>----------</td>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1) jnxSyslogEventName - .1.3.6.1.4.1.2636.3.35.1.1.1.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) jnxSyslogTimestamp- .1.3.6.1.4.1.2636.3.35.1.1.1.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3) jnxSyslogSeverity- .1.3.6.1.4.1.2636.3.35.1.1.1.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4) jnxSyslogFacility- .1.3.6.1.4.1.2636.3.35.1.1.1.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5) jnxSyslogProcessId- .1.3.6.1.4.1.2636.3.35.1.1.1.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6) jnxSyslogProcessName- .1.3.6.1.4.1.2636.3.35.1.1.1.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7) jnxSyslogHostName- .1.3.6.1.4.1.2636.3.35.1.1.1.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8) jnxSyslogMessage- .1.3.6.1.4.1.2636.3.35.1.1.1.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9) Apart from the jnxSyslogTrap objects, this notification can include one or more attribute-value pairs. The attribute-value pairs shall be identified by objects jnxSyslogAvAttribute and jnxSyslogAvValue.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>jnxEventTrap</th>
<th>.1.3.6.1.4.1.2636.4.1.3.0.1</th>
<th>Services</th>
<th>All Junos OS devices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1) jnxEventTrapDescr - .1.3.6.1.4.1.2636.3.37.1.1.0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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. See sample in the Sylog and event trap samples section.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>jnxJsAvPatternUpdateTrap</th>
<th>.1.3.6.1.4.1.2636.3.3.9.1.13.1.0.1</th>
<th>Configuration</th>
<th>J Series and SRX Series</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1) jnxAVPatternVersionString- .1.3.6.1.4.1.2636.3.39.1.13.1.3.1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) jnxAVPatternTimestamp- .1.3.6.1.4.1.2636.3.39.1.13.1.3.2.0</td>
</tr>
<tr>
<td>Metric</td>
<td>OID</td>
<td>Description</td>
<td>SRX 5600, SRX 5800, and J Series</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------</td>
<td>------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>jnxJsChassisClusterSwitchover</td>
<td>1.3.6.1.4.1.2636.3.3</td>
<td>Chassis (alarm conditions)</td>
<td>1) jnxJsChClusterSwitchoverInf</td>
</tr>
<tr>
<td></td>
<td>9.1.14.1.0.1</td>
<td></td>
<td>oRedundancyGroup-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.3.39.1.14.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) jnxJsChClusterSwitchoverInf</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>oClusterId-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.3.39.1.14.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.2.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3) jnxJsChClusterSwitchoverInf</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>oNodeId-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.3.39.1.14.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4) jnxJsChClusterSwitchoverInf</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>oPreviousState-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.3.39.1.14.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.4.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5) jnxJsChClusterSwitchoverInf</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>oCurrentState-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.3.39.1.14.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.5.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6) jnxJsChClusterSwitchoverInf</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>oReason-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.3.39.1.14.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.6.0</td>
</tr>
<tr>
<td>bfdSessUp</td>
<td>1.3.6.1.4.1.2636.5.3</td>
<td>Routing</td>
<td>All Junos OS devices</td>
</tr>
<tr>
<td></td>
<td>.1.0.1</td>
<td></td>
<td>1) bfdSessDiag, -- low range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>value-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.5.3.1.1.2.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) bfdSessDiag -- high range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>value-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.5.3.1.1.2.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.8</td>
</tr>
<tr>
<td>bfdSessDown</td>
<td>1.3.6.1.4.1.2636.5.3</td>
<td>Routing</td>
<td>All Junos OS devices</td>
</tr>
<tr>
<td></td>
<td>.1.0.2</td>
<td></td>
<td>1) bfdSessDiag, -- low range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>value-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.5.3.1.1.2.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) bfdSessDiag -- high range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>value-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.5.3.1.1.2.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.8</td>
</tr>
<tr>
<td>jnxBfdSessTxIntervalHigh</td>
<td>1.3.6.1.4.1.2636.3.4</td>
<td>Routing</td>
<td>All Junos OS devices</td>
</tr>
<tr>
<td></td>
<td>5.1.0.1</td>
<td></td>
<td>1) jnxBfdSessThreshTxInterval-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.3.45.1.1.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) jnxBfdSessCurrTxInterval-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.3.45.1.1.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.2</td>
</tr>
<tr>
<td>jnxBfdSessDetectionTimeHigh</td>
<td>1.3.6.1.4.1.2636.3.4</td>
<td>Routing</td>
<td>All Junos OS devices</td>
</tr>
<tr>
<td></td>
<td>5.1.0.2</td>
<td></td>
<td>1) jnxBfdSessThreshDectTime-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.3.45.1.1.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) jnxBfdSessCurrDectTime-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.3.6.1.4.1.2636.3.45.1.1.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1.4</td>
</tr>
</tbody>
</table>
### Best Practices for SRX Cluster Management

