Engineering
Simplicity

Junos® OS

Traffic Sampling, Forwarding, and Monitoring User Guide

Published
2025-12-24

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

YEAR 2000 NOTICE

END USER LICENSE AGREEMENT

Table of Contents

About This Guide

This guide provides information about traffic sampling, which allows you to sample IP traffic based on particular input interfaces and various fields in the packet header. You can also use traffic sampling to monitor interfaces, protocols, and addresses.

The guide also offers information about per-flow load balancing, port mirroring, and domain name system (DNS) or Trivial File Transfer Protocol (TFTP) forwarding.

Traffic sampling and forwarding are supported only on routers equipped with an Internet Processor II application-specific integrated circuit (ASIC). To determine whether a routing platform has an Internet Processor II ASIC, use the `show chassis hardware` command.

1

CHAPTER

Overview

IN THIS CHAPTER

Traffic Sampling, Forwarding, and Monitoring Overview

Traffic sampling allows you to sample IP traffic based on particular input interfaces and various fields in the packet header. You can also use traffic sampling to monitor any combination of specific logical interfaces, specific protocols on one or more interfaces, a range of addresses on a *logical interface*, or individual IP addresses. Information about the sampled packets is saved to files on the router's hard disk.

Traffic sampling is not meant to capture all packets received by a router. We do not recommend excessive sampling (a rate greater than 1/1000 packets), because it can increase the load on your processor. If you need to set a higher sampling rate to diagnose a particular problem or type of traffic received, we recommend that you revert to a lower sampling rate after you discover the problem or troublesome traffic. In addition, traffic sampling and forwarding are supported only on routers equipped with an Internet Processor II application-specific integrated circuit (ASIC). To determine whether a routing platform has an Internet Processor II ASIC, use the `show chassis hardware` command.

Junos OS supports both per-packet and per-flow load balancing. In Junos OS Release 9.0 and later, you can configure per-prefix load balancing. This feature enables the router to elect the next hop independent of the route chosen by other routers. The result is a better utilization of available links. Likewise, you can configure Junos OS so that, for the active route, all next-hop addresses for a destination are installed in the forwarding table. This is called per-packet load balancing, which you can use to spread traffic across multiple paths between routers.

With forwarding policies, you can configure per-flow load balancing, *port mirroring*, and domain name system (DNS) or Trivial File Transfer Protocol (TFTP) forwarding.

NOTE: On SRX devices, sampling/J-Flow cannot be done for the packets that are handled by PowerMode IPsec (PMI) or Services Offload (SOF). Sampling/J-Flow counters will not provide the exact number of packets that are received on the interface.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
9.0	In Junos OS Release 9.0 and later, you can configure per-prefix load balancing.

2

CHAPTER

Collecting Traffic Samples for Network Monitoring

IN THIS CHAPTER

Traffic Sampling Configuration

To configure traffic sampling, include the `sampling` statement at the `[edit forwarding-options]` hierarchy level:

```
[edit forwarding-options]
sampling {
    disable;
    family (inet | inet6 | mpls) {
        disable;
        output {
            aggregate-export-interval seconds;
            extension-service service-name;
            file {
                disable;
                filename filename;
                files number;
                size bytes;
                (stamp | no-stamp);
                (world-readable | no-world-readable);
            }
            flow-active-timeout seconds;
            flow-inactive-timeout seconds;
            flow-server hostname {
                aggregation {
                    autonomous-system;
                    destination-prefix;
                    protocol-port;
                    source-destination-prefix {
                        caida-compliant;
                    }
                    source-prefix;
                }
                autonomous-system-type (origin | peer);
                (local-dump | no-local-dump);
                port port-number;
                source-address address;
                version format;
                version9 {
                    template template-name;
                }
            }
            interface interface-name {
                engine-id number;
                engine-type number;
                source-address address;
            }
        }
    }
    input {
        max-packets-per-second number;
        maximum-packet-length bytes;
        rate number;
        run-length number;
    }
    traceoptions {
        file filename {
            files number;
            size bytes;
            (world-readable | no-world-readable);
        }
    }
}
```

Minimum Traffic Sampling Configuration

To configure traffic sampling, you must perform at least the following tasks:

1. Create a firewall filter to apply to the logical interfaces being sampled by including the `filter` statement at the `[edit firewall family family-name]` hierarchy level. In the filter `then` statement, you must specify the action modifier **sample** and the action **accept**.

```
[edit firewall family family-name]
filter filter-name {
    term term-name {
        then {
            sample;
            accept;
        }
    }
}
```

2. Apply the filter to the interfaces on which you want to sample traffic:

```
[edit interfaces]
interface-name {
    unit logical-unit-number {
        family family-name {
            filter {
                input filter-name;
            }
            address address {
                destination destination-address;
            }
        }
    }
}
```

3. Enable sampling and specify a nonzero sampling rate:

```
[edit forwarding-options]
sampling {
    input {
        rate number;
    }
}
```

Configuring Traffic Sampling

On routing platforms containing a Monitoring Services PIC or an Adaptive Services PIC, you can configure traffic sampling for traffic passing through the routing platform. You can also configure traffic sampling of MPLS traffic.

To configure traffic sampling on a logical interface:

1. Include the `input` statement at the `[edit forwarding-options sampling]` hierarchy level, for example:

   ```
   [edit forwarding-options sampling]
   input {
       max-packets-per-second number;
       maximum-packet-length bytes
       rate number;
       run-length number;
   }
   ```

   NOTE: On SRX platforms:

   - `max-packets-per-second` can have a range of (1..160000)
   - `rate` can have a range of (1..16000000)
   - `run-length` can have a range of (0..20)

   You can export flow records generated by inline flow monitoring to four collectors under a family with the same source IP address. The Packet Forwarding Engine (PFE) can export the flow record, flow record template, option data, and, option data template packet to all configured collectors. You can configure the multiple collectors at the `[edit forwarding-options sampling instance` *`instance name`*`]` hierarchy level.

   NOTE: You cannot change the source IP address for collectors under the same family.

2. Specify the threshold traffic value by using the `max-packets-per-second` statement. The value is the maximum number of packets to be sampled, beyond which the sampling mechanism begins dropping packets. The range is 0 through 65,535. A value of 0 instructs the Packet Forwarding Engine not to sample any packets. The default value is 1000.

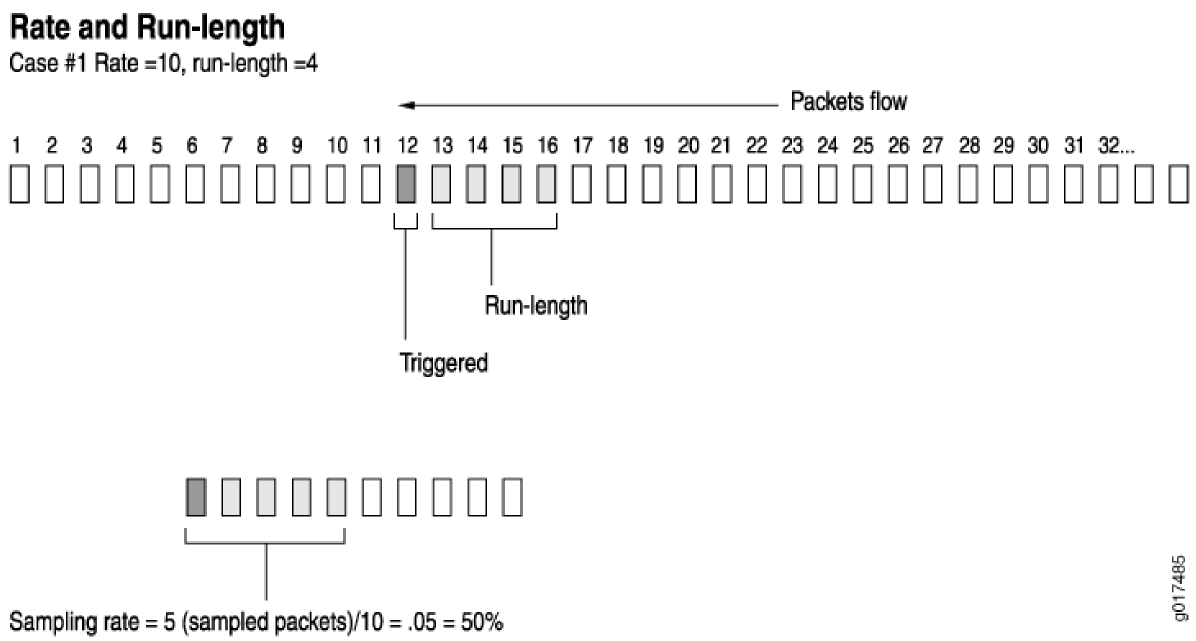
   NOTE: This statement is not valid for port mirroring.

3. Specify the maximum length of the sampled packet by using the `maximum-packet-length` *`bytes`* statement. For *bytes*, specify a value.

   NOTE: For MX-Series devices with Modular Port Concentrators (MPCs) port-mirrored or sampled packets can be truncated (or clipped) to any length in the range of 1 to 255 bytes. Only 1 to 255 are valid values for packet truncation on these devices. For other devices, the range is from 0 to 9216. A maximum-packet-length value of zero represents that truncation is disabled, and the entire packet is mirrored or sampled.

4. Specify the sampling rate by setting the values for *rate* and *run-length* (see Figure 1 on page 8).

Figure 1: Configure Sampling Rate



The forwarding plane provides support for random sampling that can be configured through the *rate* or `run-length` statement. The `rate` statement sets the ratio of the number of packets to be sampled on an average. For example, if you configure a rate of 10, on average every tenth packet (1 packet out of 10) is sampled.

The `run-length` statement specifies the number of matching packets to sample following the initial one-packet trigger event. Configuring a run length greater than 0 allows you to sample packets following those already being sampled.

NOTE: The `run-length` statement is not supported on MX Series routers with Modular Port Concentrators (MPCs).

You can also send the sampled packets to a specified host using the cflowd version 5 and 8 formats or the version 9 format as defined in RFC 3954. For more information, see "Directing Traffic Sampling Output to a Server Running the cflowd Application" on page 12 and "Collecting Traffic Sampling Output in the Cisco Systems NetFlow Services Export Version 9 Format" on page 16.

Junos OS does not sample packets originating from the router. If you configure a sampling filter and apply it to the output side of an interface, then only the transit packets going through that interface are sampled. Packets that are sent from the Routing Engine to the Packet Forwarding Engine are not sampled.

When you apply a firewall filter to a loopback interface, the filter might block responses from the Monitoring Services PIC. To allow responses from the Monitoring Services PIC to pass through for sampling purposes, configure a term in the firewall filter to include the Monitoring Services PIC's IP address.

NOTE: Targeted broadcast does not work when the targeted broadcast option `forward-and-send-to-re` and the traffic sampling option `sampling` are configured on the same egress interface of an MX960 router. To overcome this scenario, you must either disable one of the these options or enable the `sampling` option with the targeted broadcast option `forward-only` on the egress interface. For information about targeted broadcast, see *Understanding Targeted Broadcast*.

RELATED DOCUMENTATION

Guidelines for Configuring Firewall Filters

Guidelines for Applying Standard Firewall Filters

Disabling Traffic Sampling

To explicitly disable traffic sampling on the router, include the `disable` statement at the `[edit forwarding-options sampling]` hierarchy level:

```
[edit forwarding-options sampling]
disable;
```

NOTE: The `disable` statement at the `[edit forwarding-options sampling]` hierarchy level disables only Routing Engine-based sampling. To disable PIC-based sampling and inline sampling, include the `disable` statement at the `[edit forwarding-options sampling instance` *`instance-name`*`]` hierarchy level.

Collecting Traffic Sampling Output in a File

IN THIS SECTION

You configure traffic sampling results to a file in the **/var/tmp** directory. To collect the sampled packets in a file, include the `file` statement at the `[edit forwarding-options sampling output]` hierarchy level:

```
[edit forwarding-options sampling family family-name output]
file <disable> filename filename <files number> <size bytes> <stamp | no-stamp > <world-readable |
no-world-readable>;
```

To configure the period of time before an active flow is exported, include the `flow-active-timeout` statement at the `[edit forwarding-options sampling output family (inet | inet6 | mpls)]` hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output]
flow-active-timeout seconds;
```

To configure the period of time before a flow is considered inactive, include the `flow-inactive-timeout` statement at the `[edit forwarding-options sampling output]` hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output]
flow-inactive-timeout seconds;
```

To configure the interface that sends out monitored information, include the `interface` statement at the `[edit forwarding-options sampling output]` hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output]
interface interface-name {
    engine-id number;
    engine-type number;
    source-address address;
}
```

NOTE: This feature is not supported with the version 9 template format. You must send traffic flows collected using version 9 to a server. For more information see "Collecting Traffic Sampling Output in the Cisco Systems NetFlow Services Export Version 9 Format" on page 16.

Traffic Sampling Output Format

Traffic sampling output is saved to an ASCII text file. The following is an example of the traffic sampling output that is saved to a file in the **/var/tmp** directory. Each line in the output file contains information for one sampled packet. You can optionally display a timestamp for each line.

The column headers are repeated after each group of 1000 packets.

```
# Apr  7 15:48:50
Time                    Dest           Src Dest Src Proto TOS Pkt Intf  IP   TCP
                        addr          addr port port          len num frag flags
Apr 7 15:48:54 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:55 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:56 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:57 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
Apr 7 15:48:58 192.168.9.194 192.168.9.195   0    0   1   0x0  84  8   0x0   0x0
```

The output contains the following fields:

- **Time**—Time at which the packet was received (displayed only if you include the `stamp` statement in the configuration)
- **Dest addr**—Destination IP address in the packet
- **Src addr**—Source IP address in the packet
- **Dest port**—Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port for the destination address
- **Src port**—TCP or UDP port for the source address
- **Proto**—Packet's protocol type
- **TOS**—Contents of the type-of-service (ToS) field in the IP header
- **Pkt len**—Length of the sampled packet, in bytes

- **Intf num**—Unique number that identifies the sampled logical interface
- **IP frag**—IP fragment number, if applicable
- **TCP flags**—Any TCP flags found in the IP header

To set the timestamp option for the file **my-sample**, enter the following:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output file]
user@host# set filename my-sample files 5 size 2m world-readable stamp;
```

Whenever you toggle the timestamp option, a new header is included in the file. If you set the **stamp** option, the **Time** field is displayed.

```
# Apr  7 15:48:50
# Time              Dest         Src  Dest   Src Proto  TOS   Pkt  Intf    IP   TCP
#                   addr        addr  port  port              len   num  frag flags
# Feb  1 20:31:21
#                   Dest         Src  Dest   Src Proto  TOS   Pkt  Intf    IP   TCP
#                   addr        addr  port  port              len   num  frag flags
```

Directing Traffic Sampling Output to a Server Running the cflowd Application

IN THIS SECTION

You can collect an aggregate of sampled flows and send the aggregate to a specified host that runs the cflowd application available from the Cooperative Association for Internet Data Analysis (CAIDA) (**http://www.caida.org**). By using cflowd, you can obtain various types of byte and packet counts of flows through a router.

The cflowd application collects the sampled flows over a period of 1 minute. At the end of the minute, the number of samples to be exported are divided over the period of another minute and are exported over the course of the same minute.

Before you can perform flow aggregation, the routing protocol process must export the autonomous system (AS) path and routing information to the sampling process. To do this, include the `route-record` statement:

```
route-record;
```

You can include this statement at the following hierarchy levels:

- `[edit routing-options]`
- `[edit routing-instances` *`routing-instance-name`* `routing-options]`

By default, flow aggregation is disabled. To enable the collection of flow aggregates, include the `flow-server` statement at the `[edit forwarding-options sampling output]` hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output ]
flow-server hostname {
    aggregation {
        autonomous-system;
        destination-prefix;
        protocol-port;
        source-destination-prefix {
            caida-compliant;
        }
        source-prefix;
    }
    autonomous-system-type (origin | peer);
    (local-dump | no-local-dump);
    port port-number;
    source-address address;
    version format;
}
```

In the cflowd statement, specify the name, identifier, and source-address of the host that collects the flow aggregates. You must also include the UDP port number on the host and the **version**, which gives the format of the exported cflowd aggregates. To specify an IPv4 source address, include the `source-address` statement. To collect cflowd records in a log file before exporting, include the `local-dump` statement. To specify the cflowd version number, include the `version` statement. The cflowd version is either 5 or 8.

You can specify both host (cflowd) sampling and port mirroring in the same configuration. You can perform RE-sampling and port mirroring actions simultaneously. However, you cannot perform PIC-sampling and port mirroring actions simultaneously.

To specify aggregation of specific types of traffic, include the `aggregation` statement. This conserves memory and bandwidth enabling cflowd to export targeted flows rather than all the aggregated

NOTE: Aggregation is valid only if cflowd version 8 is specified.

To specify a flow type, include the `aggregation` statement at the `[edit forwarding-options sampling output cflowd` *`hostname`*`]` hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output hostname]
aggregation {
    source-destination-prefix;
}
```

You specify the aggregation type using one of the following options:

- **autonomous-system**—Aggregate by AS number; may require setting the separate cflowd `autonomous-system-type` statement to include either **origin** or **peer** AS numbers. The **origin** option specifies to use the origin AS of the packet source address in the Source Autonomous System cflowd field. The **peer** option specifies to use the peer AS through which the packet passed in the Source Autonomous System cflowd field. By default, **cflowd** exports the origin AS number.

- **destination-prefix**—Aggregate by destination prefix (only).

- **protocol-port**—Aggregate by protocol and port number; requires setting the separate `cflowd port` statement.

- **source-destination-prefix**—Aggregate by source and destination prefix. Version 2.1b1 of CAIDA's cflowd application does not record source and destination mask length values in compliance with CAIDA's *cflowd Configuration Guide*, dated August 30, 1999. If you configure the `caida-compliant` statement, Junos OS complies with Version 2.1b1 of cflowd. If you do not include the `caida-compliant` statement in the configuration, Junos OS records source and destination mask length values in compliance with the *cflowd Configuration Guide*.

- **source-prefix**—Aggregate by source prefix (only).

Collection of sampled packets in a local ASCII file is not affected by the `cflowd` statement.

Debugging cflowd Flow Aggregation

To collect the cflowd flows in a log file before they are exported, include the **local-dump** option at the `[edit forwarding-options sampling output cflowd hostname]` hierarchy level:

```
[edit forwarding-options sampling family (inet | inet6 | mpls) output flow-server hostname]
local-dump;
```

By default, the flows are collected in **/var/log/sampled**; to change the filename, include the `filename` statement at the `[edit forwarding-options sampling traceoptions]` hierarchy level. For more information about changing the filename, see "Collecting Traffic Sampling Output in a File" on page 10.

NOTE: Because the **local-dump** option adds extra overhead, you should use it only while debugging cflowd problems, not during normal operation.

The following is an example of the flow information. The AS number exported is the origin AS number. All flows that belong under a cflowd header are dumped, followed by the header itself:

```
Jun 27 18:35:43 v5 flow entry
Jun 27 18:35:43    Src addr: 10.53.127.1
Jun 27 18:35:43    Dst addr: 10.6.255.15
Jun 27 18:35:43    Nhop addr: 192.168.255.240
Jun 27 18:35:43    Input interface: 5
Jun 27 18:35:43    Output interface: 3
Jun 27 18:35:43    Pkts in flow: 15
Jun 27 18:35:43    Bytes in flow: 600
Jun 27 18:35:43    Start time of flow: 7230
Jun 27 18:35:43    End time of flow: 7271
Jun 27 18:35:43    Src port: 26629
Jun 27 18:35:43    Dst port: 179
Jun 27 18:35:43    TCP flags: 0x10
Jun 27 18:35:43    IP proto num: 6
Jun 27 18:35:43    TOS: 0xc0
Jun 27 18:35:43    Src AS: 64496
Jun 27 18:35:43    Dst AS: 64511
Jun 27 18:35:43    Src netmask len: 16
Jun 27 18:35:43    Dst netmask len: 0
```

[... 41 more **v5 flow** entries; then the following header:]

```
Jun 27 18:35:43 cflowd header:
Jun 27 18:35:43   Num-records: 42
Jun 27 18:35:43   Version: 5
Jun 27 18:35:43   Flow seq num: 118
Jun 27 18:35:43   Engine id: 0
Jun 27 18:35:43   Engine type: 3
```

Collecting Traffic Sampling Output in the Cisco Systems NetFlow Services Export Version 9 Format

IN THIS SECTION

In Junos OS Release 8.3 and later, you can collect a record of sampled flows using the version 9 format as defined in RFC 3954, *Cisco Systems NetFlow Services Export Version 9*. Version 9 uses templates to collect a set of sampled flows and send the record to a specified host.

You configure the version 9 template used to collect a record of sampled flows at the `[edit services monitoring]` hierarchy level. For more information, see the Junos OS Services Interfaces Library for Routing Devices and the Monitoring, Sampling, and Collection Services Interfaces User Guide.

To enable the collection of traffic flows using the version 9 format, include the `version9` statement at the `[edit forwarding-options sampling family family-name output flow-server hostname]` hierarchy level:

```
[edit forwarding-options sampling family family-name output flow-server hostname]
version9 {
    template template-name;
}
```

template-name is the name of the version 9 template configured at the `[edit services monitoring]` hierarchy level.

You configure traffic sampling at the `[edit forwarding-options sampling input]` hierarchy level. In Junos OS Release 8.3 and later, you can configure sampling for MPLS traffic as well as IPv4 traffic. You can define a version 9 flow record template suitable for IPv4 traffic, MPLS traffic, or a combination of the two. You can sample packets from both the **inet** and **mpls** protocol families at the same time. You can configure sampling for peer AS billing traffic for the **inet** and **ipv6** protocols only. For more information about how to configure traffic sampling, see "Configuring Traffic Sampling" on page 6.

The following restrictions apply to configuration of the version 9 format:

- You can configure only one host to collect traffic flows using the version 9 format. Configure the host at the `[edit forwarding-options sampling family` *`family-name`* `output flow-server` *`hostname`*`]` hierarchy level.
- You cannot specify both the version 9 format and cflowd versions 5 and 8 formats in the same configuration. For more information about how to configure flow monitoring using cflowd version 8, see "Directing Traffic Sampling Output to a Server Running the cflowd Application" on page 12.
- Any values for **flow-active-timeout** and **flow-inactive-timeout** that you configure at the `[edit forwarding-options sampling output]` hierarchy level are overridden by the values configured in the version 9 template.
- Version 9 does not support Routing Engine-based sampling. You cannot configure version 9 to send traffic sampling result to a file in the **/var/tmp** directory.

Example: Configuring Active Flow Monitoring Using Version 9

In this example, you enable active flow monitoring using version 9. You specify a template **mpls** that you configure at the `[edit services monitoring]` hierarchy level. You also configure the traffic family **mpls** to sample MPLS packets.

```
[edit forwarding-options]
sampling {
    input {
        rate 1;
        run-length;
    }
    family inet {
        output {
            flow-server 10.60.2.1 { # The IP address and port of the host
                port 2055; # that collects the sampled traffic flows.
                source-address 3.3.3.1;
                version9 {
                    template mpls; # Version 9 records are sent
                } # using the template named mpls
            }
        }
    }
}
```

Example: Sampling a Single SDH Interface

The following configuration gathers statistical sampling information from a small percentage of all traffic on a single SDH interface and collects it in a file named **sdh-samples.txt**.

Create the filter:

```
[edit firewall family inet]
filter {
    sample-sdh {
        then {
            sample;
            accept;
        }
    }
}
```

Apply the filter to the SDH interface:

```
[edit interfaces]
so-0/0/1 {
    unit 0 {
        family inet {
            filter {
                input sample-sdh;
            }
            address 10.127.68.254/32 {
                destination 10.127.74.7;
            }
        }
    }
}
```

Finally, configure traffic sampling:

```
[edit forwarding-options]
sampling {
    input {
        rate 100;
        run-length 2;
    }
    family inet {
        output {
            file {
                filename sdh-samples.txt;
                files 40;
                size 5m;
            }
        }
}
```

Example: Sampling All Traffic from a Single IP Address

The following configuration gathers statistical information about every packet entering the router on a specific Gigabit Ethernet port originating from a single source IP address of **10.45.92.31**, and collects it in a file named **samples-10-45-92-31.txt**.

Create the filter:

```
[edit firewall family inet]
filter one-ip {
    term get-ip {
        from {
            source-address 10.45.92.31;
        }
        then {
            sample;
            accept;
        }
    }
}
```

Apply the filter to the Gigabit Ethernet interface:

```
[edit interfaces]
ge-4/1/1 {
    unit 0 {
        family inet {
            filter {
                input one-ip;
            }
            address 10.45.92.254;
        }
    }
}
```

Finally, gather statistics on all the candidate samples; in this case, gather all statistics:

```
[edit forwarding-options]
sampling {
    input {
        rate 1;
    }
    family inet {
        output {
            file {
                filename samples-215-45-92-31.txt;
                files 100;
                size 100k;
            }
        }
    }
}
```

Example: Sampling All FTP Traffic

The following configuration gathers statistical information about a moderate percentage of packets using FTP in the output path of a specific ethernet interface, and collects the information in a file named **et-ftp-traffic.txt**.

Create a filter:

```
[edit firewall family inet]
filter ftp-stats {
    term ftp-usage {
        from {
            destination-port [ftp ftp-data];
        }
        then {
            sample;
            accept;
        }
    }
}
```

Apply the filter to the interface:

```
[edit interfaces]
et-7/0/2 {
    unit 0 {
        family inet {
            filter {
                input ftp-stats;
            }
            address 10.35.78.254/32 {
                destination 10.35.78.4;
            }
        }
    }
}
```

Finally, gather statistics on 10 percent of the candidate samples:

```
[edit forwarding-options]
sampling {
    input {
        rate 10;
    }
    family inet {
        output {
            file {
                filename et-ftp-traffic.txt;
                files 50;
                size 1m;
            }
        }
    }
}
```

Tracing Traffic-Sampling Operations

Tracing operations track all traffic-sampling operations and record them in a log file in the **/var/log** directory. By default, this file is named **/var/log/sampled**. The default file size is 128 KB, and 10 files are created before the first one gets overwritten.

To trace traffic-sampling operations, include the `traceoptions` statement at the `[edit forwarding-options sampling]` hierarchy level:

```
[edit forwarding-options sampling]
traceoptions {
    file <filename>  <files number> <size bytes> <world-readable | no-world-readable>;
    no-remote-trace;
}
```

3

CHAPTER

Configuring Traffic Forwarding for Network Monitoring

IN THIS CHAPTER

Configuring Traffic Forwarding and Monitoring

To configure forwarding options and traffic monitoring, include statements at the `[edit forwarding-options]` hierarchy level:

```
[edit forwarding-options]
accounting group-name {
    output {
        cflowd [ hostnames ] {
            aggregation {
                autonomous-system;
                destination-prefix;
                protocol-port;
                source-destination-prefix {
                    caida-compliant;
                }
                source-prefix;
            }
            autonomous-system-type (origin | peer);
            port port-number;
            version format;
        }
        flow-active-timeout seconds;
        flow-inactive-timeout seconds;
        interface interface-name {
            engine-id number;
            engine-type number;
            source-address address;
        }
    }
}
enhanced-hash-key {
    family inet {
        gtp-tunnel-endpoint-identifier;
        incoming-interface-index;
        no-destination-port;
        no-source-port;
        type-of-service;
    }
    family inet6 {
        gtp-tunnel-endpoint-identifier;
        incoming-interface-index;
        no-destination-port;
        no-source-port;
        traffic-class;
    }
    family mpls {
        incoming-interface-index;
        label-1-exp;
        no-payload;
    }
    family multiservice {
        incoming-interface-index;
        no-payload;
        outer-priority;
    }
    services-loadbalancing {
        family inet layer-3-services {
            incoming-interface-index;
            source-address;
        }
    }
}
family family-name {
    filter {
        input filter-name;
        output filter-name;
    }
    route-accounting;
}
flood {
    input filter-name;
}
hash-key {
    family inet {
        layer-3;
        layer-4;
    }
    family mpls {
        no-interface-index;
        label-1;
        label-2;
        label-3;
        no-labels;
        no-label-1-exp;
        payload {
            ether-pseudowire;
            ip {
                layer-3-only;
                port-data {
                    source-msb;
                    source-lsb;
                    destination-msb;
                    destination-lsb;
                }
            }
        }
    }
    family multiservice }
        destination-mac;
        label-1;
        label-2;
        payload {
            ip {
                layer-3-only;
            }
        }
        source-mac;
    }
}
helpers {
    bootp {
        client-response-ttl;
        description text-description;
        interface interface-group {
            client-response-ttl number;
            description text-description;
            maximum-hop-count number;
            minimum-wait-time seconds;
            no-listen;
            server address {
                logical-system logical-system-name <routing-instance [ <default> routing-
instance-names ]>;
                routing-instance [ <default> routing-instance-names ];
            }
        }
        maximum-hop-count number;
        minimum-wait-time seconds;
        relay-agent-option;
        server [ addresses ];
    }
    domain {
        description text-description;
        server < [ routing-instance routing-instance-names ] >;
        interface interface-name {
            description text-description;
            no-listen;
            server < [ routing-instance routing-instance-names ] >;
        }
    }
    tftp {
        description text-description;
        server < [ routing-instance routing-instance-names ] >;
        interface interface-name {
            description text-description;
            no-listen;
            server < [ routing-instance routing-instance-names ] >;
        }
    }
    traceoptions {
        file <filename> <files number> <match regular-expression> <size size> <world-readable |
no-world readable>;
        flag flag;
        level severity-level;
        no-remote-trace;
    }
}
load-balance {
    indexed-load-balance;
    per-flow {
        hash-seed number;
    }
    per-prefix {
        hash-seed number;
    }
}
monitoring group-name {
    family inet {
        output {
            cflowd hostname {
                port port-number;
            }
            export-format cflowd-version-5;
            flow-active-timeout seconds;
            flow-export-destination {
                cflowd-collector;
            }
            flow-inactive-timeout seconds;
             interfaceinterface-name {
                engine-id number;
                engine-type number;
                input-interface-index number;
                output-interface-index number;
                source-address address;
            }
        }
    }
}
next-hop-group [ group-names ] {
    interface interface-name {
        next-hop [ addresses ];
    }
}
port-mirroring {
    family (ccc | inet | inet6 | vpls) {
        output {
            interface interface-name {
                next-hop address;
            }
            no-filter-check;
        }
        input {
            maximum-packet-length bytes;
            rate number;
            run-length number;
        }
    }
    traceoptions {
        file <filename> <files number> <match regular-expression> <size bytes> <world-readable | no-
world-readable>;
        no-remote-trace;
    }
}
```

NOTE: When a route pointing to more than one services PIC is available, and with application layer gateways (ALGs) configured, you must always configure the distribution of traffic across PICs based on the source IP address by including the `family inet layer-3-services source-address` statement at the `[edit forwarding-options enhanced-hash-key services-loadbalancing]` hierarchy level for IPv4 traffic and the `family inet6 layer-3-services source-address` statement at the [edit forwarding-options enhanced-hash-key services-loadbalancing] hierarchy level for IPv6 traffic. With ALGs used to manage a parent-child relationship of sessions, both the parent and the child sessions must be processed by the same type of services PIC.

