Network Configuration Example

Configuring Chassis Clusters on Branch SRX Series Services Gateways

Published: 2014-01-24

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

Network Configuration Example Configuring Chassis Clusters on Branch SRX Series Services Gateways
NCE0110

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

YEAR 2000 NOTICE

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

END USER LICENSE AGREEMENT

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

Table of Contents

Introduction 1
Chassis Clustering on Branch SRX Series Services Gateways Overview 1
Understanding Chassis Clustering on Branch SRX Series Services Gateways 2
Feature Description 2
Redundant Ethernet Interfaces 6
Link Aggregation Interfaces and LACP 7
Real-Time Performance Monitoring 7
IP Monitoring 7
Feature Support and Comparison Matrix 7
Example: Configuring Chassis Clusters on an SRX Services Gateway for the Branch 8
Example: Configuring an Active/Passive Chassis Cluster 19
Example: Configuring an Active/Passive Chassis Cluster with Asymmetric Routing 32
Configuring an Active/Active Full Mesh Chassis Cluster 38
Upgrading a Chassis Cluster 41
In-Band Management of Chassis Clusters 41
Understanding the Chassis Cluster High Availability Model 41
Managing a Chassis Cluster 43
Connecting to a Cluster Using SSH/Telnet 43
Using In-Band Management Through Network and Security Manager 45
Using SNMP to Manage a Chassis Cluster 47
Upgrading the Software Image on a Chassis Cluster 49

Introduction

This document reviews the high availability chassis clustering feature, together with its limitations and design considerations. It also discusses some common use cases and how they relate to the Juniper Networks® ScreenOS® Software NetScreen Redundancy Protocol (NSRP) counterparts.

Chassis Clustering on Branch SRX Series Services Gateways Overview

Modern networks require high availability. To accommodate this requirement, the Juniper Networks SRX Series Services Gateways and J Series Services Routers can be configured to operate in cluster mode, where a pair of devices can be connected and configured to operate like a single node, providing device, interface, and service-level redundancy. Starting with the 9.0 release of the Juniper Networks Junos® operating system (Junos OS), J Series routers and SRX Series devices can be deployed using the chassis cluster feature to provide high availability. For the J Series, this feature is only available with the flow-enabled version of Junos OS. With the introduction of the SRX Series services gateways for the branch in Junos OS Release 9.5, high availability is supported on branch SRX Series devices.

High availability between devices is easily incorporated into enterprise designs and is particularly relevant when architecting branch and remote site links to larger corporate offices. By leveraging the high availability feature, enterprises can ensure connectivity in the event of device or link failure.

Chassis clustering is supported on identical pairs of:

- Juniper Networks J2320 Services Routers, J2350 Services Routers, J4350 Services Routers, or J6350 Services Routers running flow-enabled Junos OS release 9.0 or later for J Series routers.
- SRX100 Services Gateways, SRX210 Services Gateways, SRX220 Services Gateways, SRX240 Services Gateways, or SRX650 Services Gateways running Junos OS Release 9.5 or later for SRX Series Services Gateways.

Chassis clustering is a simple feature to implement that ensures reliable enterprise connectivity between branch sites and corporate headquarters or regional offices. It provides stateful traffic failover between two Juniper Networks security devices while maintaining the abstraction of a single device, which simplifies network design. The feature has been carefully designed to address many common connectivity challenges such as asymmetric traffic, VPNs, and mixed LAN/WAN environments. Juniper Networks SRX Series for the branch and J Series Services Routers employing chassis cluster provide a foundation for reliable and high-performance network deployments.

Related Documentation

- Example: Configuring Chassis Clusters on an SRX Services Gateway for the Branch on page 8
- Example: Configuring an Active/Passive Chassis Cluster on page 19
- Example: Configuring an Active/Passive Chassis Cluster with Asymmetric Routing on page 32

- Configuring an Active/Active Full Mesh Chassis Cluster on page 38

Understanding Chassis Clustering on Branch SRX Series Services Gateways

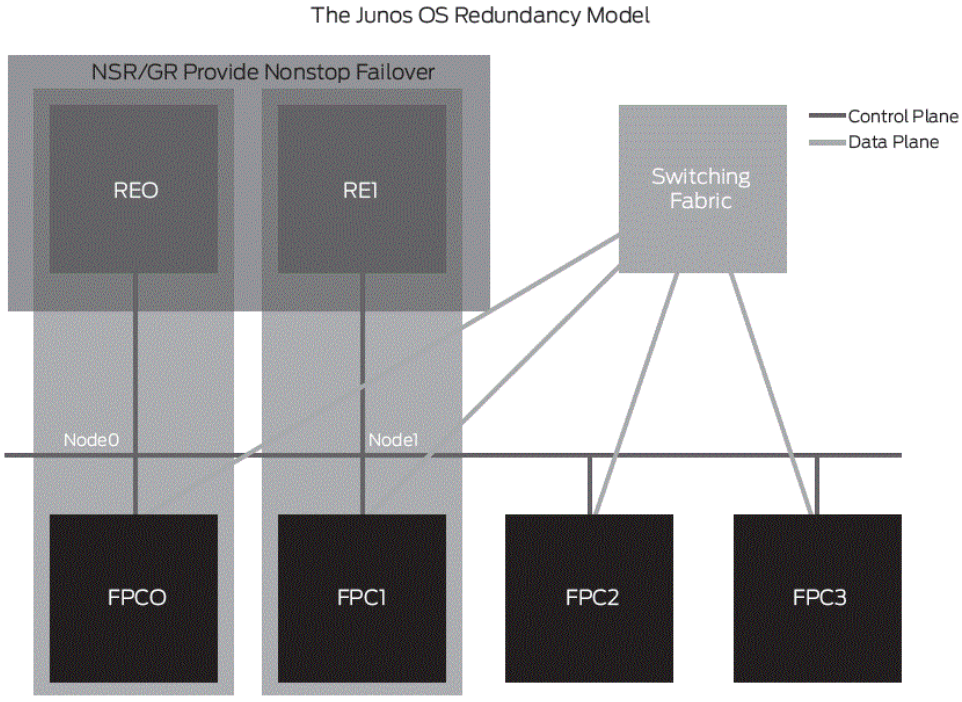
Chassis clustering between devices can be deployed in either active/passive or active/active scenarios. Junos OS allows a high availability cluster to additionally be used in asymmetric routing scenarios.

- Feature Description on page 2
- Redundant Ethernet Interfaces on page 6
- Link Aggregation Interfaces and LACP on page 7
- Real-Time Performance Monitoring on page 7
- IP Monitoring on page 7
- Feature Support and Comparison Matrix on page 7

Feature Description

The high availability feature is modeled after redundancy features first introduced in Juniper Networks M Series Multiservice Edge Routers and T Series Core Routers. This topic first gives a brief overview of the way Junos OS redundancy works, so that you can better understand how this model is applied when clustering devices. Because Junos OS is designed with separate control and data planes, redundancy must operate in both. The control plane in Junos OS is managed by Routing Engines, which perform all the routing and forwarding computations (among many other functions). Once the control plane converges, forwarding entries are pushed to all Packet Forwarding Engines, which are virtualized on J Series routers. Packet Forwarding Engines then perform route-based lookups to determine the appropriate destination for each packet independent of the Routing Engines. This simplistic view of the Junos OS forwarding paradigm is represented in Figure 1 on page 3.

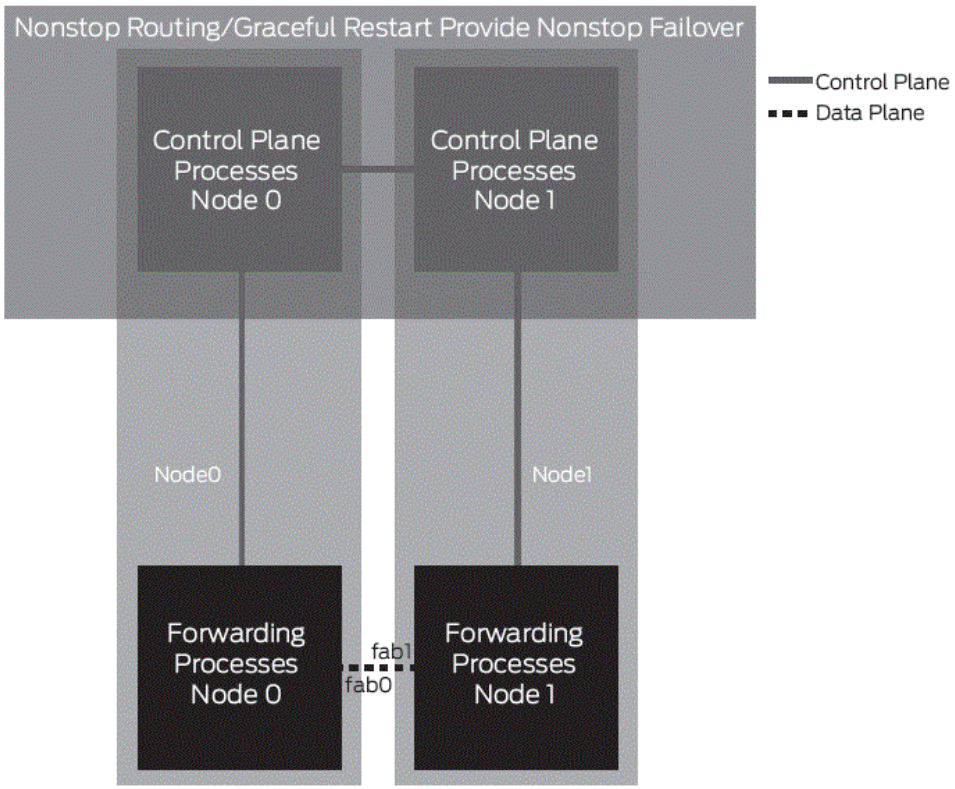
Figure 1: Junos OS Redundancy Model



Control plane failover is provided in Junos OS by using graceful restart or nonstop active routing (NSR). In the former, the router signals a control plane failure to the rest of the network, while continuing to forward traffic on the data plane (since a control plane failure does not affect the forwarding plane). The rest of the network continues to use the restarting router (for a grace period), while the restarting router forms new adjacencies. The backup Routing Engine in this scenario detects the entire configuration, but not the runtime state of the control plane. In a failure, the backup Routing Engine has to recalculate all routing/forwarding tables.

Nonstop active routing leverages state replication between Routing Engines. In this case, a restarting router handles control plane failures transparently, as the backup Routing Engine takes control of the router without any assistance from the rest of the network. Routing protocols handle data plane failures, while interface, Packet Forwarding Engine, or Flexible PIC Concentrator failovers are handled by diverting traffic through other interfaces, which can be achieved by using conventional routing protocols, the Virtual Router Redundancy Protocol (VRRP), or aggregate interfaces. When enabling a chassis cluster for J Series routers, Junos OS uses a similar model, without support for nonstop active routing state replication, to provide control plane redundancy as shown in Figure 2 on page 4.

Figure 2: Device Clustering



The chassis clustering feature supports clustering of two devices and requires two connections between the devices as previously illustrated. The chassis cluster is seen as a single device by both external devices and administrators of the cluster. When clustering is enabled, node1 of the cluster renumbers its interfaces to avoid collisions with node0. Depending on the model used (only two devices of the same model can be clustered), node1 renumbers its interfaces by adding the total number of system Flexible PIC Concentrators to the original Flexible PIC Concentrator number of the interface. (On a J Series router, the onboard ports and each Physical Interface Module (PIM) slot correspond to a Flexible PIC Concentrator.) Accordingly, when clustering two J2320 routers, node1 renumbers its interfaces as ge-4/0/0 to ge-7/0/0, because a J2320 has three PIM slots and four standard Gigabit Ethernet ports on the system board acting as FPC0.

Depending on the device used, node1 renumbers its interfaces by adding the total number of system Flexible PIC Concentrators to the original Flexible PIC Concentrator number of the interface. See Table 1 on page 4 for interface renumbering on the SRX Series device.

Table 1: SRX Series Services Gateways Interface Renumbering

SRX Series Services Gateway	Control Link Name	Renumbering Constant	Node0 Interface Name	Node1 Interface Name
SRX100	**fe-0/0/7**	1	**fe-0/0/0**	**fe-1/0/0**

Table 1: SRX Series Services Gateways Interface Renumbering *(continued)*

SRX Series Services Gateway	Control Link Name	Renumbering Constant	Node0 Interface Name	Node1 Interface Name
SRX210	**fe-0/0/7**	2	**ge-0/0/0**	**ge-2/0/0**
SRX220	**ge-0/0/7**	3	**ge-0/0/0**	**ge-3/0/0**
SRX240	**ge-0/0/1**	5	**ge-0/0/0**	**ge-5/0/0**
SRX550	**ge-0/0/1**	9	**ge-0/0/0**	**ge-9/0/0**
SRX650	**ge-0/0/1**	9	**ge-0/0/0**	**ge-9/0/0**

After clustering is enabled, the system creates fxp0, fxp1, and fabric interfaces. Depending on the platform, fxp0 and fxp1 are mapped to a physical interface. This is not user configurable. The fabric interface is user configurable.

However, the fabric interface is user defined. See Table 2 on page 5 for mapping of the fxp0 and fxp1 interfaces on the SRX Series devices.