<table>
<thead>
<tr>
<th>jnxBgpM2Established</th>
<th>1.3.6.1.4.1.2636.5.1.1.0.1</th>
<th>Routing</th>
<th>All Junos OS devices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1) jnxBgpM2PeerLocalAddrType-1.3.6.1.4.1.2636.5.1.1.2.1.1.1.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) jnxBgpM2PeerLocalAddr-1.3.6.1.4.1.2636.5.1.1.2.1.1.1.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3) jnxBgpM2PeerRemoteAddrType-1.3.6.1.4.1.2636.5.1.1.2.1.1.1.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4) jnxBgpM2PeerRemoteAddr-1.3.6.1.4.1.2636.5.1.1.2.1.1.1.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5) jnxBgpM2PeerLastErrorReceived-1.3.6.1.4.1.2636.5.1.1.2.2.1.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6) jnxBgpM2PeerState-1.3.6.1.4.1.2636.5.1.1.2.1.1.1.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>jnxBgpM2BackwardTransition</th>
<th>1.3.6.1.4.1.2636.5.1.1.0.2</th>
<th>Routing</th>
<th>All Junos OS devices</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1) jnxBgpM2PeerLocalAddrType-1.3.6.1.4.1.2636.5.1.1.2.1.1.1.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2) jnxBgpM2PeerLocalAddr-1.3.6.1.4.1.2636.5.1.1.2.1.1.1.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3) jnxBgpM2PeerRemoteAddrType-1.3.6.1.4.1.2636.5.1.1.2.1.1.1.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4) jnxBgpM2PeerRemoteAddr-1.3.6.1.4.1.2636.5.1.1.2.1.1.1.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5) jnxBgpM2PeerLastErrorReceived-1.3.6.1.4.1.2636.5.1.1.2.2.1.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6) jnxBgpM2PeerLastErrorReceivedText-1.3.6.1.4.1.2636.5.1.1.2.2.1.1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7) jnxBgpM2PeerState-1.3.6.1.4.1.2636.5.1.1.2.1.1.1.2</td>
</tr>
</tbody>
</table>

Note that if the fxp0 interface goes down on the backup Routing Engine, it does not send any traps. The system logging feature (syslog) 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 as previously mentioned. You can configure the system to send specific syslog messages to the external syslog servers based on regular expressions or severity. See the *Junos OS System Log Messages Reference* to check for syslogs you are interested in.
A SYSLOG event can be converted to an SNMP trap using an event policy. Event policies instruct the eventd process to select specific events, correlate the events, and perform a set of actions.

The following sample shows the jnxSyslog trap configuration for a ui-commit_progress (configuration commit in progress) event.

**Configuration**

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

```
JUNIPER NETWORKS

© Juniper Networks, Inc.
67

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] <<< Community: petblr
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] <<< 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] <<< 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 45 43
Jul 6 13:31:21 snmpd[0] <<< 53 53
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 00 00
Jul 6 13:31:21 snmpd[0] <<< 2b 00 00
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] <<< OID : jnxSyslogFacility.83
Jul 6 13:31:21 snmpd[0] <<< type : Number
Jul 6 13:31:21 snmpd[0] <<< value: 24
```
Detecting a Failover or Switchover

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:

- The cluster ID and node ID
- The reason for the failover
- The redundancy group that is involved in the failover
- The redundancy group’s previous state and current state