Configuring IPv4 and IPv6 Accounting

IPv4 and IPv6 accounting are disabled by default. But you can enable the accounting by including the `route-accounting` statements at the `[edit forwarding-options family inet4]` and `[edit forwarding-options family inet6]` hierarchy level, as shown here:

```
[edit]
forwarding-options {
    family inet4 {
        route-accounting;
    }
]
```

```
[edit]
forwarding-options {
    family inet6 {
        route-accounting;
    }
]
```

To view the IPv4 or IPv6 statistics for a given physical or logical interface, run the operational command `show interfaces extensive` *`interface`* `| find IPv4` or `show interfaces extensive` *`interface`* `| find IPv6`. Note that the output displays packet and byte counts for transit traffic only. Locally generated packets are not included in the metrics.

```
show interfaces ge-2/0/9 detail | find IPv6
IPv6 transit statistics:
Input  bytes  :         8576802312
Output bytes  :        8991637500
Input  packets:        5787313
Output packets:      5994425
```

Follow the guidelines given below to configure IPv4 and IPv6 statistics accounting:

- The transit rate is calculated in software and not in hardware. So, you can expect some deviation from the line rate. You do not receive the expected rate with lower fps (for example, 5 fps).
- The traffic must be IPv4 or IPv6 at both ingress and egress. Any form of encapsulated traffic or traffic translating to IPv4 is not supported on PTX routers.
- If the GRE tunnel is IPv4 or IPv6 based, this traffic is not accounted in ingress statistics.
- Statistics accounting is supported only for `et` and `ae` interfaces on PTX routers.
- The virtual, software, pseudo, Layer 2, and special type Layer 3 virtual interfaces are not supported. For example, interfaces such as the following:
  - `irb`
  - `rvi`
  - `lsi`
  - `fti`
  - `vtep`
  - `vt`
- IPv4 transit statistics is not displayed on Routine Engine show commands.
- No interoperability between SCU and IPv4 statistics accounting on PTX routers.

  The following functionalities are not supported:
  - lo0 traffic
  - IPv4 statistics query over SNMP
  - Aggregated Ethernet child link IPv4 statistics accounting

RELATED DOCUMENTATION

route-accounting

show interfaces statistics

Configuring Discard Accounting

On routing platforms containing a Monitoring Services PIC or an Adaptive Services PIC, you can configure accounting for traffic passing through the routing platform.

To configure discard accounting, include the `accounting group` *`group-name`* statement at the `[edit forwarding-options]` hierarchy level:

```
[edit forwarding-options]
accounting group-name {
    output {
        cflowd [ hostnames ] {
            aggregation {
                autonomous-system;
                destination-prefix;
                protocol-port;
                source-destination-prefix {
                    caida-compliant;
                }
                source-prefix;
            }
            autonomous-system-type (origin | peer);
            port port-number;
            version format;
        }
        flow-active-timeout seconds;
        flow-inactive-timeout seconds;
        interface {
            engine-id number;
            engine-type number;
            source-address address;
        }
    }
}
```

To configure the output flow aggregation, include the `cflowd` statement. For more information about flow aggregation, see "Directing Traffic Sampling Output to a Server Running the cflowd Application" on page 12. To configure the interval before exporting an active flow, include the `flow-active-timeout` statement. The default value for **flow-active-timeout** is **1800** seconds. To configure the interval before a flow is considered inactive, include the `flow-inactive-timeout` statement. The default value for **flow-inactive-timeout** is 60 seconds. To configure the interface that sends out monitored information, include the `interface` statement. Discard accounting is supported for the Monitoring Services PIC only.

When you apply a firewall filter to a loopback interface, the filter might block responses from the Monitoring Services PIC. To allow responses from the Monitoring Services PIC to pass through for accounting purposes, configure a term in the firewall filter to include the Monitoring Services PIC IP address. For more detailed information about configuring firewall filters, see *Guidelines for Configuring Firewall Filters* and *Guidelines for Applying Standard Firewall Filters*.

You can use discard accounting for passive and active flow monitoring.

RELATED DOCUMENTATION

Monitoring, Sampling, and Collection Services Interfaces User Guide

Junos OS Class of Service User Guide for Routing Devices

Configuring Active Flow Monitoring on PTX Series Packet Transport Routers

You can use flow monitoring to help with network administration. Active flow monitoring on PTX Series routers allows you to collect sampled packets, then the router does GRE encapsulation of the packets and sends them to a remote server for flow processing. The GRE encapsulation includes an interface index and GRE key field. The GRE encapsulation removes MPLS tags. You configure one or more port-mirroring instances to define which traffic to sample and configure a server to receive the GRE encapsulated packets. You configure a firewall filter on interfaces where you want to capture flows. You can configure as many as 48 port-mirroring instances.

To configure the router to do GRE encapsulation of sampled packets and send them to a remote server for flow processing:

1. Configure one or more server profiles that specify a host where GRE encapsulated sampled packets are sent, and optionally, a source address to include in the header of each sampled packet.

a. Specify a name for each server profile and an IP address of the host where sampled packets are sent:

```
[edit services hosted-services]
user@host# set server-profile server-profile-name server-address ipv4-address
```

b. (Optional) For each server profile, specify a source address to include in the header of each sampled packet:

```
[edit services hosted-services server-profile server-profile-name]
user@host# set client-address ipv4-address
```

NOTE: The default client address is 0.0.0.0. You must specify an IPv4 address as the client address. You can also specify the loopback address or management interface address as the client address.

2. Configure one or more port-mirroring instances.

a. Specify a name for each port-mirroring instance:

```
[edit forwarding-options port-mirroring]
user@host# set instance instance-name
```

NOTE: You can configure a maximum of 48 port-mirroring instances.

b. Specify a protocol family for each port-mirroring instance:

```
[edit forwarding-options port-mirroring instance instance-name]
user@host# set family (inet | inet6 )
```

3. To set the ratio of the number of packets to sample, specify a value from 1 through 65,535 for each port-mirroring instance:

```
[edit forwarding-options port-mirroring instance instance-name input]
user@host# set rate number
```

NOTE: You must specify a value for the `rate` statement. The default value is zero, which effectively disables sampling. If, for example, you specify a rate value of 4, every fourth packet (1 packet out of 4) is sampled.

4. (Optional) Specify the number of samples to collect after the initial trigger event for each port-mirroring instance:

```
[edit forwarding-options port-mirroring instance instance-name input]
user@host# set run-length number
```

NOTE: The default value is zero. You can specify a number up to 20.

5. To designate a host where sampled traffic is sent, specify the name of server profile configured at the `[edit services hosted-services]` hierarchy level for each port-mirroring instance:

```
[edit forwarding-options port-mirroring instance instance-name family ( inet | inet6) output]
user@host# set server-profile server-profile-name
```

6. Configure one or more firewall filters.

   a. For each firewall filter, specify a protocol family, filter name, and match conditions:

   ```
   [edit firewall]
   user@host# set filter family (inet | inet6) filter filter-name term term-name from match-
   condtions
   ```

   b. For each firewall filter you configure, specify the name of a port-mirroring instance you configured at the `[edit forwarding-options]` hierarchy level as a nonterminating action so that the traffic that matches that instance is sampled:

   ```
   [edit firewall family (inet | inet6) filter filter-name term term-name]
   user@host# set then port-mirroring instance instance-name
   ```

7. Apply each firewall filter to an interface to evaluate incoming traffic:

```
[edit interfaces interface-name unit logical-unit-number]
user@host# set family (inet | inet6) filter input firewall-filter-name
```

NOTE: Active flow monitoring is supported only on incoming traffic. You cannot apply firewall filters to evaluate outgoing traffic.

8. Configure the remote server, where GRE encapsulated packets are sent, to perform flow processing.

RELATED DOCUMENTATION

Configuring Port Mirroring | 37

hosted-services

port-mirroring

server-profile (Active Flow Monitoring)

Firewall Filter Nonterminating Actions

Configuring Passive Flow Monitoring

On routing platforms containing the Monitoring Services PIC or the Monitoring Services II PIC, you can configure flow monitoring for traffic passing through the routing platform. This type of monitoring method is passive monitoring.

To configure flow monitoring, include the `monitoring` statement at the `[edit forwarding-options` hierarchy level:

```
[edit forwarding-options]
monitoring group-name {
    family inet {
        output {
            cflowd hostname {
                port port-number;
            }
            export-format cflowd-version-5;
            flow-active-timeout seconds;
            flow-export-destination {
                cflowd-collector;
            }
            flow-inactive-timeout seconds;
            interface interface-name {
                engine-id number;
                engine-type number;
                input-interface-index number;
                output-interface-index number;
                source-address address;
            }
        }
    }
}
```

To configure a passive monitoring group, include the `monitoring` statement and specify a group name. To configure monitoring on a specified address family, include the `family` statement and specify an address family. To specify an interface to monitor incoming traffic, include the `input` statement. To configure the monitoring information that is sent out, include the `output` statement. To configure the output flow aggregation, include the `cflowd` statement. For more information about flow aggregation, see "Directing Traffic Sampling Output to a Server Running the cflowd Application" on page 12. To specify the format of the monitoring information sent out, include the `export-format` statement and specify a version number. To configure the interval before exporting an active flow, include the `flow-active-timeout` statement. The default value for **flow-active-timeout** is 1800 seconds. To enable flow collection, include the `flow-export-destination` statement. To configure the interval before a flow is considered inactive, include the `flow-inactive-timeout` statement. The default value for **flow-inactive-timeout** is 60 seconds. To configure the interface that sends out the monitored information, include the `interface` statement. Flow monitoring is supported for Monitoring Services PIC interfaces only.

When you apply a firewall filter to a loopback interface, the filter might block responses from the Monitoring Services PIC. To allow responses from the Monitoring Services PIC to pass through for monitoring purposes, configure a term in the firewall filter to include the Monitoring Services PIC's IP address. For more detailed information about configuring firewall filters, see *Guidelines for Configuring Firewall Filters* and *Guidelines for Applying Standard Firewall Filters*.

RELATED DOCUMENTATION

Monitoring, Sampling, and Collection Services Interfaces User Guide

Junos OS Class of Service User Guide for Routing Devices

Configuring Port Mirroring

IN THIS SECTION

All Reviewers, this PDF is long, but you only have to pay attention to highlighted text and text between the points marked as Start or Restart and Pause or End for a PR number.

SME reviewers—Please review the highlighted sections called out by either "All SME reviewers" or "<your-platform> reviewers",

PR reviewers—Please review the doc PRs that you originated.]

Port mirroring is the ability of a router to send a copy of an IPv4 or IPv6 packet to an external host address or a packet analyzer for analysis.

See Feature Explorer for the latest list of supported platforms and Junos releases that support port mirroring.

All SME reviewers: Please review the following new paragraph:

This topic, Configuring Port Mirroring, refers to port mirroring on MX Series and PTX Series routers and EX9200 switches. For specific configuration differences, see examples that pertain to port mirroring on these individual platforms—for example, Example: Configuring Remote Port Mirroring on PTX Routers and Example: Configuring Multiple Port Mirroring with Next-Hop Groups on M, MX and T Series Routers.

Port mirroring is different from traffic sampling. In traffic sampling, a sampling key based on the packet header is sent to the Routing Engine. There, the key can be placed in a file, or cflowd packets based on the key can be sent to a cflowd server. In port mirroring, the entire packet is copied and sent out through a next-hop interface.

[PR1723958 update--START--

We use the term *mirrored packet* rather than *sampled packet* in this document. Port mirroring is a type of sampling.

One application for port mirroring sends a duplicate packet to a virtual tunnel. A next-hop group can then be configured to forward copies of this duplicate packet to several interfaces. For more information about next-hop groups, see https://www.juniper.net/documentation/us/en/software/junos/sampling-forwarding-monitoring/topics/concept/policy-configuring-next-hop-groups.html**Configuring Next-Hop Groups to Use Multiple Interfaces to Forward Packets Used in Port Mirroring**.

PR1723958 update--PAUSE]

[PTX and EX9200 SME reviewers: Are the following statements—up to the next heading, Port Mirroring Configuration Guidelines—true for PTX and EX9200?]

All MX Series 5G Universal Routing Platforms support port mirroring for IPv4 or IPv6.

Port mirroring for VPLS traffic is supported on MX Series routers.

Port mirroring is supported for Layer 2 traffic on MX Series routers. For information about how to configure port mirroring for Layer 2 traffic, see the Network Management and Monitoring Guide.

In the MPCs on MX Series routers, GRE and MPLS header information is not contained in the port-mirrored traffic corresponding to MPLS packets transmitted through IP-GRE tunnels.

[PR1730861 START--

PTX Reviewers, I deleted the following statement from the topic because several platforms and **all** the named line cards are EOL—

PTX1K, PTX10002, PTX5K, PTX3K, and PTX10K platforms with first-generation line cards (LC1101, LC1102, LC1104 and LC1105) do not support egress port mirroring.

[Also, same reviewers, the PR asks for more info re configuring port mirroring on PTX10001-MR. Please see "Example: Configuring Remote Port Mirroring on PTX Routers" on page 59

.]

[PR1730861 END]

Port Mirroring Configuration Guidelines

[MX, PTX, and EX9200 SME reviewers: Are the following config guidelines also true for PTX and EX9200?]

When configuring port mirroring, the following restrictions apply:

- Only transit data is supported.
- The port mirror output interface MTU value should be big enough to accommodate the mirrored packets.
- [PR1723958 update--RESTART –-changed to "port mirror output interface" from "port mirror or analyzer output interface"

  PR1723958 update--PAUSE]

  A standalone trunk port is not supported as a port mirror output interface for MX Series routers and EX9200 switches. If you want to use a trunk port as a mirror output port, you must use a bridge domain (MX) or a VLAN (EX) as port mirror output, then attach the trunk port to the respective bridge domain or VLAN as an output port.
- You can configure port mirroring for IPv4 and IPv6 simultaneously on the MX Series routers.
- Port mirroring in the ingress and egress direction is not supported for link services IQ (lsq-) interfaces.
- Ingress filtering of multicast packets is supported on all Dense Port Concentrators (DPCs) in MX Series routers. Egress filtering of multicast packets is supported for interfaces on MPCs in MX Series routers. Filtering of multicast packets based on destination address is not supported for interfaces on I-chip ASIC-based DPCs in MX Series routers.

  For Layer 3 port mirroring (`family inet` and `family inet6`), if the traffic being mirrored is multicast (in other words, if the packet's destination IP address is a multicast address), the destination MAC address in the mirrored copy corresponds to this multicast destination IP address and not to the unicast MAC address specified in the `[edit forwarding-options port-mirroring family (inet | inet6) output]` configuration.
- By default, firewall filters cannot be applied to port-mirroring destination interfaces. To enable port-mirroring destination interfaces to support firewall filters, use the `no-filter-check` statement to disable filter checking on the interfaces. You can include the `no-filter-check` statement at the following hierarchy levels:
  - `[edit forwarding-options port-mirroring family (inet | inet6 | ccc | vpls) output]`
  - `[edit forwarding-options port-mirroring instance` *`instance-name`* `family (inet | ccc | vpls) output]`
- You must include a firewall filter with both the `accept` action and the `port-mirror` action modifier on the inbound interface.
- The interface you configure for port mirroring should not participate in any kind of routing activity.
- [PR1723958 update--RESTART---changed to "mirrored ...packets" from "sampled ... packets", 2X]

  The destination address you specify should not have a route to the ultimate traffic destination. For example, if the mirrored IPv4 packets have a destination address of `192.68.9.10` and the port-mirrored traffic is sent to `192.68.20.15` for analysis, the device associated with the latter address should not know a route to `192.68.9.10`. Also, it should not send the mirrored packets back to the source address.

[PR1723958 update--PAUSE]

- MX Series routers support more than one port-mirroring interface per router.
- You can configure multiple port-mirroring instances on MX Series routers.
- You can specify both host (cflowd) sampling and port mirroring in the same configuration. You can perform Routing Engine–sampling and port mirroring actions simultaneously. However, you cannot perform PIC-sampling and port mirroring actions simultaneously.
- [PR1723958 update--RESTART---changed to "mirrored ...packets" from "sampled ... packets",1X].

  In typical applications, you send the mirrored packets to an analyzer or a workstation for analysis, not to another router. If you must send this traffic over a network, you should use tunnels.

  [PR1723958 update--PAUSE]

  [PR1744110 START]

- On PTX Series routers that support port mirroring, IPv4 packets are dropped if the packet length exceeds the configured maximum packet length.

  [PR1744110 END---But see more text entries for this PR in the last topic in this document, *Example: Configuring Port Mirroring with Family any and a Firewall Filter*.]

NOTE: In a firewall filter configured with a `port-mirror-instance` or `port-mirror` action, if `l2-mirror` action is also configured, then port-mirroring instance family should be `any`. In the absence of the `l2-mirror` action, port-mirroring instance family should be the firewall filter family.

In the following example, `port-mirroring instance pm1` is `inet`, which is the firewall filter family, and because `l2-mirror` action is not present.

```
set chassis network-services enhanced-ip
set interfaces xe-0/0/0:0 encapsulation extended-vlan-bridge
set interfaces xe-0/0/0:0 unit 0 family bridge interface-mode access
set interfaces xe-0/0/0:0 unit 0 family bridge vlan-id 100
set forwarding-options port-mirroring instance pm1 input rate 1
set forwarding-options port-mirroring instance pm1 family inet output interface
xe-0/0/0:0.0
```

```
set firewall family inet filter f1 term t1 from source-address 10.1.1.1/32
set firewall family inet filter f1 term t1 from source-address 10.1.1.2/32
set firewall family inet filter f1 term t1 then count t1
set firewall family inet filter f1 term t1 then port-mirror-instance pm1
set firewall family inet filter f1 term t1 then accept
```

In the following example, because `l2-mirror` action is also present alongside `port-mirror-instance` action, `port-mirroring instance pm1 family` is `any`.

```
set chassis network-services enhanced-ip
set interfaces xe-0/0/0:0 unit 0 family bridge interface-mode access
set interfaces xe-0/0/0:0 unit 0 family bridge vlan-id 100
set forwarding-options port-mirroring instance pm1 input rate 1
set forwarding-options port-mirroring instance pm1 family any output interface
xe-0/0/0:0.0
set firewall family inet filter f1 term t1 from source-address 10.1.1.1/32
set firewall family inet filter f1 term t1 from source-address 10.1.1.2/32
set firewall family inet filter f1 term t1 then count t1
set firewall family inet filter f1 term t1 then port-mirror-instance pm1
set firewall family inet filter f1 term t1 then l2-mirror
set firewall family inet filter f1 term t1 then accept
```

[MX, PTX, and EX9200 SME reviewers: This is the endpoint of "Port Mirroring Configuration Guidelines."]

Configuring Port Mirroring

To configure port mirroring, include the `port-mirroring` statement at the `[edit forwarding-options]` hierarchy level:

```
[edit forwarding-options]
port-mirroring {
    family (ccc | inet | inet6 | vpls) {
        output {
            interface interface-name {
                next-hop address;
            }
```

```
                no-filter-check;
            }
            input {
                maximum-packet-length bytes;
                rate number;
                run-length number;
            }
        }
    }
```

Configuring the Port-Mirroring Address Family and Interface

[PR1723958 update--RESTART---I changed "traffic to sample" to "traffic to mirror" —

To configure port mirroring, include the `port-mirroring` statement. To configure the address family type of traffic to mirror, include the `family` statement. To configure the rate of sampling, length of sampling, and the maximum size for the mirrored packet, include the `input` statement. To specify on which interface to send duplicate packets and the next-hop address to send packets, include the `output` statement. To determine whether there are any filters on the specified interface, include the `no-filter-check` statement.

[PR1723958 update--PAUSE]

For information about the `rate` and `run-length` statements, see Configuring Traffic Sampling.

Configuring MX Series Routers to Mirror Traffic Only Once

[PR1723958 update--RESTART---changed "sampled" to "mirrored".]

On MX Series routers, you can configure port mirroring so that the router mirrors traffic only once. If you configure port mirroring on both ingress and egress interfaces, the same packet could be mirrored twice. To mirror packets only once and prevent the router from sending duplicate mirrored packets to the same mirroring destination, include the `mirror-once` statement at the `[edit forwarding-options port-mirroring]` hierarchy level:

[PR1723958 update--PAUSE]

```
[edit forwarding-options port-mirroring]
mirror-once;
```

NOTE: The `mirror-once` statement is supported only in the global port-mirroring instance.

Configuring Port-Mirroring Instances

Instances enable you to mirror packets to different destinations from the same PFE and also use different sampling parameters for each instance.

You can configure multiple port-mirroring instances on MX Series routers. For information about configuring multiple port-mirroring instances, see the Network Management and Monitoring Guide.

To configure a port-mirroring instance, include the `instance` *`port-mirroring-instance`* statement at the `[edit forwarding-options port-mirroring]` hierarchy level:

```
[edit forwarding-options port-mirroring]
instance port-mirroring-instance-name {
    family (ccc | inet | inet6 | vpls) {
        output {
            interface interface-name {
                next-hop address;
            }
            no-filter-check;
        }
    }
    input {
        maximum-packet-length bytes;
        rate number;
        run-length number;
    }
}
```

Associating a Port-Mirroring Instance with an FPC or a PIC on MX Series Routers

[PR1723958 RESTART---see highlighted text in the paras immediately below.]

You must associate port-mirroring instances with FPCs or PICs. You can associate a port-mirroring instance with a specific FPC or with a specific PIC on an MX Series router. "Associating" a port-mirroring instance to an FPC or a PIC is sometimes referred to as "binding" the instance to the FPC or PIC.

A PIC-level instance overrides an FPC-level instance, and an FPC-level instance overrides a global instance.

The number of instances supported are:

- On an MX DPC, a maximum of 2 instances can be bound to (associated with) a PIC. Because you can can have 4 PICs per FPC, each FPC supports 8 instances.

- On an MX MPC, a maximum of 2 instances are supported and they can only be bound at the FPC level under the `[edit chassis]` configuration hierarchy.

To check the association of port-mirroring instances with an FPC, issue the configuration-mode command `show chassis fpc` *`fpc-number`*.

To associate a port-mirroring instance with an FPC or PIC on an MX Series router, you must include the `port-mirror-instance` *`port-mirroring-instance-name`* statement at the `[edit chassis fpc` *`slot-number`*`]` hierarchy level (replace `fpc` with `pic` to configure the binding on a PIC).

```
[edit chassis]
fpc slot-number {
    port-mirror-instance port-mirroring-instance-name;
}
```

NOTE: You do not need to include this `[edit chassis]` configuration in a **global** port-mirroring configuration.

[PR1723958 **END**]

For *`slot-number`*, specify the slot number of the FPC or PIC you want to associate with the port-mirroring instance. For *`port-mirroring-instance-name`*, specify the name of a port-mirroring instance you configured at the `[edit forwarding-options port-mirroring]` hierarchy level.

Platform-Specific Behavior

Use Feature Explorer to confirm platform and release support for specific features.

Use the following tables to review platform-specific behavior for your platform:

Platform	Difference
PTX Series of Routers	The `no-filter-check` statement to disable filter checking on the interfaces is unsupported.

Example: Configuring Local Port Mirroring on PTX Routers

IN THIS SECTION

This example shows you how to configure and verify local port mirroring on PTX platforms running Junos Evolved. The PTX platforms include PTX10001-36MR, LC1201 and LC1202 in PTX10004, PTX10008 and PTX10016 chassis

Before You Begin

Hardware and Software requirements	Junos OS Evolved Release 22.2R1.12-EVO or later. PTX10001-36MR **See Feature Explorer for a complete listing of supported platforms and Junos OS versions.**
Estimated reading time	Fifteen minutes.
Estimated configuration time	Thirty minutes

Business impact	Use this configuration example to configure local port mirroring feature. Port mirror is a critical tool for debugging and security related tasks. Mirror traffic can be analyzed offline by a variety of tools to either see protocol interactions or for anomaly detection.
Know more	To better understand Port Mirroring, see Port Mirroring and Analyzers
Learn more	Learning Portal

Functional Overview

Table 1 on page 46 provides a quick summary of the protocols and technologies deployed in this example.

Table 1: Local Port Mirroring Functional Overview

Routing and Signaling protocols	
OSPF and OSPF3	All routers run OSPF and OSPF3 as the IGP. All routers belong to area 0 (also called the backbone area). The OSPF/OSPF3 routing domains provide internal reachability to all networks and interfaces in the topology. In this example the CE and PE/P devices are part of the same IGP routing domain. As a result, tunnels are not needed between the PE devices to transport CE traffic over the core. In addition, because this is a local mirror use case GRE encapsulation is not needed when sending mirrored traffic to the monitoring station.
Routing Protocols	
IPv4 and IPv6	All devices are configured to support routing of both IPv4 and IPv6.
Analyzer (monitoring station)	
Centos and Wireshark	The analyzer runs Centos 7.x with a GUI version of Wireshark.

Topology Overview

In this example, the R3 device functions as the Device Under Test (DUT) as this is where port mirroring is configured. The device uses firewall filters to match the IP addresses associated with the CE devices to trigger the port mirror action. A combination of ingress and egress filters are employed to mirror both request and response traffic flowing between the CE devices (R1 and R5) .

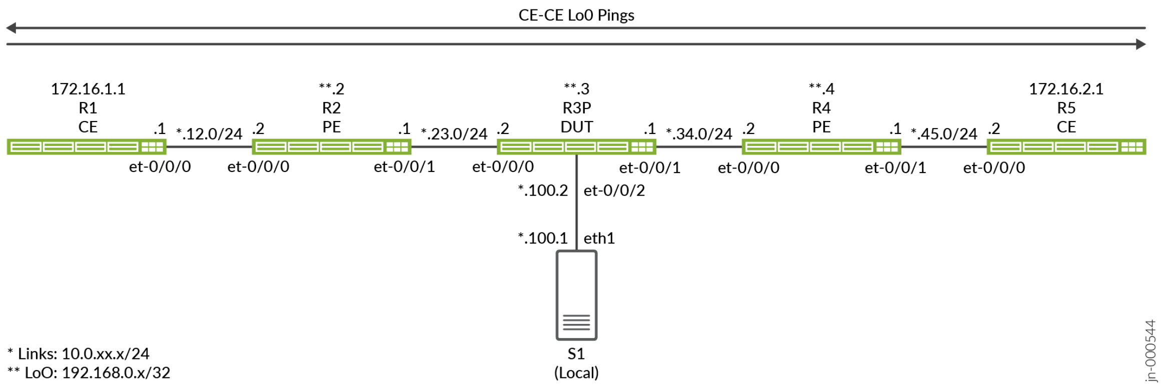
The firewall filters that evoke packet sampling are applied to one or more of the transit interfaces on the R3 device.