Table 2: SRX Series Services Gateways fxp0 and fxp1 Interfaces Mapping

SRX Series Services Gateway	fxp0 Interface	fxp1 Interface	fabric Interface
SRX100	**fe-0/0/6**	**fe-0/0/7**	user defined
SRX210	**fe-0/0/6**	**fe-0/0/7**	user defined
SRX220	**fe-0/0/6**	**fe-0/0/7**	user defined
SRX240	**ge-0/0/0**	**ge-0/0/1**	user defined
SRX550	**ge-0/0/0**	**ge-0/0/1**	user defined
SRX650	**ge-0/0/0**	**ge-0/0/1**	user defined

As seen in Figure 2 on page 4, fxp1 (the high availability link) provides control plane communication between the nodes in the cluster, and the fxp0 interface provides management access. The fxp0 interface is limited to host traffic only. Traffic received through the fxp0 interface is not forwarded to any other interface in the system. Fabric interfaces are used to exchange data plane information and traffic between devices. As opposed to the fxp0 and fxp1 interfaces, the fabric interface can be mapped to any Ethernet interface in the system.

The control plane redundancy of the cluster is similar to that used within single M Series and T Series routers. Each device acts as a Routing Engine in a system with redundant Routing Engines. Graceful restart is used to provide control plane failover with minimal traffic impact on the network. The control plane redundancy model is active/passive,

where a node in the cluster is designated as the active device and performs all cluster routing calculations. Except for a few key processes required for managing clustering, most of the processes are running only on the primary Routing Engine. When the primary node fails, the routing process and other processes in the backup device become active and assume control plane operations.

Data plane redundancy is somewhat more involved. Juniper Networks M Series and T Series routers perform traffic forwarding on a packet by packet basis. There is no concept of flow, and each Packet Forwarding Engine maintains a copy of the forwarding table that was distributed by the active Routing Engine. The forwarding table allows each Packet Forwarding Engine to perform traffic forwarding independent of other system Packet Forwarding Engines. If a Packet Forwarding Engine fails, the rest of the Packet Forwarding Engines in the system are unaffected, allowing the control plane to reroute the traffic to a working Packet Forwarding Engine.

In contrast, J Series routers and the SRX Series devices inspect all traffic and keep a table of all active sessions. Whenever a new connection is allowed through the system, the device makes note of the 5-tuple that identifies a particular connection (source and destination IP addresses, source and destination ports as applicable, and protocol) and updates the table with session details such as next hop, session timeouts, sequence numbers (if the protocol is TCP), and other session-specific information required to guarantee that no packets are forwarded from unknown or undesired protocols (or users). Session information is updated as traffic traverses the device and is required on both devices in a cluster to guarantee that established sessions are not dropped when a failover occurs.

The control plane Routing Engines function in active/backup mode while the data plane (Packet Forwarding Engines) function in active/active mode. With active/active Packet Forwarding Engines, it is possible for traffic to ingress the cluster on one node and egress from the other node, which means that both nodes must be able to create and synchronize sessions. For example, when return traffic arrives asymmetrically at the node that did not record the initial session, the chassis cluster feature gracefully forwards the traffic to the original node for processing, which prevents security features from being compromised. Be aware that the previous discussion applies only to routed traffic.

Redundant Ethernet Interfaces

As previously discussed, control plane failures are detected by member nodes, causing the backup node to take control of the cluster. Conversely, data plane failures rely on routing protocols to reroute traffic or redundant Ethernet interfaces to overcome interface failures. The concept of redundant Ethernet is fairly simple; two Ethernet interfaces (one from each node in a cluster) are configured as part of the same redundant Ethernet (reth) interface. The reth interface is then configured as part of a redundancy group. A redundancy group is active on only one of the nodes in the cluster, and the redundant Ethernet interfaces that are members of that group send (and normally receive) traffic only through the physical interfaces on the active node.

A redundancy group can be configured to monitor one or more physical interfaces. Each monitored interface is given a weight, which is subtracted from the redundancy group threshold if the interface fails. If the threshold becomes less than zero due to interface failure, the redundancy group transitions state, causing the other node in the cluster to

become active for the group. Consequently, all the redundant Ethernet interfaces that are part of this redundancy group use the interfaces on the new node to send (and normally receive) traffic, thus routing traffic around the failure.

Redundant Ethernet interfaces share the same IP and media access control (MAC) addresses between the different physical interfaces that are members of the reth. The redundant interfaces send gratuitous Address Resolution Protocol (ARP) messages when failing over and appear as a single interface to the rest of the network.

Link Aggregation Interfaces and LACP

As of Junos OS Release 11.2, reth interfaces can contain LAG interface groups as members. Additionally, the physical interfaces contained in the LAG group can cross members of the SRX Series chassis cluster. This allows multiple active physical interfaces between cluster members to participate in the redundant Ethernet (reth) and the redundancy protocol.

Real-Time Performance Monitoring

All Junos OS-based devices have the ability to perform real-time performance monitoring (RPM), a task running on the router that monitors hosts using either ICMP, TCP, or HTTP, that periodically checks the remote hosts and keeps a log history of the packet loss and latency results. This information can be used to monitor upstream routers in a high availability design, and together with IP monitoring, can enable backup interfaces or modify the active routing table based on the probe-results from RPM.

IP Monitoring

IP monitoring is the Junos OS equivalent of the ScreenOS Track-IP feature. This allows an SRX Series device to monitor upstream hosts using RPM and dynamically modify the routing table, based on the availability of the hosts being monitored with RPM.

Feature Support and Comparison Matrix

Although both protocols were designed to provide the same services, NSRP and JSRP (the protocol used in Junos OS) do not operate in the same manner and do not provide the same set of features. Table 3 on page 7 summarizes the main differences between the protocols.

Table 3: Feature Comparison

Feature	JSRP	NSRP
Session replication	Yes	Yes
Application-level gateway (ALG) replication	Yes	Yes
Network Address Translation (NAT) session replication	Yes	Yes
IPsec session replication (policy-based VPN)	Yes	Yes
IPsec session replication (route-based VPN)	Yes	Yes

Table 3: Feature Comparison *(continued)*

Feature	JSRP	NSRP
Route synchronization	N/A	Yes
Interface monitoring	Yes	Yes
Zone monitoring	No	Yes
Track IP / IP monitoring	Yes Renamed to IP monitoring	Yes
Real-time performance monitoring (RPM)	Yes	No
Asymmetric routing	Yes	No
Load balancing	Yes	No
Graceful restart	Yes	No
Layer 2 mode	Yes Junos OS 11.2R2 or later	Yes

Related Documentation

- Example: Configuring Chassis Clusters on an SRX Services Gateway for the Branch on page 8
- Example: Configuring an Active/Passive Chassis Cluster on page 19
- Example: Configuring an Active/Passive Chassis Cluster with Asymmetric Routing on page 32
- Configuring an Active/Active Full Mesh Chassis Cluster on page 38

Example: Configuring Chassis Clusters on an SRX Services Gateway for the Branch

This example shows how to set up chassis clustering on a pair of SRX Series Services Gateways for the branch.

- Requirements on page 8
- Overview on page 9
- Configuration on page 11

Requirements

This example uses the following hardware and software components:

- Two Juniper Networks SRX210 Services Gateways with identical hardware configurations running Junos OS Release 9.0 or later.

- Four Juniper Networks EX Series Ethernet Switches running Junos OS Release 12.1R7.9 or later.

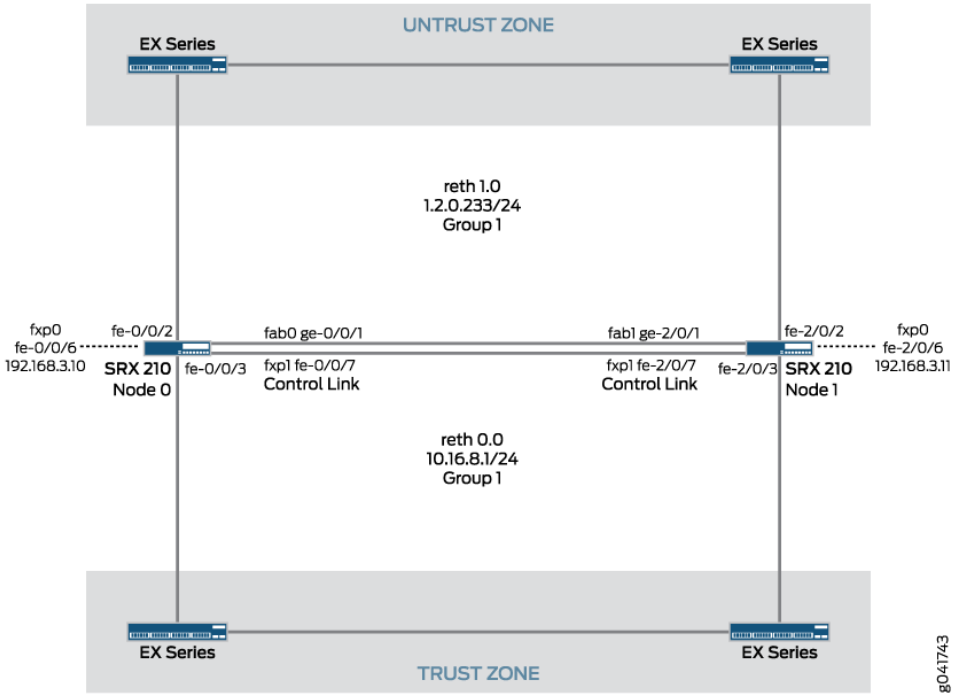
NOTE: This configuration example has been tested to work using Junos OS Release 12.1R7.9 and is assumed to work on all later releases.

Overview

This example shows how to configure chassis clustering using an identical pair of Juniper Networks SRX210 Services Gateways named Node_0 and Node_1.

NOTE: In the system command-line interface, the nodes in the cluster are identified as node0 and node1. In this example, you configure the device hostnames to be Node_0 and Node_1.

Figure 3: SRX Series for the Branch Chassis Cluster Physical Topology



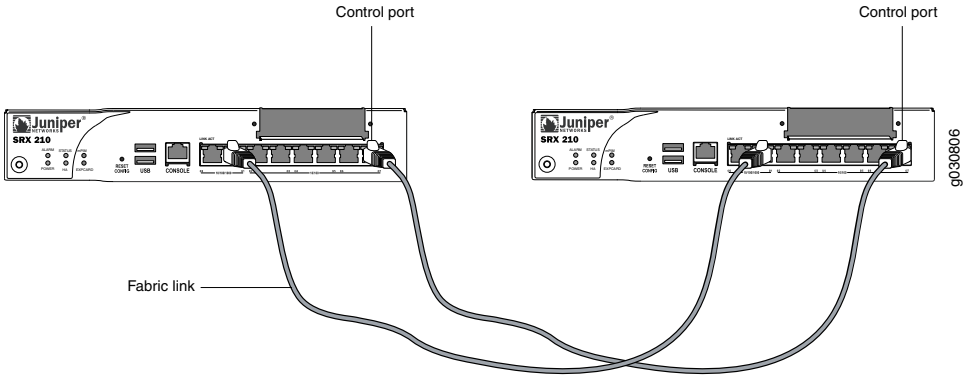
NOTE:

Before you begin, physically connect the two SRX210 devices as follows:

- Connect interface fe-0/0/7 on Router Node_0 to interface fe-0/0/7 on Router Node_1 for the fxp1 control link.

- Connect interface ge-0/0/1 on Router Node_0 to interface ge-0/0/1 on Router Node_1 for the fabric link.
- If you want to use out-of-band management, connect interface fe-0/0/6 on Router Node_0 and interface fe-0/0/6 on Router Node_1 to an out-of-band network management system.

Figure 4: Connecting SRX 210 Device in a Chassis Cluster



NOTE: After the cluster is formed, node1 renumbers its interfaces. In this example interface fe-0/0/7 becomes interface fe-2/0/7, interface ge-0/0/1 becomes interface ge-2/0/1, and interface fe-0/0/6 becomes interface fe-2/0/6 on node1.

Configuration

Step-by-Step Procedure

This procedure begins with the minimum required configuration to enable basic chassis clustering. After the minimal configuration, the two SRX210 devices operate as a single device controlling all interfaces in both nodes.

The procedure continues with the configuration statements needed to specify the IP addresses of the fxp0 management interface and the host name of each cluster node. (node0 and node1 have different management IP addresses and hostnames (Node_0, Node_1)).

The procedure concludes with the configuration needed to add redundant Ethernet interfaces and the associated redundancy groups.

NOTE: If you are starting with the default factory configuration, you need to configure the root password using the **set system root-authentication plain-text-password** command, the **set system root-authentication encrypted-password** command or an equivalent procedure.

To configure basic chassis clustering:

1. If you are starting with the default factory configuration, you must delete some logical interface units and VLANs, and modify or delete the security zones.

   Delete any configuration used for the fxp0 management interface, control link, and fabric link interfaces. The following example uses the factory default configuration for an SRX210 device.

   Enter the following on both nodes:

   ```
   root# delete interfaces
   root# delete vlans
   root# delete security zones security-zone trust interfaces vlan.0
   root# delete security zones security-zone untrust interfaces ge-0/0/0.0
   ```

2. Commit the configuration.

   ```
   root# commit
   commit complete
   ```

3. Enable clustering on Router Node_0 by setting the appropriate cluster ID in the EEPROM.

   Enter the command in operational mode, not in configuration mode.