These are the different states that a cluster can be in 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 any event, 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 that 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

```
regress@SRX5600-1> 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

<table>
<thead>
<tr>
<th>Time</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jul 6 05:14:57</td>
<td>V2 Trap</td>
<td>Source: 192.168.29.2</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>Destination</td>
<td>10.204.132.188</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>Version</td>
<td>SNMPv2</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>Community</td>
<td>test</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>OID</td>
<td>sysUpTime.0</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>type</td>
<td>TimeTicks</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>value</td>
<td>(754507) 2:05:45.07</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>OID</td>
<td>snmpTrapOID.0</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>type</td>
<td>Object</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>value</td>
<td>jnxJsChassisClusterSwitchover</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>OID</td>
<td>jnxJsChClusterSwitchoverInfoRedundancyGroup.0</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>type</td>
<td>OctetString</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>value</td>
<td>&quot;1&quot;</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>HEX</td>
<td>31</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>OID</td>
<td>jnxJsChClusterSwitchoverInfoClusterId.0</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>type</td>
<td>OctetString</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>value</td>
<td>&quot;12&quot;</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>HEX</td>
<td>31 32</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>OID</td>
<td>jnxJsChClusterSwitchoverInfoNodeId.0</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>type</td>
<td>OctetString</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>value</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>HEX</td>
<td>30</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>OID</td>
<td>jnxJsChClusterSwitchoverInfoPreviousState.0</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>type</td>
<td>OctetString</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>value</td>
<td>&quot;primary&quot;</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>HEX</td>
<td>70 72 69 6d 61 72 79</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>OID</td>
<td>jnxJsChClusterSwitchoverInfoCurrentState.0</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>type</td>
<td>OctetString</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>value</td>
<td>&quot;secondary-hold&quot;</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>HEX</td>
<td>73 65 63 6f 6e 64 61 72</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>OID</td>
<td>jnxJsChClusterSwitchoverInfoReason.0</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>type</td>
<td>OctetString</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>value</td>
<td>&quot;manual failover&quot;</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>HEX</td>
<td>6d 61 6e 75 61 6c 20 66</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>OID</td>
<td>snmpTrapEnterprise.0</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>type</td>
<td>Object</td>
</tr>
<tr>
<td>Jul 6 05:14:57</td>
<td>value</td>
<td>jnxProductNameSRX5600</td>
</tr>
</tbody>
</table>
Automatic Failover Trap

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

Using Operational and Event Scripts for Management and Monitoring
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. They are simply the op scripts triggered by event policies.

The following shows an example of a jnx event trap. In the example, the ev-syslog-trap event script raises a jnxEvent trap whenever an alarm is triggered on the device.

Configuration

```bash
regress@SRX-3400-1 > show configuration event-options policy trap
{
    events SYSTEM;
    attributes-match {
        SYSTEM.message matches "Alarm set";
    }
    then {
        event-script ev-syslog-trap.slax {
            arguments {
                event SYSTEM;
                message "($$.message)";
            }
        }
    }
}
```
SCRIPT

See APPENDIX C: Event Script for information about generating the `jnxEventTrap` SNMP traps and to see the script used to generate the trap shown in the following example.

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

```plaintext
 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 74 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 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"
Utility MIB for Monitoring

The Juniper Networks jnx-utility MIB 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 via the jcs:invoke RPC API.

One of the examples we use for reading power on the device, which is not available using SNMP, is the jnxUtility MIB. For example, the following command gives the power readings on SRX Series devices. With a simple event script, you can read the power output every minute and populate the jnx-utility MIB. Similarly, you can write op scripts or event scripts that can populate a variety of data of different types. See APPENDIX D: Utility MIB example for sample scripts and usage of the utility MIB.

SRX Log Messages

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, you can use the show security flow session command:

- To display a list of incoming and outgoing IP flows, including services
- To show the security attributes associated with a flow, for example, the policies that apply to traffic belonging to that flow
- To display the session timeout value, when the session became active, for 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. Both the traditional syslog, and the structured syslog are supported. The SRX3000 and SRX5000 lines support 1000 log messages per second, and the management station needs to be able 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. Additional care should be taken so 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 recommend logging messages to the control plane, except under certain circumstances, which are discussed later in this document.

For SRX Series branch devices running Junos OS Release 9.6 and later and high-end SRX 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 via 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 both at the same time.

**High-End SRX Series Devices Data Plane Logging to Syslog Servers**

This section discusses various configurations for the high-end SRX Series devices for different deployment models.

High-end SRX Series devices must log messages directly to syslog if the logging rate is above 1,000 log messages per second. Today, only UDP is supported for syslog due to the overhead associated with TCP and the extremely high rate of log messages that can be generated on the SRX Series devices. Data plane log messages cannot be sent out the fxp0 interface because that would require the data plane to send log messages through the control plane. The one exception would be if event logging was configured to log all data plane log messages to the control plane.

**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 would configure standard syslog.

   ```
   user@SRX# 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.

   ```
   user@SRX# set security log source-address <ip-address>
   ```

3. Configuring 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@SRX# set security log stream <name> host <ip-address>
   ```

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

   ```
   user@SRX# 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 dataplane, 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.
   ```bash
   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 dataplane 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 1,000 log messages per second to the control plane.
   ```bash
   set security log mode event event-rate <logs per second>
   ```

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

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

**Configuration**

```bash
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-
```
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 logs, the control plane and logs 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 config 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 (via 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 logs and logs sent to the Routing Engine are sent to a syslog server. The following is a sample configuration to enable this type of logging.

**Configuration**

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 like 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 section offers an overview on 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@SRX# 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@SRX# set system syslog host <syslog server> source-address <source-address>
```

3. Configure regular expression matching (optional).

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@SRX# set system syslog host <syslog server> <facility> <severity> match IDP
```

Branch SRX Series Device Logging

You can configure the SRX Series device to send only traffic logs to the syslog server via 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.

```
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
```
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.

```
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 Dataplane 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 1,000 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 dataplane 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 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 routing-instances Logging instance-type virtual-router.
set routing-instances Logging interface ge-0/0/7.0.
```
**Note:** AppA is now able to manage the ge-0/0/7 interface since AppA is managing 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 via SNMP.