Table 2: Local Port Mirroring Topology Overview

Device Name	Role	Function
CE	Customer Edge (CE) device that sends test traffic to confirm sampling works properly.	These devices are designated as CE devices. In most cases a CE device is part of a VPN service. Here, we let the CE share the same OSPF Area 0 as the provider devices to provide main instance IP connectivity.
PE	Provider Edge (PE) device that attaches to the CE.	Devices at the edge of a provider network. Our PEs run only OSPF. BGP and VPNs are not deployed.
P	A Provider (P) core router.	We opt to demonstrate port mirroring at a P router. You can configure port mirroring on any of the provider devices as needed.
Analyzer	The analyzer device received the mirror traffic for storage and analysis.	The specifics of the analyzer are outside the scope of this document. There are a number of open source and commercial options available. Our analyzer happens to be running Centos 7.x with a Gnome desktop supporting a GUIO version of Wireshark.

Topology Illustrations

Figure 2: Local Port Mirroring



R3 Configuration Steps

For information about navigating the CLI, see Using the CLI Editor in Configuration Mode

NOTE: For complete configuration on all devices see: "Appendix 2: Set Commands on All Devices" on page 55

This section highlights the main configuration tasks needed to configure the DUT, which is the P device (R3) in this example. Excluding the specifics used for sampling, all devices have a similar baseline configuration that supports main instance IPv6 and IPv4 connectivity.

1. Configure the IPv4 and IPv6 routing baseline. This includes numbering the loopback and core facing interfaces for both IPv4 and IPv6. You also define the OSPF and OSPFv3 routing protocol to provide reachability between all network interfaces.

   A passive IGP instance is provisioned for the interface attached to the analyzer. This provides reachability for diagnostic purposes without having hello packets generated on the interface. An OSPF adjacency is not expected or needed to the analyzer device

```
[edit]
set interfaces et-0/0/0 unit 0 family inet address 10.0.23.2/24
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:23::2/64
set interfaces et-0/0/1 unit 0 family inet address 10.0.34.1/24
set interfaces et-0/0/1 unit 0 family inet6 address 2001:db8:10:0:34::1/64
set interfaces et-0/0/2 unit 0 family inet address 10.0.100.2/24
set interfaces et-0/0/2 unit 0 family inet6 address 2001:db8:10:0:100::2/64
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:192:168:0::3/128

set routing-options router-id 192.168.0.3
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0 interface et-0/0/2.0 passive

set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0 interface et-0/0/2.0 passive
```

NOTE: For the local mirror use case IP connectivity is only needed between the analyzer and the device doing the port mirroring. In this example we run a passive IGP on the interface attached to the analyzer. We also configure a default route on the analyzer to provide IP connectivity between it and the other devices. This provides the ability to test connectivity between the analyzer and all other devices. in the topology.

This capability is most useful in a remote port mirror case where there is a need for Layer 3 reachability between the sampling device and the analyzer.

2. Configure the sampling rate. We use a rate of 1 to select and sample all matching packets. The default `run-length` of 0 is left in place given all matching traffic is already sampled. You must also specify the egress interface and next hop address that the mirrored traffic is sent to. In this example of local port mirror, it should be noted that the interface and next hop addresses specified are directly attached to the DUT. As a result, no tunnels are needed or used when sending the mirrored traffic to the analyzer.

```
[edit]
set forwarding-options port-mirroring input rate 1
set forwarding-options port-mirroring family inet output interface et-0/0/2.0 next-hop
10.0.100.1
set forwarding-options port-mirroring family inet6 output interface et-0/0/2.0 next-hop
2001:db8:10:0:100::1
```

NOTE: This configuration assumes that the analyzer replies to ARP and ND request sent by the DUT for MAC address resolution. If this is not the case, or if you wish that ARP traffic is not part of your packet captures, you should configure a static ARP entry. Be sure to specify the correct MAC address for the interface on the analyzer device that is attached to the DUT.

3. Define the firewall filter to match on and then mirror IPv4 packets. Note that the filter's action specifies a port mirror action. This action directs matching traffic to the port mirroring instances you configured previously. Two filters are defined, one each for the source and destination addresses of CE1 and CE2, respectively. The filters include a count function to assist in confirmation of proper operation.

   Don't overlook the final `accept-all` term that overrides the default `deny-all` action of a Junos firewall filter!

```
[edit]
set firewall filter mirror_ce1 term term1 from source-address 172.16.1.1/32
set firewall filter mirror_ce1 term term1 from destination-address 172.16.2.1/32
set firewall filter mirror_ce1 term term1 then count mirror_ce1
set firewall filter mirror_ce1 term term1 then port-mirror
set firewall filter mirror_ce1 term term1 then accept
set firewall filter mirror_ce1 term accept-all then accept
set firewall filter mirror_ce2 term term1 from source-address 172.16.2.1/32
set firewall filter mirror_ce2 term term1 from destination-address 172.16.1.1/32
set firewall filter mirror_ce2 term term1 then count mirror_ce2
set firewall filter mirror_ce2 term term1 then port-mirror
set firewall filter mirror_ce2 term term1 then accept
set firewall filter mirror_ce2 term accept-all then accept
```

4. Define the firewall filter to match and mirror IPv6 packets.

```
[edit]
set firewall family inet6 filter ce1_v6 term 1 from source-address 2001:db8:172:16:1::1/128
set firewall family inet6 filter ce1_v6 term 1 from destination-address
2001:db8:172:16:2::1/128
set firewall family inet6 filter ce1_v6 term 1 then count ce1_v6
set firewall family inet6 filter ce1_v6 term 1 then port-mirror
set firewall family inet6 filter ce1_v6 term 1 then accept
set firewall family inet6 filter ce1_v6 term accept-all then accept
set firewall family inet6 filter ce2_v6 term 1 from source-address 2001:db8:172:16:2::1/128
set firewall family inet6 filter ce2_v6 term 1 from destination-address
2001:db8:172:16:1::1/128
set firewall family inet6 filter ce2_v6 term 1 then count ce2_v6
set firewall family inet6 filter ce2_v6 term 1 then port-mirror
set firewall family inet6 filter ce2_v6 term 1 then accept
set firewall family inet6 filter ce2_v6 term accept-all then accept
```

5. Apply the IPv4 and IPv6 filters to the desired interfaces. In our example, we apply both filters to the et-0/0/0 interface. Note the directionality of the filter application. For each CE traffic flow (IPv4 or IPv6), we apply one filter as ingress and the other as egress. This method of application is compatible with the way the filters are written given the address assignments and directionality of the traffic.

```
[edit]
set interfaces et-0/0/0 unit 0 family inet filter input mirror_ce1
set interfaces et-0/0/0 unit 0 family inet filter output mirror_ce2

set interfaces et-0/0/0 unit 0 family inet6 filter input ce1_v6
set interfaces et-0/0/0 unit 0 family inet6 filter output ce2_v6
```

Verification

1. Confirm OSPF and OSPF3 neighbors and routes to all loopback addresses.

```
user@r3-ptx> show ospf neighbor

Address          Interface              State           ID               Pri  Dead
10.0.23.1        et-0/0/0.0             Full            192.168.0.2      128    31
10.0.34.2        et-0/0/1.0             Full            192.168.0.4      128    38

user@r3-ptx> show ospf3 neighbor

ID               Interface              State     Pri   Dead
192.168.0.2      et-0/0/0.0             Full      128     30
  Neighbor-address fe80::c6ba:25ff:fe48:9
192.168.0.4      et-0/0/1.0             Full      128     32
  Neighbor-address fe80::6204:30ff:fe6e:ffff

regress@r3-ptx> show route protocol ospf | match /32
```

```
172.16.1.1/32      *[OSPF/10] 01:04:02, metric 2
172.16.2.1/32      *[OSPF/10] 6d 00:47:07, metric 2
192.168.0.2/32     *[OSPF/10] 01:04:02, metric 1
192.168.0.4/32     *[OSPF/10] 6d 00:47:12, metric 1
224.0.0.5/32       *[OSPF/10] 6d 00:48:28, metric 1
224.0.0.6/32       *[OSPF/10] 6d 00:48:28, metric 1

regress@r3-ptx> show route protocol ospf3 | match /128
2001:db8:172:16:1::1/128
2001:db8:172:16:2::1/128
2001:db8:192:168::2/128*[OSPF3/10] 01:04:09, metric 1
2001:db8:192:168::4/128*[OSPF3/10] 6d 00:47:15, metric 1
ff02::5/128        *[OSPF3/10] 6d 00:48:35, metric 1
ff02::6/128        *[OSPF3/10] 6d 00:48:35, metric 1
```

2. Confirm the port mirroring instance on R3. Verify that the port mirroring state is `up` for the mirroring interface. Be sure to confirm the `up` state for both the IPv4 and IPv6 families. While here, it is a good idea to confirm IP connectivity between the DUT and the analyzer. In our setup, a default route is configured on the analyzer to permit ping testing from all points of the network. Technically, the analyzer only has to be reachable by the DUT (R3), as this is an example of local port mirroring.

```
user@r3-ptx> show forwarding-options port-mirroring
Instance Name: &global_instance
  Instance Id: 1
  Input parameters:
    Rate                  : 1
    Run-length            : 0
    Maximum-packet-length : 0
  Output parameters:
    Family              State     Destination          Next-hop
    inet                up        et-0/0/2.0           10.0.100.1
    inet6               up        et-0/0/2.0           2001:db8:10:0:100::1
```

3. Clear the firewall counters and interface statistics on R3. Next, generate IPv4 and IPv6 test traffic between the CE devices and display the firewall counters on R3. Verify the filters applied to R3 correctly reflect the test traffic.

```
user@r3-ptx> clear firewall all
user@r3-ptx> clear interfaces statistics all
```

```
user@r1-ptx> ping 172.16.2.1 source 172.16.1.1 count 10 rapid
PING 172.16.2.1 (172.16.2.1) from 172.16.1.1 : 56(84) bytes of data.
--- 172.16.2.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 711ms
rtt min/avg/max/mdev = 11.161/72.078/364.497/121.714 ms, ipg/ewma 78.945/100.962 ms

user@r1-ptx> ping 2001:db8:172:16:2::1 source 2001:db8:172:16:1::1 count 10 rapid
 ping 2001:db8:172:16:2::1 source 2001:db8:172:16:1::1 count 10 rapid
PING 2001:db8:172:16:2::1(2001:db8:172:16:2::1) from 2001:db8:172:16:1::1 : 56 data bytes

--- 2001:db8:172:16:2::1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 2436ms
rtt min/avg/max/mdev = 11.363/247.188/518.314/226.132 ms, pipe 2, ipg/ewma 270.652/201.439 ms
```

4. Display the firewall counters on R3. Verify if the filters applied to R3 correctly reflect the test traffic you generated.

```
user@r3-ptx> show firewall
Filter: mirror_ce1
Counters:
Name                                                Bytes              Packets
mirror_ce1                                            840                   10

Filter: mirror_ce2
Counters:
Name                                                Bytes              Packets
mirror_ce2                                            840                   10

Filter: ce1_v6
Counters:
Name                                                Bytes              Packets
ce1_v6                                               1040                   10
```

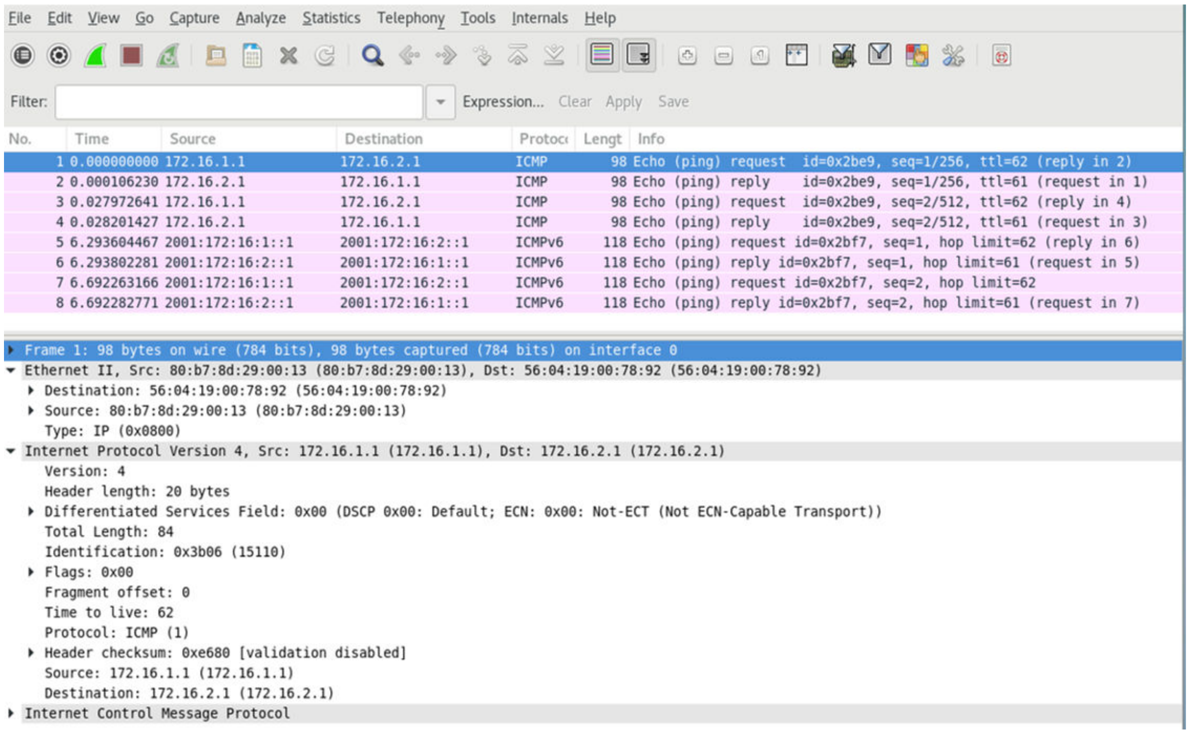
```
Filter: ce2_v6
Counters:
Name                                                Bytes              Packets
ce2_v6                                                1040                   10
```

5. Display interface statistics for R3's et-0/0/2.0 interface that is attached to the analyzer. The goal is to confirm output traffic counters that correlate to the test traffic generated. With ten pings for both IPv4 and IPv6, and given that we mirror both request and replies, you can expect to see about 40 output packets.

```
user@r3-ptx> show interfaces et-0/0/2.0 detail
Logical interface et-0/0/2.0 (Index 1017) (SNMP ifIndex 541) (Generation 704374637676)
    Flags: Up SNMP-Traps Encapsulation: ENET2
  Traffic statistics:
   Input  bytes  :                  0
   Output bytes  :               3760
   Input  packets:                  0
   Output packets:                 40
    Local statistics:
     Input  bytes  :                  0
     Output bytes  :                  0
     Input  packets:                  0
     Output packets:                  0
    Transit statistics:
     Input  bytes  :                  0                    0 bps
     Output bytes  :               3760                    0 bps
     Input  packets:                  0                    0 pps
     Output packets:                 40                    0 pps
```

6. Run tcpdump or the analysis application of your choice on the monitoring station to confirm receipt and processing of the mirrored test traffic. To keep the size of the capture smaller, we generated new test traffic with only two ping requests for each IPv4 and IPv6. The capture and decode confirms that port mirroring of IPv4 and IPv6, based on a firewall filter matching, is working as expected. Note that both request and response traffic is shown.

   Also, in the capture, note that only the Layer 3 traffic is mirrored. The Layer 2 encapsulation shown is generated by the DUT (R3) when forwarding the mirrored traffic to the analyzer. You can configure port mirroring for Layer 2 services like Ethernet switching or VXLAN when you need to preserve the original Layer 2 frame.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.   Time        Source             Destination        Protoco Lengt Info
    1 0.000000000 172.16.1.1         172.16.2.1         ICMP      98 Echo (ping) request  id=0x2be9, seq=1/256, ttl=62 (reply in 2)
    2 0.000106230 172.16.2.1         172.16.1.1         ICMP      98 Echo (ping) reply    id=0x2be9, seq=1/256, ttl=61 (request in 1)
    3 0.027972641 172.16.1.1         172.16.2.1         ICMP      98 Echo (ping) request  id=0x2be9, seq=2/512, ttl=62 (reply in 4)
    4 0.028201427 172.16.2.1         172.16.1.1         ICMP      98 Echo (ping) reply    id=0x2be9, seq=2/512, ttl=61 (request in 3)
    5 6.293604467 2001:172:16:1::1   2001:172:16:2::1   ICMPv6   118 Echo (ping) request id=0x2bf7, seq=1, hop limit=62 (reply in 6)
    6 6.293802281 2001:172:16:2::1   2001:172:16:1::1   ICMPv6   118 Echo (ping) reply id=0x2bf7, seq=1, hop limit=61 (request in 5)
    7 6.692263166 2001:172:16:1::1   2001:172:16:2::1   ICMPv6   118 Echo (ping) request id=0x2bf7, seq=2, hop limit=62
    8 6.692282771 2001:172:16:2::1   2001:172:16:1::1   ICMPv6   118 Echo (ping) reply id=0x2bf7, seq=2, hop limit=61 (request in 7)

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▾ Ethernet II, Src: 80:b7:8d:29:00:13 (80:b7:8d:29:00:13), Dst: 56:04:19:00:78:92 (56:04:19:00:78:92)
  ▸ Destination: 56:04:19:00:78:92 (56:04:19:00:78:92)
  ▸ Source: 80:b7:8d:29:00:13 (80:b7:8d:29:00:13)
    Type: IP (0x0800)
▾ Internet Protocol Version 4, Src: 172.16.1.1 (172.16.1.1), Dst: 172.16.2.1 (172.16.2.1)
    Version: 4
    Header length: 20 bytes
  ▸ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 84
    Identification: 0x3b06 (15110)
  ▸ Flags: 0x00
    Fragment offset: 0
    Time to live: 62
    Protocol: ICMP (1)
  ▸ Header checksum: 0xe680 [validation disabled]
    Source: 172.16.1.1 (172.16.1.1)
    Destination: 172.16.2.1 (172.16.2.1)
▸ Internet Control Message Protocol
```

Appendix: Set Commands on All Devices

IN THIS SECTION

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the [edit] hierarchy level.

R1 (CE)

```
set system host-name r1-ptx
set interfaces et-0/0/0 unit 0 family inet address 10.0.12.1/24
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:12::1/64
set interfaces lo0 unit 0 family inet address 172.16.1.1/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:172:16:1::1/128
set routing-options router-id 172.16.1.1
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
```

R2 (PE)

```
set system host-name r2-ptx
set interfaces et-0/0/0 unit 0 family inet address 10.0.12.2/24
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:12::2/64
set interfaces et-0/0/1 unit 0 family inet address 10.0.23.1/24
set interfaces et-0/0/1 unit 0 family inet6 address 2001:db8:10:0:23::1/64
set interfaces et-0/0/2 unit 0 family inet tunnel-termination
set interfaces et-0/0/2 unit 0 family inet address 10.0.100.2/24
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:192:168:0::2/128
set routing-options router-id 192.168.0.2
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
```

R3 (DUT)

```
set system host-name r3-ptx
set interfaces et-0/0/0 unit 0 family inet filter input mirror_ce1
set interfaces et-0/0/0 unit 0 family inet filter output mirror_ce2
set interfaces et-0/0/0 unit 0 family inet address 10.0.23.2/24
set interfaces et-0/0/0 unit 0 family inet6 filter input ce1_v6
set interfaces et-0/0/0 unit 0 family inet6 filter output ce2_v6
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:23::2/64
set interfaces et-0/0/1 unit 0 family inet address 10.0.34.1/24
set interfaces et-0/0/1 unit 0 family inet6 address 2001:db8:10:0:34::1/64
set interfaces et-0/0/2 unit 0 family inet address 10.0.100.2/24
set interfaces et-0/0/2 unit 0 family inet6 address 2001:db8:10:0:100::2/64
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:192:168:0::3/128
set forwarding-options port-mirroring input rate 1
set forwarding-options port-mirroring input run-length 0
set forwarding-options port-mirroring family inet output interface et-0/0/2.0 next-hop 10.0.100.1
set forwarding-options port-mirroring family inet6 output interface et-0/0/2.0 next-hop
2001:db8:10:0:100::1
set firewall family inet6 filter ce1_v6 term 1 from source-address 2001:db8:172:16:1::1/128
set firewall family inet6 filter ce1_v6 term 1 from destination-address 2001:db8:172:16:2::1/128
set firewall family inet6 filter ce1_v6 term 1 then count ce1_v6
set firewall family inet6 filter ce1_v6 term 1 then port-mirror
set firewall family inet6 filter ce1_v6 term 1 then accept
set firewall family inet6 filter ce1_v6 term accept-all then accept
set firewall family inet6 filter ce2_v6 term 1 from source-address 2001:db8:172:16:2::1/128
set firewall family inet6 filter ce2_v6 term 1 from destination-address 2001:db8:172:16:1::1/128
set firewall family inet6 filter ce2_v6 term 1 then count ce2_v6
set firewall family inet6 filter ce2_v6 term 1 then port-mirror
set firewall family inet6 filter ce2_v6 term 1 then accept
set firewall family inet6 filter ce2_v6 term accept-all then accept
set firewall filter mirror_ce1 term 1 from source-address 172.16.1.1/32
set firewall filter mirror_ce1 term 1 from destination-address 172.16.2.1/32
set firewall filter mirror_ce1 term 1 then count mirror_ce1
set firewall filter mirror_ce1 term 1 then port-mirror
set firewall filter mirror_ce1 term 1 then accept
set firewall filter mirror_ce1 term accept-all then accept
set firewall filter mirror_ce2 term term1 from source-address 172.16.2.1/32
set firewall filter mirror_ce2 term 1 from destination-address 172.16.1.1/32
set firewall filter mirror_ce2 term 1 then count mirror_ce2
set firewall filter mirror_ce2 term 1 then port-mirror
set firewall filter mirror_ce2 term 1 then accept
set firewall filter mirror_ce2 term accept-all  then accept
set routing-options router-id 192.168.0.3
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface et-0/0/2.0 passive
set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface et-0/0/2.0 passive
```

R4 (PE)

```
set system host-name r4-ptx
set interfaces et-0/0/0 unit 0 family inet address 10.0.34.2/24
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:34::2/64
set interfaces et-0/0/1 unit 0 family inet address 10.0.45.1/24
set interfaces et-0/0/1 unit 0 family inet6 address 2001:db8:10:0:45::1/64
set interfaces et-0/0/2 unit 0 family inet address 10.0.200.2/24
set interfaces et-0/0/2 unit 0 family inet6 address 2001:db8:10:0:200::2/64
set interfaces lo0 unit 0 family inet address 192.168.0.4/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:192:168:0::4/128
set routing-options router-id 192.168.0.4
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
```

R5 (CE)

```
set system host-name r5-ptx
set interfaces et-0/0/0 unit 0 family inet address 10.0.45.2/24
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:45::2/64
set interfaces lo0 unit 0 family inet address 172.16.2.1/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:172:16:2::1/128
set routing-options router-id 172.16.2.1
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
```

Example: Configuring Remote Port Mirroring on PTX Routers

IN THIS SECTION

This example shows you how to configure and verify remote port mirroring on PTX platforms running Junos Evolved. The PTX platforms include PTX10001-36MR, LC1201 and LC1202 in PTX10004, PTX10008 and PTX10016 chassis.

Before You Begin

Hardware and Software requirements	Junos OS Evolved Release 22.2R1.12-EVO or later. PTX10001-36MR See Feature Explorer for a complete listing of supported platforms and Junos OS versions.
Estimated reading time	Fifteen minutes.
Estimated configuration time	Thirty minutes

Business impact	Port mirror is a critical tool for debugging and security related tasks. Mirrored traffic can be analyzed offline by a variety of tools to analyze protocol interactions, anomaly detection, or lawful intercept wire tap operations.
Know more	To better understand Port Mirroring, see Port Mirroring and Analyzers
Learn more	Learning Portal

Functional Overview

Remote Port Mirroring Functional Overview on page 60 provides a quick summary of the protocols and technologies deployed in this example.

Table 3: Remote Port Mirroring Functional Overview

Routing and Signaling protocols	
OSPF and OSPF3	All routers run OSPF and OSPF3 as the IGP. All provider routers belong to area 0 (also called the backbone area). The OSPF/OSPF3 routing domains provide internal reachability to all networks and interfaces in the topology. The CE routers use OSPF and OSPF3 to exchange routes with the PEs.
MPLS and RSVP	The provider routers signal MPLS LSPs using the RSVP protocol. IPv6 tunneling is enabled to support IPv6 over MPLS. MPLS is used to support a Layer 3 VPN.
MP-BGP	Multiprotocol BGP is used between the PE routers to advertise customer VPN routes.
Layer 3 VPN	The PE routers use a VRF routing instance to support as Layer 3 VPN serve for the CE routers. The customer traffic is transported over the core inside of RSVP signaled LSPs. For more details on the operational of a MPLS-Based L3 VPN, see Example: Configure a Basic MPLS-Based Layer 3 VPN.
Routed Protocols	

IPv4 and IPv6	All routers are configured to support routing of both IPv4 and IPv6.
Analyzer (monitoring station)	
Centos and Wireshark	The analyzer runs Centos 7.x with a GUI version of Wireshark.

Topology Overview

This example uses the context of a MPLS-based L3 VPN to demonstrate the remote port mirroring feature on PTX routers. The L3 VPN is configured to support both IPv4 and IPv6 traffic between the customer edge (CE) and provider edge (PE) routers.

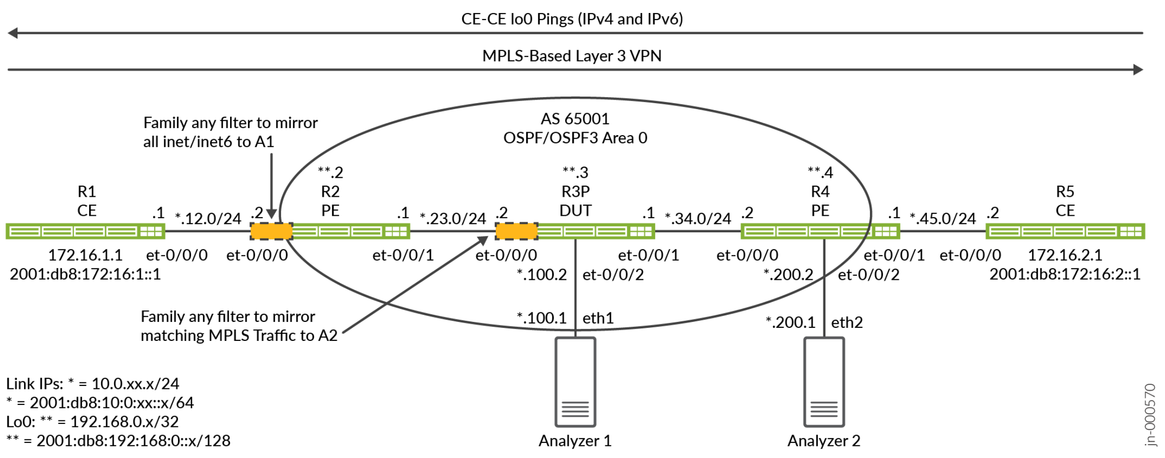
Table 4: Remote Port Mirroring Sample Topology Overview

Router Name	Role	Function
CE	Customer Edge (CE) router that sends test traffic to confirm port mirroring works properly.	These routers are designated as CE routers. The CE routers obtain L3 VPN service from the provider network. The CEs don't share the same OSPF routing domain as the provider routers.
PE	Provider Edge (PE) router attached to the CE.	The PEs are found at the edge of a provider network. Our PEs support Layer 3 VPNs through the use of routing instances, MP-BGP, RSVP, and an MPLS data plane. The PE1 router functions as one of the remote port mirroring DUTs.
P	A Provider (P) core router.	The P router represents a BGP-free provider core router. It supports OSPF, OSPF3, and MPLS transport. It does not run BGP or carry VPN state. The P router functions as one of the remote port mirroring DUTs.

Analyzer	The analyzer device receives the mirror traffic for storage and analysis.	The specifics of the analyzer are outside the scope of this document. There are a number of open source and commercial options available. Our analyzer happens to be running Centos 7.x with a Gnome desktop supporting a GUI version of Wireshark.

Topology Illustrations

Figure 3: Remote Port Mirroring



This example demonstrates two ways of mirroring CE traffic sent over the provider network:

- The first method uses a match-all filter on the PE-CE VRF interface.
- The second method demonstrates a MPLS label matching filter that is applied on the provider (P) router.

The PE1 router (R2) and the P router (R3) are the routers where remote port mirroring is configured and function as our DUTs. These routers use family `any` firewall filters to match on select traffic for port mirroring. A combination of ingress and egress filters are employed to mirror both request and response traffic flowing between the CE routers (R1 and R5) .

Remote port mirroring uses a tunnel for GRE encapsulation to send mirrored traffic to a remote analyzer device. Our topology has two analyzers. One is attached to the R2/PE1 router and the other to the R3/P router. This allows us to demonstrate two ways of mirroring CE traffic, one at a PE and the other on a core P router. We use a Centos host with Wireshark for packet capture and analysis.

PTX platforms use the Flexible Tunnel Interface (fti) infrastructure to support variety of tunneling applications. In the case of remote port mirrors, GRE tunnels are configured on fti interfaces to transport mirrored traffic to a remote analyzer device . As part of this example, you'll configure fti based GRE tunnels, a mirroring instance, and the firewall filters that select traffic to be mirrored.

R2/PE1 Configuration Steps

For information about navigating the CLI, see Using the CLI Editor in Configuration Mode

NOTE: For complete configuration on all routers see: "Appendix 2: Set Commands on All Routers" on page 89

This section highlights the configuration tasks needed to configure the PE1/R2 router in this example. We provide full configurations for all routers in the appendix. Step 1 summarizes the example's baseline. This baseline consists of IPv4 and IPv6 connectivity, MPLS, and a Layer 3 VPN. Our example places the focus on configuring and verifying remote port mirroring.

NOTE: For details on the L3 VPN operation and baseline configuration see Example: Configure a Basic MPLS-Based Layer 3 VPN.

1. Configure the IP routing and L3 VPN baseline on the PE1 router. This involves the following:

   a. Numbering the interfaces for both IPv4 and IPv6 and including `family mpls` support on core facing interfaces.

   b. Configuring the OSPF and OSPFv3 routing protocols to provide reachability between all network interfaces.

   c. RSVP and MPLS label switched paths (LSPs) to support L3 VPN traffic.

   d. Configuring VRF and BGP peering with the `inet-vpn and inet6-vpn` address families for a PE router. Our VRF example uses OSPF and OSPF3 as the PE-CE routing protocol.