   ```
   root> set chassis cluster cluster-id 1 node 0 reboot
   Successfully enabled chassis cluster. Going to reboot now
   ```

4. Enable clustering on Router Node_1 by setting the appropriate cluster ID in the EEPROM.

   ```
   root> set chassis cluster cluster-id 1 node 1 reboot
   Successfully enabled chassis cluster. Going to reboot now
   ```

NOTE: A reboot is required for this setting to take effect. After the devices reboot, the interfaces on node1 (Node_1) are renumbered as shown in Table 1 on page 4. Only node0 and node1 can be configured, because the implementation is limited to two nodes in a cluster.

5. Define the interfaces used for the fabric connection.

NOTE: Enter the following from configuration mode on either node0 or node1. The configuration entered on one node is synchronized with the other node in the cluster. In this example the commands are entered on node0.

Configure the fab0 interface as the fabric interface of node0. Configure the fab1 interface as the fabric interface of node1. These interfaces must be connected back-to-back, or through a Layer 2 infrastructure, as shown in Figure 3 on page 9.

```
root# set interface fab0 fabric-options member-interfaces ge-0/0/1
root# set interface fab1 fabric-options member-interfaces ge-2/0/1
```

6. Configure the node name and the out-of-band management interface on each device using configuration groups.

```
root# set groups node0 system host-name Node_0
root# set groups node0 interfaces fxp0 unit 0 family inet address 192.168.3.10/24
root# set groups node1 system host-name Node_1
root# set groups node1 interfaces fxp0 unit 0 family inet address 192.168.3.11/24
```

7. (Optional) Configure device-specific options such as the SNMP description.

```
root@Node_0# set groups node0 snmp description Node_0
root@Node_0# set groups node1 snmp description Node_1
```

8. Apply the group configuration.

```
root@Node_0# set apply-groups “${node}”
```

NOTE: Because this command uses special characters, you might not be able to copy and paste this command. You need to type the command.

9. Commit the configuration.

After the configuration is committed, the prompt changes to display the hostname of the node.

```
root@# commit
node0:
configuration check succeeds
node1:
```

```
commit complete
node0:
commit complete
```

10. (Optional) If you are using redundant Ethernet interfaces as in this example, define two reth interfaces for the cluster by including the **reth-count** statement.

```
root@Node_0# set chassis cluster reth-count 2
```

11. If you are using redundant Ethernet interfaces as in this example, create the reth0 and reth1 redundant interfaces.

```
root@Node_0# set interfaces reth0 redundant-ether-options redundancy-group 1
root@Node_0# set interfaces reth1 redundant-ether-options redundancy-group 1
```

12. (Optional) If you are using redundant Ethernet interfaces as in this example, add interfaces fe-0/0/2 (in node0) and fe-2/0/2 (fe-0/0/2 in node1) to the reth0 interface and interfaces fe-0/0/3 and fe-2/0/3 to the reth 0 interface.

```
root@Node_0# set interface fe-0/0/2 fastether-options redundant-parent reth1
root@Node_0# set interface fe-2/0/2 fastether-options redundant-parent reth1
root@Node_0# set interface fe-0/0/3 fastether-options redundant-parent reth0
root@Node_0# set interface fe-2/0/3 fastether-options redundant-parent reth0
```

13. (Optional) Define node0 as the primary node for redundancy group 1 and redundancy group 0.

    This configuration defines which device has priority (for chassis cluster, high priority is preferred) for the control plane, and which device is preferred to be active for the data plane.

```
root@Node_0# set chassis cluster redundancy-group 1 node 0 priority 100
root@Node_0# set chassis cluster redundancy-group 1 node 1 priority 1
root@Node_0# set chassis cluster redundancy-group 0 node 0 priority 100
root@Node_0# set chassis cluster redundancy-group 0 node 1 priority 1
```

14. Configure IP addresses for the reth interfaces.

```
root@Node_0# set interfaces reth1 unit 0 family inet address 1.2.0.233/24
root@Node_0# set interfaces reth0 unit 0 family inet address 10.16.8.1/24
```

NOTE: If you need to disable clustering, set the cluster ID of each node to 0, and reboot the nodes using the **set chassis cluster cluster-id 0 node 0 reboot** command:

15. Commit the configuration.

```
root@# commit
node0:
commit complete
```

Verification

Confirm that the example is working properly.

- Viewing the Chassis Cluster Status on page 14
- Viewing the Chassis Cluster Statistics on page 14
- Viewing the Control Link Status on page 16
- Viewing the Session on page 16

Viewing the Chassis Cluster Status

Purpose Verify the status of a cluster and present a view of the cluster from the node's perspective. Statistics are not synchronized between the nodes in the cluster. When debugging clusters, it is useful to log in to each member node and analyze the output from each.

Action Use the **show chassis cluster status** command to determine which reth interfaces are active on each node.

```
root@Node_0> show chassis cluster status

Cluster ID: 1
Node                 Priority          Status    Preempt  Manual failover

Redundancy group: 0 , Failover count: 1
    node0                   100         primary        no       no
    node1                   1           secondary      no       no

Redundancy group: 1 , Failover count: 1
    node0                   100         primary        no       no
    node1                   1           secondary      no       no
```

Meaning The command shows the different redundancy groups configured in the cluster, together with their specified priorities and the status of each node. The **no** value under the Manual column shows that there are no manual failovers. The special redundancy group 0 refers to the status of the control plane. In this example, node0 is the primary node for this group and, therefore, performs all control plane calculations and runs the control plane processes such as **rpd**, **kmd**, **dhcpd**, **pppd**, and others.

Viewing the Chassis Cluster Statistics

Purpose Display the statistics of the different objects being synchronized and the fabric and control interface hello messages.

Action Enter the **show chassis cluster statistics** command.

```
root@Node_0> show chassis cluster statistics

Control link statistics:
    Control link 0:
        Heartbeat packets sent: 3411
        Heartbeat packets received: 3391
        Heartbeat packet errors: 0
Fabric link statistics:
    Child link 0
```

```
        Probes sent: 6149
        Probes received: 6148
    Child link 1
        Probes sent: 0
        Probes received: 0
Services Synchronized:
    Service name                              RTOs sent    RTOs received
    Translation context                       0            0
    Incoming NAT                              0            0
    Resource manager                          0            0
    DS-LITE create                            0            0
    Session create                            0            0
    IPv6 session create                       0            0
    Session close                             0            0
    IPv6 session close                        0            0
    Session change                            0            0
    IPv6 session change                       0            0
    Gate create                               0            0
    Session ageout refresh requests           0            0
    IPv6 session ageout refresh requests      0            0
    Session ageout refresh replies            0            0
    IPv6 session ageout refresh replies       0            0
    IPSec VPN                                 0            0
    Firewall user authentication              0            0
    MGCP ALG                                  0            0
    H323 ALG                                  0            0
    SIP ALG                                   0            0
    SCCP ALG                                  0            0
    PPTP ALG                                  0            0
    JSF PPTP ALG                              0            0
    RPC ALG                                   0            0
    RTSP ALG                                  0            0
    RAS ALG                                   0            0
    MAC address learning                      0            0
    GPRS GTP                                  0            0
    GPRS SCTP                                 0            0
    GPRS FRAMEWORK                            0            0
    JSF RTSP ALG                              0            0
    JSF SUNRPC MAP                            0            0
    JSF MSRPC MAP                             0            0
    DS-LITE delete                            0            0
    JSF SLB                                   0            0
    APPID                                     0            0
    JSF MGCP MAP                              0            0
    JSF H323 ALG                              0            0
    JSF RAS ALG                               0            0
    JSF SCCP MAP                              0            0
    JSF SIP MAP                               0            0
    PST_NAT_CREATE                            0            0
    PST_NAT_CLOSE                             0            0
    PST_NAT_UPDATE                            0            0
    JSF TCP STACK                             0            0
    JSF IKE ALG                               0            0
```

Meaning Use the sample output to:

- Verify that the **Heartbeat packets sent** is incrementing.
- Verify that the **Heartbeat packets received** is a number close to the number of **Heartbeat packets sent**.

Viewing the Control Link Status

Purpose Display the status of the control interface (fxp1) of this particular node.

Action Enter the **show chassis cluster interface** command.

```
root@Node_0> show chassis cluster interface
Control link status: Up

Control interfaces:
    Index   Interface        Status
    0       fxp1             Up

Fabric link status: Up

Fabric interfaces:
    Name    Child-interface    Status
                               (Physical/Monitored)
    fab0    ge-0/0/1           Up   / Up
    fab0
    fab1    ge-2/0/1           Up   / Up
    fab1

Redundant-ethernet Information:
    Name         Status      Redundancy-group
    reth0        Up          1
    reth1        Up          1
```

Meaning The sample output shows that the physical interface is **Up**.

Viewing the Session

Purpose Display the sessions in the session table of each node by specifying the node number. Synchronized sessions are seen in both nodes, where they appear as active in one node and backup in the other. A detailed view of a session can be obtained by specifying the session ID.

Action Enter the **show security flow session node0** and **show security flow session session-identifier 2** commands.

```
root@Node_0> show security flow session node0

Session ID: 2, Policy name: self-traffic-policy/1, State: Active, Timeout: 1800
  In: 172.24.241.53/50045 --> 172.19.101.34/22;tcp, If: ge-0/0/0.0
  Out: 172.19.101.34/22 --> 172.24.241.53/50045;tcp, If: .local..0

1 sessions displayed

root@Node_0> show security flow session session-identifier 2

Session ID: 2, Status: Normal, State: Active
Flag: 0x40
Virtual system: root, Policy name: self-traffic-policy/1
Maximum timeout: 1800, Current timeout: 1800
Start time: 1900, Duration: 256
    In: 172.24.241.53/50045 --> 172.19.101.34/22;tcp,
     Interface: ge-0/0/0.0,
     Session token: 0xa, Flag: 0x4097
```



```
    Route: 0x20010, Gateway: 172.19.101.1, Tunnel: 0
    Port sequence: 0, FIN sequence: 0,
    FIN state: 0,
   Out: 172.19.101.34/22 --> 172.24.241.53/50045;tcp,
    Interface: .local..0,
    Session token: 0x4, Flag: 0x4112
    Route: 0xfffb0006, Gateway: 172.19.101.34, Tunnel: 0
    Port sequence: 0, FIN sequence: 0,
    FIN state: 0,

1 sessions displayed
```

Meaning TCP sequence numbers are not synchronized. However, the active node for a given session keeps track of the sequence numbers. When a session is migrated due to a failure (for example, failures that cause the egress interface of a session or group of sessions to be in a different node than prior to the failure), the sequence number counting resumes on the new node based on the sequence numbers of the packets going through the new active node for the session or sessions.

Results

The following is a sample configuration. It is not the complete device configuration. The output has been truncated for brevity.

```
groups {
  node0 {
    system {
      host-name Node_0;
    }
    interfaces {
      fxp0 {
        unit 0 {
          family inet {
            address 192.168.3.10/24;
          }
        }
      }
    }
    snmp {
      description Node_0;
    }
  }
  node1 {
    system {
      host-name Node_1;
    }
    interfaces {
      fxp0 {
        unit 0 {
          family inet {
            address 192.168.3.11/24;
          }
        }
      }
    }
```

```
      snmp {
        description Node_1;
      }
    }
}
apply-groups "${node}";
system {
    root-authentication {
      encrypted-password "$1$6OK0wOML$uUhXEdlQXpApkbXGu735h0"; ##
        SECRET-DATA
    }
    services {
      ...
      dhcp {
        router {
          192.168.1.1;
        }
        pool 192.168.1.0/24 {
          address-range low 192.168.1.2 high 192.168.1.254;
        }
        propagate-settings ge-0/0/0.0;
      }
    }
    ...
}
chassis {
    cluster {
      reth-count 2;
      redundancy-group 1 {
        node 0 priority 100;
        node 1 priority 1;
      }
      redundancy-group 0 {
        node 0 priority 100;
        node 1 priority 1;
      }
    }
}
interfaces {
    fe-0/0/2 {
      fastether-options {
        redundant-parent reth1;
      }
    }
    fe-0/0/3 {
      fastether-options {
        redundant-parent reth0;
      }
    }
    fe-2/0/2 {
      fastether-options {
        redundant-parent reth1;
      }
    }
    fe-2/0/3 {
      fastether-options {
```

```
        redundant-parent reth0;
      }
    }
    fab0 {
      fabric-options {
        member-interfaces {
          ge-0/0/1;
        }
      }
    }
    fab1 {
      fabric-options {
        member-interfaces {
          ge-2/0/1;
        }
      }
    }
    reth0 {
      redundant-ether-options {
        redundancy-group 1;
      }
      unit 0 {
        family inet {
          address 10.16.8.1/24;
        }
      }
    }
    reth1 {
      redundant-ether-options {
        redundancy-group 1;
      }
      unit 0 {
        family inet {
          address 1.2.0.233/24;
        }
      }
    }
  }
...
```

Related Documentation

- Understanding Chassis Clustering on Branch SRX Series Services Gateways on page 2
- Example: Configuring an Active/Passive Chassis Cluster on page 19
- Example: Configuring an Active/Passive Chassis Cluster with Asymmetric Routing on page 32
- Configuring an Active/Active Full Mesh Chassis Cluster on page 38
- Upgrading the Software Image on a Chassis Cluster on page 49

Example: Configuring an Active/Passive Chassis Cluster

This example shows how to set up a basic active/passive chassis cluster on a pair of branch SRX Series devices. In this case, a single device in the cluster is used to route all

traffic, while the other device is used only in the event of a failure. When a failure occurs, the backup device becomes the primary and takes over all forwarding tasks.