**Application Tracking**

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.

See the *Junos OS Security Configuration Guide* for configuring application identification and tracking. Management stations can subscribe for the log messages for application tracking. An SRX Series device can support a high volume of these log messages (minimum 1000 log messages/sec). 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.

**Summary**

This document has provided best practices for managing SRX Series chassis clusters using SNMP, NETCONF XML management protocol, Junos OS XML Management Protocol, and other instrumentation provided by Junos OS. It can be used to effectively manage the SRX Series chassis clusters.
APPENDIX A: XML RPC Sample Response
This appendix provides details related to managing SRX Series chassis clusters using remote procedure calls (RPCs).

Chassis Inventory Sample RPC Response

```
<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>
        <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>
        </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>
        </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>
        </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>
        </chassis-module>
        <chassis-module>
          <name>Routing Engine 0</name>
          <version>REV 03</version>
          <part-number>740-023530</part-number>
          <serial-number>9009008946</serial-number>
```
```
Best Practices for SRX Cluster Management

<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>
</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>
</chassis-module>
<chassis-module>
<name>CPU</name>
<version>REV 02</version>
<part-number>710-024633</part-number>
<serial-number>WY0854</serial-number>
<description>SRX5k DPC FMB</description>
</chassis-module>
<chassis-module>
<name>PIC 0</name>
<part-number>BUILTIN</part-number>
<serial-number>BUILTIN</serial-number>
<description>SPU Cp-Flow</description>
</chassis-module>
<chassis-module>
<name>PIC 1</name>
<part-number>BUILTIN</part-number>
<serial-number>BUILTIN</serial-number>
<description>SPU Flow</description>
</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-module>
<chassis-module>
<name>CPU</name>
<version>REV 03</version>
<part-number>710-024633</part-number>
<serial-number>XH8755</serial-number>
<description>SRX5k DPC FMB</description>
</chassis-module>
<chassis-module>
<name>PIC 0</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>

<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>
Best Practices for SRX Cluster Management

JUNIPER NETWORKS

© Juniper Networks, Inc.
<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>
APPENDIX B: Sample SNMP Responses

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

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

```
root@SRX3400-1> 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/*

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
```
Redundant Ethernet Interface Statistics in MIB II Tables

```
rout@SRX3400-1> 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}
rout@SRX3400-1> 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
```
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
APPENDIX C: Event Script for Generating jnxEventTrap SNMP Traps

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 utility 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-generate-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;
    }
}
### APPENDIX D: Utility MIB Examples

This appendix 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

```
regress@PE3> 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

```
router> 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

```bash
host> show configuration event-options
  generate-event {
      1-min time-interval 60;
  }
policy powerUtil {
    events 1-min;
    then {
      event-script power.slax;
    }
}
```

General Disclaimer

Although Juniper Networks has attempted to provide accurate information in this guide, Juniper Networks does not warrant or guarantee the accuracy of the information provided herein. Juniper Networks may change the programs or products mentioned at any time without prior notice. Mention of non-Juniper Networks products or services is for information purposes only and constitutes neither an endorsement nor a recommendation of such products or services or of any company that develops or sells such products or services.

ALL INFORMATION PROVIDED IN THIS GUIDE IS PROVIDED “AS IS”, WITH ALL FAULTS, AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED OR STATUTORY. JUNIPER NETWORKS AND ITS SUPPLIERS HEREBY DISCLAIM ALL WARRANTIES RELATED TO THIS GUIDE AND THE INFORMATION CONTAINED HEREIN, WHETHER EXPRESSED OR IMPLIED OR STATUTORY INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT, OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

JUNIPER NETWORKS AND ITS SUPPLIERS SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR REVENUES, COSTS OF REPLACEMENT GOODS OR SERVICES, LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OF THE GUIDE OR ANY JUNIPER NETWORKS PRODUCT OR SERVICE, OR DAMAGES RESULTING FROM USE OF OR RELIANCE ON THE INFORMATION PROVIDED IN THIS GUIDE, EVEN IF JUNIPER NETWORKS OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Many of the Juniper Networks products and services identified in this guide are provided with, and subject to, written software licenses and limited warranties. Those licenses and warranties provide the purchasers of those products with certain rights. Nothing in this guide shall be deemed to expand, alter, or modify any warranty or license or any other agreement provided by Juniper Networks with any Juniper Networks product, or to create any new or additional warranties or licenses.