```
[edit]
set interfaces et-0/0/0 unit 0 family inet address 10.0.12.2/24
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:12::2/64
set interfaces et-0/0/1 unit 0 family inet address 10.0.23.1/24
set interfaces et-0/0/1 unit 0 family inet6 address 2001:db8:10:0:23::1/64
set interfaces et-0/0/1 unit 0 family mpls

set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:192:168:0::2/128

set policy-options policy-statement ospf-export term 1 from protocol bgp
set policy-options policy-statement ospf-export term 1 then accept

set routing-instances ce1 instance-type vrf
set routing-instances ce1 protocols ospf area 0.0.0.0 interface all
set routing-instances ce1 protocols ospf export ospf-export
set routing-instances ce1 protocols ospf3 area 0.0.0.0 interface all
set routing-instances ce1 protocols ospf3 export ospf-export
set routing-instances ce1 interface et-0/0/0.0
set routing-instances ce1 route-distinguisher 65001:1
set routing-instances ce1 vrf-target target:65001:100
set routing-instances ce1 vrf-table-label

set routing-options router-id 192.168.0.2
set routing-options autonomous-system 65001

set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 192.168.0.2
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family inet6-vpn unicast
set protocols bgp group ibgp neighbor 192.168.0.4

set protocols mpls label-switched-path pe2 to 192.168.0.4
set protocols mpls label-switched-path pe2 no-cspf
set protocols mpls ipv6-tunneling
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable

set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable

set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
set protocols rsvp interface all
set protocols rsvp interface fxp0 disable
```

With the baseline covered the remaining steps focus on configuring the R2/PE1 router to port mirror all traffic sent and received on its CE1 facing VRF interface.

2. You begin with configuration of the GRE tunnel. On a PTX router the tunnel is implemented over an fti interface. The source address for the GRE tunnel does not have to be routable, unless you expect to perform diagnostic ping testing sourced from or destined to the GRE tunnel source. The GRE destination for traffic mirrored by PE1 is the analyzer 1 router. This destination must be reachable for mirror traffic to be sent to the analyzer. We use a passive IGP instance to ensure IGP reachability to the analyzer networks.

   The destination address maps to the Analyzer 1 device attached to the **et-0/0/2** interface on the P/R3 router. You must configure the fti logical interface with `family ccc` to support remote port mirroring on the PTX. This is because the mirror action occurs at Layer 2.

   ```
   set interfaces fti0 unit 1 description "GRE tunnel for remote port mirror"
   set interfaces fti0 unit 1 tunnel encapsulation gre source address 10.100.0.1
   set interfaces fti0 unit 1 tunnel encapsulation gre destination address 10.0.100.1
   set interfaces fti0 unit 1 family ccc
   ```

   NOTE: A passive IGP instance is provisioned for the interfaces attached to the analyzer devices. This provides IGP reachability to the analyzer ports for the GRE encapsulated mirrored traffic. The passive setting prevents the generation of hello packets to the analyzers as this would just clutter up the captures.

   In addition, we configured a static route on the analyzer device to allow it to respond to pings as an aid in diagnostic testing. Strictly speaking, only simplex connectivity or routing is needed between the DUTs and the analyzer for remote port mirroring to function.

3. Configure the sampling instance. We use a rate of 1 to sample all matching packets. The default `run-length` of 1 is left in place given all matching traffic is already sampled. You must specify the logical interface on the fti router that is used to transport the mirror traffic. You configured `unit 1` on the fti0

interface for the GRE tunnel in the previous step, so that same interface and unit is specified as the output interface in the mirror instance.

```
[edit]
set forwarding-options port-mirroring instance pe-mirror input rate 1
set forwarding-options port-mirroring instance pe-mirror family any output interface fti0.1
```

NOTE: You can specify a maximum packet length for mirrored traffic at the `[edit forwarding-options port-mirroring instance *instance-name* input]` hierarchy with the `maximum-packet-length` option. By default the packet length is 0, which means the entire packet is mirrored.

4. Define two family `any` firewall filters to match on and mirror the CE traffic. Two filters are defined, one for mirroring CE1 to CE2 traffic, and the other for mirroring CE2 to CE1. The filters include a count function to assist in verification. The port mirror action directs matching traffic to the port mirroring instance you configured previously.

   Family `any` filters support both Layer 2 and Layer 3 matching. For the former, you can match on VLAN ID, interface, MAC address, or MPLS label. For the latter, you can match on standard IPv4 or IPv6 headers fields.

   Given our topology we use a match all term that catches **all** traffic sent or received by the CEs. This includes IPv4, IPv6, ARP, LLDP, and any routing protocols such as OSPF.

```
[edit]
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 then count ce1-ce2-mirror
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 then port-mirror-instance pe-mirror
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 then accept
set firewall family any filter ce1-ce2 term else then accept

set firewall family any filter ce2-ce1 term mirror-ce2-ce1 then count ce2-ce1-mirror
set firewall family any filter ce2-ce1 term mirror-ce2-ce1 then port-mirror-instance pe-mirror
set firewall family any filter ce2-ce1 term mirror-ce2-ce1 then accept
set firewall family any filter ce2-ce1 term else then accept
```

   Both filters end with a match all accept term to override the default deny all action at the end of a Junos filter. In this way traffic that is not matched in the first term is accepted. This term is added as a safeguard in the event that later a specific match condition is added to the first term.

If desired, you can evoke a policer action in your filter to limit the number of mirrored packets sent over the GRE tunnel. The policer is defined with a bandwidth and burst limit along with a discard action for traffic that exceeds the policer.

You cannot apply a PM filter with a policer action in the egress direction.

NOTE: If you wanted to match only ICMP traffic sent between the IPv4 loopback addresses of the CE devices add Layer 3 match criteria to your filter.

```
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 from ip-version ipv4 ip-protocol icmp
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 from ip-version ipv4 ip-source-address 172.16.1.1/32
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 from ip-version ipv4 ip-destination-address 172.16.2.1/32
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 then count ce1-ce2-mirror
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 then port-mirror-instance pe-mirror
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 then accept
set firewall family any filter ce1-ce2 term else then accept
```

5. Apply the filters to the CE facing interface on R2/PE1. For our example, that means application of the filters to the **et-0/0/0** interface on PE1. Note the directionality of the filter application. An ingress and egress filter is needed to mirror both directions of CE traffic flow.

```
[edit]
set interfaces et-0/0/0 unit 0 filter input ce1-ce2
set interfaces et-0/0/0 unit 0 filter output ce2-ce1
```

For completeness we show the configuration of R2/PE1 in curly brace format.

```
system {
    host-name r2-ptx;
}
interfaces {
    et-0/0/0 {
        unit 0 {
            filter {
                input ce1-ce2;
                output ce2-ce1;
            }
            family inet {
                address 10.0.12.2/24;
            }
            family inet6 {
                address 2001:db8:10:0:12::2/64;
            }
        }
    }
    et-0/0/1 {
        unit 0 {
            family inet {
                address 10.0.23.1/24;
            }
            family inet6 {
                address 2001:db8:10:0:23::1/64;
            }
            family mpls;
        }
    }
    fti0 {
        unit 1 {
            description "GRE tunnel for remote port mirror";
            tunnel {
                encapsulation gre {
                    source {
                        address 10.100.0.1;
                    }
                    destination {
                        address 10.0.100.1;
                    }
                }
            }
            family ccc;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.168.0.2/32;
            }
            family inet6 {
                address 2001:db8:192:168:0::2/128;
            }
        }
    }
}
forwarding-options {
    port-mirroring {
        instance {
            pe-mirror {
                input {
                    rate 1;
                }
                family any {
                    output {
                        interface fti0.1;
                    }
                }
            }
        }
    }
}
policy-options {
    policy-statement ospf-export {
        term 1 {
            from protocol bgp;
            then accept;
        }
    }
}
firewall {
    family any {
        filter ce1-ce2 {
            term mirror-ce1-ce2 {
                then {
                    count ce1-ce2-mirror;
                    port-mirror-instance pe-mirror;
                    accept;
                }
            }
        }
        filter ce2-ce1 {
            term mirror-ce2-ce1 {
                then {
                    count ce2-ce1-mirror;
                    port-mirror-instance pe-mirror;
                    accept;
                }
            }
        }
    }
}
routing-instances {
    ce1 {
        instance-type vrf;
        protocols {
            ospf {
                area 0.0.0.0 {
                    interface all;
                }
                export ospf-export;
            }
            ospf3 {
                area 0.0.0.0 {
                    interface all;
                }
                export ospf-export;
            }
        }
        interface et-0/0/0.0;
        route-distinguisher 65001:1;
        vrf-target target:65001:100;
        vrf-table-label;
    }
}
routing-options {
    router-id 192.168.0.2;
    autonomous-system 65001;
}
protocols {
    bgp {
        group ibgp {
            type internal;
            local-address 192.168.0.2;
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            family inet6-vpn {
                unicast;
            }
            neighbor 192.168.0.4;
        }
    }
    mpls {
        label-switched-path pe2 {
            to 192.168.0.4;
            no-cspf;
        }
        ipv6-tunneling;
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
        }
    }

    ospf3 {
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
            interface et-0/0/2.0 {
                passive;
            }
        }
    }
    rsvp {
        interface all;
    }
}
```

R3 Configuration Steps

For information about navigating the CLI, see Using the CLI Editor in Configuration Mode

NOTE: For complete configuration on all routers see: "Appendix 2: Set Commands on All Routers" on page 89

This section highlights the main configuration tasks needed to configure the P/R3 router in this example. We start with the MPLS-based L3 VPN baseline. Then, we show the steps needed to configure remote port mirroring on a P router to match on and mirror MPLS traffic.

1. Configure the IPv4 and IPv6 routing and MPLS baseline on the P/R3 router. This involves a few things:

   a. Numbering the interfaces for both IPv4 and IPv6 and including `family mpls` support on core facing interfaces.

   b. Configure the OSPF and OSPFv3 routing protocols to provide reachability between all network interfaces.

   c. RSVP and MPLS to support the L3 VPN data plane. As a P router, BGP peering and VRF definitions are not present.

   ```
   [edit]
   set interfaces et-0/0/0 description "R3-R2 P-PE"
   set interfaces et-0/0/0 unit 0 family inet address 10.0.23.2/24
   set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:23::2/64
   set interfaces et-0/0/0 unit 0 family mpls
   set interfaces et-0/0/1 unit 0 family inet address 10.0.34.1/24
   set interfaces et-0/0/1 unit 0 family inet6 address 2001:db8:10:0:34::1/64
   set interfaces et-0/0/1 unit 0 family mpls
   set interfaces et-0/0/2 unit 0 family inet address 10.0.100.2/24
   set interfaces lo0 unit 0 family inet address 192.168.0.3/32
   set interfaces lo0 unit 0 family inet6 address 2001:db8:192:168:0::3/128

   set routing-options router-id 192.168.0.3
   ```

```
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols mpls interface et-0/0/2.0 disable

set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface et-0/0/2.0 passive

set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface et-0/0/2.0 passive

set protocols rsvp interface all
set protocols rsvp interface fxp0 disable
set protocols rsvp interface et-0/02.0 disable
```

The P1/R3 router attaches to Analyzer 1 over the **et-0/0/2** interface. We disable the RSVP protocol and MPLS support on this interface as the mirrored traffic arrives as IPv4 using GRE encapsulation. LSPs don't extend to the analyzer.

NOTE: The **et-0/0/2** interface configuration used here assumes that the analyzer replies to ARP and ND request sent by the DUT for MAC address resolution. If this is not the case, or if you wish that ARP traffic is not part of your packet captures, you should configure a static ARP entry. Be sure to specify the correct MAC address for the interface on the analyzer device that is attached to the DUT.

With the baseline covered, the following steps focus on configuring remote port mirroring for MPLS traffic on a P router.

2. You begin with the definition of the GRE tunnel. On PTX platforms tunnels are implemented on an **fti** interface. The source address for the GRE tunnel does not have to be routable, unless you expect to perform diagnostic ping testing sourced from or destined to the GRE tunnel source. The GRE destination for mirrored traffic is the Analyzer 2 device attached to PE2/R4. This destination must be reachable for mirror traffic to be sent to the analyzer. We use a passive IGP instance to ensure IGP reachability to the analyzer networks.

   You must configure the fti logical interface with `family ccc` to support remote port mirroring on the PTX. This is because the mirror action occurs at Layer 2.

```
[edit]
set interfaces fti0 unit 1 tunnel encapsulation gre source address 10.100.0.3
set interfaces fti0 unit 1 tunnel encapsulation gre destination address 10.0.200.1
set interfaces fti0 unit 1 family ccc
```

NOTE: A passive IGP instance is provisioned for the interfaces attached to the analyzer devices. This provides IGP reachability to the analyzer ports for the GRE encapsulated mirrored traffic. The passive setting prevents the generation of hello packets to the analyzers as this would just clutter up the packet captures.

In addition, we configured a static route on the analyzer device so it can respond to pings as an aid in diagnostic testing. Strictly speaking, only simplex connectivity or routing is needed between the DUTs and the analyzer for remote port mirroring to function.

3. Configure the sampling instance. We use a rate of 1 to select and sample all matching packets. By default the `run-length` is 1. This is fine given all matching traffic is sampled with a rate of 1. You must specify the logical interface on the fti router that is used to send the mirror traffic. You configured unit 1 for the fti interface in the previous step so that same unit is specified as the output interface for the mirror instance.

```
[edit]
set forwarding-options port-mirroring instance p-router-mirror input rate 1
set forwarding-options port-mirroring instance p-router-mirror family any output interface
fti0.1
```

4. Define the firewall filters to mirror traffic sent between the CE routers.

   Family `any` filters only allow Layer 2 match types. For example, VLAN ID, interface, MAC address, or MPLS label. As a result you are not able to use IPv4 or IPv4 specific match conditions.

   Two filters are defined, one for each direction of traffic flow between the CEs.

   Given our goal of mirroring VPN traffic at a P router, the filters are written to match on the specific labels that identify MPLS traffic flows between the two PE routers.

   For the CE1 to CE2 direction, the filter matches on the RSVP transport label used by PE1 to reach PE2. Because of PHP and egress application, the filter used in the CE2 to CE1 direction matches on the VRF label advertised by PE1 to PE2. Matching traffic evokes the port mirror action to the mirroring instances defined previously. The filters include a count function to assist in confirmation of proper operation.

   We determined the correct labels using these commands:

a. For traffic sent by CE1 to CE2, the current RSVP transport label is displayed with a `show rsvp session ingress detail` command. This is the RSVP assigned label used by PE1 to reach PE2 using MPLS. It should be noted that all VPN traffic sent between this PE pair uses the same RSVP transport. The resulting filter is not specific to the CE1 VRF on the PE router.

```
user@r2-ptx> show rsvp session ingress detail

Ingress RSVP: 1 sessions

192.168.0.4
  From: 192.168.0.2, LSPstate: Up, ActiveRoute: 0
  LSPname: pe2, LSPpath: Primary
  LSPtype: Static Configured
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: 22
  Resv style: 1 FF, Label in: -, Label out: 22
  Time left:    -, Since: Tue Apr 18 12:58:47 2023
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 2 receiver 65196 protocol 0
  PATH rcvfrom: localclient
  Adspec: sent MTU 1500
  Path MTU: received 1500
  PATH sentto: 10.0.23.2 (et-0/0/1.0) 1 pkts
  RESV rcvfrom: 10.0.23.2 (et-0/0/1.0) 1 pkts, Entropy label: Yes
  Record route: <self> 10.0.23.2 10.0.34.2
Total 1 displayed, Up 1, Down 0
```

In this case the output shows that RSVP label 22 is assigned to the PE1/R2 router to reach the LSP egress at the PE2/R4 router.

b. For traffic sent by CE2 to CE1 you must match on the VRF label. This is because the P router is the penultimate hop node and on the egress interface it will have popped the RSVP transport label. This leaves on the VRF label as the bottom of the stack. You confirm the VRF label advertised by PE1 to the PE2 with a `show route advertising-protocol bgp` *`remote-peer-address`* `detail` command. This command shows the routes advertised by the local PE along with the VRF label that is bound to the routing instance.

```
user@r2-ptx> show route advertising-protocol bgp 192.168.0.4 detail

ce1.inet.0: 7 destinations, 8 routes (7 active, 0 holddown, 0 hidden)
* 10.0.12.0/24 (1 entry, 1 announced)
 BGP group ibgp type Internal
```

```
     Route Distinguisher: 65001:1
     VPN Label: 16
     Nexthop: Self
     Flags: Nexthop Change
     Localpref: 100
     AS path: [65001] I
     Communities: target:65001:100

* 172.16.1.1/32 (1 entry, 1 announced)
 BGP group ibgp type Internal
     Route Distinguisher: 65001:1
     VPN Label: 16
     Nexthop: Self
     Flags: Nexthop Change
     MED: 1
     Localpref: 100
     AS path: [65001] I
     Communities: target:65001:100 rte-type:0.0.0.0:1:0
. . .
```

The output shows that PE1 signaled VRF label 16 to the PE2 router for the routes associated with the CE1 routing instance.

Using the information from the previous commands we know which RSVP/VRF labels to match on in the firewall filter.

```
[edit]
set firewall family any filter ce1-ce2 term ce1-ce2-mirror from mpls-label 22
set firewall family any filter ce1-ce2 term ce1-ce2-mirror then count ce1-ce2-traffic
set firewall family any filter ce1-ce2 term ce1-ce2-mirror then port-mirror-instance p-
router-mirror
set firewall family any filter ce1-ce2 term else then accept

set firewall family any filter ce2-ce1 term ce2-ce1-mirror from mpls-label 16
set firewall family any filter ce2-ce1 term ce2-ce1-mirror then count ce2-ce1-traffic
set firewall family any filter ce2-ce1 term ce2-ce1-mirror then port-mirror-instance p-
router-mirror
set firewall family any filter ce2-ce1 term else then accept
```

The filters end with a match all accept term to override the default deny all at the end of a Junos filter. In this way traffic that is not matched in the first term is accepted. This is critical to avoiding disrupting all other traffic using this interface!

NOTE: RSVP labels can change due to LSP re-signaling caused by link outages or other configuration changes. We demonstrate how a filter can be used to match on specific labels to help constrain the traffic that is mirrored. You can always apply a match all filter to ensure that MPLS label changes does not affect mirroring. The disadvantage of the match all approach is you will mirror all traffic received on the P router interface to include the core protocols and non-VPN traffic.

5. Apply the filters to the PE1 facing interface on the P/R3 router. The directionality of the filter application is important. Our filters are designed to operate in the input direction for CE1 to CE2 traffic, and in the egress direction for CE2 to CE1 traffic. Because these are family any filters, they are applied at the unit level independent of either IPv4 or IPv6. Family any filters operate at Layer 2 which is independent of any protocol family.

```
[edit]
set interfaces et-0/0/0 unit 0 filter input ce1-ce2
set interfaces et-0/0/0 unit 0 filter output ce2-ce1
```

For completeness we show the configuration of R2/PE1 in curly brace format.

```
system {
    host-name r3-ptx;
}
interfaces {
    et-0/0/0 {
        description "R3-R2 P-PE";
        unit 0 {
            filter {
                input ce1-ce2;
                output ce2-ce1;
            }
            family inet {
                address 10.0.23.2/24;
            }
            family inet6 {
                address 2001:db8:10:0:23::2/64;
            }
            family mpls;
        }
    }
    et-0/0/1 {
        unit 0 {
            family inet {
                address 10.0.34.1/24;
            }
            family inet6 {
                address 2001:db8:10:0:34::1/64;
            }
            family mpls;
        }
    }
    et-0/0/2 {
        unit 0 {
            family inet {
                address 10.0.100.2/24;
            }
        }
    }
    fti0 {
        unit 1 {
            tunnel {
                encapsulation gre {
                    source {
                        address 10.100.0.3;
                    }
                    destination {
                        address 10.0.200.1;
                    }
                }
            }
            family ccc;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 192.168.0.3/32;
            }
            family inet6 {
                address 2001:db8:192:168:0::3/128;
            }
        }
    }
}
```

```
forwarding-options {
    port-mirroring {
        instance {
            p-router-mirror {
                input {
                    rate 1;
                }
                family any {
                    output {
                        interface fti0.1;
                    }
                }
            }
        }
    }
}
firewall {
    family any {
        filter ce1-ce2 {
            term ce1-ce2-mirror {
                from {
                    mpls-label 22;
                }
                then {
                    count ce1-ce2-traffic;
                    port-mirror-instance p-router-mirror;
                }
            }
            term else {
                then accept;
            }
        }
        filter ce2-ce1 {
            term ce2-ce1-mirror {
                from {
                    mpls-label 16;
                }
                then {
                    count ce2-ce1-traffic;
                    port-mirror-instance p-router-mirror;
                }
            }
            term else {
                then accept;
            }
        }
    }
}
routing-options {
    router-id 192.168.0.3;
}
protocols {
    mpls {
        interface all;
        interface fxp0.0 {
            disable;
        }
        interface et-0/0/2.0 {
            disable;
        }
    }
    ospf {
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
            interface et-0/0/2.0 {
                passive;
            }
        }
    }
    ospf3 {
        area 0.0.0.0 {
            interface all;
            interface fxp0.0 {
                disable;
            }
            interface et-0/0/2.0 {
                passive;
            }
        }
    }
    rsvp {
        interface all;
        interface et-0/0/2.0 {
            disable;
        }
    }
}
```

Verification

In this example, there are two DUTs where port mirroring is configured. The verification mechanisms are the same. In both cases you'll confirm that the filter is matching on expected traffic, and that mirrored packets are sent to the associated analyzer.

With our configuration PE mirrored traffic is sent to Analyzer 1 while P router mirrored traffic is sent to Analyzer 2. If desired all mirrored traffic can be sent to the same destination but, this results in the interspersing of traffic if port mirroring occurs in multiple places simultaneously.

These steps can be performed on either or both of the DUT routers as desired. R2/PE1 is the first DUT and the P router/R3 is the second.

1. Confirm OSPF and OSPF3 neighbors and routes to all loopback addresses. Also, verify the route to the remote analyzer IP address. You must be able to send IPv4 packets to the remote analyzer. Optionally, the analyzer can be configured with static routes so it can reply.

```
user@r2-ptx> show ospf neighbor

Address          Interface              State          ID               Pri  Dead
10.0.23.2        et-0/0/1.0             Full           192.168.0.3      128    36

user@r2-ptx> show ospf3 neighbor

ID               Interface              State     Pri   Dead
192.168.0.3      et-0/0/1.0             Full      128     36
  Neighbor-address fe80::569e:18ff:fe45:ffff



user@r2-ptx> show route protocol ospf | match /32
192.168.0.3/32      *[OSPF/10] 2d 22:41:49, metric 1
192.168.0.4/32      *[OSPF/10] 2d 22:41:49, metric 2
224.0.0.5/32        *[OSPF/10] 2d 22:43:37, metric 1
172.16.1.1/32       *[OSPF/10] 2d 22:41:45, metric 1
224.0.0.5/32        *[OSPF/10] 2d 22:43:37, metric 1
```

```
user@r2-ptx> show route protocol ospf3 | match /128
2001:db8:192:168::3/128
2001:db8:192:168::4/128
ff02::5/128        *[OSPF3/10] 2d 22:47:10, metric 1
2001:db8:172:16:1::1/128
ff02::5/128        *[OSPF3/10] 2d 21:38:06, metric 1
```

NOTE: In the above output, the 172.16.1.1/32 route is learned in the *ce1* routing instance. Thus, with this command you have confirmed proper OSPF operation in both the core and customer edge! Only the local CE1 loopback is listed because the remote CE loopback is learned as a BGP route.

The passive IGP instance configured on the interface connected to the analyzers provided the needed IP connectivity. Again, we add static routes to the analyzers to permit return traffic for diagnostic testing.

```
user@r2-ptx> show route 10.0.100.1

inet.0: 15 destinations, 15 routes (15 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.100.0/24      *[OSPF/10] 2d 22:49:10, metric 2
                    >  to 10.0.23.2 via et-0/0/1.0
user@r2-ptx> show route 10.0.200.1

inet.0: 15 destinations, 15 routes (15 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.200.0/24      *[OSPF/10] 1d 20:36:13, metric 3
                    >  to 10.0.23.2 via et-0/0/1.0

user@r2-ptx> ping 10.0.100.1 count 2
PING 10.0.100.1 (10.0.100.1) 56(84) bytes of data.
64 bytes from 10.0.100.1: icmp_seq=1 ttl=63 time=5.48 ms
64 bytes from 10.0.100.1: icmp_seq=2 ttl=63 time=4.91 ms

--- 10.0.100.1 ping statistics ---
```

```
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 4.908/5.196/5.484/0.288 ms
```

2. Confirm fti interface and GRE tunnel status.

```
user@r2-ptx> show interfaces fti0.1 detail
  Logical interface fti0.1 (Index 1000) (SNMP ifIndex 543) (Generation 609885357005)
    Description: GRE tunnel for remote port mirror
    Flags: Up Point-To-Point Encapsulation: GRE DF
          , Source address: 10.100.0.1/32, Destination address: 10.0.100.1
  Traffic statistics:
   Input  bytes  :                    0
   Output bytes  :             12887342
   Input  packets:                    0
   Output packets:                99020
    Local statistics:
     Input  bytes  :                    0
     Output bytes  :                    0
     Input  packets:                    0
     Output packets:                    0
    Transit statistics:
     Input  bytes  :                    0                    0 bps
     Output bytes  :             12887342                    0 bps
     Input  packets:                    0                    0 pps
     Output packets:                99020                    0 pps
    Protocol ccc, MTU: 1476, Generation: 609885357006, Route table: 0
      Flags: Is-Primary
```

The highlights in the output indicate the tunnel interface and GRE tunnel are operational. The non-0 output packet counter is a good sign that traffic is being mirrored. We don't expect any input packets as this GRE tunnel is used only to send mirrored traffic to a remote analyzer.

3. Confirm the port mirroring instance. The port mirroring state should be `up`, and the correct mirroring interface should be listed under the destination. The family `any` indicates this a Layer 2 port mirror instance that is protocol family agnostic. Mirrored traffic includes the original Layer 2 frame along with its contents.