- Requirements on page 20
- Overview on page 20
- Configuration on page 22

Requirements

This example uses the following hardware and software components:

- Two Juniper Networks SRX210 Services Gateways with identical hardware configurations running Junos OS Release 9.0 or later.
- Four Juniper Networks EX Series Ethernet Switches running Junos OS Release 9.6 or later.

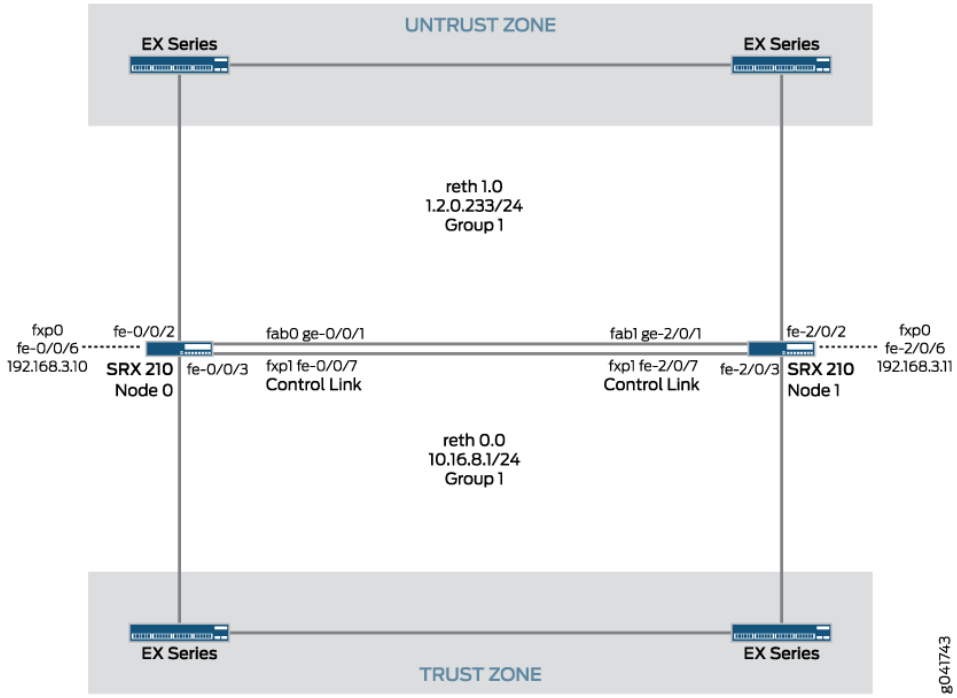
NOTE: This configuration example has been tested using Junos OS Release 12.1R7.9 and is assumed to work on all later releases.

Overview

This example shows how to configure an active/pasive chassis cluster using an identical pair of Juniper Networks SRX210 Services Gateways named Node_0 and Node_1.

NOTE: In the system command-line interface, the nodes in the cluster are identified as node0 and node1. In this example, you configure the device hostnames to be Node_0 and Node_1.

Figure 5: SRX Series for the Branch Active/Passive Chassis Cluster Physical Topology



Active/passive clusters can be created using redundant Ethernet (reth) interfaces. The redundancy group determines the reth state by monitoring the state of the physical interfaces in reth0 and reth1. If any of these interfaces fail, the group is declared inactive by the system that hosts the failing interface. When a failure occurs, both reth interfaces fail over simultaneously, because they belong to the same redundancy group. This configuration minimizes the traffic on the fabric link, because only one node in the cluster is forwarding traffic at any given time.

Before you begin, physically connect the two SRX210 devices as follows:

- Connect interface fe-0/0/7 on Router Node_0 to interface fe-0/0/7 on Router Node_1 for the fxp1 control link.
- Connect interface ge-0/0/1 on Router Node_0 to interface ge-0/0/1 on Router Node_1 for the fabric link.
- If you want to use out-of-band management, connect interface fe-0/0/6 on Router Node_0 and interface fe-0/0/6 on Router Node_1 to an out-of-band network management system.

NOTE: After the cluster is formed, node1 renumbers its interfaces. In this example interface fe-0/0/7 becomes interface fe-2/0/7, interface ge-0/0/1 becomes interface ge-2/0/1, and interface fe-0/0/6 becomes interface fe-2/0/6 on node1.

Configuration

Step-by-Step Procedure

To configure active/passive chassis clustering:

1. If you are starting with the default factory configuration, you must delete some logical interface units and VLANs, and modify or delete the security zones.

   Delete any configuration used for the fxp0 management interface, control link, and fabric link interfaces. The following example uses the factory default configuration for an SRX210 device.

   Enter the following on both nodes:

   ```
   root# delete interfaces
   root# delete vlans
   root# delete security zones security-zone trust interfaces vlan.0
   root# delete security zones security-zone untrust interfaces ge-0/0/0.0
   ```

2. Commit the configuration.

   ```
   root# commit
   commit complete
   ```

3. Enable clustering on Router Node_0 by setting the appropriate cluster ID in the EEPROM.

   Enter the command in operational mode, not in configuration mode.

   ```
   root> set chassis cluster cluster-id 1 node 0 reboot
   Successfully enabled chassis cluster. Going to reboot now
   ```

4. Enable clustering on Router Node_1 by setting the appropriate cluster ID in the EEPROM.

   ```
   root> set chassis cluster cluster-id 1 node 1 reboot
   Successfully enabled chassis cluster. Going to reboot now
   ```

   NOTE: A reboot is required for this setting to take effect. After the devices reboot, the interfaces on node1 (Node_1) are renumbered as shown in Table 1 on page 4. Only node0 and node1 can be configured, because the implementation is limited to two nodes in a cluster.

5. Define the interfaces used for the fabric connection.

   NOTE: Enter the following from configuration mode on either node0 or node1. The configuration entered on one node is synchronized with the other node in the cluster. In this example the commands are entered on node0.

Configure the fab0 interface as the fabric interface of node0. Configure the fab1 interface as the fabric interface of node1. These interfaces must be connected back-to-back, or through a Layer 2 infrastructure, as shown in Figure 3 on page 9.

```
root# set interface fab0 fabric-options member-interfaces ge-0/0/1
root# set interface fab1 fabric-options member-interfaces ge-2/0/1
```

6. Create configuration groups for each node.

   Because the SRX Series Services Gateway chassis cluster configuration is contained within a single common configuration, use the Junos OS node-specific configuration method called *groups* to assign some elements of the configuration to a specific member only.

   ```
   root# set groups node0 system host-name Node_0
   root# set groups node0 interfaces fxp0 unit 0 family inet address 192.168.3.10/24
   root# set groups node1 system host-name Node_1
   root# set groups node1 interfaces fxp0 unit 0 family inet address 192.168.3.11/24
   ```

7. (Optional) Configure device-specific options such as the SNMP description.

   ```
   root# set groups node0 snmp description Node_0
   root# set groups node1 snmp description Node_1
   ```

8. Apply the group configuration.

   ```
   root# set apply-groups “${node}”
   ```

   NOTE: Because this command uses special characters, you might not be able to copy and paste this command. You need to type the command.

9. Commit the configuration.

   After the configuration is committed, the prompt changes to display the hostname of the node.

   ```
   root@# commit
   node0:
   configuration check succeeds
   node1:
   commit complete
   node0:
   commit complete
   ```

10. Configure the heartbeat threshold.

    The heartbeat threshold sets the number of consecutive missed heartbeat messages that a device in a chassis cluster must exceed to trigger failover of the active node.

    ```
    root@Node_0# set chassis cluster heartbeat-threshold 3
    ```

11. Define two redundant Ethernet (reth) interfaces for the cluster by including the **reth-count** statement.

    ```
    root@Node_0# set chassis cluster reth-count 2
    ```

12. Create the reth0 and reth1 redundant interfaces.

```
root@Node_0# set interfaces reth0 redundant-ether-options redundancy-group 1
root@Node_0# set interfaces reth1 redundant-ether-options redundancy-group 1
```

13. Assign the member interfaces to the parent reth interface.

    Add interfaces fe-0/0/2 (in node0) and fe-2/0/2 (fe-0/0/2 in node1) to the reth1 interface and interfaces fe-0/0/3 and fe-2/0/3 to the reth0 interface.

```
root@Node_0# set interface fe-0/0/2 fastether-options redundant-parent reth1
root@Node_0# set interface fe-2/0/2 fastether-options redundant-parent reth1
root@Node_0# set interface fe-0/0/3 fastether-options redundant-parent reth0
root@Node_0# set interface fe-2/0/3 fastether-options redundant-parent reth0
```

14. Define node0 as the primary node for redundancy group 1.

    This configuration defines which device has priority (for chassis cluster, high priority is preferred) for the control plane, and which device is preferred to be active for the data plane.

    Redundancy group 0 determines the status of the node as primary or secondary.

```
root@Node_0# set chassis cluster redundancy-group 1 node 0 priority 100
root@Node_0# set chassis cluster redundancy-group 1 node 1 priority 1
root@Node_0# set chassis cluster redundancy-group 0 node 0 priority 100
root@Node_0# set chassis cluster redundancy-group 0 node 1 priority 1
```

15. Configure IP addresses for the reth interfaces.

```
root@Node_0# set interfaces reth1 unit 0 family inet address 1.2.0.233/24
root@Node_0# set interfaces reth0 unit 0 family inet address 10.16.8.1/24
```

16. Configure the chassis cluster behavior in case of a failure.

    The **preempt** keyword causes the control to be reverted back to the primary node (node0, which has a higher priority) for the group when the failing interface causing the failover is operational again.

```
root@Node_0# set chassis cluster redundancy-group 1 preempt
```

17. Enable monitoring of the physical interfaces that are used for redundancy group 1.

```
root@Node_0# set chassis cluster redundancy-group 1 interface-monitor fe-0/0/2
  weight 255
root@Node_0# set chassis cluster redundancy-group 1 interface-monitor fe-2/0/2
  weight 255
root@Node_0# set chassis cluster redundancy-group 1 interface-monitor fe-0/0/3
  weight 255
root@Node_0# set chassis cluster redundancy-group 1 interface-monitor fe-2/0/3
  weight 255
```

NOTE: Interface monitoring is not supported on redundancy group 0 for SRX Series branch devices.

18. Assign the reth interfaces to the appropriate security zone.

    Just as with physical interfaces, reth interfaces must be part of a security zone.

```
root@Node_0# set security zones security-zone untrust interfaces reth1.0
root@Node_0# set security zones security-zone trust interfaces reth0.0
```

19. Commit the configuration.

```
root@Node_0# commit
node0:
configuration check succeeds
node1:
commit complete
node0:
commit complete
```

Verification

Confirm that the example is working properly.

- Viewing the Chassis Cluster Status on page 25
- Viewing the Chassis Cluster Statistics on page 25
- Viewing the Control Link Status on page 27

Viewing the Chassis Cluster Status

Purpose Verify the status of the cluster. Statistics are not synchronized between the nodes in the cluster. When debugging clusters, it is useful to log in to each member node and analyze the output from each.

Action Use the **show chassis cluster status** command to determine which reth interfaces are active on each node.

```
root@Node_0> show chassis cluster status

Cluster ID: 1
Node                 Priority          Status    Preempt  Manual failover

Redundancy group: 0 , Failover count: 0
    node0                  100         primary        no       no
    node1                  1           secondary      no       no

Redundancy group: 1 , Failover count: 0
    node0                  100         primary        yes      no
    node1                  1           secondary      yes      no
```

Meaning The command shows the different redundancy groups configured in the cluster, together with their specified priorities and the status of each node. The **no** value under the Manual column shows that there are no manual failovers. The special redundancy group 0 refers to the status of the control plane. In this example, node0 is the primary node for this group and, therefore, performs all control plane calculations and runs the control plane processes such as **rpd**, **kmd**, **dhcpd**, **pppd**, and others.

Viewing the Chassis Cluster Statistics

Purpose Display the statistics of the different objects being synchronized and the fabric and control interface hello messages

Action Enter the **show chassis cluster statistics** command.

```
root@Node_0> show chassis cluster statistics

Control link statistics:
    Control link 0:
        Heartbeat packets sent: 70
        Heartbeat packets received: 70
        Heartbeat packet errors: 0
Fabric link statistics:
    Child link 0
        Probes sent: 140
        Probes received: 140
    Child link 1
        Probes sent: 0
        Probes received: 0
Services Synchronized:
    Service name                                RTOs sent    RTOs received
    Translation context                         0            0
    Incoming NAT                                0            0
    Resource manager                            0            0
    DS-LITE create                              0            0
    Session create                              0            0
    IPv6 session create                         0            0
    Session close                               0            0
    IPv6 session close                          0            0
    Session change                              0            0
    IPv6 session change                         0            0
    Gate create                                 0            0
    Session ageout refresh requests             0            0
    IPv6 session ageout refresh requests        0            0
    Session ageout refresh replies              0            0
    IPv6 session ageout refresh replies         0            0
    IPSec VPN                                   0            0
    Firewall user authentication                0            0
    MGCP ALG                                    0            0
    H323 ALG                                    0            0
    SIP ALG                                     0            0
    SCCP ALG                                    0            0
    PPTP ALG                                    0            0
    JSF PPTP ALG                                0            0
    RPC ALG                                     0            0
    RTSP ALG                                    0            0
    RAS ALG                                     0            0
    MAC address learning                        0            0
    GPRS GTP                                    0            0
    GPRS SCTP                                   0            0
    GPRS FRAMEWORK                              0            0
    JSF RTSP ALG                                0            0
    JSF SUNRPC MAP                              0            0
    JSF MSRPC MAP                               0            0
    DS-LITE delete                              0            0
    JSF SLB                                     0            0
    APPID                                       0            0
    JSF MGCP MAP                                0            0
    JSF H323 ALG                                0            0
    JSF RAS ALG                                 0            0
    JSF SCCP MAP                                0            0
    JSF SIP MAP                                 0            0
    PST_NAT_CREATE                              0            0
    PST_NAT_CLOSE                               0            0
```



```
    PST_NAT_UPDATE                          0           0
    JSF TCP STACK                           0           0
    JSF IKE ALG                             0           0
```

Meaning Use the sample output to:

- Verify that the **Heartbeat packets sent** is incrementing.
- Verify that the **Heartbeat packets received** is a number close to the number of **Heartbeat packets sent**.