```
user@r2-ptx> show forwarding-options port-mirroring
Instance Name: pe-mirror
  Instance Id: 1
  Input parameters:
    Rate                  : 1
    Run-length            : 1
    Maximum-packet-length : 0
  Output parameters:
    Family              State     Destination          Next-hop
    any                 up        fti0.1               NA
```

4. Clear the firewall counters and interface statistics on the DUT. Then generate a known number of IPv4 and IPv6 test packets between the CE router loopback addresses.

```
user@r2-ptx> clear firewall all
user@r2-ptx> clear interfaces statistics all
```

```
user@r1-ptx> ping 172.16.2.1 source 172.16.1.1 count 4
PING 172.16.2.1 (172.16.2.1) from 172.16.1.1 : 56(84) bytes of data.
64 bytes from 172.16.2.1: icmp_seq=1 ttl=61 time=1237 ms
64 bytes from 172.16.2.1: icmp_seq=2 ttl=61 time=178 ms
64 bytes from 172.16.2.1: icmp_seq=3 ttl=61 time=507 ms
64 bytes from 172.16.2.1: icmp_seq=4 ttl=61 time=417 ms

--- 172.16.2.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3066ms
rtt min/avg/max/mdev = 177.830/584.676/1237.435/395.575 ms, pipe 2

user@r1-ptx> ping 2001:db8:172:16:2::1 source 2001:db8:172:16:1::1 count 4
64 bytes from 2001:db8:172:16:2::1: icmp_seq=1 ttl=62 time=978 ms
64 bytes from 2001:db8:172:16:2::1: icmp_seq=2 ttl=62 time=621 ms
64 bytes from 2001:db8:172:16:2::1: icmp_seq=3 ttl=62 time=782 ms
64 bytes from 2001:db8:172:16:2::1: icmp_seq=4 ttl=62 time=920 ms

--- 2001:db8:172:16:2::1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 621.122/825.114/978.021/137.715 ms
```

5. Back on R2/PE1, display firewall counters and interface statistics to confirm they reflect the test traffic. You may see some extra packets counted in the CE1 to CE 2 direction that reflect the OSPF, OSPF3, or ARP exchanges between the CE12 and PE1 routers. Note that in the CE12 to CE1

direction the filter application in the egress direction only catches end-to-end traffic. Therefore the CE2-CE1 counter does reflect the test traffic generated.

```
user@r2-ptx> show firewall

Filter: ce1-ce2
Counters:
Name
Bytes              Packets
ce1-ce2-mirror
1353                   13

Filter: ce2-ce1
Counters:
Name
Bytes              Packets
ce2-ce1-mirror
752                    8
```

Display interface statistics for the fti0./1 interface used to mirror traffic to the remote analyzer.

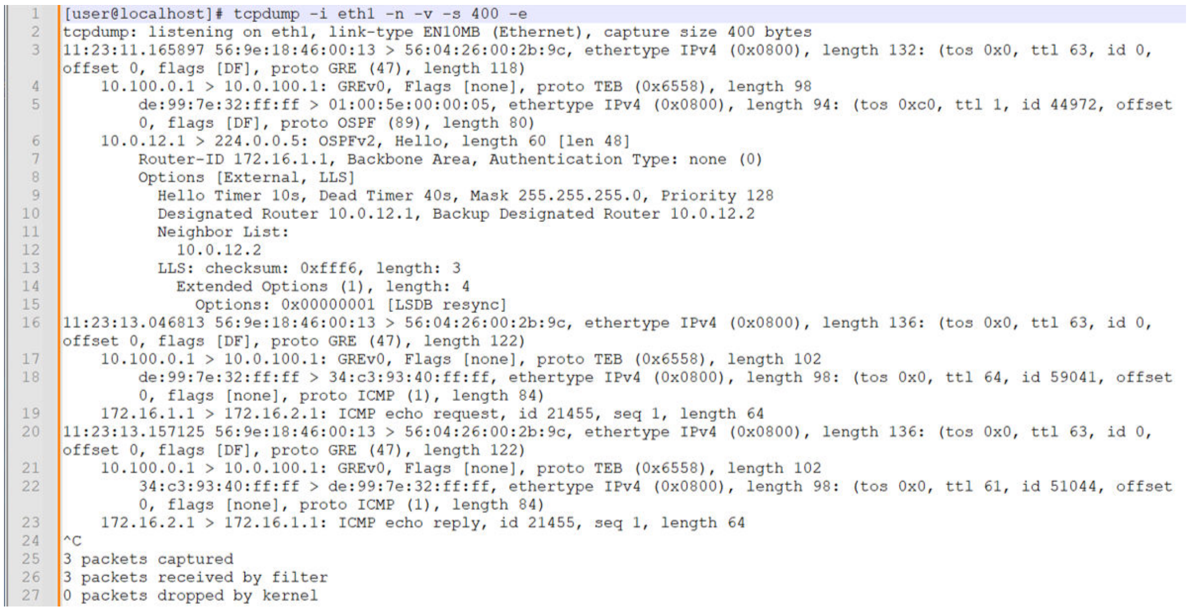
```
user@r3-ptx> show interfaces fti0.1 detail
  Logical interface fti0.1 (Index 1000) (SNMP ifIndex 543) (Generation 609885357005)
    Description: GRE tunnel for remote port mirror
    Flags: Up Point-To-Point Encapsulation: GRE DF
          , Source address: 10.100.0.1/32, Destination address: 10.0.100.1
  Traffic statistics:
   Input  bytes  :                   0
   Output bytes  :                2424
   Input  packets:                   0
   Output packets:                  20
    Local statistics:
     Input  bytes  :                   0
     Output bytes  :                   0
     Input  packets:                   0
     Output packets:                   0
    Transit statistics:
     Input  bytes  :                   0                    0 bps
     Output bytes  :                2424                 2096 bps
     Input  packets:                   0                    0 pps
     Output packets:                  20                    2 pps
    Protocol ccc, MTU: 1476, Generation: 609885357006, Route table: 0
      Flags: Is-Primary
```

With 8 test packets generated you may be surprised to see a packet count on the egress of the fti0.1 interface of 20. First, recall that both OSPF and OSPF3 hellos packets sent by the CE1 router are being mirrored. Second, consider that both ping request and replies are being mirrored at PE1/R2. That means there are 8 ping request, and 8 replies, for a total of 16 ICMP test packets.

6. Run tcpdump or the analysis application of your choice on the monitoring station to confirm receipt and processing of the mirrored test traffic. Our setup has two analyzer devices, or precisely, one analyzer host with two interfaces. You can perform this step on both of the analyzer interfaces simultaneously, if desired. Recall that the traffic mirrored to the **eth2** interface on Analyzer 2 is mirrored at the P router/R3, and therefore includes MPLS encapsulation. Traffic mirrored from R1/PE1 does not include the MPLS encapsulation.

   We begin with a capture on the **eth1** interface that receives mirrored traffic from R2/PE1. After starting the capture we perform a single IPv4 ping between the CE routers.

   NOTE: After the capture, we pasted the text-based tcpdump output into a text application that displays line numbers. This makes it easier to call out key portions of the capture. We also enabled line wrap to improve visibility.



```
 1  [user@localhost]# tcpdump -i eth1 -n -v -s 400 -e
 2  tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 400 bytes
 3  11:23:11.165897 56:9e:18:46:00:13 > 56:04:26:00:2b:9c, ethertype IPv4 (0x0800), length 132: (tos 0x0, ttl 63, id 0,
    offset 0, flags [DF], proto GRE (47), length 118)
 4      10.100.0.1 > 10.0.100.1: GREv0, Flags [none], proto TEB (0x6558), length 98
 5          de:99:7e:32:ff:ff > 01:00:5e:00:00:05, ethertype IPv4 (0x0800), length 94: (tos 0xc0, ttl 1, id 44972, offset
            0, flags [DF], proto OSPF (89), length 80)
 6      10.0.12.1 > 224.0.0.5: OSPFv2, Hello, length 60 [len 48]
 7          Router-ID 172.16.1.1, Backbone Area, Authentication Type: none (0)
 8          Options [External, LLS]
 9            Hello Timer 10s, Dead Timer 40s, Mask 255.255.255.0, Priority 128
10            Designated Router 10.0.12.1, Backup Designated Router 10.0.12.2
11            Neighbor List:
12              10.0.12.2
13            LLS: checksum: 0xfff6, length: 3
14              Extended Options (1), length: 4
15                Options: 0x00000001 [LSDB resync]
16  11:23:13.046813 56:9e:18:46:00:13 > 56:04:26:00:2b:9c, ethertype IPv4 (0x0800), length 136: (tos 0x0, ttl 63, id 0,
    offset 0, flags [DF], proto GRE (47), length 122)
17      10.100.0.1 > 10.0.100.1: GREv0, Flags [none], proto TEB (0x6558), length 102
18          de:99:7e:32:ff:ff > 34:c3:93:40:ff:ff, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 64, id 59041, offset
            0, flags [none], proto ICMP (1), length 84)
19      172.16.1.1 > 172.16.2.1: ICMP echo request, id 21455, seq 1, length 64
20  11:23:13.157125 56:9e:18:46:00:13 > 56:04:26:00:2b:9c, ethertype IPv4 (0x0800), length 136: (tos 0x0, ttl 63, id 0,
    offset 0, flags [DF], proto GRE (47), length 122)
21      10.100.0.1 > 10.0.100.1: GREv0, Flags [none], proto TEB (0x6558), length 102
22          34:c3:93:40:ff:ff > de:99:7e:32:ff:ff, ethertype IPv4 (0x0800), length 98: (tos 0x0, ttl 61, id 51044, offset
            0, flags [none], proto ICMP (1), length 84)
23      172.16.2.1 > 172.16.1.1: ICMP echo reply, id 21455, seq 1, length 64
24  ^C
25  3 packets captured
26  3 packets received by filter
27  0 packets dropped by kernel
```

   Areas to note in the capture include:

   a. We evoke tcpdump on the **eth1** interface and include flags to prevent name resolution, provide detail, capture up to 400 bytes, and to include Layer 2 headers.

b. Line 3 is the start of the first Layer 2 frame and IP packet. The Ethernet frame is the encapsulation used by the R3/P1 router to send traffic to the locally attached analyzer device. The destination MAC address is owned by the **eth1** interface on Analyzer 1. The 100.0.100.1 IP to MAC address resolution is performed through ARP. The Ethernet frame indicates it carries the IP protocol. At the IP layer we see the packet has the Don't Fragment bit set and identifies that the payload is GRE.

c. Line 4 shows the decode of the outer IP packet and its GRE payload. The source and destination IP addresses reflect the GRE tunnel configured ion the **fti0.1** interface at R2/PE1. The GRE header identifies that its payload is a Layer 2 frame via the transparent Ethernet bridging (TEB) protocol ID. This confirms that remote port mirroring on PTX platforms with a family `any` filter results in the mirroring of Layer 2 frames.

d. Line 5 is the decode for the GRE packet's payload . The source and destination MAC addresses (de:99:7e:32:ff:ff and 01:00:5e:00:00:05, respectively) reflect those used for OSPF hello multicast at the link layer. Line 5 also shows that the payload of the GRE encapsulated Layer 2 frame is IP, and that the payload of the IP packet is OSPF.

NOTE: The OSPF hello exchange between CE and PE is local in a Layer 3 VPN. We see the OSPF packets sent by the local CE because the port mirror action at ingress captured all traffic. The OSPF hello packets generated by the remote CE are not transported over the core and therefore not seen as egress in the capture.

e. Line 6 decodes the OSPF hello sent by the CE1 router. The source IP address is assigned to the **et-0/0/0.0** interface of CE1. The destination IP address is used for OSPF multicast.

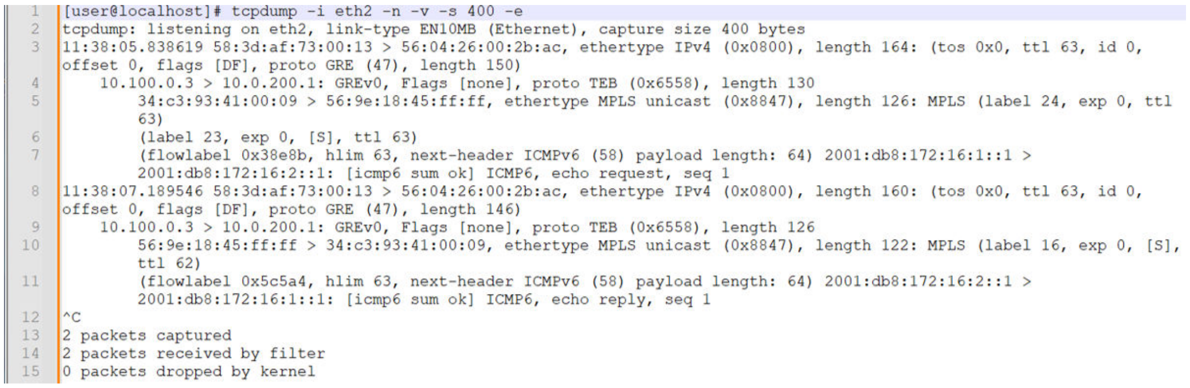
f. We skip over the OSPF decode and land on line 16. This is the second frame in the capture and reflects an IPv4 ICMP echo request. Once again, the Layer 2 frame reflects the MAC addresses of the P1/R3 and Analyzer 1 devices. We see that the outer frame carries an IP packet with a GRE payload. The source and destination IP address of the outer IP packet reflect the GRE tunnel configured at R2/PE2.

g. Line 18 begins the decode of GRE payload. We again see the MAC addresses of the CE1 and PE1 routers. At the IP layer we see the packet is sent from the loopback address of the CE1 router. The destination IP is the loopback address of CE2. The payload of the inner IP packet is the ICMP echo request as sent from CE1 to CE2.

h. Line 20 decodes the ICMP echo reply sent by CE2. This confirms the port mirroring is working in both the CE1 transmits and receive direction.

7. Next, we generate a single IPv6 ping between CE routers while capturing on the **eth2** interface of Analyzer 2. This confirms the port mirror configuration on the R3/P router as well as IPv6 port mirroring support.

NOTE: After the capture, we pasted the text-based tcpdump output into a text application that displays line numbers. This makes it easier to call out key portions of the capture. We also enabled line wrap to improve visibility.



```
 1  [user@localhost]# tcpdump -i eth2 -n -v -s 400 -e
 2  tcpdump: listening on eth2, link-type EN10MB (Ethernet), capture size 400 bytes
 3  11:38:05.838619 58:3d:af:73:00:13 > 56:04:26:00:2b:ac, ethertype IPv4 (0x0800), length 164: (tos 0x0, ttl 63, id 0,
    offset 0, flags [DF], proto GRE (47), length 150)
 4      10.100.0.3 > 10.0.200.1: GREv0, Flags [none], proto TEB (0x6558), length 130
 5          34:c3:93:41:00:09 > 56:9e:18:45:ff:ff, ethertype MPLS unicast (0x8847), length 126: MPLS (label 24, exp 0, ttl
            63)
 6          (label 23, exp 0, [S], ttl 63)
 7          (flowlabel 0x38e8b, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:db8:172:16:1::1 >
            2001:db8:172:16:2::1: [icmp6 sum ok] ICMP6, echo request, seq 1
 8  11:38:07.189546 58:3d:af:73:00:13 > 56:04:26:00:2b:ac, ethertype IPv4 (0x0800), length 160: (tos 0x0, ttl 63, id 0,
    offset 0, flags [DF], proto GRE (47), length 146)
 9      10.100.0.3 > 10.0.200.1: GREv0, Flags [none], proto TEB (0x6558), length 126
10          56:9e:18:45:ff:ff > 34:c3:93:41:00:09, ethertype MPLS unicast (0x8847), length 122: MPLS (label 16, exp 0, [S],
            ttl 62)
11          (flowlabel 0x5c5a4, hlim 63, next-header ICMPv6 (58) payload length: 64) 2001:db8:172:16:2::1 >
            2001:db8:172:16:1::1: [icmp6 sum ok] ICMP6, echo reply, seq 1
12  ^C
13  2 packets captured
14  2 packets received by filter
15  0 packets dropped by kernel
```

Note that both request and response traffic is shown. Given that this port mirror occurs on a P-router, OSPF packets between the CE routers are not mirrored as they are not sent over the provider core. Things to note in this capture include:

a. At line 3 the outer Ethernet frame is decoded. The source and destination MAC addresses now reflect the R4/PE2 and Analyzer devices, respectively.

b. At line 4 the inner IP packet is decoded. The frame indicates GRE encapsulation, and the source and destination IPs confirm this traffic is mirrored over the **fti0.1** GRE tunnel configured at the R3/P1 router. The GRE encapsulation shows the TEB protocol, indicating that a Layer 2 Ethernet frame is encapsulated.

c. Line 5 begins the decode of the inner Ethernet frame and its MPLS payload. The source MAC address is assigned to the **et-0/0/1** interface on the R2/PE1 router. The destination MAC is associated with the **et-0/0/0** interface on the R3/P1 router.

   The inner Ethernet frame identifies a payload of MPLS. This is in keeping with Layer 2 port mirroring performed at a P router with filter terms matching on MPLS labels.

   Note that in the CE1 to CE2 direction, the mirrored traffic shows two MPLS labels. The RSVP transport label is 24 (this can change due to LSP re-signaling), and the inner VRF label set to 23, which is the VRF label associated with the CE1 routing instance at the R2/PE1 router.

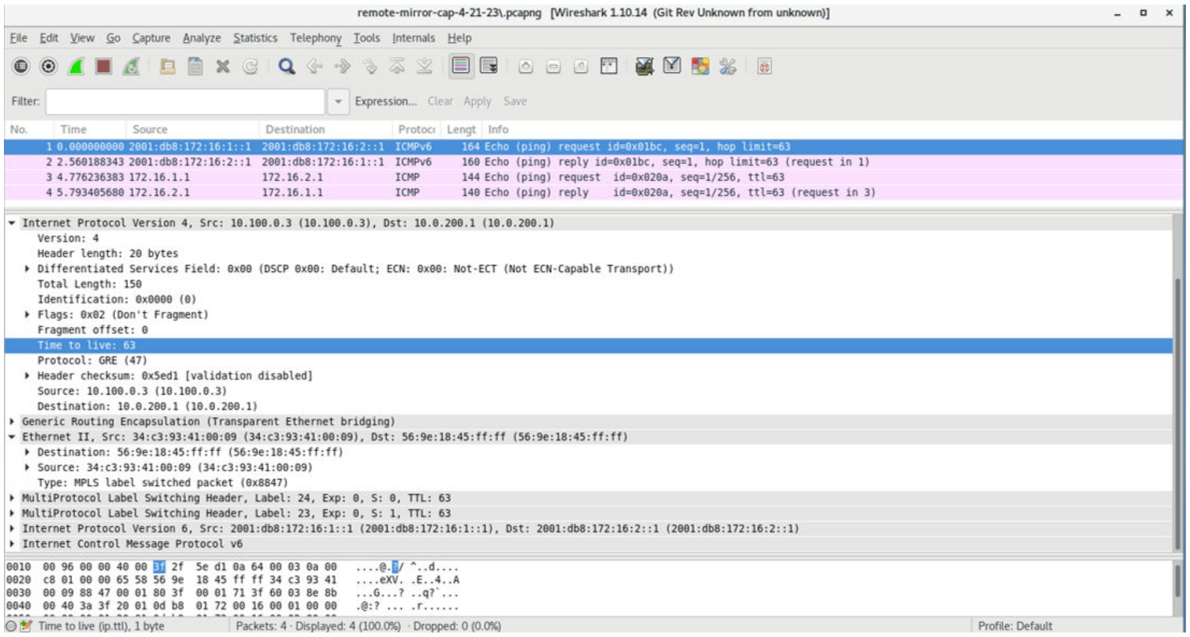
   NOTE: At the time of this capture the MPLS transport label used by PE1 to reach PE2 had changed. We updated the filter definition at R3/P1 to reflect the current RSVP transport label value of 24 for the captures in this section.

d. Line 7 decodes the IPv6 payload of the MPLS frame. This is the IPv6 packet sent by CE1 to the CE2. The IPv6 packet identifies its payload as ICMP6 and shows this is an echo request.

e. Line 8 begins the decode of the reply traffic from CE2. In the CE2 to CE1 direction only a single label is present in the mirror traffic. This is the VRF label that remains after the R3/PE1 router performs penultimate hop popping (PHP) before sending the traffic to the R2/PE1 router. The traffic mirrored at egress in the P1 to PE2 direction.

   The captures at both analyzer devices confirm that remote port mirroring is working as expected.

8. We conclude with a GUI decode of the same CE to CE test traffic as captured on the Analyzer 2 device. Note again the presence of MPLS labels reflecting a port mirror operation on a MPLS-based PE router.


remote-mirror-cap-4-21-23\.pcapng [Wireshark 1.10.14 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
Profile: Default

```
No.  Time        Source                Destination           Protoc Lengt Info
   1 0.000000000 2001:db8:172:16:1::1  2001:db8:172:16:2::1  ICMPv6   164 Echo (ping) request id=0x01bc, seq=1, hop limit=63
   2 2.560188343 2001:db8:172:16:2::1  2001:db8:172:16:1::1  ICMPv6   160 Echo (ping) reply id=0x01bc, seq=1, hop limit=63 (request in 1)
   3 4.776236383 172.16.1.1            172.16.2.1            ICMP     144 Echo (ping) request  id=0x020a, seq=1/256, ttl=63
   4 5.793405680 172.16.2.1            172.16.1.1            ICMP     140 Echo (ping) reply    id=0x020a, seq=1/256, ttl=63 (request in 3)

▾ Internet Protocol Version 4, Src: 10.100.0.3 (10.100.0.3), Dst: 10.0.200.1 (10.0.200.1)
    Version: 4
    Header length: 20 bytes
  ▸ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 150
    Identification: 0x0000 (0)
  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 63
    Protocol: GRE (47)
  ▸ Header checksum: 0x5ed1 [validation disabled]
    Source: 10.100.0.3 (10.100.0.3)
    Destination: 10.0.200.1 (10.0.200.1)
▸ Generic Routing Encapsulation (Transparent Ethernet bridging)
▾ Ethernet II, Src: 34:c3:93:41:00:09 (34:c3:93:41:00:09), Dst: 56:9e:18:45:ff:ff (56:9e:18:45:ff:ff)
  ▸ Destination: 56:9e:18:45:ff:ff (56:9e:18:45:ff:ff)
  ▸ Source: 34:c3:93:41:00:09 (34:c3:93:41:00:09)
    Type: MPLS label switched packet (0x8847)
▸ MultiProtocol Label Switching Header, Label: 24, Exp: 0, S: 0, TTL: 63
▸ MultiProtocol Label Switching Header, Label: 23, Exp: 0, S: 1, TTL: 63
▸ Internet Protocol Version 6, Src: 2001:db8:172:16:1::1 (2001:db8:172:16:1::1), Dst: 2001:db8:172:16:2::1 (2001:db8:172:16:2::1)
▸ Internet Control Message Protocol v6

0010  00 96 00 00 40 00 3f 2f  5e d1 0a 64 00 03 0a 00   ....@.?/ ^..d....
0020  c8 01 00 00 65 58 56 9e  18 45 ff ff 34 c3 93 41   ....eXV. .E..4..A
0030  00 09 88 47 00 01 80 3f  00 01 71 3f 60 03 8e 8b   ...G...? ..q?`...
0040  00 40 3a 3f 20 01 0d b8  01 72 00 16 00 01 00 00   .@:? ... .r......

Time to live (ip.ttl), 1 byte    Packets: 4 · Displayed: 4 (100.0%) · Dropped: 0 (0.0%)
```

The capture shows both IPv4 and IPv6 test traffic being mirrored. This capture reflects traffic that is mirrored by the P router. As a result MPLS encapsulation is present.

Appendix: Set Commands on All Routers

IN THIS SECTION

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the [edit] hierarchy level.

R1 (CE1)

```
del interfaces et-0/0/0
set interfaces et-0/0/0 unit 0 family inet address 10.0.12.1/24
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:12::1/64
set interfaces lo0 unit 0 family inet address 172.16.1.1/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:172:16:1::1/128
set routing-options router-id 172.16.1.1
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
```

R2 (PE1) DUT1

```
set interfaces et-0/0/0 unit 0 filter input ce1-ce2
set interfaces et-0/0/0 unit 0 filter output ce2-ce1
set interfaces et-0/0/0 unit 0 family inet address 10.0.12.2/24
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:12::2/64
set interfaces et-0/0/1 unit 0 family inet address 10.0.23.1/24
set interfaces et-0/0/1 unit 0 family inet6 address 2001:db8:10:0:23::1/64
set interfaces et-0/0/1 unit 0 family mpls
set interfaces fti0 unit 1 description "GRE tunnel for remote port mirror"
set interfaces fti0 unit 1 tunnel encapsulation gre source address 10.100.0.1
set interfaces fti0 unit 1 tunnel encapsulation gre destination address 10.0.100.1
set interfaces fti0 unit 1 family ccc
set interfaces lo0 unit 0 family inet address 192.168.0.2/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:192:168:0::2/128
set forwarding-options port-mirroring instance pe-mirror input rate 1
set forwarding-options port-mirroring instance pe-mirror input run-length 1
set forwarding-options port-mirroring instance pe-mirror family any output interface fti0.1
set policy-options policy-statement ospf-export term 1 from protocol bgp
set policy-options policy-statement ospf-export term 1 then accept
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 then count ce1-ce2-mirror
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 then port-mirror-instance pe-mirror
set firewall family any filter ce1-ce2 term mirror-ce1-ce2 then accept
set firewall family any filter ce2-ce1 term mirror-ce2-ce1 then count ce2-ce1-mirror
set firewall family any filter ce2-ce1 term mirror-ce2-ce1 then port-mirror-instance pe-mirror
set firewall family any filter ce2-ce1 term mirror-ce2-ce1 then accept
set routing-instances ce1 instance-type vrf
set routing-instances ce1 protocols ospf area 0.0.0.0 interface all
set routing-instances ce1 protocols ospf export ospf-export
set routing-instances ce1 protocols ospf3 area 0.0.0.0 interface all
set routing-instances ce1 protocols ospf3 export ospf-export
set routing-instances ce1 interface et-0/0/0.0
set routing-instances ce1 route-distinguisher 65001:1
set routing-instances ce1 vrf-target target:65001:100
set routing-instances ce1 vrf-table-label
set routing-options router-id 192.168.0.2
set routing-options autonomous-system 65001
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 192.168.0.2
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family inet6-vpn unicast
set protocols bgp group ibgp neighbor 192.168.0.4
set protocols mpls label-switched-path pe2 to 192.168.0.4
set protocols mpls label-switched-path pe2 no-cspf
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface et-0/0/2.0 passive
set protocols rsvp interface all
set protocols rsvp interface fxp0 disable
```

R3 (P Router) DUT 2

```
set interfaces et-0/0/0 description "R3-R2 P-PE"
set interfaces et-0/0/0 unit 0 filter input ce1-ce2
set interfaces et-0/0/0 unit 0 filter output ce2-ce1
set interfaces et-0/0/0 unit 0 family inet address 10.0.23.2/24
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:23::2/64
set interfaces et-0/0/0 unit 0 family mpls
set interfaces et-0/0/1 unit 0 family inet address 10.0.34.1/24
set interfaces et-0/0/1 unit 0 family inet6 address 2001:db8:10:0:34::1/64
set interfaces et-0/0/1 unit 0 family mpls
set interfaces et-0/0/2 unit 0 family inet address 10.0.100.2/24
set interfaces fti0 unit 1 tunnel encapsulation gre source address 10.100.0.3
set interfaces fti0 unit 1 tunnel encapsulation gre destination address 10.0.200.1
set interfaces fti0 unit 1 family ccc
set interfaces lo0 unit 0 family inet address 192.168.0.3/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:192:168:0::3/128
set forwarding-options port-mirroring instance p-router-mirror input rate 1
set forwarding-options port-mirroring instance p-router-mirror input run-length 0
set forwarding-options port-mirroring instance p-router-mirror family any output interface fti0.1
set firewall family any filter ce1-ce2 term ce1-ce2-mirror from mpls-label 22
set firewall family any filter ce1-ce2 term ce1-ce2-mirror then count ce1-ce2-traffic
set firewall family any filter ce1-ce2 term ce1-ce2-mirror then port-mirror-instance p-router-
mirror
set firewall family any filter ce1-ce2 term else then accept
set firewall family any filter ce2-ce1 term ce2-ce1-mirror from mpls-label 16
set firewall family any filter ce2-ce1 term ce2-ce1-mirror then count ce2-ce1-traffic
set firewall family any filter ce2-ce1 term ce2-ce1-mirror then port-mirror-instance p-router-
mirror
set firewall family any filter ce2-ce1 term else then accept
set routing-options router-id 192.168.0.3
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols mpls interface et-0/0/2.0 disable
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface et-0/0/2.0 passive
set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface et-0/0/2.0 passive
set protocols rsvp interface all
set protocols rsvp interface fxp0 disable
```

R4 (PE2)

```
set interfaces et-0/0/0 unit 0 family inet address 10.0.34.2/24
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:34::2/64
set interfaces et-0/0/0 unit 0 family mpls
set interfaces et-0/0/1 unit 0 family inet address 10.0.45.1/24
set interfaces et-0/0/1 unit 0 family inet6 address 2001:db8:10:0:45::1/64
set interfaces et-0/0/2 unit 0 family inet address 10.0.200.2/24
set interfaces lo0 unit 0 family inet address 192.168.0.4/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:192:168:0::4/128
set policy-options policy-statement ospf-export term 1 from protocol bgp
set policy-options policy-statement ospf-export term 1 then accept
set routing-instances ce2 instance-type vrf
set routing-instances ce2 protocols ospf area 0.0.0.0 interface all
set routing-instances ce2 protocols ospf export ospf-export
set routing-instances ce2 protocols ospf3 area 0.0.0.0 interface all
set routing-instances ce2 protocols ospf3 export ospf-export
set routing-instances ce2 interface et-0/0/1.0
set routing-instances ce2 route-distinguisher 65001:2
set routing-instances ce2 vrf-target target:65001:100
set routing-options router-id 192.168.0.4
set routing-options autonomous-system 65001
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 192.168.0.4
set protocols bgp group ibgp family inet unicast
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp family inet6-vpn unicast
set protocols bgp group ibgp neighbor 192.168.0.2
set protocols mpls label-switched-path pe1 to 192.168.0.2
set protocols mpls label-switched-path pe1 no-cspf
set protocols mpls ipv6-tunneling
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface et-0/0/2.0 passive
set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface et-0/0/2.0 passive
set protocols rsvp interface all
```

R5 (CE2)

```
set interfaces et-0/0/0 unit 0 family inet address 10.0.45.2/24
set interfaces et-0/0/0 unit 0 family inet6 address 2001:db8:10:0:45::2/64
set interfaces lo0 unit 0 family inet address 172.16.2.1/32
set interfaces lo0 unit 0 family inet6 address 2001:db8:172:16:2::1/128
set routing-options router-id 172.16.2.1
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf3 area 0.0.0.0 interface all
set protocols ospf3 area 0.0.0.0 interface fxp0.0 disable
```

Caveats and Limitations

This section lists the caveats and limitations for remote port mirroring on PTX platforms.

1. If the output portion of the port mirroring instance configuration has to be changed, the existing output configuration should be deleted first and the change committed, before the new configuration is added.
2. A total of 15 mirror instances are supported. There is no commit error if the number of remote port mirror instances exceeds 15.
3. A given mirrored packet can be sent to only one remote analyzer.
4. The maximum packet length can be configured as multiple of 128 bytes. The exported packet will be 22 bytes less than the configured value.
5. Multiple output interfaces are not supported in a given mirroring instance. There is no commit error if multiple output interfaces are configured.
6. The sampling process is not GRES supported. There will be drops of mirrored traffic in the event of a GRES event or restart of the `mirrord` process.
7. Tunnel traffic that terminates on the local router cannot be mirrored in the egress direction.
8. You cannot use port mirroring with a filter that evokes a policer action in the egress direction.

9. Statistics related to mirrored packets must be verified through firewall counters or FTI interface statistics.

Configuring Next-Hop Groups to Use Multiple Interfaces to Forward Packets Used in Port Mirroring

Next-hop groups allow you to include multiple interfaces used to forward duplicate packets used in port mirroring.

To configure a next-hop group, include the `next-hop-group` statement at the `[edit forwarding-options]` hierarchy level:

```
[edit forwarding-options]
next-hop-group [ group-names ] {
    interface interface-name {
        next-hop [ addresses ];
    }
}
```

or

```
[edit forwarding-options port-mirroring family inet6 output]
next-hop-group group-name{
    group-type inet6;
    interface interface-name {
        next-hop ipv6-address;
    }
    next-hop-subgroup group-name{
        interface interface-name {
            next-hop ipv6-address;
        }
    }
}
```

You can specify one or more group names. To configure the interface that sends out sampled information, include the `interface` statement and specify an interface. To specify a next-hop address to send sampled information, include the `next-hop` statement and specify an IP address.

Next-hop groups have the following restrictions:

- Next-hop groups support up to 16 next-hop addresses.
- You can configure up to 30 next-hop groups.
- Each next-hop group must have at least two next-hop addresses.
- When a firewall filter with next-hop-group action is applied on an interface in egress, the redirected copy does not retain any packet headers added while forwarding the packet to that interface. For example, if a filter with action next-hop-group is applied in egress of a GRE interface, the redirected copies received on the next-hop-group member interfaces do not contain a GRE header.

Next-hop groups can be used for port mirroring.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
14.2	Starting with release 14.2, next-hop groups are supported for M Series and MX Series routers only.

RELATED DOCUMENTATION

Configuring Port Mirroring | 37

Defining a Port-Mirroring Firewall Filter

On routers containing an Internet Processor II application-specific integrated circuit (ASIC) you can send a copy of an IP version 4 (IPv4) or IP version 6 (IPv6) packet from the router to an external host address or a packet analyzer for analysis. This is known as *port mirroring*.

Port mirroring is different from traffic sampling. In traffic sampling, a sampling key based on the IPv4 header is sent to the Routing Engine. There, the key can be placed in a file, or cflowd packets based on the key can be sent to a cflowd server. In port mirroring, the entire packet is copied and sent out through a next-hop interface.

You can configure simultaneous use of sampling and port mirroring, and set an independent sampling rate and run-length for port-mirrored packets. However, if a packet is selected for both sampling and port mirroring, only one action can be performed and port mirroring takes precedence. For example, if you configure an interface to sample every packet input to the interface and a filter also selects the packet to be port mirrored to another interface, only the port mirroring would take effect. All other packets not matching the explicit filter port-mirroring criteria continue to be sampled when forwarded to their final destination.

Firewall filters provide a means of protecting your router from excessive traffic transiting the router to a network destination or destined for the Routing Engine. Firewall filters that control local packets can also protect your router from external incidents.

You can configure a *firewall filter* to do the following:

- Restrict traffic destined for the Routing Engine based on its source, protocol, and application.
- Limit the traffic rate of packets destined for the Routing Engine to protect against flood, or denial-of-service (DoS) attacks.
- Address special circumstances associated with fragmented packets destined for the Routing Engine. Because the device evaluates every packet against a firewall filter (including fragments), you must configure the filter to accommodate fragments that do not contain packet header information. Otherwise, the filter discards all but the first fragment of a fragmented packet.

For information about configuring firewall filters in general (including in a Layer 3 environment), see *Stateless Firewall Filter Overview* and *How Standard Firewall Filters Evaluate Packets* in the Routing Policies, Firewall Filters, and Traffic Policers User Guide.

{PR 1723959 START]

NOTE: This task shows the generic steps you use to define a firewall filter with a port-mirroring action—values are shown as variables; for example, *term-name*. See the topics listed at the end of the task for examples that show real values used in setting up a firewall filter for port mirroring.

{PR 1723959 PAUSE]

To define a firewall filter with a port-mirroring action:

1. Prepare traffic for port mirroring by including the `filter` statement at the `[edit firewall family (inet | inet6)]` hierarchy level.

```
filter filter-name;
```

This filter at the `[edit firewall family (inet | inet6)]` hierarchy level selects traffic to be port-mirrored:

```
filter filter-name {
    term term-name {
        then {
            port-mirror;
            accept;
        }
    }
}
```

2. Enable configuration of firewall filters.

```
[edit]
user@host# edit firewall family family
```

   The value of the *family* option can be `inet` or `inet6`.

3. Enable configuration of a firewall filter ***filter-name***.

```
[edit firewall family family]
user@host# edit filter filter-name
```

4. Enable configuration of a firewall filter term ***filter-term-name***.

```
[edit firewall family family filter filter-name]
user@host# edit term filter-term-name
```

   For more information about firewall filter terms, see *Guidelines for Configuring Firewall Filters* in the Junos OS Routing Policies, Firewall Filters and Traffic Policers User Guide for Routing Devices.

5. Specify the firewall filter match conditions based on the route source address to mirror a subset of the sampled packets.

   For information about configuring firewall filter match conditions, see *Firewall Filter Match Conditions Based on Numbers or Text Aliases*, *Firewall Filter Match Conditions Based on Bit-Field Values*, *Firewall Filter Match Conditions Based on Address Fields*, and *Firewall Filter Match Conditions Based on Address Classes*in the Junos OS Routing Policies, Firewall Filters and Traffic Policers User Guide for Routing Devices.

6. Enable configuration of the ***action*** and ***action-modifier*** to apply to the matching packets.

```
[edit firewall family family filter filter-name term filter-term-name]
user@host# edit then
```

7. Specify the actions to be taken on matching packets.

```
[edit firewall family family filter filter-name term filter-term-name then]
user@host# set action
```

   The recommended value for the ***action*** is **accept**. If you do not specify an action, or if you omit the `then` statement entirely, all packets that match the conditions in the `from` statement are accepted.

8. Specify port-mirror as the ***action-modifier***.

   When the filter action is `port-mirror`, the packet is copied to a local interface for local or remote monitoring.

```
[edit firewall family family filter filter-name term filter-term-name then]
user@host# set port-mirror
```

9. Verify the minimum configuration of the firewall filter.

```
[edit firewall ... ]
user@host# top
[edit]
user@host# show firewall

family (inet | inet6) { # Type of packets to mirror
    filter filter-name { # Firewall filter name
        term filter-term-name {
            from { # Do not specify match conditions based on route source address
            }
            then {
                port-mirror;
                accept;
            }
        }
    }
}
```

   {PR 1723959 RESTART]

The following topics provide examples of firewall filter configuration with real values:

- Example: Configuring Layer 2 Port Mirroring to Remote VLAN
- Example: Layer 2 Port Mirroring to Multiple Destinations

[PR 1723959 END]

RELATED DOCUMENTATION

Configuring Port Mirroring

Defining a Next-Hop Group on MX Series Routers for Port Mirroring

On routers containing an Internet Processor II application-specific integrated circuit (ASIC) you can send a copy of an IP version 4 (IPv4) or IP version 6 (IPv6) packet from the router to an external host address or a packet analyzer for analysis. This is known as *port mirroring*.

Port mirroring is different from traffic sampling. In traffic sampling, a sampling key based on the IPv4 header is sent to the Routing Engine. There, the key can be placed in a file, or cflowd packets based on the key can be sent to a cflowd server. In port mirroring, the entire packet is copied and sent out through a next-hop interface.

You can configure simultaneous use of sampling and port mirroring, and set an independent sampling rate and run-length for port-mirrored packets. However, if a packet is selected for both sampling and port mirroring, only one action can be performed, and port mirroring takes precedence. For example, if you configure an interface to sample every packet input to the interface and a filter also selects the packet to be port mirrored to another interface, only the port mirroring takes effect. All other packets not matching the explicit filter port-mirroring criteria continue to be sampled when forwarded to their final destination.

Next-hop groups allow you to include port mirroring on multiple interfaces.

On MX Series routers, you can mirror tunnel interface input traffic to multiple destinations. To this form of multipacket port mirroring, you specify two or more destinations in a next-hop group, define a firewall filter that references the next-hop group as the filter action, and then apply the filter to a logical tunnel interface `lt-`) or virtual tunnel interfaces (`vt-` on the MX Series router.

To define a next-hop group for a Layer 2 port-mirroring firewall filter action:

1. Enable the configuration of forwarding options.

```
[edit]
user@host set forwarding-options port-mirroring family (inet | inet6) output
```

2. Enable configuration of a next-hop-group for Layer 2 port mirroring.

```
[edit forwarding-options port-mirroring ... family (inet | inet6) output]
user@host# set next-hop-group next-hop-group-name
```

3. Specify the type of addresses to be used in the next-hop group configuration.

```
[edit forwarding-options port-mirroring ... family (inet | inet6) output next-hop-group next-
hop-group-name]
user@host# set group-type inet6
```

4. Specify the interfaces of the next-hop route.

```
[edit forwarding-options port-mirroring ... family (inet | inet6) output next-hop-group next-
hop-group-name]
user@host# set interface logical-interface-name-1
user@host# set interface logical-interface-name-2
```

   or

```
[edit forwarding-options port-mirroring ... family (inet | inet6) output next-hop-group next-
hop-group-name]
user@host# set interface interface-name next-hop next-hop-address
```

   The MX Series router supports up to 30 next-hop groups. Each next-hop group supports up to 16 next-hop addresses. Each next-hop group must specify at least two addresses. The *next-hop-address* can be an IPv4 or IPv6 address.

5. (Optional) Specify the next-hop subgroup.

```
[edit forwarding-options port-mirroring ... family (inet | inet6) output next-hop-group next-
hop-group-name]
user@host# set next-hop-subgroup subgroup-name interface interface-name next-hop next-hop-
address
```

6. Verify the configuration of the next-hop group.

```
[edit forwarding-options port-mirroring ... family (inet | inet6) output next-hop-group next-
hop-group-name]
user@host# top
[edit]
user@host# show forwarding-options

...
next-hop-group next-hop-group-name {
    group-type inet6;
    interface logical-interface-name-1;
    interface interface-name{
        next-hop next-hop-address;
    }
    next-hop-subgroup subgroup-name{
        interface interface-name{
            next-hop next-hop-address;
        }
    }
}
...
```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
14.2	Starting with release 14.2, on routers containing an Internet Processor II application-specific integrated circuit (ASIC) you can send a copy of an IP version 4 (IPv4) or IP version 6 (IPv6) packet from the router to an external host address or a packet analyzer for analysis.

4

CHAPTER

Configuring Forwarding Table Filters to Efficiently Route Traffic

IN THIS CHAPTER

Configuring Forwarding Table Filters

Forwarding table filters are defined the same as other firewall filters, but you apply them differently. Instead of applying forwarding table filters to interfaces, you apply them to forwarding tables, each of which is associated with a routing instance and a virtual private network (VPN).

All packets are subjected to the input forwarding table filter that applies to the forwarding table. A forwarding table filter controls which packets the router accepts and then performs a lookup for the forwarding table, thereby controlling which packets the router forwards on the interfaces.

When the router receives a packet, it determines the best route to the ultimate destination by looking in a forwarding table, which is associated with the VPN on which the packet is to be sent. The router then forwards the packet toward its destination through the appropriate interface.

NOTE: For transit packets exiting the router through the tunnel, forwarding table filtering is not supported on the interfaces you configure as the output interface for tunnel traffic.

A forwarding table filter allows you to filter data packets based on their components and to perform an action on packets that match the filter; it essentially controls which bearer packets the router accepts and forwards. To configure a forwarding table filter, include the `firewall` statement at the `[edit]` hierarchy level:

```
[edit]
firewall {
    family family-name {
        filter filter-name {
            term term-name {
                from {
                    match-conditions;
                }
                then {
                    action;
                    action-modifiers;
                }
            }
        }
    }
}
```

family-name is the family address type: IPv4 (**inet**), IPv6 (**inet6**), Layer 2 traffic (**bridge**), or MPLS (**mpls**).

term-name is a named structure in which match conditions and actions are defined.

match-conditions are the criteria against which a bearer packet is compared; for example, the IP address of a source device or a destination device. You can specify multiple criteria in a match condition.

action specifies what happens if a packet matches all criteria; for example, the gateway GPRS support node (GGSN) accepting the bearer packet, performing a lookup in the forwarding table, and forwarding the packet to its destination; discarding the packet; and discarding the packet and returning a rejection message.

action-modifiers are actions that are taken in addition to the GGSN accepting or discarding a packet when all criteria match; for example, counting the packets and logging a packet.

To create a forwarding table, include the `instance-type` statement with the **forwarding** option at the `[edit routing-instances` *`instance-name`*`]` hierarchy level:

```
[edit]
routing-instances instance-name {
    instance-type forwarding;
}
```

To apply a forwarding table filter to a VPN routing and forwarding (VRF) table, include the **filter** and `input` statements at the `[edit routing-instance` *`instance-name`* `forwarding-options family` *`family-name`*`]` hierarchy level:

```
[edit routing-instances instance-name]
instance-type forwarding;
forwarding-options {
    family family-name {
        filter {
            input filter-name;
        }
    }
}
```

To apply a forwarding table filter to a forwarding table, include the **filter** and `input` statements at the `[edit forwarding-options family` *`family-name`*`]` hierarchy level:

```
[edit forwarding-options family family-name]
filter {
    input filter-name;
}
```

To apply a forwarding table filter to the default forwarding table **inet.0**, which is not associated with a specific routing instance, include the **filter** and `input` statements at the `[edit forwarding-options family inet]` hierarchy level:

```
[edit forwarding-options family inet]
filter {
    input filter-name;
}
```

RELATED DOCUMENTATION

Guidelines for Configuring Firewall Filters

Guidelines for Applying Standard Firewall Filters

Applying Forwarding Table Filters | 107

Forwarding Table Filters for Routing Instances on ACX Series Routers

Forwarding table filter is a mechanism by which all the packets forwarded by a certain forwarding table are subjected to filtering and if a packet matches the filter condition, the configured action is applied on the packet. You can use the forwarding table filter mechanism to apply a filter on all interfaces associated with a single routing instance with a simple configuration. You can apply a forwarding table filter to a routing instance of type forwarding and also to the default routing instance `inet.0`. To configure a forwarding table filter, include the `filter *filter-name*` statement at the `[edit firewall family <inet | inet6>]` hierarchy level.

The following limitations apply to forwarding table filters configured on routing instances:

- You cannot attach the same filter to more than one routing instance.
- You cannot attach the same filter at both the `[edit interfaces *interface-name* family <inet | inet6> filter input *filter-name*]` and `[edit routing-instances *instance-name* forwarding-options family <inet | inet6> filter input *filter-name*]` hierarchy level.

- You cannot attach a filter that is either interface-specific or a physical interface filter.
- You cannot attach a filter to the egress direction of routing instances.

RELATED DOCUMENTATION

Configuring Forwarding Table Filters

Applying Forwarding Table Filters

A forwarding table filter allows you to filter data packets based on their components and perform an action on packets that match the filter. You can apply a filter on the ingress or egress packets of a forwarding table. You configure the filter at the `[edit firewall family family-name]` hierarchy level; for more information, see *Configuring Forwarding Table Filters*.

To apply a forwarding table filter on ingress packets of a forwarding table, include the **filter** and `input` statements at the `[edit forwarding-options family family-name]` hierarchy level:

```
[edit forwarding-options family family-name]
filter {
    input filter-name;
}
```

You can filter based upon destination-class information by applying a firewall filter on the egress packets of the forwarding table. By applying firewall filters to packets that have been forwarded by a routing table, you can match based on certain parameters that are decided by the route lookup. For example, routes can be classified into specific destination and source classes. Firewall filters used for policing and mirroring are able to match based upon these classes.

To apply a firewall filter on egress packets of a forwarding table, include the **filter** and `output` statements at the `[edit forwarding-options family family-name]` hierarchy level:

```
[edit forwarding-options family family-name]
filter {
    output filter-name;
}
```

NOTE: You cannot have a firewall filter that includes an `interface-group` match condition if you are also using an egress forwarding table filter. This is because the `interface-group` match condition uses the logical interface on which the packet was received to match the interface group (or set of interface groups), while the forwarding table filters apply only to local host traffic and transit packets.

To apply a forwarding table filter to a flood table, include the **flood** and `input` statements at the `[edit forwarding-options family` *`family-name`*`]` hierarchy level as shown below. The `flood` statement is valid for the **vpls** protocol family only.

```
[edit forwarding-options family vpls]
flood {
    input filter-name;
}
```

On the MX Series router only, to apply a forwarding table filter for a virtual switch, include the **filter** and `input` statements at the `[edit routing-instances` *`routing-instance-name`* `bridge-domains` *`bridge-domain-name`* `forwarding-options]` hierarchy level:

```
[edit routing-instances routing-instance-name bridge-domains bridge-domain-name forwarding-
options]
    filter {
        input filter-name;
    }
```

For more information about how to configure a virtual switch, see the Junos OS Layer 2 Switching and Bridging Library for Routing Devices.

On MX Series 3D Universal Edge Routers, you can apply a forwarding table filter by using the `soure-checking` statement at the `[edit forwarding-options family inet6]` hierarchy level:

```
[edit forwarding-options family inet6]
    family inet6 {
        source-checking;
    }
}
```

This discards IPv6 packets when the source address type is unspecified, loopback, multicast or link-local.

RFC 4291, *IP Version 6 Addressing Architecture*, refers to four address types that require special treatment when they are used as source addresses. The four address types are:

- Unspecified
- Loopack
- Multicast
- Link-Local Unicast

The loopback and multicast addresses must never be used as a source address in IPv6 packets. The unspecified and link-local addresses can be used as source addresses but routers must never forward packets that have these addresses as source addresses. Typically, packets that contain unspecified or link-local addresses as source addresses are delivered to the local host. If the destination is not the local host, then the packet must not be forwarded. Configuring this statement filters or discards IPv6 packets of these four address types.

NOTE: The egress forwarding table filter is applied on the ingress interface of the FPC. If different packets to the same destination arrive on different FPCs, they might encounter different policers.

NOTE: You cannot configure this output statement for VPLS. You can continue to configure ingress forwarding table filters with the input statement at the `[edit forwarding-options family vpls filter]` hierarchy level.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
14.2	In versions 14.2 and prior, the egress forwarding table filter is not supported for the J Series Service Routers.
8.4	In Junos OS Release 8.4 and later, you can no longer configure this output statement for VPLS.

5

CHAPTER

Configuring Forwarding Options for Load Balancing Traffic

IN THIS CHAPTER

Configuring Load Balancing for Ethernet Pseudowires

You can configure load balancing for IPv4 traffic over Layer 2 Ethernet pseudowires. You can also configure load balancing for Ethernet pseudowires based on IP information. The option to include IP information in the hash key provides support for Ethernet circuit cross-connect (CCC) connections.

NOTE: This feature is supported only on M120, M320, MX Series, and T Series routers.

To configure load balancing for IPv4 traffic over Layer 2 Ethernet pseudowires, include the `ether-pseudowire` statement at the `[edit forwarding-options hash-key family mpls payload]` hierarchy level:

```
[edit forwarding-options]
hash-key {
    family mpls {
        (label-1 | no-labels);
        payload {
            ether-pseudowire;
        }
    }
}
```

NOTE: You must also configure either the `label-1` or the `no-labels` statement at the `[edit forwarding-options hash-key family mpls]` hierarchy level.

You can also configure load balancing for Ethernet pseudowires based on IP information. This functionality provides support for load balancing for Ethernet cross-circuit connect (CCC) connections. To include IP information in the hash key, include the `ip` statement at the `[edit forwarding-options hash-key family mpls payload]` hierarchy level:

```
[edit forwarding-options]
hash-key {
    family mpls {
        (label-1 | no-labels);
        payload {
            ip;
        }
    }
}
```

NOTE: You must also configure either the `label-1` or `no-labels` statement at the `[edit forwarding-options hash-key family mpls]` hierarchy level.

You can configure load balancing for IPv4 traffic over Ethernet pseudowires to include only Layer 3 IP information in the hash key. To include only Layer 3 IP information, include the `layer-3-only` option at the `[edit forwarding-options family mpls hash-key payload ip]` hierarchy level:

```
[edit forwarding-options]
hash-key {
    family mpls {
        (label-1 | no-labels);
        payload {
            ip {
                layer-3-only;
            }
        }
    }
}
```

NOTE: You must also configure either the `label-1` or `no-labels` statement at the `[edit forwarding-options hash-key family mpls]` hierarchy level.

Configuring Load-Balance Groups

In addition to including policers in firewall filters, you can configure a load-balance group that is not part of a firewall filter configuration. A load-balance group contains interfaces that all use the same next-hop group characteristic to load-balance the traffic.

To configure a load-balance group, include the `load-balance-group` statement at the `[edit firewall]` hierarchy level:

```
[edit firewall]
load-balance-group group-name {
    next-hop-group[ group-names ];
}
```

Next-hop groups allow you to include multiple interfaces used to forward duplicate packets used in port mirroring. For more information about next-hop groups, see "Configuring Next-Hop Groups to Use Multiple Interfaces to Forward Packets Used in Port Mirroring" on page 95.

Understanding the Algorithm Used to Load Balance Traffic on MX Series Routers

IN THIS SECTION

When a packet is received on the ingress interface of a device, the packet forwarding engine (PFE) performs a look up to identify the forwarding next hop. If there are multiple equal-cost paths (ECMPs) to the same next-hop destination, the ingress PFE can be configured to distribute the flow between the next hops. Likewise, distribution of traffic may be required between the member links of an aggregated interface such as aggregated Ethernet. The selection of the actual forwarding next-hop is based on the hash computation result over select packet header fields and several internal fields such as **interface index**. You can configure some of the fields that are used by the hashing algorithm.

- For MX series routers with Modular Port Concentrators (MPCs) and Type 5 FPCs, configure the hash for the supported traffic types at the `forwarding-options enhanced-hash-key` hierarchy level. Details on which fields are included by default for which traffic family can be found below.

  In Junos OS Release 18.3R1, the default method for calculating the enhanced-hash was changed to provide improved entropy for IP tunnels, IPv6 flows and PPPoE payloads transmitted as family multiservice. These defaults can be disabled by setting their respective no- commands.

- For MX series routers with DPCs, configure the hash for the supported traffic types at the `forwarding-options hash-key` hierarchy level.

Junos supports different types of load balancing.

- *Per-prefix load balancing* –Each prefix is mapped to only one forwarding next-hop.

- *Per-packet load balancing*–All next-hop addresses for a destination in the active route are installed in the forwarding table (the term *per-packet* load balancing in Junos is equivalent to what other vendors may call *per-flow* load balancing). See "Configuring Per-Packet Load Balancing" on page 130 for more information.

- *Random packet load balancing*–Next-hops are picked randomly for each packet. This method is available on MX routers with MPC line cards for Aggregated Ethernet interfaces and ECMP paths. To configure per-packet random spray load balancing, include the `per-packet` statement at the `[edit interfaces aex aggregated-ether-options load-balance]` hierarchy level. See *Example: Configuring Aggregated Ethernet Load Balancing* for more information.

- *Per-Packet Random Spray Load Balancing* –When the adaptive load-balancing option fails, per-packet random spray load balancing serves as a last resort. It ensures that the members of ECMP are equally loaded without taking bandwidth into consideration. Per packet causes packet reordering and hence is recommended only if the applications absorb reordering. Per-packet random spray eliminates traffic imbalance that occurs as a result of software errors, except for packet hash.

  Starting in Junos OS Release 20.2R1, you can configure per packet random load balancing on MX240, MX480, and MX960 routers with MPC10E (MPC10E-15C-MRATE and MPC10E-10C-MRATE) line card and MX2010 and MX2020 routers with MX2K-MPC11E line card.

- *Adaptive Load Balancing* - Adaptive Load Balancing (ALB) is a method that corrects a genuine traffic imbalance by using a feedback mechanism to distribute the traffic across the links in an aggregated Ethernet bundle and on equal-cost multipath (ECMP) next hops. ALB optimizes traffic distribution when packet flows have widely varying traffic rates. ALB uses a feedback mechanism to correct traffic load imbalance by adjusting the bandwidth and packet streams on links within an AE bundle.

- *ALB on multiple Packet Forwarding Engines for aggregated Ethernet bundles*

  Starting in Junos OS Release 20.1R1, on MX Series MPCs, on aggregate Ethernet Bundles ALB redistributes the traffic evenly across multiple ingress Packet Forwarding Engines (PFE) on the same line card. In earlier releases, ALB was limited to a single PFE while redistributing traffic in an AE bundle. This impacted flexibility and redundancy. ALB is disabled by default.

  You can configure ALB by setting the `adaptive` statement at the `[edit interfaces ae-interface aggregated-ether-options load-balance]` hierarchy level.

  See *Configuring Adaptive Load Balancing* for more information.

- *ALB on multiple PFEs for ECMP next hops*

  Starting in Junos OS Release 20.1R1, you can configure ALB for ECMP next hops across multiple ingress PFEs on the same line card for even distribution of the traffic and redundancy. In earlier releases, ALB for ECMP next hops was limited to a single PFE. This limitation impacted flexibility and redundancy. ALB dynamically monitors the traffic load contributed by each flow in relation to overall ECMP link loading levels, and then takes corrective action when the threshold is reached.

  You can configure ALB for ECMP next hops by configuring the `ecmp-alb` command under the `[edit chassis]` hierarchy level.

  See *ecmp-alb* for more information.

  NOTE: ALB will work for multiple PFEs residing on the same line card. This feature will not be supported for PFEs residing on different line cards.
  For PFEs residing on different line cards, ingress traffic can cause an uneven load on the egress ports, even if the ALB is enabled.

Several additional configuration options are also available:

- *Per-slot hash function configuration* –This method is based on a unique, load-balance hash value for each PIC slot and is only valid for M120, M320, and MX Series routers with DPCE and MS-DPC line cards.

- *Symmetrical load balancing* –This method provides symmetrical load balancing on an 802.3ad LAG. The hash used for symmetrical load balancing is set at the `interface` level of the hierarchy. It ensures that a given flow of duplex traffic traverses the same devices in both directions, and is available on MX Series routers.

  NOTE: On MX Series devices, starting in Junos OS Release 25.2R1, when symmetrical load balancing is enabled and ECMP uses multi-level load balancing over AE bundles, traffic across the AE referenced by the ECMP links might not be evenly distributed. First-level ECMP load balancing will still be distributed evenly.

MX MPC and T-Series Type 5 FPC Specifics

The hash computation algorithm on MX MPC and T Series Type 5 FPCs produces identical results for packets with swapped layer 3 addresses or layer 4 transport ports. For example, the hash computation result for a packet with source address 192.0.2.1 and destination address 203.0.113.1 is identical to the hash computation result for a packet with source address 203.0.113.1 and destination address 192.0.2.1.

To avoid possible packet re-ordering, layer 4 transport protocol ports are never used in hash computation for fragmented IPv4 packets. This is true for the first fragment of the flow, identified by the `more fragment` bit in a header, and all subsequent fragments, identified by non-zero fragment offset. The first fragment and subsequent fragments are always forwarded over same next-hop.

Hashing Algorithm Used in Junos 18.3R1 and later

In most cases, including layer 3 and layer 4 field information in the hash calculation produces results that are good enough for equitable distribution for traffic. However, in cases such as IP-in-IP or GRE tunneling, layer 3 and layer 4 field information alone may not be enough to produce a hash with sufficient entropy for load balancing. For example, in a deployment where MX series routers transit GRE flows, the GRE encapsulation tunnels typically occur as a single flow with the same source and destination, and same GRE key. Fat flows can also markedly increase the imbalance in link utilization, as traffic volume over the tunnels increases. Another example is when MX PE routers are being used as VPLS PE devices in a subscriber edge deployment where the routers back-haul broadband subscriber traffic from the access devices to a central broadband network gateway (BNG). In such a case, only the subscriber MAC addresses and the BNG router MAC addresses are available for hashing. But with few BNG MACs and relatively few subscriber MACs, the typical layer 3 and layer 4 fields are not sufficient to create a hash for optimal load balancing.

Therefore, for MX series routers with Trio MPCs and running Junos OS Release 18.3R1 or later, the default `enhanced-hash-key` calculation has changed. A summary of the changes is listed here:

- For GRE packets, if the outer IP packet is not a fragmented packet (first fragment or any subsequent fragment), and the inner packet is IPv4 or IPv6, then the source and destination addresses from the inner packet are used in the hash computation in addition to the outer source and destination addresses. Layer 4 ports of the inner packet are also included if the protocol of the inner IP packet is TCP or UDP, and the inner IP packet is not a fragment (first fragment or any subsequent fragment).

Likewise, if the outer IP packet is not a fragment packet, and the inner packet is MPLS, then the top inner label is included in the hash computation.

- For PPPoE packets, if the inner packet is IPv4 or IPv6, then the source and destination addresses from the inner packet are included. Layer 4 ports are included if the protocol of the inner IP packet is TCP or UDP, and the inner IP packet is not a fragment. Inclusion of the PPPoE inner packet fields can be disabled by configuring the `no-payload` option at the `forwarding-options enhanced-hash-key family multiservice` hierarchy level.

- For IPv6, the IPv6 header flow label field is included in the hash computation. RFC 6437 describes the 20-bit flow label field in the IPv6 header. Set the `no-flow-label` option at the `forwarding-options enhanced-hash-key family inet6` hierarchy to disable the new default.

Hash fields used for GRE traffic sent over IPv4

The lists show the fields used in the hash calculation, for non-fragmented packets, in Junos 18.3R1 and later. By default, the field is used in the hash calculation unless otherwise noted. Also where noted, the IP and port fields used in the hash is symmetric, that is, swapping the fields does not change the hash result.

- IPv4, GRE
  - GRE Key
  - Source and destination address; symmetric
  - Protocol
  - DSCP (disabled)
  - Incoming Interface Index (disabled)
- IPv4 in IPv4, GRE
  - Payload (inner IPv4: source and destination ports, IP addresses); symmetric
  - GRE Key

    GRE Protocol = IPv4
  - Source and destination address; symmetric
  - Protocol
  - DSCP (disabled)

- Incoming Interface Index (disabled)

- IPv6 in IPv4, GRE
  - Payload (inner IPv6: source and destination ports, IP addresses); symmetric
  - GRE Key

    GRE Protocol = IPv6
  - Source and destination address; symmetric
  - Protocol
  - DSCP (disabled)
  - Incoming Interface Index (disabled)
- MPLS in IPv4, GRE
  - Payload (inner MPLS: top label)
  - GRE Key

    GRE Protocol = MPLS
  - Source and destination address; symmetric
  - Protocol
  - DSCP (disabled)
  - Incoming Interface Index (disabled)
- IPv4, L2TPv2 used in Junos 17.2 and later

  Inclusion of the L2TPv2 tunnel ID and session ID can be enabled by configuring the `forwarding-options enhanced-hash-key family inet l2tp-tunnel-session-identifier` option. Note that Juniper does not recommend enabling this option by default. This is because L2TP session identification is based on the destination UDP port match (1701), and this port may not be exclusively used for L2TP transport so the extraction of the tunnel and session ID fields from the packet may not always be accurate.
  - Session ID
  - Tunnel ID
  - Source and destination port
  - Source and destination address; symmetric
  - Protocol (UDP)

    - DSCP (disabled)
    - Incoming Interface Index (disabled)

Hash fields used for GRE traffic sent over IPv6

The list shows the fields used in the hash calculation for non-fragmented packets. By default, the field is used in the hash calculation unless otherwise noted. Also where noted, the IP and port fields used in the hash is symmetric, that is, swapping the fields does not change the hash result.

- IPv6, GRE
    - GRE Key
    - Source and destination address; symmetric
    - Next header
    - Flow label (Junos 18.3 and later)
    - Traffic class (disabled)
    - Incoming Interface Index (disabled)
- IPv4 in IPv6, GRE (Junos 18.3 and later)
    - Payload (inner IPv4: source and destination ports, IP addresses); symmetric
    - GRE Key

      GRE Protocol = IPv4
    - Source and destination address; symmetric
    - Next header
    - Flow label (Junos 18.3 and later)
    - Traffic class (disabled)
    - Incoming Interface Index (disabled)
- IPv6 in IPv6, GRE (Junos 18.3 and later)
    - Payload (inner IPv6: source and destination ports, IP addresses); symmetric
    - GRE Key

    GRE Protocol = IPv6

  - Source and destination address; symmetric
  - Next header
  - Flow label (Junos 18.3 and later)
  - Traffic class (disabled)
  - Incoming Interface Index (disabled)

- **MPLS in IPv6, GRE (Junos 18.3 and later)**
  - Payload (inner MPLS: top labels); symmetric
  - GRE Key

    GRE Protocol = MPLS

  - Source and destination address; symmetric
  - Next header
  - Flow label
  - Traffic class (disabled)
  - Incoming Interface Index (disabled)

Hash fields used for IPv4

The list shows the fields used in the hash calculation for non-fragmented packets, except where noted. By default, the field is used in the hash calculation unless otherwise noted. Also where noted, the IP and port fields hash is symmetric, that is, swapping the fields does not change the hash result.

- **IPv4, not TCP or UDP, or fragmented packets**
  - Source and destination address; symmetric
  - Protocol
  - DSCP (disabled)
  - Incoming Interface Index (disabled)
- **IPv4, TCP and UDP, non fragmented packets**

    - Source and destination port; symmetric
    - Source and destination address; symmetric
    - Protocol
    - DSCP (disabled)
    - Incoming Interface Index (disabled)

- **IPv4, PPTP**
    - 16 least significant bits of the GRE Key
    - Source and destination address; symmetric
    - Protocol
    - DSCP (disabled)
    - Incoming Interface Index (disabled)

- **IPv4, GTP, UDP traffic to destination port 2152**

    Inclusion of GPRS tunneling protocol (GTP) tunnel endpoint identifier (TEID) can be enabled at the `forwarding-options enhanced-hash-key family inet gtp-tunnel-endpoint-identifier` option. Note that Juniper does not recommend enabling this option by default. This is because GTP session identification is based on the destination UDP port match (2152), and this port may not be exclusively used for GTP transport, so the extraction of TEID field from the packet may not always be accurate.

    - GTP TEID (disabled)
    - Source and destination port
    - Source and destination address; symmetric
    - Protocol
    - DSCP (disabled)
    - Incoming Interface Index (disabled)

Hash fields used for IPv6

The list shows the fields used in the hash calculation for non-fragmented packets, except where noted. By default, the field is used in the hash calculation unless otherwise noted. Also where noted, the IP and port fields hash is symmetric, that is, swapping the fields does not change the hash result.

- **IPv6, non TCP and UDP packet, or TCP and UDP packet fragmented by the originator**
  - Source and destination address; symmetric
  - Next header
  - Flow label (Junos 18.3 and later)
  - Traffic class (disabled)
  - Incoming Interface Index (disabled)
- **IPv6, non fragmented TCP and UDP packet**
  - Source and destination port; symmetric
  - Source and destination address; symmetric
  - Next header
  - Flow label (Junos 18.3 and later)
  - Traffic class (disabled)
  - Incoming Interface Index (disabled)
- **IPv6, PPTP**
  - 16 least significant bits of the GRE Key
  - Source and destination address; symmetric
  - Next header
  - Flow label (Junos 18.3 and later)
  - Traffic class (disabled)
  - Incoming Interface Index (disabled)
- **IPv6, GTP**

  Inclusion of GPRS tunneling protocol (GTP) tunnel endpoint identifier (TEID) can be enabled at the `forwarding-options enhanced-hash-key family inet gtp-tunnel-endpoint-identifier` hierarchy level. Note that Juniper does not recommend enabling this option by default. This is because GTP session identification is based on the destination UDP port match (2152), and this port may not be exclusively used for GTP transport, so the extraction of TEID field from the packet may not always be accurate.

- GTP TEID (disabled by default; enable at the `forwarding-options enhanced-hash-key family inet gtp-tunnel-endpoint-identifier` hierarchy level.
- Source and destination port
- Source and destination address; symmetric
- Next header
- Flow label (Junos 18.3 and later)
- Traffic class (disabled)
- Incoming Interface Index (disabled)

Hash fields used for multiservice

Family multiservice hash configuration applies to packets entering into the router as `family ccc`, `vpls`, or `bridge`. The list shows the fields used in the hash calculation for non-fragmented packets. By default, the field is used in the hash calculation unless otherwise noted. Also where noted, the IP and port fields used in the hash is symmetric, that is, swapping the fields does not change the hash result.

- **Ethernet, non-IP or non-MPLS**

  If configured, payload information is extracted from untagged packets or packets with up to two VLAN tags.

  - Outer 802.1p (disabled)
  - Source and destination MAC; symmetric
  - Incoming Interface Index (disabled)
- **Ethernet, IPv4**
  - Payload (inner IPv4: source and destination ports, IP addresses); symmetric
  - Outer 802.1p (disabled)
  - Source and destination MAC; symmetric
  - Incoming Interface Index (disabled)
- **Ethernet, IPv6**
  - Payload (inner IPv6: source and destination ports, IP addresses); symmetric

  - Outer 802.1p (disabled)
  - Source and destination MAC; symmetric
  - Incoming Interface Index (disabled)
- **Ethernet, MPLS**
  - Payload (inner MPLS: top labels plus inner IPv4 and IPv6 fields); symmetric. See *Hash fields used for MPLS, Junos 18.3 and later*, below, for related information.
  - Outer 802.1p (disabled)
  - Source and destination MAC; symmetric
  - Incoming Interface Index (disabled)
- **IPv4 in PPPoE (data packet)**
  - Payload (inner IPv4: source and destination ports, IP addresses); symmetric
  - PPP protocol IPv4 version 0x1, type 0x1
  - Outer 802.1p (disabled)
  - Source and destination MAC; symmetric
  - Incoming Interface Index (disabled)
- **IPv6 in PPPoE (data packet)**
  - Payload (inner IPv6: source and destination ports, IP addresses); symmetric
  - PPP protocol IPv6 version 0x1, type 0x1
  - Outer 802.1p (disabled)
  - Source and destination MAC; symmetric
  - Incoming Interface Index (disabled)

Hash fields used for MPLS, Junos 18.3 and later

The list shows the fields used in the hash calculation for non-fragmented packets. By default, the field is used in the hash calculation unless otherwise noted. Also where noted, the IP and port fields used in the hash is symmetric, that is, swapping the fields does not change the hash result.

- **MPLS, Encapsulated IPv4 or IPv6**
    - Payload (inner IPv4: source and destination ports, IP addresses); symmetric
    - Payload (inner IPv6: source and destination ports, IP addresses, next header); symmetric
    - Label 1..16 (20 bits)
    - Outer Label EXP (disabled)
    - Incoming Interface Index (disabled)
- **MPLS, IPv4 or IPv6 in Ethernet pseudo-wire**
    - Payload (IPv4/IPv6 in Ethernet pseudo-wire)
    - Label 2..16 (20 bits)
    - Outer Label EXP (disabled)
    - Label 1 (20 bits)
    - Incoming Interface Index (disabled)
- **MPLS, MPLS in Ethernet pseudo-wire**
    - Payload (two top labels of MPLS label stack entry in Ethernet pseudo-wire)
    - Label 2..16 (20 bits)
    - Outer Label EXP (disabled)
    - Label 1 (20 bits)
    - Incoming Interface Index (disabled)
- **MPLS, entropy label**

  When an entropy label is detected, the payload field is not processed, and the indicator is not included into hash computation

    - Label 1..16 (20 bits)
    - Outer Label EXP (disabled)
    - Incoming Interface Index (disabled)

Hash fields used for MPLS from Junos 14.1 to Junos 18.3

The list shows the fields used in the hash calculation for non-fragmented packets. By default, the field is used in the hash calculation unless otherwise noted. Also where noted, the IP and port fields used in the hash is symmetric, that is, swapping the fields does not change the hash result.

- MPLS, Encapsulated IPv4 or IPv6
  - Payload (inner IPv4: source and destination ports, IP addresses); symmetric

    Payload (inner IPv6: source and destination ports, IP addresses, next header); symmetric
  - Label 2.8 (20 bits)

    Outer Label EXP (disabled)

    Label 1 (20 bits)
  - Incoming Interface Index (disabled)
- MPLS, IPv4 or IPv6 in Ethernet pseudo-wire
  - Payload (IPv4/IPv6 in Ethernet pseudo-wire)
  - Label 2.8 (20 bits)

    Outer Label EXP (disabled)

    Label 1 (20 bits)
  - Incoming Interface Index (disabled)
- MPLS, MPLS in Ethernet pseudo-wire
  - Payload (two top labels of MPLS label stack entry in Ethernet pseudo-wire)
  - Label 2..16 (20 bits)
  - Outer Label EXP (disabled)
  - Label 1 (20 bits)
  - Incoming Interface Index (disabled)
- MPLS, entropy label

  When an entropy label is detected, the payload field is not processed, and the indicator is not included into hash computation
  - Label 2.8 (20 bits)

Outer Label EXP (disabled)

Label 1 (20 bits)

- Incoming Interface Index (disabled)

List of Junos Updates for Hash Calculation and Load Balancing for MX series routers with MPCs

Table 5: List of updates for MX series routers

Junos Release	Change
18.3R1	Includes IPv6 flow label, inner GRE header, and inner PPPoE in default hash computation. Increases MPLS label stack depth to 16 labels.
17.2R1	Load balancing for L2TP encapsulated IPv4 and IPv6 packets.
16.1R1	Includes EoMPLS payload hash with control word. Introduces source-only and destination-only based hashing.
15.1R1	Provides targeted distribution of static interfaces across AE member links. Includes source, destination, and MAC of MPLS encapsulated PPPoE payload in the default hash computation.
14.2R3	Increases scaling of LAG and MC-LAG.
14.2R2	Provides aggregate Ethernet bundle with 10G, 40G and 100G links.
14.1R1	Decouples aeX interface creation from `ch agg eth dev`. Increases aggregate Ethernet interface name space. Provides adaptive load balancing for ECMP next hops.

13.3R1	Includes enhancements for adaptive, per-packet-random, and periodic-rebalance load balancing.
11.4R1	provides load sharing across ECMP next hops.

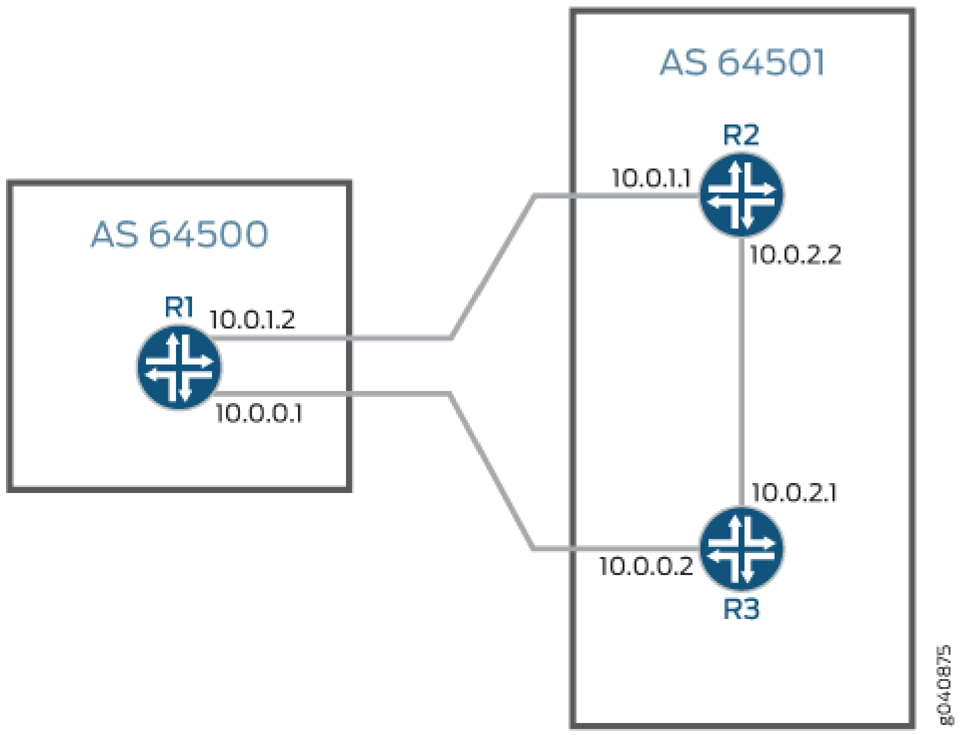
Understanding Per-Packet Load Balancing

By default, when there are multiple equal-cost paths to the same destination for the active route, Junos OS uses a hash algorithm to choose one of the next-hop addresses to install in the forwarding table. Whenever the set of next hops for a destination changes in any way, the next-hop address is re-chosen using the hash algorithm. Starting in Junos OS Release 18.3R1, for MX series routers, the default behavior for IPv6, GRE, and PPPoE packet hash computation was modified to include the flow-label field for improved load-balancing in certain cases (you can use the `no-payload` option to revert to the previous method for hash computation). See "Understanding the Algorithm Used to Load Balance Traffic on MX Series Routers" on page 113 for details.

You can configure Junos OS so that, for the active route, all next-hop addresses for a destination are installed in the forwarding table. This feature is called *per-packet load balancing*. The naming may be counter-intuitive. However, Junos *per-packet* load balancing is functionally equivalent to what other vendors may term *per-flow* load balancing. You can use load balancing to spread traffic across multiple paths between routers.

Figure 4 on page 129 shows a simple load balancing scenario. Device R1 is in AS 64500 and is connected to both Device R2 and Device R3, which are in AS 64501. Device R1 can be configured to load balance traffic across the two links.

Figure 4: Simple Load Balancing Scenario



Starting in Junos OS 13.3R3, for MX Series 5G Universal Routing Platforms with modular port concentrators (MPCs) only, you can configure consistent load balancing, which prevents the reordering of all flows to active paths in an equal-cost multipath (ECMP) group when one or more next-hop paths fail. Only flows for paths that are inactive are redirected to another active next-hop path. Flows mapped to servers that remain active are maintained. This feature applies only to external BGP peers.

Starting in Junos OS Release 19.1R1,on QFX10000 switches, you can configure load balancing of IPv4 or IPv6 packets by using GPRS Tunneling Protocol-tunnel endpoint identifier (GTP-TEID) field hash calculations. The GTP-TEID hashing is added to the Layer 2 and Layer 3 field hashing that you have already configured. To enable this feature on QFX10000 switches, configure the `gtp-tunnel-endpoint-identifier` statement at the `[edit forwarding-options enhanced-hash-key family inet]` or the `[edit forwarding-options enhanced-hash-key family inet6]` hierarchy Level. GTP versions 1 and 2 are supported; they support only user data. You must use UDP port number 2152 for both GTP versions.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
19.1R1	Starting in Junos OS Release 19.1R1,on QFX10000 switches, you can configure load balancing of IPv4 or IPv6 packets by using GPRS Tunneling Protocol-tunnel endpoint identifier (GTP-TEID) field hash calculations
18.3R1	Starting in Junos OS Release 18.3R1, for MX series routers, the default behavior for IPv6, GRE, and PPPoE packet hash computation was modified to include the flow-label field for improved load-balancing in certain cases (you can use the `no-payload` option to revert to the previous method for hash computation).
13.3R3	Starting in Junos OS 13.3R3, for MX Series 5G Universal Routing Platforms with modular port concentrators (MPCs) only, you can configure consistent load balancing, which prevents the reordering of all flows to active paths in an equal-cost multipath (ECMP) group when one or more next-hop paths fail.

RELATED DOCUMENTATION

Example: Load Balancing BGP Traffic

Configuring Per-Packet Load Balancing | **130**

Configuring Load Balancing Based on MPLS Labels

Configuring Load Balancing for Ethernet Pseudowires

Configuring Load Balancing Based on MAC Addresses

Configuring VPLS Load Balancing Based on IP and MPLS Information

Configuring VPLS Load Balancing on MX Series 5G Universal Routing Platforms

Configuring Consistent Load Balancing for ECMP Groups

Configuring Per-Packet Load Balancing

IN THIS SECTION

In Junos OS, you enable per-flow load balancing by setting the **load-balance per-packet** action in the routing policy configuration. The naming may be counter-intuitive, because in Junos, *per-packet* load balancing is functionally equivalent to what other vendors may term *per-flow* load balancing.

To configure per-packet load balancing, include the `load-balance per-packet` statement either as an option of the `route-filter` statement at the `[edit policy-options policy-statement policy-name term term-name from]` hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name from]
route-filter destination-prefix match-type {
    load-balance per-packet;
}
```

or at the `[edit policy-options policy-statement policy-name term term-name then]` hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name then]
load-balance per-packet;
```

To complete the configuration you must apply the routing policy to routes exported from the routing table to the forwarding table, by including the policy name in the list specified by the `export` statement:

```
export [ policy-names ];
```

You can include this statement at the following hierarchy levels:

- `[edit routing-options forwarding-table]`
- `[edit logical-systems logical-system-name routing-options forwarding-table]`

By default, Junos ignores port data when determining flows. To include port data in the flow determination, include the `family inet` statement at the `[edit forwarding-options hash-key]` hierarchy level:

```
[edit forwarding-options hash-key]
family inet {
    layer-3;
    layer-4;
}
```

If you include both the **layer 3** and **layer 4** statements, the device uses the following Layer 3 and Layer 4 information to load-balance:

- Source IP address
- Destination IP address
- Protocol
- Source port number
- Destination port number
- Incoming interface index
- IP type of service

When all of the **layer 3** and **layer 4** parameters are identical, the device sends packets in the flow through the same interface, which in turn helps prevent out of order delivery for TCP and UDP flows..

Internet Control Message Protocol (ICMP) packets are handled differently because the field location offset is the checksum field, which makes each ping packet a separate "flow." There are other protocols that can be encapsulated in IP that may have a varying value in the 32-bit offset. This may also be problematic because these protocols are seen as a separate flow.

By default, or if you include only the **layer 3** statement, the router uses the incoming interface index as well as the following Layer 3 information in the packet header to load balance traffic:

- Source IP address
- Destination IP address
- Protocol

By default, IP version 6 (IPv6) packets are automatically load-balanced based on the following Layer 3 and Layer 4 information:

- Source IP address
- Destination IP address
- Protocol
- Source port number
- Destination port number
- Incoming interface index
- Traffic class

Per-Packet Load Balancing Examples

Perform per-packet load balancing for all routes:

```
[edit]
policy-options {
    policy-statement load-balancing-policy {
        then {
            load-balance per-packet;
        }
    }
}
routing-options {
    forwarding-table {
        export load-balancing-policy;
    }
}
```

Perform per-packet load balancing only for a limited set of routes:

```
[edit]
policy-options {
    policy-statement load-balancing-polic {
        from {
            route-filter 192.168.10/24 orlonger;
            route-filter 10.114/16 orlonger;
        }
        then {
            load-balance per-packet;
        }
    }
}
routing-options {
    forwarding-table {
        export load-balancing-policy;
    }
}
```

To configure per-packet random spray load balancing, include the `load-balance random` statement at the `[edit policy-options policy-statement` *`policy-name`* `term` *`term-name`* `then]` hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name then]
load-balance random;
```

To complete the configuration you must apply the routing policy to routes exported from the routing table to the forwarding table, by including the policy name in the list specified by the `export` statement at the `[edit routing-options forwarding-table]` hierarchy level

```
[edit routing-options forwarding-table]
export [ policy-names ];
```

RELATED DOCUMENTATION

Understanding Per-Packet Load Balancing | 128

Configuring Per-Flow Load Balancing

IN THIS SECTION

In Junos OS, you enable per-flow load balancing by setting the **load-balance per-flow** option in the routing policy configuration. Note that the existing **load-balance per-packet** option is also supported on Junos OS and Junos OS Evolved software.

When there are multiple equal-cost paths to the same destination for the active route, Junos OS uses a hash algorithm by default to choose one of the next-hop addresses to install in the forwarding table. When you use per-flow load balancing, all next-hop addresses for a destination destination are installed to the forwarding table. The load is thus balanced to spread traffic across multiple paths between routes.