Viewing the Control Link Status

Purpose Display the status of the control interface (fxp1) of this particular node and the status of the monitored interfaces in the cluster.

Action Enter the **show chassis cluster interface** command.

```
root@Node_0> show chassis cluster interface
Control link status: Up

Control interfaces:
    Index   Interface        Status
    0       fxp1             Up

Fabric link status: Up

Fabric interfaces:
    Name    Child-interface    Status
                               (Physical/Monitored)
    fab0    ge-0/0/1           Up   / Up
    fab0
    fab1    ge-2/0/1           Up   / Up
    fab1

Redundant-ethernet Information:
    Name         Status      Redundancy-group
    reth0        Up          1
    reth1        Up          1

Interface Monitoring:
    Interface         Weight    Status    Redundancy-group
    fe-2/0/3          255       Up        1
    fe-0/0/3          255       Up        1
    fe-2/0/2          255       Up        1
    fe-0/0/2          255       Up        1
```

Meaning The sample output shows that the fxp1 interface is **Up** and the physical interfaces are being monitored.

Results

The following is a sample configuration. It is not the complete device configuration. The output has been truncated for brevity.

```
groups {
  node0 {
```

```
        system {
            host-name Node_0;
        }
        interfaces {
            fxp0 {
                unit 0 {
                    family inet {
                        address 192.168.3.10/24;
                    }
                }
            }
        }
    }
    node1 {
        system {
            host-name Node_1;
        }
        interfaces {
            fxp0 {
                unit 0 {
                    family inet {
                        address 192.168.3.11/24;
                    }
                }
            }
        }
    }
}
apply-groups "${node}";
system {
    root-authentication {
        encrypted-password "$1$XE1VTzge$1Cxn1eaJT/2bZBxo9Uzyj/"; ## SECRET-DATA
    }
    name-server {
        208.67.222.222;
        208.67.220.220;
    }
    services {
        ssh;
        telnet;
        xnm-clear-text;
        web-management {
            http {
                interface vlan.0;
            }
            https {
                system-generated-certificate;
                interface vlan.0;
            }
        }
        dhcp {
            router {
                192.168.1.1;
            }
            pool 192.168.1.0/24 {
                address-range low 192.168.1.2 high 192.168.1.254;
```

```
        }
        propagate-settings ge-0/0/0.0;
      }
    }
    syslog {
      archive size 100k files 3;
      user * {
        any emergency;
      }
      file messages {
        any critical;
        authorization info;
      }
      file interactive-commands {
        interactive-commands error;
      }
    }
    max-configurations-on-flash 5;
    max-configuration-rollbacks 5;
    license {
      autoupdate {
        url https://ae1.juniper.net/junos/key_retrieval;
      }
    }
}
chassis {
    cluster {
      reth-count 2;
      heartbeat-threshold 3;
      redundancy-group 1 {
        node 0 priority 100;
        node 1 priority 1;
        preempt;
        interface-monitor {
          fe-0/0/2 weight 255;
          fe-2/0/2 weight 255;
          fe-0/0/3 weight 255;
          fe-2/0/3 weight 255;
        }
      }
      redundancy-group 0 {
        node 0 priority 100;
        node 1 priority 1;
      }
    }
}
interfaces {
    fe-0/0/2 {
      fastether-options {
        redundant-parent reth1;
      }
    }
    fe-0/0/3 {
      fastether-options {
        redundant-parent reth0;
      }
```

```
    }
    fe-2/0/2 {
      fastether-options {
        redundant-parent reth1;
      }
    }
    fe-2/0/3 {
      fastether-options {
        redundant-parent reth0;
      }
    }
    fab0 {
      fabric-options {
        member-interfaces {
          ge-0/0/1;
        }
      }
    }
    fab1 {
      fabric-options {
        member-interfaces {
          ge-2/0/1;
        }
      }
    }
    reth0 {
      redundant-ether-options {
        redundancy-group 1;
      }
      unit 0 {
        family inet {
          address 10.16.8.1/24;
        }
      }
    }
    reth1 {
      redundant-ether-options {
        redundancy-group 1;
      }
      unit 0 {
        family inet {
          address 1.2.0.233/24;
        }
      }
    }
  }
  protocols {
    stp;
  }
  security {
    screen {
      ids-option untrust-screen {
        icmp {
          ping-death;
        }
        ip {
```

```
        source-route-option;
        tear-drop;
      }
      tcp {
        syn-flood {
          alarm-threshold 1024;
          attack-threshold 200;
          source-threshold 1024;
          destination-threshold 2048;
          timeout 20;
        }
        land;
      }
    }
  }
  nat {
    source {
      rule-set trust-to-untrust {
        from zone trust;
        to zone untrust;
        rule source-nat-rule {
          match {
            source-address 0.0.0.0/0;
          }
          then {
            source-nat {
              interface;
            }
          }
        }
      }
    }
  }
  policies {
    from-zone trust to-zone untrust {
      policy trust-to-untrust {
        match {
          source-address any;
          destination-address any;
          application any;
        }
        then {
          permit;
        }
      }
    }
  }
  zones {
    security-zone trust {
      host-inbound-traffic {
        system-services {
          all;
        }
        protocols {
          all;
        }
```

```
        }
        interfaces {
            reth0.0;
        }
    }
    security-zone untrust {
        screen untrust-screen;
        interfaces {
            reth1.0;
        }
    }
  }
}
```

Related Documentation

- Understanding Chassis Clustering on Branch SRX Series Services Gateways on page 2
- Example: Configuring Chassis Clusters on an SRX Services Gateway for the Branch on page 8
- Example: Configuring an Active/Passive Chassis Cluster with Asymmetric Routing on page 32
- Configuring an Active/Active Full Mesh Chassis Cluster on page 38
- Upgrading the Software Image on a Chassis Cluster on page 49

Example: Configuring an Active/Passive Chassis Cluster with Asymmetric Routing

This example configures the Junos OS asymmetric routing feature on a pair of SRX Series devices configured as an active/passive cluster.

- Requirements on page 32
- Overview on page 32
- Configuration on page 35

Requirements

This example uses the following hardware and software components:

- Two Juniper Networks SRX210 Services Gateways with identical hardware configurations running Junos OS Release 9.0 or later.
- Four Juniper Networks EX Series Ethernet Switches running Junos OS Release 12.1R7.9 or later.

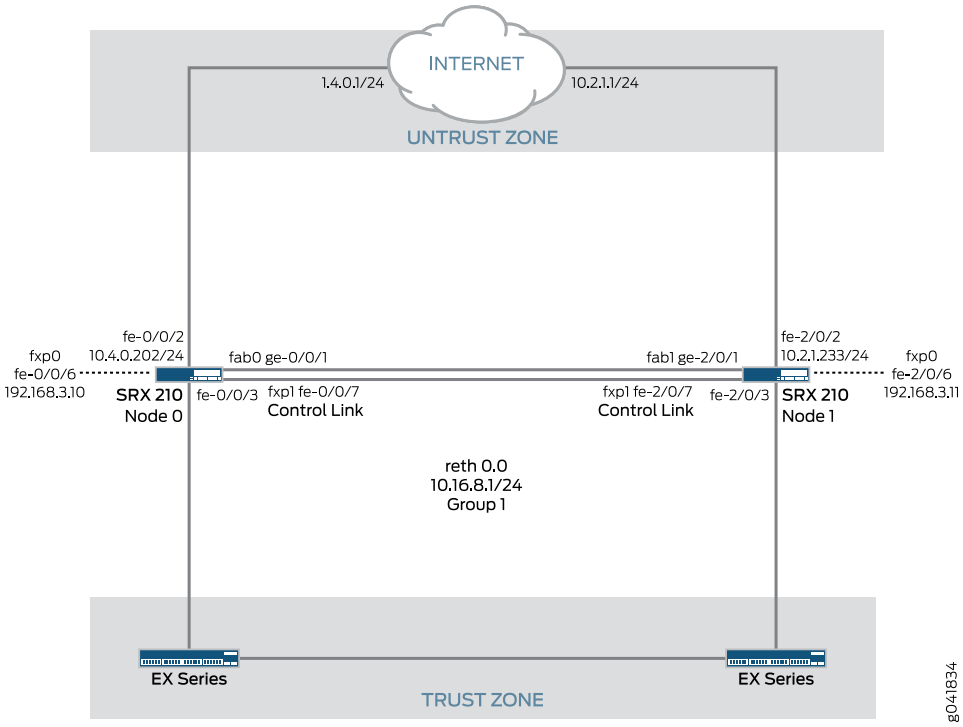
Overview

In this scenario, traffic received by a node is matched against that node's session table. The result of this lookup determines whether that node processes the session or forwards the traffic to the other node through the fabric link. This capability enables you to anchor the session to either device in the cluster. As long as the session tables are replicated, the traffic is correctly processed. To minimize fabric traffic, sessions are always anchored to the node hosting the egress interface for that particular connection.

NOTE: In the system command-line interface, the nodes in the cluster are identified as node0 and node1. In this example, you configure the device hostnames to be Node_0 and Node_1.

Figure 6 on page 33 shows the topology used in this example.

Figure 6: Asymmetric Routing Scenario Topology



Under normal operating conditions, traffic flows between the trust zone (reth0.0 in node0) and the fe-0/0/2 interface. The flow sessions are created in both node0 and node1. The sessions are only active in node0 since the primary Internet connection resides in node0 and the egress interface for all of these sessions is fe-0/0/2 on node0.

An active/passive chassis cluster with asymmetric routing can recover from a failure of the redundant Ethernet interface in the trust zone or from an physical interface failure in the untrust zone.

Trust Zone Redundant Ethernet Interface Failure Scenario

A failure of the fe-0/0/3 interface on node0 triggers a failover of the redundancy group, causing interface fe-2/0/3 on node1 to become active.

After the failover:

1. Traffic from the Internet arrives at node0 to be processed (since the session is anchored to this node).

2. The traffic is sent to node1 through the fabric interface where node1 forwards it through the fe-2/0/3 interface.

The return traffic follows a similar process:

1. Traffic arrives at node1 on interface fe-2/0/3.
2. After session lookup, the traffic is sent to node0 across the fabric link because the egress interface is on node0 and the session is active on node0.
3. Node0 processes the traffic and forwards it to the Internet through the fe-0/0/2 interface.

Untrust Zone Interfaces Failure Scenario

In the case of an interface in the untrust zone failing, the sessions are migrated from node0 to node1.

After a failure of interface fe-0/0/2 that is connected to the Internet:

1. A change in the routing table is made. The node installs a default route with the fe-2/0/2 interface on node1 as the egress interface.
2. The sessions in node0 become inactive (since the egress interface now resides in node1), and the backup sessions in node1 become active.
3. Traffic arriving from the trust zone received on interface fe-0/0/3 on node0 is forwarded to node1 across the fabric link for processing.
4. After traffic is processed in node1, it is forwarded to the Internet through the fe-2/0/2 interface.

NOTE: If this scenario is used with source NAT, to accommodate different address spaces assigned by different providers, the scenario described will not work because the egress sessions is NAT translated differently after the failover. This is not a limitation of the high availability implementation, but a consequence of the fact that if two Internet service providers (ISPs) are used, the customer does not own a public address space, and a failure in one of the ISPs results in the loss of connectivity of all IP addresses belonging to the failed service provider.

Before you begin, physically connect the two SRX210 devices as follows:

- Connect interface fe-0/0/7 on Router Node_0 to interface fe-0/0/7 on Router Node_1 for the fxp1 control link.
- Connect interface ge-0/0/1 on Router Node_0 to interface ge-0/0/1 on Router Node_1 for the fabric link.
- If you want to use out-of-band management, connect interface fe-0/0/6 on Router Node_0 and interface fe-0/0/6 on Router Node_1 to an out-of-band network management system.

NOTE: After the cluster is formed, node1 renumbers its interfaces. In this example interface fe-0/0/7 becomes interface fe-2/0/7, interface ge-0/0/1 becomes interface ge-2/0/1, and interface fe-0/0/6 becomes interface fe-2/0/6 on node1.

Configuration

Step-by-Step Procedure

This example configures the asymmetric routing feature in a active/passive chassis cluster.

1. If you are starting with the default factory configuration, you must delete some logical interface units and VLANs, and modify or delete the security zones.

   Delete any configuration used for the fxp0 management interface, high availability control link and fabric link interfaces. The following example uses the factory default configuration for an SRX210 device.

   Enter the following on both nodes:

   ```
   root# delete interfaces
   root# delete vlans
   root# delete security zones security-zone trust interfaces vlan.0
   root# delete security zones security-zone untrust interfaces ge-0/0/0.0
   ```

2. Commit the configuration.

   ```
   root# commit
   commit complete
   ```

3. Enable clustering on Router Node_0 by setting the appropriate cluster ID in the EEPROM.

   Enter the command in operational mode, not in configuration mode.

   ```
   root> set chassis cluster cluster-id 1 node 0 reboot
   Successfully enabled chassis cluster. Going to reboot now
   ```

4. Enable clustering on Router Node_1 by setting the appropriate cluster ID in the EEPROM.

   ```
   root> set chassis cluster cluster-id 1 node 1 reboot
   Successfully enabled chassis cluster. Going to reboot now
   ```

   NOTE: A reboot is required for this setting to take effect. After the devices reboot, the interfaces on node1 (Node_1) are renumbered as shown in Table 1 on page 4. Only node0 and node1 can be configured, because the implementation is limited to two nodes in a cluster.

5. Define the interfaces used for the fabric connection.

NOTE: Enter the following from configuration mode on either node0 or node1. The configuration entered on one node is synchronized with the other node in the cluster. In this example the commands are entered on node0.

Configure the fab0 interface as the fabric interface of node0. Configure the fab1 interface as the fabric interface of node1. These interfaces must be connected back-to-back, or through a Layer 2 infrastructure, as shown in Figure 3 on page 9.

```
root# set interface fab0 fabric-options member-interfaces ge-0/0/1
root# set interface fab1 fabric-options member-interfaces ge-2/0/1
```

6. Create configuration groups for each node.

   Because the SRX Services Gateway chassis cluster configuration is contained within a single common configuration, use the Junos OS node-specific configuration method called *groups* to assign some elements of the configuration to a specific member only.

   ```
   root# set groups node0 system host-name Node_0
   root# set groups node0 interfaces fxp0 unit 0 family inet address 192.168.3.10/24
   root# set groups node1 system host-name Node_1
   root# set groups node1 interfaces fxp0 unit 0 family inet address 192.168.3.11/24
   ```

7. (Optional) Configure device-specific options such as the SNMP description.

   ```
   root# set groups node0 snmp description Node_0
   root# set groups node1 snmp description Node_1
   ```

8. Apply the group configuration.

   ```
   root# set apply-groups “${node}”
   ```

   NOTE: Because this command uses special characters, you might not be able to copy and paste this command. You need to type the command.

9. Commit the configuration.

   After the configuration is committed, the prompt changes to display the hostname of the node.

   ```
   root@# commit
   node0:
   configuration check succeeds
   node1:
   commit complete
   node0:
   commit complete
   ```

10. Configure the heartbeat threshold.

The heartbeat threshold sets the number of consecutive missed heartbeat messages that a device in a chassis cluster must exceed to trigger failover of the active node.

```
root@Node_0# set chassis cluster heartbeat-threshold 3
```

11. Define two redundant Ethernet (reth) interfaces for the cluster by including the **reth-count** statement.

```
root@Node_0# set chassis cluster reth-count 1
```

12. Create the reth0 redundant interface.

```
root@Node_0# set interfaces reth0 redundant-ether-options redundancy-group 1
```

13. Assign the member interfaces to the parent reth interface.

Add interfaces fe-0/0/3 and fe-2/0/3 to the reth0 interface.

```
root@Node_0# set interfaces fe-0/0/3 fastether-options redundant-parent reth0
root@Node_0# set interfaces fe-2/0/3 fastether-options redundant-parent reth0
```

14. Define node0 as the primary node for redundancy group 1.

Redundancy-group 1 is used to control the reth interface connected to the trust zone. Note that the redundancy group (and therefore reth0) only fails over if either interface fe-0/0/3 or fe-2/0/3 fails, but not if any of the interfaces connected to the Internet fail.

```
root@Node_0# set chassis cluster redundancy-group 1 node 0 priority 100
root@Node_0# set chassis cluster redundancy-group 1 node 1 priority 1
```

15. Configure the IP addresses of the physical interfaces connected to the Internet and for the reth interface.

```
root@Node_0# set interfaces fe-0/0/2 unit 0 family inet address 1.4.0.202/24
root@Node_0# set interfaces fe-2/0/2 unit 0 family inet address 10.2.1.233/24
root@Node_0# set interfaces reth0 unit 0 family inet address 10.16.8.1/24
```

16. Configure the chassis cluster behavior in case of a failure.

The **preempt** keyword causes the control to be reverted back to the primary node (node0, which has a higher priority) for the group when the failing interface causing the failover is operational again.

```
root@Node_0# set chassis cluster redundancy-group 1 preempt
```

17. Enable monitoring of the physical interfaces that are used for redundancy group 1.

```
root@Node_0# set chassis cluster redundancy-group 1 interface-monitor fe-0/0/3
  weight 255
root@Node_0# set chassis cluster redundancy-group 1 interface-monitor fe-2/0/3
  weight 255
```

NOTE: Interface monitoring is not supported on redundancy group 0 for SRX Series branch devices.

18. Configure two static routes, one to each ISP.

Configure the route metric to make IP address 1.4.0.1 the preferred next hop router.

```
root# set routing-options static route 0.0.0.0/0 qualified-next-hop 1.4.0.1 metric
  10
root# set routing-options static route 0.0.0.0/0 qualified-next-hop 1.2.1.1 metric
  100
```

19. Configure the zone definitions.

```
root# set security zones security-zone untrust interfaces fe-0/0/3
  host-inbound-traffic system-services dhcp
root# set del security zones security-zone untrust interfaces fe-2/0/3
  host-inbound-traffic system-services dhcp
root# set security zones security-zone trust interfaces reth0.0
```

20. Configure a security policy that permits any source address, any destination address, and any application from zone trust to zone untrust.

```
root# set security policies from-zone trust to-zone untrust policy ANY match
  source-address any
root# set security policies from-zone trust to-zone untrust policy ANY match
  destination-address any
root# set security policies from-zone trust to-zone untrust policy ANY match
  application any
root# set security policies from-zone trust to-zone untrust policy ANY then permit
```

Related Documentation

- Understanding Chassis Clustering on Branch SRX Series Services Gateways on page 2
- Example: Configuring Chassis Clusters on an SRX Services Gateway for the Branch on page 8
- Example: Configuring an Active/Passive Chassis Cluster on page 19
- Configuring an Active/Active Full Mesh Chassis Cluster on page 38
- Upgrading the Software Image on a Chassis Cluster on page 49

Configuring an Active/Active Full Mesh Chassis Cluster

Active/active clustering on SRX Series devices is used to maintain traffic flows on both chassis cluster members whenever possible.

This scenario is found in medium to large deployments. OSPF is used to control the traffic flow through the nodes in the cluster, and the Junos OS Services Redundancy Protocol (JSRP) is used to synchronize the sessions between the two nodes. Since asymmetric routing is supported, you do not need to force the traffic in both directions to a particular node. If a failure occurs and return traffic for a session arrives at a node different from the node that created the session, the fabric link is used to send the traffic back to the node where sessions are active (this is the node hosting the egress interface for that particular session).

This scenario benefits from the use of full mesh connectivity between the devices (thus improving the resiliency of the network). It eliminates the need to add extra switches between the firewalls and the routers, and it reduces the number of potential points of failure in the network.

Figure 7 on page 39 shows the physical topology of a common SRX Series Services Gateway for the branch chassis cluster.

Figure 7: Active/Active Full Mesh Physical Topology

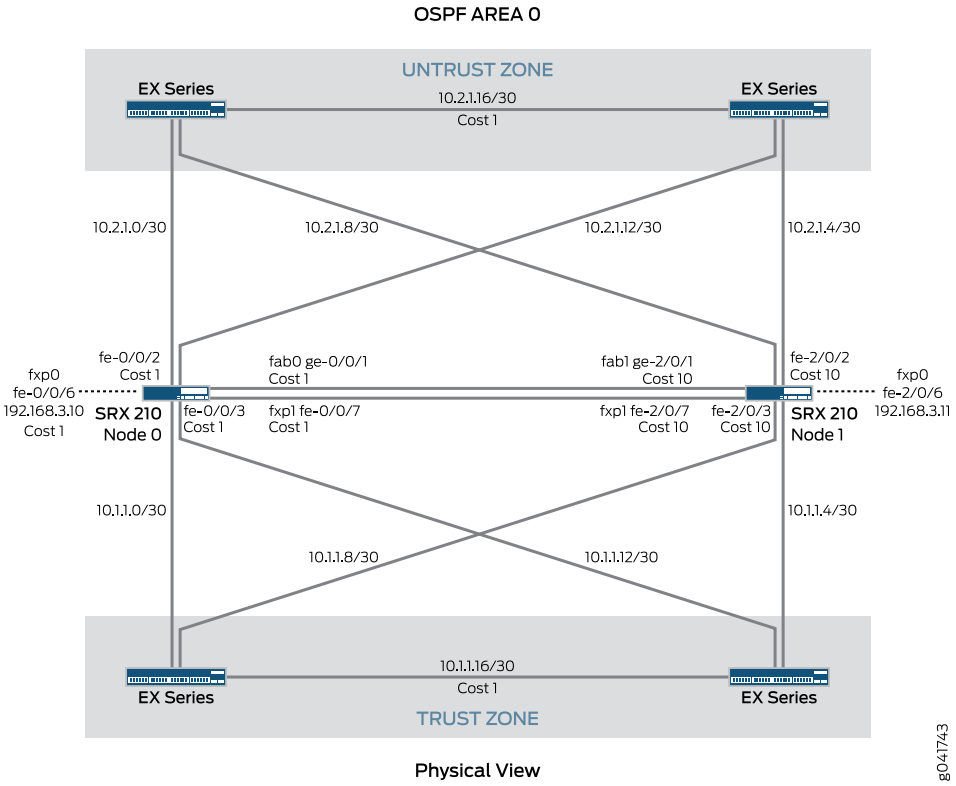
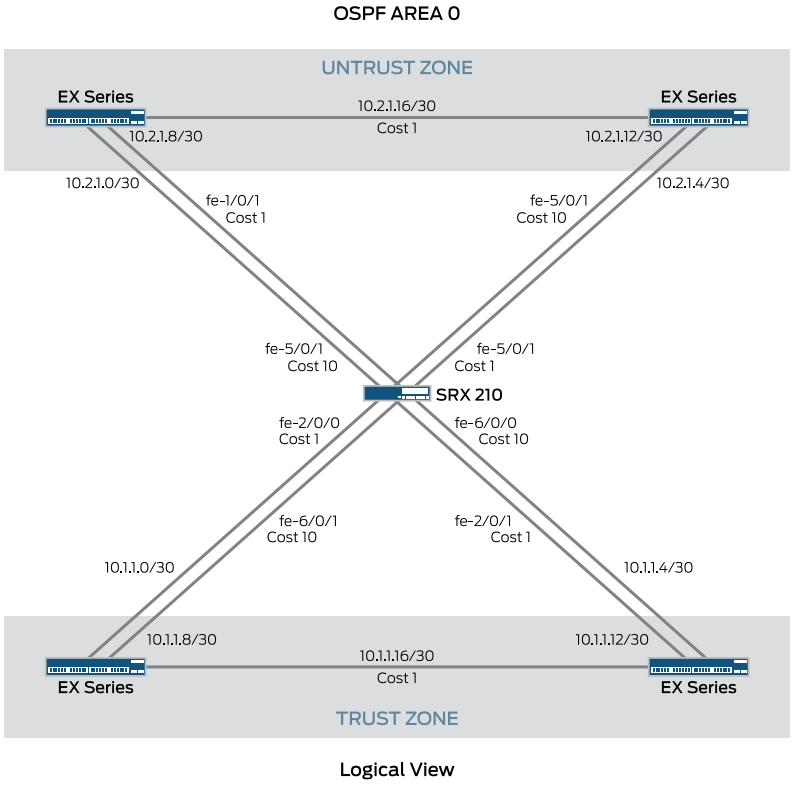


Figure 8 on page 40 shows the logical topology of a common SRX Series Services Gateway for the branch chassis cluster.

Figure 8: Active/Active Full Mesh Logical Topology



Logical View

The following should be taken into account when using the chassis cluster feature:

- Errors in either the fabric or fxp1 links (but not both) cause the backup node to become disabled (single failure point). If a backup node detects errors in both the fabric and fxp1 links, it becomes the primary node (dual failure point).
- In the event of a control link failure, the system tries to avoid a *dual primary node* scenario by monitoring the fabric link. If hello messages are received though this link, the secondary node becomes disabled, while the primary remains active. If neither control link nor fabric link hello messages are received, the secondary node transitions to active.
- When a fabric link failure is detected, the nodes try to avoid a dual primary node scenario just like in the case of a control link failure. If the fabric link fails but the control link is still operational, the backup node becomes disabled. This prevents the situation where there are two active primary nodes.
- Failover times are in the order of a few seconds. A failure is detected in 3 seconds or more (because the minimum hello time is 1000 ms, and the smallest threshold is three consecutive lost hello messages).

For the most up-to-date information about feature support, use the Juniper Networks Feature Explorer application at http://pathfinder.juniper.net/feature-explorer/.

Related Documentation

- Understanding Chassis Clustering on Branch SRX Series Services Gateways on page 2

- Example: Configuring Chassis Clusters on an SRX Services Gateway for the Branch on page 8
- Example: Configuring an Active/Passive Chassis Cluster on page 19
- Example: Configuring an Active/Passive Chassis Cluster with Asymmetric Routing on page 32
- Upgrading the Software Image on a Chassis Cluster on page 49

Upgrading a Chassis Cluster

Upgrading a chassis cluster is a simple procedure, but note that a service disruption of about 3 to 5 minutes occurs during this process. To upgrade the cluster, perform the following tasks:

1. Load the new image file on node0.
2. Perform the image upgrade, without rebooting the node by entering the **request system software add <image name>** command.
3. Load the new image file in node1.
4. Perform the image upgrade in node1 by entering the **request system software add <image name>** command.
5. Reboot both nodes simultaneously by entering the **request system reboot** command.

In-Band Management of Chassis Clusters

SRX Series Services Gateways for the branch can be managed in-band or out-of-band (through the use of the fxp0 interface) when deployed in a cluster configuration. This assumes that the cluster can be reached from the management stations through revenue ports only.

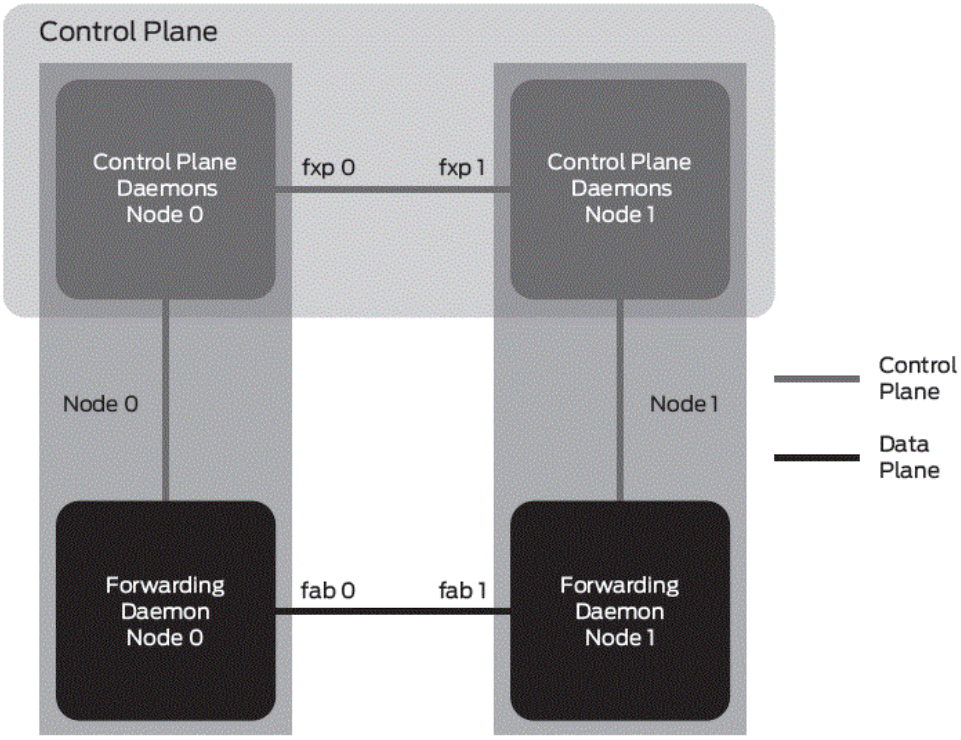
For more information about using in-band management connections, see *Best Practices for SRX Series Chassis Cluster Management* at http://www.juniper.net/techpubs/en_US/release-independent/nce/information-products/topic-collections/nce/srx-poly-memory-best/srx-poly-memory-best.pdf

Understanding the Chassis Cluster High Availability Model

The high availability feature available in Junos OS for SRX Series 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 routing and forwarding computations (among many other things). After 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 Routing Engine intervention.

When enabling a chassis cluster in SRX Series gateways, the same model is used to provide control plane redundancy as is shown in Figure 9 on page 42.

Figure 9: SRX Series Clustering Model



Just like in a router with two Routing Engines, the control plane of SRX Series clusters 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 daemon, such as BGP traffic, OSPF, IS-IS, RIP, and PIM.
- Internet Key Exchange (IKE) negotiation messages.
- Traffic directed to management daemons like SSH, Telnet, SNMP, and the NETCONF XML management protocol.
- Monitoring protocols like Bidirectional Forwarding Detection (BFD), or real-time performance monitoring (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, the fxp0 interface is used 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.

In releases earlier than Junos OS Release 10.1R2, the management of a chassis cluster using Network and Security Manager (and other management interfaces) required connectivity to the control plane of both members of a cluster, therefore requiring access to the fxp0 interface of each node.

The *Managing a Chassis Cluster* topic explains how to manage a chassis cluster through the primary node without requiring the use of the fxp0 interfaces.

Related Documentation

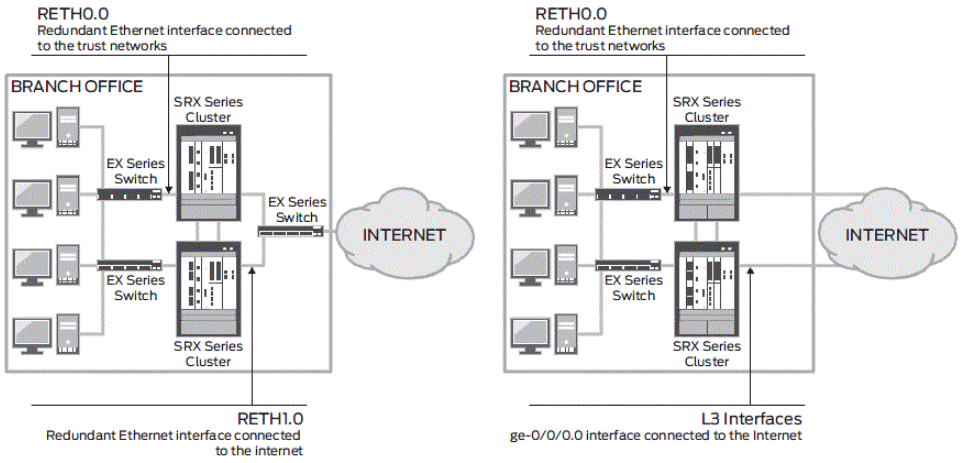
- Understanding Chassis Clustering on Branch SRX Series Services Gateways on page 2
- Example: Configuring Chassis Clusters on an SRX Services Gateway for the Branch on page 8
- Example: Configuring an Active/Passive Chassis Cluster on page 19
- Example: Configuring an Active/Passive Chassis Cluster with Asymmetric Routing on page 32
- Upgrading the Software Image on a Chassis Cluster on page 49

Managing a Chassis Cluster

Connecting to a Cluster Using SSH/Telnet

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. Either Layer 3 or redundant ethernet (reth) interfaces always direct the traffic to the primary node, whichever node that is. Both deployment scenarios depicted in the following diagrams are common.

Figure 10: Common Branch Deployment Scenarios for SRX Series Chassis Clusters



In both cases, establishing a connection to any of the local addresses connects to the primary node of redundancy group 0. For example, you can connect to the primary node even when the reth interface that is a member of 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.

To establish a connection to the node's interfaces:

1. Log in to the cluster.

   This example uses SSH. SSH management of a cluster is a good example of how all management protocols behave. It is simple to connect to the primary node, and connecting to the secondary node must be done through the primary.

   ```
   $ssh 10.1.1.34
   labuser@10.1.1.34's password:
   --- JUNOS 10.2R1.3 built 2010-05-14 15:13:40 UTC
   {primary:node1}
   ```

2. Display the cluster status by entering the **show chassis cluster status** command.

   ```
   user@host> show chassis cluster status

   luster ID: 3
   Node                 Priority          Status      Preempt      Manual failover
   Redundancy group: 0 , Failover count: 3
   node0                     200           secondary     no              yes
   node1                     255            primary      no              yes
   Redundancy group: 1 , Failover count: 4
   node0                     254            primary      yes             no
   node1                      1             secondary    yes             no
   ```

3. Log in to the secondary node from the primary node.

   Most monitoring commands show the status of both nodes. When needed, it is possible to connect to the secondary node from the primary node by entering the **request routing-engine login node *node-id*** command.

   ```
   user@host> request routing-engine login node 0

   -- JUNOS 10.2R1.3 built 2010-05-14 15:13:40 UTC
   {secondary:node0}
   Exiting the session brings us back to the primary node:
   {secondary:node0}
   labuser@BranchGW> exit
   rlogin: connection closed
   {primary:node1}
   labuser@BranchGW>
   ```

Network and Security Manager management of a cluster is no different than this example. Network and Security Manager versions prior to 2010.2 require NETCONF connections to both nodes.

Related Documentation

- Understanding Chassis Clustering on Branch SRX Series Services Gateways on page 2
- Upgrading the Software Image on a Chassis Cluster on page 49
- Using SNMP to Manage a Chassis Cluster on page 47
- Using In-band Management Through Network and Security Manager on page 45
- Upgrading a Chassis Cluster on page 41

Using In-Band Management Through Network and Security Manager

Management of SRX Series gateways in cluster configurations using Network and Security Manager is modeled after the management of ScreenOS devices connected using the NetScreen Redundancy Protocol (NSRP), where Network and Security Manager connects to each member forming a high availability pair independently. However, other Junos OS-based devices running in high availability mode can be managed through Network and Security Manager using a single connection. In particular, Network and Security Manager can manage Juniper Networks EX Series Ethernet Switches with Virtual Chassis technology by connecting to the primary node only. In this case, configuration and monitoring of the chassis is done through this single connection.

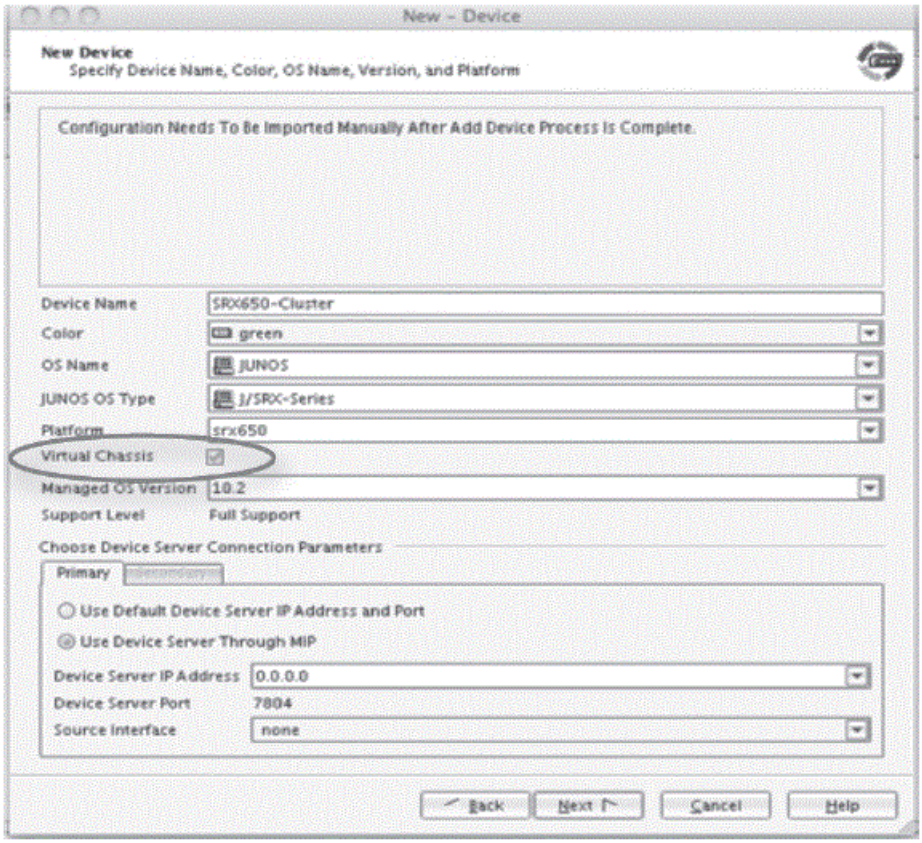
Network and Security Manager versions 2010.2 and later have the ability to manage a branch SRX Series cluster using only a single connection to the primary node. This change requires modifications to both chassis cluster devices so that they identify themselves to Network and Security Manager as a Virtual Chassis. For backwards compatibility purposes, clusters identify to Network and Security Manager as a chassis cluster by default, and it is expected that they are managed through the fxp0 interfaces.

The default behavior can be changed in the device by adding the following configuration to the cluster:

```
user@host# set chassis cluster network-management cluster-master
```

Adding the device to Network and Security Manager is similar to adding an EX Series Virtual Chassis. Simply check the **virtual-chassis** checkbox when adding the cluster. Note that the cluster must be added as a single node, and not as a chassis cluster.

Figure 11: Adding a Cluster as a Virtual Chassis in Network and Security Manager



The hardware inventory displays the chassis serial number of the primary node, and a failover results in an update reflecting the serial number change.

Most configuration and monitoring options are supported, with the following exceptions:

- Chassis inventory displays **sub-component** instead of **FPC**.
- The **chassis serial number** as obtained from cached copy in Network and Security Manager from **get-system-information** contains old information and is not correct.
- A software update of both devices through Network and Security Manager is not supported.
- The Virtual Chassis status view shows no valid information.
- The license inventory shows information only about the primary node.
- The hardware inventory gets out of sync when the primary node is rebooted.
- Reboot commands sent through Network and Security Manager are only applied on the primary node.

- Only control plane logs from the primary node are sent to Network and Security Manager.
- Data plane logs (like session logs or IDP attacks) can be sent from both nodes directly to Network and Security Manager in structured-syslog format.

When updating IDP signatures, Network and Security Manager pushes the security package to the primary node, after which it sends a remote procedure call (RPC) to the cluster to trigger an upgrade. Under normal circumstances, only the primary node gets updated. To overcome this limitation, a Junos OS script has been developed that takes care of updating the secondary node automatically, after the primary has been updated.

For more information about managing chassis clusters, see *Best Practices for SRX Series Chassis Cluster Management* at http://www.juniper.net/techpubs/en_US/release-independent/nce/information-products/topic-collections/nce/srx-policy-memory-best/srx-policy-memory-best.pdf

For more information about sending system log files to Network and Security Manager, see *Configuring a Branch SRX Virtual Chassis to Send Data Plane System Log Messages to NSM* at http://www.juniper.net/techpubs/en_US/release-independent/nce/information-products/topic-collections/nce/srx-data-plane-syslog-to-nsm/srx-data-plane-syslog-to-nsm.pdf.

Related Documentation

- Understanding Chassis Clustering on Branch SRX Series Services Gateways on page 2
- Example: Configuring Chassis Clusters on an SRX Services Gateway for the Branch on page 8
- Example: Configuring an Active/Passive Chassis Cluster on page 19
- Example: Configuring an Active/Passive Chassis Cluster with Asymmetric Routing on page 32
- Upgrading the Software Image on a Chassis Cluster on page 49

Using SNMP to Manage a Chassis Cluster

Just like when you use SSH or Telnet, the primary device in a chassis cluster can answer SNMP queries and generate SNMP traps for both nodes.

To display the cluster information, use the **snmpwalk** command on a network management station. The command returns information about both nodes.

To display the interface descriptions of a cluster, enter the following:

```
[labuser@centos-1 ~]$ snmpwalk -v 2c -c public 10.1.1.34 ifDescr
IF-MIB::ifDescr.1 = STRING: fxp0
IF-MIB::ifDescr.2 = STRING: fxp1
IF-MIB::ifDescr.4 = STRING: lsi
IF-MIB::ifDescr.5 = STRING: dsc
IF-MIB::ifDescr.6 = STRING: lo0
IF-MIB::ifDescr.7 = STRING: tap
IF-MIB::ifDescr.8 = STRING: gre
IF-MIB::ifDescr.9 = STRING: ipip
IF-MIB::ifDescr.10 = STRING: pime
```

```
IF-MIB::ifDescr.11 = STRING: pimd
IF-MIB::ifDescr.12 = STRING: mtun
IF-MIB::ifDescr.13 = STRING: fxp0.0
IF-MIB::ifDescr.14 = STRING: fxp1.0
IF-MIB::ifDescr.21 = STRING: lo0.16384
IF-MIB::ifDescr.22 = STRING: lo0.16385
IF-MIB::ifDescr.116 = STRING: pp0
IF-MIB::ifDescr.123 = STRING: st0
IF-MIB::ifDescr.159 = STRING: reth1.0
IF-MIB::ifDescr.160 = STRING: reth0.0
IF-MIB::ifDescr.162 = STRING: reth0
IF-MIB::ifDescr.163 = STRING: reth1
IF-MIB::ifDescr.172 = STRING: vlan
IF-MIB::ifDescr.501 = STRING: ge-0/0/0
IF-MIB::ifDescr.502 = STRING: ge-0/0/1
IF-MIB::ifDescr.503 = STRING: ge-0/0/1.0
IF-MIB::ifDescr.504 = STRING: ge-3/0/0
IF-MIB::ifDescr.505 = STRING: ge-3/0/0.0
IF-MIB::ifDescr.506 = STRING: ge-3/0/1
IF-MIB::ifDescr.507 = STRING: ge-3/0/1.0
IF-MIB::ifDescr.508 = STRING: ge-3/0/2
IF-MIB::ifDescr.509 = STRING: ge-3/0/3
IF-MIB::ifDescr.510 = STRING: ge-3/0/4
IF-MIB::ifDescr.511 = STRING: ge-3/0/5
IF-MIB::ifDescr.512 = STRING: ge-3/0/6
IF-MIB::ifDescr.513 = STRING: ge-3/0/7
IF-MIB::ifDescr.514 = STRING: fab1.0
IF-MIB::ifDescr.515 = STRING: fab1
IF-MIB::ifDescr.516 = STRING: ge-4/0/0
IF-MIB::ifDescr.517 = STRING: ge-4/0/1
IF-MIB::ifDescr.518 = STRING: ge-4/0/1.0
IF-MIB::ifDescr.519 = STRING: ge-7/0/0
IF-MIB::ifDescr.520 = STRING: ge-7/0/1
IF-MIB::ifDescr.521 = STRING: ge-7/0/0.0
IF-MIB::ifDescr.522 = STRING: ge-7/0/2
IF-MIB::ifDescr.523 = STRING: ge-7/0/3
IF-MIB::ifDescr.524 = STRING: ge-7/0/4
IF-MIB::ifDescr.525 = STRING: ge-7/0/5
IF-MIB::ifDescr.526 = STRING: ge-7/0/1.0
IF-MIB::ifDescr.527 = STRING: ge-7/0/6
IF-MIB::ifDescr.528 = STRING: ge-7/0/7
IF-MIB::ifDescr.529 = STRING: t1-6/0/0
IF-MIB::ifDescr.530 = STRING: t1-6/0/1
IF-MIB::ifDescr.531 = STRING: fab0
IF-MIB::ifDescr.532 = STRING: fab0.0
```

Related Documentation

- Understanding Chassis Clustering on Branch SRX Series Services Gateways on page 2
- Example: Configuring Chassis Clusters on an SRX Services Gateway for the Branch on page 8
- Example: Configuring an Active/Passive Chassis Cluster on page 19
- Example: Configuring an Active/Passive Chassis Cluster with Asymmetric Routing on page 32
- Upgrading the Software Image on a Chassis Cluster on page 49

- In-Band Management of Chassis Clusters on page 41
- Managing a Chassis Cluster on page 43

Upgrading the Software Image on a Chassis Cluster

Upgrading a chassis cluster is a simple procedure, but note that a service disruption of about 3 to 5 minutes occurs during this process. To upgrade the cluster, perform the following tasks:

For information about an in-service software upgrade procedure, see https://www.juniper.net/techpubs/en_US/junos12.2/topics/reference/command-summary/request-system-software-in-service-upgrade.html

- Load the new image file on node0.
- Perform the image upgrade, without rebooting the node by entering the **request system software add** ***image name*** command.
- Copy the new image file to node1 by entering the **file copy** ***image name*** command.
- Perform the image upgrade in node1 by entering the **request system software add** ***image name*** command.
- Reboot each node by entering the **request system reboot** command.

To upgrade Junos OS, connect to each node individually and copy the image to the primary node using FTP or SCP (provided that FTP or SSH are enabled). After the image is copied to the primary node, copy the file into the secondary node.

The following procedure details how to upgrade both nodes of a cluster managed in-band:

1. Copy the software image into the primary node using your preferred method..
2. Copy the files from the primary node to the backup node using the **file copy** command (it might take a few minutes).

   In this example the image is copied to the **/var/tmp** directory in node0.

   user@node0# run file copy /var/tmp/junos-jsr-10.2R1.3-domestic.tgz node1:/var/tmp
3. Log in to the backup node and load the new image.

```
user@node0#  run request routing-engine login node 1
--- JUNOS 10.1-20100515.0 built 2010-05-15 06:07:46 UTC

{secondary:node1}

user@node1> request system software add /var/tmp/junos-jsr-10.2R1.3domestic.tgz no-copy
unlink

NOTICE: Validating configuration against junos-jsr-10.2R1.3-domestic.tgz.
NOTICE: Use the 'no-validate' option to skip this if desired.
Checking compatibility with configuration
Initializing...
Verified manifest signed by PackageProduction_10_1_0
Verified junos-10.1-20100515.0-domestic signed by PackageProduction_10_1_0
Using /var/tmp/junos-jsr-10.2R1.3-domestic.tgz
Checking junos requirements on /
```

```
Saving boot file package in /var/sw/pkg/junos-boot-jsr-10.2R1.3.tgz
Verified manifest signed by PackageProduction_10_2_0
Hardware Database regeneration succeeded
Validating against /config/juniper.conf.gz
cp: /cf/var/validate/chroot/var/etc/resolv.conf and /etc/resolv.conf are
identical (not copied).
cp: /cf/var/validate/chroot/var/etc/hosts and /etc/hosts are identical (not
copied).
Network security daemon: warning: You have enabled/disabled inet6 flow.
Network security daemon: You must reboot the system for your change to take
effect.
Network security daemon: If you have deployed a cluster, be sure to reboot all
nodes.
mgd: commit complete
Validation succeeded
Validating against /config/rescue.conf.gz
Network security daemon: warning: You have enabled/disabled inet6 flow.
Network security daemon: You must reboot the system for your change to take
effect.
Network security daemon: If you have deployed a cluster, be sure to reboot all
nodes.
mgd: commit complete
Validation succeeded
Installing package '/var/tmp/junos-jsr-10.2R1.3-domestic.tgz' ...
Verified junos-boot-jsr-10.2R1.3.tgz signed by PackageProduction_10_2_0
Verified junos-jsr-10.2R1.3-domestic signed by PackageProduction_10_2_0
Available space: 333778 require: 4160
Saving boot file package in /var/sw/pkg/junos-boot-jsr-10.2R1.3.tgz
JUNOS 10.2R1.3 will become active at next reboot
WARNING: A reboot is required to load this software correctly
WARNING: Use the 'request system reboot' command
WARNING: when software installation is complete
Saving state for rollback ...
Removing /var/tmp/junos-jsr-10.2R1.3-domestic.tgz
{secondary:node1}
```

```
user@node1>exit
```

4. Upgrade the primary node by entering the **request system software add** command.

```
user@node0# run request system software add /var/tmp/junos-jsr-10.2R1.3- domestic.tgz
no-copy unlink

NOTICE: Validating configuration against junos-jsr-10.2R1.3-domestic.tgz.
NOTICE: Use the 'no-validate' option to skip this if desired.
Checking compatibility with configuration
Initializing...
Verified manifest signed by PackageProduction_10_1_0
Verified junos-10.1-20100515.0-domestic signed by PackageProduction_10_1_0
Using /var/tmp/junos-jsr-10.2R1.3-domestic.tgz
Checking junos requirements on /
Saving boot file package in /var/sw/pkg/junos-boot-jsr-10.2R1.3.tgz
Verified manifest signed by PackageProduction_10_2_0
Hardware Database regeneration succeeded
Validating against /config/juniper.conf.gz
cp: /cf/var/validate/chroot/var/etc/resolv.conf and /etc/resolv.conf are
identical (not copied).
cp: /cf/var/validate/chroot/var/etc/hosts and /etc/hosts are identical (not
copied).
Network security daemon: warning: You have enabled/disabled inet6 flow.
Network security daemon: You must reboot the system for your change to take
effect.
Network security daemon: If you have deployed a cluster, be sure to reboot all
```



```
nodes.
mgd: commit complete
Validation succeeded
Validating against /config/rescue.conf.gz
mgd: commit complete
Validation succeeded
Installing package '/var/tmp/junos-jsr-10.2R1.3-domestic.tgz' ...
Verified junos-boot-jsr-10.2R1.3.tgz signed by PackageProduction_10_2_0
Verified junos-jsr-10.2R1.3-domestic signed by PackageProduction_10_2_0
Available space: 332709 require: 4160
Saving boot file package in /var/sw/pkg/junos-boot-jsr-10.2R1.3.tgz
JUNOS 10.2R1.3 will become active at next reboot
WARNING: A reboot is required to load this software correctly
WARNING: Use the 'request system reboot' command
WARNING: when software installation is complete
Saving state for rollback ...
Removing /var/tmp/junos-jsr-10.2R1.3-domestic.tgz

{primary:node0}[edit]

user@node0#
```

5. Login to node1.

```
user@node0# run request routing-engine login node 1

--- JUNOS 10.1-20100515.0 built 2010-05-15 06:07:46 UTC
{secondary:node1}
```

6. Reboot node1 by entering the **request system reboot** command.

```
user@node1> request system reboot

Reboot the system ? [yes,no] (no) yes

Shutdown NOW!
[pid 6456]

{secondary:node1}
user@node1>
*** FINAL System shutdown message from labuser@J2320-2 ***
System going down IMMEDIATELY

{secondary:node1}
```

7. Log out of node1 by entering the **exit** command.

```
user@node1> exit

rlogin: connection closed

{primary:node0}[edit]
```

8. Reboot node0 by entering the **request system reboot** command.

```
user@node1# run request system reboot

Reboot the system ? [yes,no] (no) yes

Shutdown NOW!
[pid 7048]

{primary:node0}[edit]
user@node0#
```

```
*** FINAL System shutdown message from user@node0 ***
System going down IMMEDIATELY
```

After both nodes are rebooted, the cluster restarts with the new image.

Related Documentation

- Understanding Chassis Clustering on Branch SRX Series Services Gateways on page 2
- Upgrading a Chassis Cluster on page 41
- Understanding the Chassis Cluster High Availability Model on page 41
- Example: Configuring Chassis Clusters on an SRX Services Gateway for the Branch on page 8
- Example: Configuring an Active/Passive Chassis Cluster on page 19
- Example: Configuring an Active/Passive Chassis Cluster with Asymmetric Routing on page 32