To configure per-flow load balancing, include the `load-balance per-packet` or `load-balance per-flow` statement either as an option of the `route-filter` statement at the `[edit policy-options policy-statement` *`policy-name`* `term` *`term-name`* `from]` hierarchy level:

NOTE: You can use the `load-balance per-packet` or `load-balance per-flow` option, and both have the same functionality.

```
[edit policy-options policy-statement policy-name term term-name from]
route-filter destination-prefix match-type {
    load-balance per-flow;
}
```

or at the `[edit policy-options policy-statement` *`policy-name`* `term` *`term-name`* `then]` hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name then]
load-balance per-flow;
```

To complete the configuration you must apply the routing policy to routes exported from the routing table to the forwarding table, by including the policy name in the list specified by the `export` statement:

```
export [ policy-names ];
```

You can include this statement at the following hierarchy levels:

- `[edit routing-options forwarding-table]`
- `[edit logical-systems` *`logical-system-name`* `routing-options forwarding-table]`

By default, Junos ignores port data when determining flows. To include port data in the flow determination, include the `family inet` statement at the `[edit forwarding-options hash-key]` hierarchy level:

```
[edit forwarding-options hash-key]
family inet {
    layer-3;
    layer-4;
}
```

If you include both the **layer 3** and **layer 4** statements, the device uses the following Layer 3 and Layer 4 information to load-balance:

- Source IP address
- Destination IP address
- Protocol
- Source port number
- Destination port number
- Incoming interface index
- IP type of service

When all of the **layer 3** and **layer 4** parameters are identical, the device sends packets in the flow through the same interface, which in turn helps prevent out of order delivery for TCP and UDP flows..

Internet Control Message Protocol (ICMP) packets are handled differently because the field location offset is the checksum field, which makes each ping packet a separate “flow.” There are other protocols that can be encapsulated in IP that may have a varying value in the 32-bit offset. This may also be problematic because these protocols are seen as a separate flow.

With M Series (with the exception of the M120 router) and T Series routers, the first fragment is mapped to the same load-balanced destination as the unfragmented packets. The other fragments can be mapped to other load-balanced destinations.

For the M120 router only, all fragments are mapped to the same load-balanced destination. This destination is not necessarily the same as that for unfragmented packets.

By default, or if you include only the **layer 3** statement, the router uses the incoming interface index as well as the following Layer 3 information in the packet header to load balance traffic:

NOTE: The per-flow option can be used instead of the per-packet option on these platforms:

- PTX-1000
- PTX-10002-60
- QFX-10001-I-20c
- QFX-10002-60c
- QFX-10003-160c
- QFX-10003-80c

  - Junos Evo PTX default configuration file
  - Junos PTX default configuration file

- Source IP address
- Destination IP address
- Protocol

By default, IP version 6 (IPv6) packets are automatically load-balanced based on the following Layer 3 and Layer 4 information:

- Source IP address
- Destination IP address
- Protocol
- Source port number
- Destination port number
- Incoming interface index
- Traffic class

Per-Packet Load Balancing Examples

Perform per-packet load balancing for all routes:

```
[edit]
policy-options {
    policy-statement load-balancing-policy {
        then {
            load-balance per-flow;
        }
    }
}
routing-options {
    forwarding-table {
        export load-balancing-policy;
    }
}
```

Perform per-packet load balancing only for a limited set of routes:

```
[edit]
policy-options {
    policy-statement load-balancing-polic {
        from {
            route-filter 192.168.10/24 orlonger;
            route-filter 10.114/16 orlonger;
        }
        then {
            load-balance per-flow;
        }
    }
}
routing-options {
    forwarding-table {
        export load-balancing-policy;
    }
}
```

To configure per-packet random spray load balancing, include the `load-balance random` statement at the `[edit policy-options policy-statement` *`policy-name`* `term` *`term-name`* `then]` hierarchy level:

```
[edit policy-options policy-statement policy-name term term-name then]
load-balance random;
```

To complete the configuration you must apply the routing policy to routes exported from the routing table to the forwarding table, by including the policy name in the list specified by the `export` statement at the `[edit routing-options forwarding-table]` hierarchy level

```
[edit routing-options forwarding-table]
export [ policy-names ];
```

RELATED DOCUMENTATION

Configuring Per-Packet Load Balancing

Understanding Load Balancing for BGP Traffic with Unequal Bandwidth Allocated to the Paths

The multipath option removes the tiebreakers from the active route decision process, thereby allowing otherwise equal cost BGP routes learned from multiple sources to be installed into the forwarding table. However, when the available paths are not equal cost, you may wish to load balance the traffic asymmetrically.

Once multiple next hops are installed in the forwarding table, a specific forwarding next hop is selected by the Junos OS per-prefix load-balancing algorithm. This process hashes against a packet's source and destination addresses to deterministically map the prefix pairing onto one of the available next hops. Per-prefix mapping works best when the hash function is presented with a large number of prefixes, such as might occur on an Internet peering exchange, and it serves to prevent packet reordering among pairs of communicating nodes.

An enterprise network normally wants to alter the default behavior to evoke a *per-packet* load-balancing algorithm. Per-packet is emphasized here because its use is a misnomer that stems from the historic behavior of the original Internet Processor ASIC. In reality, current Juniper Networks routers support per-prefix (default) and per-flow load balancing. The latter involves hashing against various Layer 3 and Layer 4 headers, including portions of the source address, destination address, transport protocol, incoming interface, and application ports. The effect is that now individual flows are hashed to a specific next hop, resulting in a more even distribution across available next hops, especially when routing between fewer source and destination pairs.

With per-packet load balancing, packets comprising a communication stream between two endpoints might be resequenced, but packets within individual flows maintain correct sequencing. Whether you opt for per-prefix or per-packet load balancing, asymmetry of access links can present a technical challenge. Either way, the prefixes or flows that are mapped to, for example, a T1 link will exhibit degraded performance when compared to those flows that map to, for example, a Fast Ethernet access link. Worse yet, with heavy traffic loads, any attempt at equal load balancing is likely to result in total saturation of the T1 link and session disruption stemming from packet loss.

Fortunately, the Juniper Networks BGP implementation supports the notion of a bandwidth community. This extended community encodes the bandwidth of a given next hop, and when combined with multipath, the load-balancing algorithm distributes flows across the set of next hops proportional to their relative bandwidths. Put another way, if you have a 10-Mbps and a 1-Mbps next hop, on average nine flows will map to the high-speed next hop for every one that uses the low speed.

Use of BGP bandwidth community is supported only with per-packet load balancing.

The configuration task has two parts:

- Configure the external BGP (EBGP) peering sessions, enable multipath, and define an import policy to tag routes with a bandwidth community that reflects link speed.
- Enable per-packet (really per-flow) load balancing for optimal distribution of traffic.

RELATED DOCUMENTATION

Understanding Per-Packet Load Balancing | 128

Understanding the Default BGP Routing Policy on Packet Transport Routers (PTX Series)

On PTX Series Packet Transport routers, the default BGP routing policy differs from that of other Junos OS routing devices. The default routing policy of the PTX Series 3000 and 5000 Series routers will not install BGP routes in the forwarding table, unless you configure another policy to override it. All other PTX Series routers will install BGP learned routes to the forwarding information base (FIB) and packet forwading engine (PFE) without the need for a policy.

The PTX Series routers are MPLS transit platforms that do IP forwarding, typically using interior gateway protocol (IGP) routes. The PTX Series Packet Forwarding Engine can accommodate a relatively small number of variable-length prefixes.

NOTE: A PTX Series router can support full BGP routes in the control plane so that it can be used as a route reflector (RR). It can do exact-length lookup multicast forwarding and can build the multicast forwarding plane for use by the unicast control plane (for example. to perform a reverse-path forwarding lookup for multicast).

Given the PFE limitation, the default routing policy for PTX Series routers is for BGP routes not to be installed in the forwarding table. You can override the default routing policy and select certain BGP routes to install in the forwarding table.

The default behavior for load balancing and BGP routes on PTX Series routers is as follows. It has the following desirable characteristics:

- Allows you to override the default behavior without needing to alter the default policy directly
- Reduces the chance of accidental changes that nullify the defaults
- Sets no flow-control actions, such as accept and reject

The default routing policy on the PTX Series routers is as follows:

```
user@host# show policy-options | display inheritance defaults no-comments
policy-options {
    policy-statement junos-ptx-series-default {
        term t1 {
            from {
                protocol bgp;
                rib inet.0;
            }
            then no-install-to-fib;
        }
        term t2 {
            from {
                protocol bgp;
                rib inet6.0;
            }
            then no-install-to-fib;
        }
        term t3 {
            then load-balance per-packet;
        }
    }
}
routing-options {
    forwarding-table {
        default-export junos-ptx-series-default;
    }
}
user@host# show routing-options forwarding-table default-export | display inheritance defaults 
no-comments
default-export junos-ptx-series-default;
```

As shown here, the `junos-ptx-series-default` policy is defined in `[edit policy-options]`. The policy is applied in `[edit routing-options forwarding-table]`, using the `default-export` statement. You can view these default configurations by using the `| display inheritance` flag.

Also, you can use the `show policy` command to view the default policy.

```
user@host> show policy junos-ptx-series-default
Policy junos-ptx-series-default:
    Term t1:
        from proto BGP
         inet.0
        then install-to-fib no
    Term t2:
        from proto BGP
         inet6.0
        then install-to-fib no
    Term t3:
        then load-balance per-packet
```

CAUTION: We strongly recommend that you do not alter the `junos-ptx-series-default` routing policy directly.

Junos OS chains the `junos-ptx-series-default` policy and any user-configured export policy. Because the `junos-ptx-series-default` policy does not use flow-control actions, any export policy that you configure is executed (by way of the implicit next-policy action) for every route. Thus you can override any actions set by the `junos-ptx-series-default` policy. If you do not configure an export policy, the actions set by `junos-ptx-series-default` policy are the only actions.

You can use the policy action `install-to-fib` to override the `no-install-to-fib` action.

Similarly, you can set the `load-balance per-prefix` action to override the `load-balance per-packet` action.

RELATED DOCUMENTATION

Conditional Advertisement and Import Policy (Routing Table) with certain match conditions

ECMP Flow-Based Forwarding on ACX Series Routers

IN THIS SECTION

An equal-cost multipath (ECMP) set is formed when the routing table contains multiple next-hop addresses for the same destination with equal cost. (Routes of equal cost have the same preference and metric values.) If there is an ECMP set for the active route, Junos OS uses a hash algorithm to choose *one* of the next-hop addresses in the ECMP set to install in the forwarding table.

You can configure Junos OS so that multiple next-hop entries in an ECMP set are installed in the forwarding table. On ACX Series routers, per-flow load balancing can be performed to spread traffic across multiple paths between routing devices. ECMP flow-based forwarding is supported for IPv4, IPv6, and MPLS packets on aggregated Ethernet (ae) interfaces.

Load balancing is used to evenly distribute traffic when there are multiple equal-cost next hops over different interfaces or a single next hop over an aggregated interface. In order to enable hashing, configure the `hash-key` statement at the the `[edit forwarding-options]` hierarchy level.

If a next-hop address is no longer part of the ECMP set or if it is removed from the routing table because of a route change, a flow that uses the next hop is rerouted and the session is not affected. Rerouting of the flow also occurs if there is a configuration change that takes away the next-hop address or if an administrator takes down the next-hop interface without deleting it. If a next-hop address is removed from the routing table because the interface is deleted or the session is intentionally cleared, the session is killed without being rerouted.

To select which packet header data to use for per-flow load balancing, include the `hash-key` statement at the `[edit forwarding-options]` hierarchy level. To load-balance IPv4 traffic by using the port data into the hash key, include the `familyinet` statement at the `[edit forwarding-options hash-key]` hierarchy level. You can incorporate either the Layer 3 IP port data, or the Layer 4 TCP or UDP port data into the hash key. To load-balance based on MPLS label information, configure the `family mpls` statement at the `[edit forwarding-options hash-key]` hierarchy level.

Forwarding of MPLS traffic by using penultimate-hop popping (PHP) and label-switched routing (LSR) is not supported on ACX Series routers. For ECMP flow-based forwarding over pseudowires, MPLS flows are assigned to one of the ECMP routes by using the hashing algorithm based on user-to-network interface (UNI) index.

To configure ECMP flow-based forwarding on ACX Series routers, first define a load-balancing routing policy by including one or more policy-statement configuration statements at the `[edit policy-options]` hierarchy level, with the action `load-balance per-packet`. Then apply the routing policy to routes exported from the routing table to the forwarding table. To do this, include the forwarding-table and export configuration statements at the `[edit routing-options]` hierarchy level.

To view the details of the ECMP next hops and to obtain information for debugging any problem with the ECMP functionality, issue the `show route` or the `show route summary` command.

See hash-key (Forwarding Options) to understand the list of hash-key statements that are not supported on the ACX 7000 Series of routers.

Platform-Specific ECMP Load Balancing Behavior

Use Feature Explorer to confirm platform and release support for specific features.

Platform	Difference
ACX Series	ACX7000 Series of routers do not support ECMP load balancing for unknown unicast traffic.

RELATED DOCUMENTATION

Understanding Routing Policies

Summary of Routing Policy Actions

Per-Flow and Per-Prefix Load Balancing Overview

By default, when there are multiple equal-cost paths to the same destination, Junos OS chooses one of the next-hop addresses at random.

On all M Series Multiservice Edge Routers, MX Series 5G Universal Routing Platforms, and T Series Core Routers, you have the additional option of configuring per-prefix load balancing based on a specified hash value that enables the router to elect a next hop independently of the route chosen by other routers.

On the M120, M320, and MX Series routers only, you have the additional option of enabling per-flow load balancing based on a unique, load-balance hash value for each Packet Forwarding Engine slot.

RELATED DOCUMENTATION

Configuring Per-Prefix Load Balancing | **145**

Configuring Per-Flow Load Balancing Based on Hash Values | **146**

Configuring Per-Prefix Load Balancing

By default, Junos OS uses a hashing method based only on the destination address to elect a forwarding next hop when multiple equal-cost paths are available. As a result, when multiple routers or switches share the same set of forwarding next hops for a given destination, they can elect the same forwarding next hop.

You can enable router-specific or switch-specific load balancing by including a per-prefix hash value. However, this method applies only to indirect next hops. In other words, when we have a route with a protocol next hop that is not directly connected, it can be resolved over a set of equal-cost forwarding next hops. Only in this case, we use the hashing algorithm to elect a forwarding next hop. An example of this is routes learned from an IBGP neighbor. The protocol next hop for those routes might not be directly reachable and would be resolved through some IGP or static routes. The result could be a set of equal-cost forwarding next hops to reach that protocol next hop. Per-prefix load balancing thus leads to better utilization of the available links.

To configure per-prefix load balancing, include the `load-balance` statement at the `[edit forwarding-options]` hierarchy level:

```
[edit forwarding-options]
load-balance {
    indexed-load-balance;
    per-prefix {
        hash-seed number;
    }
}
```

To enable per-prefix load balancing, you must include the `hash-seed` *`number`* statement. The range that you can configure is 0 (the default) through 65,535. If no hash seed is configured, the elected forwarding next hop is the same as in previous releases.

If you notice an issue with the load-balance distribution, try including the `indexed-load-balance` statement at the `[edit forwarding-options load-balance]` hierarchy level. This statement causes the creation of a nexthop structure that is both a function of the hash, and a function of the low-order bits of the IP address.

For MPC line cards in MX routers, `indexed-load-balance` has been superseded by an internal hash-rotation mechanism to reduce polarization.

CAUTION: Including the `indexed-load-balance` statement causes an increase in memory usage on the device.

```
indexed-load-balance;
```

RELATED DOCUMENTATION

Understanding the Algorithm Used to Load Balance Traffic on MX Series Routers | 113

enhanced-hash-key

Configuring Per-Flow Load Balancing Based on Hash Values

By default, Junos OS uses a hashing method based only on the destination address to elect a forwarding next hop when multiple equal-cost paths are available. All Packet Forwarding Engine slots are assigned the same hash value by default.

You can enable router-specific or switch-specific load balancing by configuring the router or switch to assign a unique, load-balance hash value for each Packet Forwarding Engine slot.

To configure per-flow load balancing. include the `load-balance` statement at the `[edit forwarding-options]` hierarchy level:

```
[edit forwarding-options]
load-balance {
    indexed-load-balance;
    per-flow {
        hash-seed;
    }
}
```

To enable per-flow load balancing, you must include the `hash-seed` statement. Junos OS automatically chooses a value for the hashing algorithm. You cannot configure a specific value for the `hash-seed` statement when you enable per-flow load balancing.

Configuring Load Balancing Based on MAC Addresses

The hash key mechanism for load-balancing uses Layer 2 media access control (MAC) information such as frame source and destination address. To load-balance traffic based on Layer 2 MAC information, include the `family multiservice` statement at the `[edit forwarding-options hash-key]` hierarchy level:

```
family multiservice {
    destination-mac;
    source-mac;
}
```

To include the destination-address MAC information in the hash key, include the **destination-mac** option. To include the source-address MAC information in the hash key, include the **source-mac** option.

NOTE: Any packets that have the same source and destination address will be sent over the same path.

NOTE: You can configure per-packet load balancing to optimize VPLS traffic flows across multiple paths.

NOTE: Aggregated Ethernet member links will now use the physical MAC address as the source MAC address in 802.3ah OAM packets.

NOTE: ACX Series routers do not support VPLS.

Load Balancing VPLS Non-Unicast Traffic Across Member Links of an Aggregate Interface

By default, VPLS non-unicast (or BUM — broadcast, unknown, and multicast) traffic sent across aggregate Ethernet interfaces is sent across only one member link of the aggregate interface. You can configure each VPLS instance to load balance BUM traffic across all members of an aggregate interface. This is referred to as BUM hashing.

To enable BUM hashing for an VPLS instance, add `bum-hashing` to the routing instance at the `[edit routing-instances *instance-name* protocols vpls]` hierarchy level. For example:

```
[edit routing-instances]
instance-name {
    protocols {
        vpls {
            bum-hashing;
        }
    }
}
```

WARNING: Enabling or disabling BUM hashing on a VPLS routing instance causes the routing instance to be destroyed and re-created when the configuration change is committed.

You can also specify which forwarding class to use for forwarding BUM traffic. When CoS-based forwarding (CBF) is configured on a VPLS PE router, BUM traffic uses the default forwarding class to select the label-switched path (LSP). You can associate an LSP with the default forwarding class.

To associate an LSP with the default forwarding class, add the `forwarding-class-default` statement at the `[edit class-of-service forwarding-policy next-hop-map *next-hop-map-name*]` hierarchy level. For example:

```
[edit class-of-service forwarding-policy next-hop-map next-hop-map-name]
forwarding-class-default {
    lsp-next-hop value;
}
```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
14.1	Starting with Junos OS Release 14.1, you can configure each VPLS instance to load balance BUM traffic across all members of an aggregate interface.

14.1	Starting with Junos OS Release 14.1, you can associate an LSP with the default forwarding class.

RELATED DOCUMENTATION

bum-hashing

Example: Configuring Multicast Load Balancing over Aggregated Ethernet Links | 149

Example: Configuring Multicast Load Balancing over Aggregated Ethernet Links

IN THIS SECTION

This example shows how to configure point-to-multipoint LSPs to load balance across aggregated Ethernet links. The load balancing applies to all traffic types, including multicast. .

Requirements

Before you begin:

1. Configure the router interfaces.

2. Configure an interior gateway protocol or static routing. See the Junos OS Routing Protocols Library for Routing Devices.

Overview

IN THIS SECTION

This example shows a sample topology and configuration to perform the following tasks:

- Load balancing VPLS multicast traffic over link aggregation
- Load balancing point-to-multipoint multicast traffic over link aggregation
- Re-load balancing after a change in the next-hop topology

  Next-hop topology changes might include but are not limited to:

  - Layer 2 membership change in the link aggregation
  - Indirect next-hop change
  - Composite next-hop change

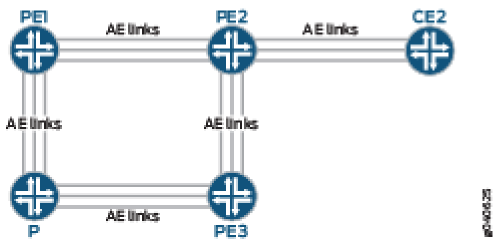
Load balancing is hash-based, so the higher the number of flows, the better. As is the case with unicast, you can also configure the hash key to be based on Layer 3 and Layer 4 information to achieve a better load-balancing result. There are a few exceptions that are specific to multicast traffic, which might lead to uneven load balancing—for example, when the outgoing interface list includes multiple aggregated interfaces with an unequal number of child links.

NOTE: For Draft Rosen multicast VPNs (MVPNs), load balancing over aggregated Ethernet interfaces is uneven when the LAGs are all core interfaces. In the case of Next-Generation MBGP MPVNs, multicast traffic is sent over point-to-multipoint and RSVP, and the hash is computed up to the IP headers. In the Draft Rosen case, multicast traffic is tunneled over GRE tunnels, and the hash is used only on GRE tunnel headers. This is why load balancing is not even for Draft Rosen when the LAGs are all core interfaces.

Topology

Figure 5 on page 151 shows the topology for this example. The example includes the configuration for Devices PE1 and PE2.

Figure 5: Multicast Load Balancing over Aggregated Ethernet Links



Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

Device PE1

```
[edit]
set forwarding-options hash-key family multiservice source-mac
set forwarding-options hash-key family multiservice destination-mac
set forwarding-options hash-key family multiservice payload ip layer-3
set interfaces ge-0/0/6 gigether-options 802.3ad ae0
set interfaces ge-0/1/6 gigether-options 802.3ad ae0
set interfaces ge-0/2/2 encapsulation ethernet-vpls
set interfaces ge-0/2/2 unit 0 family vpls
set interfaces ge-0/2/3 gigether-options 802.3ad ae0
set interfaces ge-0/2/6 gigether-options 802.3ad ae0
set interfaces ge-0/3/0 gigether-options 802.3ad ae0
set interfaces ge-0/3/1 gigether-options 802.3ad ae0
set interfaces ge-0/3/6 gigether-options 802.3ad ae0
set interfaces ge-1/0/6 gigether-options 802.3ad ae0
set interfaces ge-1/2/6 unit 0 family inet address 10.13.1.2/30
set interfaces ae0 unit 0 family inet address 10.11.11.1/30
set interfaces ae0 unit 0 family iso
set interfaces ae0 unit 0 family mpls
set policy-options policy-statement exp-to-fwd term a from community grn-com
set policy-options policy-statement exp-to-fwd term a then install-nexthop lsp PE1-to-PE2
set policy-options policy-statement exp-to-fwd term a then accept
set policy-options community grn-com members target:65000:1
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path PE1-to-PE2 to 10.255.19.77
set protocols mpls label-switched-path PE1-to-PE3 to 10.255.19.79
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group int type internal
set protocols bgp group int local-address 10.255.71.214
set protocols bgp group int family inet any
set protocols bgp group int family l2vpn signaling
set protocols bgp group int neighbor 10.255.19.77
set protocols bgp group int neighbor 10.255.19.79
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set routing-instances vpls instance-type vpls
set routing-instances vpls interface ge-0/2/2.0
set routing-instances vpls route-distinguisher 65000:1
set routing-instances vpls vrf-target target:65000:1
set routing-instances vpls protocols vpls site-range 3
set routing-instances vpls protocols vpls no-tunnel-services
set routing-instances vpls protocols vpls site asia site-identifier 1
set routing-instances vpls protocols vpls site asia interface ge-0/2/2.0
set routing-instances vpls protocols vpls vpls-id 100
set routing-instances vpls protocols vpls bum-hashing
```

Device PE2

```
set interfaces ge-0/0/7 gigether-options 802.3ad ae0
set interfaces ge-0/1/7 gigether-options 802.3ad ae0
set interfaces ge-0/2/3 gigether-options 802.3ad ae0
set interfaces ge-0/2/7 gigether-options 802.3ad ae0
set interfaces ge-2/0/0 gigether-options 802.3ad ae1
set interfaces ge-2/0/1 gigether-options 802.3ad ae1
set interfaces ge-2/0/2 gigether-options 802.3ad ae1
set interfaces ge-2/0/4 encapsulation ethernet-vpls
set interfaces ge-2/0/4 unit 0 family vpls
set interfaces ge-2/0/7 gigether-options 802.3ad ae0
set interfaces ge-2/0/9 unit 0 family inet address 10.10.1.1/30
set interfaces ge-2/0/9 unit 0 family mpls
set interfaces ge-2/1/7 gigether-options 802.3ad ae0
set interfaces ge-2/2/7 gigether-options 802.3ad ae0
set interfaces ge-2/3/7 gigether-options 802.3ad ae0
set interfaces ae0 unit 0 family inet address 10.11.11.2/30
set interfaces ae0 unit 0 family iso
set interfaces ae0 unit 0 family mpls
set interfaces ae1 unit 0 family inet address 10.1.1.1/30
set interfaces ae1 unit 0 family mpls
set protocols rsvp interface all
set protocols rsvp interface fxp0.0 disable
set protocols mpls label-switched-path PE2-to-PE3 from 10.255.19.77
set protocols mpls label-switched-path PE2-to-PE3 to 10.255.19.79
set protocols mpls label-switched-path PE2-to-PE1 to 10.255.71.214
set protocols mpls interface all
set protocols mpls interface fxp0.0 disable
set protocols bgp group int type internal
set protocols bgp group int local-address 10.255.19.77
set protocols bgp group int family inet any
set protocols bgp group int family l2vpn signaling
set protocols bgp group int neighbor 10.255.71.214
set protocols bgp group int neighbor 10.255.19.79
set protocols ospf traffic-engineering
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface ge-2/0/0.0
set protocols ospf area 0.0.0.0 interface ge-2/0/1.0
set protocols ospf area 0.0.0.0 interface ge-2/0/2.0
set protocols ospf area 0.0.0.0 interface ae0.0
set protocols ospf area 0.0.0.0 interface ae1.0
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set routing-instances vpls instance-type vpls
set routing-instances vpls interface ge-2/0/4.0
set routing-instances vpls route-distinguisher 65000:1
set routing-instances vpls vrf-target target:65000:1
set routing-instances vpls protocols vpls site-range 3
set routing-instances vpls protocols vpls no-tunnel-services
set routing-instances vpls protocols vpls site 2 site-identifier 2
set routing-instances vpls protocols vpls site 2 interface ge-2/0/4.0
set routing-instances vpls protocols vpls vpls-id 100
set routing-instances vpls protocols vpls bum-hashing
```

Step-by-Step Procedure

To configure Device PE1:

1. Configure Device PE1 interfaces.

```
[edit interfaces]
user@PE1# set ge-0/0/6 gigether-options 802.3ad ae0
user@PE1# set ge-0/1/6 gigether-options 802.3ad ae0
user@PE1# set ge-0/2/2 encapsulation ethernet-vpls
user@PE1# set ge-0/2/2 unit 0 family vpls
user@PE1# set ge-0/2/3 gigether-options 802.3ad ae0
user@PE1# set ge-0/2/6 gigether-options 802.3ad ae0
user@PE1# set ge-0/3/0 gigether-options 802.3ad ae0
user@PE1# set ge-0/3/1 gigether-options 802.3ad ae0
user@PE1# set ge-0/3/6 gigether-options 802.3ad ae0
user@PE1# set ge-1/0/6 gigether-options 802.3ad ae0
user@PE1# set ge-1/2/6 unit 0 family inet address 10.1.1.2/30
user@PE1# set ae0 unit 0 family inet address 10.11.11.1/30
user@PE1# set ae0 unit 0 family iso
user@PE1# set ae0 unit 0 family mpls
```

2. On Device PE1, configure the packet header data to be used for per-flow load balancing.

```
[edit forwarding-options hash-key family multiservice]
user@PE1# set source-mac
user@PE1# set destination-mac
user@PE1# set payload ip layer-3
```

3. Configure the routing policy on Device PE1.

```
[edit policy-options]
user@PE1# set policy-statement exp-to-fwd term a from community grn-com
user@PE1# set policy-statement exp-to-fwd term a then install-nexthop lsp PE1-to-PE2
user@PE1# set policy-statement exp-to-fwd term a then accept
user@PE1# set policy-options community grn-com members target:65000:1
```

4. Configure Device PE1 routing protocols and MPLS.

```
[edit protocols]
user@PE1# set rsvp interface all
user@PE1# set rsvp interface fxp0.0 disable
user@PE1# set mpls label-switched-path PE1-to-PE2 to 10.255.19.77
user@PE1# set mpls label-switched-path PE1-to-PE3 to 10.255.19.79
user@PE1# set mpls interface all
user@PE1# set mpls interface fxp0.0 disable
user@PE1# set bgp group int type internal
user@PE1# set bgp group int local-address 10.255.71.214
user@PE1# set bgp group int family inet any
user@PE1# set bgp group int family l2vpn signaling
user@PE1# set bgp group int neighbor 10.255.19.77
user@PE1# set bgp group int neighbor 10.255.19.79
user@PE1# set ospf traffic-engineering
user@PE1# set ospf area 0.0.0.0 interface all
user@PE1# set ospf area 0.0.0.0 interface fxp0.0 disable
```

5. Configure VPLS on Device PE1.

```
[edit routing-instances vpls]
user@PE1# set instance-type vpls
user@PE1# set interface ge-0/2/2.0
user@PE1# set route-distinguisher 65000:1
user@PE1# set vrf-target target:65000:1
user@PE1# set protocols vpls site-range 3
user@PE1# set protocols vpls no-tunnel-services
user@PE1# set protocols vpls site asia site-identifier 1
user@PE1# set protocols vpls site asia interface ge-0/2/2.0
```

```
user@PE1# set protocols vpls vpls-id 100
user@PE1# set protocols vpls bum-hashing
```

Step-by-Step Procedure

To configure Device PE2:

1. Configure Device PE2 interfaces.

   ```
   [edit interfaces]
   user@PE2# set ge-0/0/7 gigether-options 802.3ad ae0
   user@PE2# set ge-0/1/7 gigether-options 802.3ad ae0
   user@PE2# set ge-0/2/3 gigether-options 802.3ad ae0
   user@PE2# set ge-0/2/7 gigether-options 802.3ad ae0
   user@PE2# set ge-2/0/0 gigether-options 802.3ad ae1
   user@PE2# set ge-2/0/1 gigether-options 802.3ad ae1
   user@PE2# set ge-2/0/2 gigether-options 802.3ad ae1
   user@PE2# set ge-2/0/4 encapsulation ethernet-vpls
   user@PE2# set ge-2/0/4 unit 0 family vpls
   user@PE2# set ge-2/0/7 gigether-options 802.3ad ae0
   user@PE2# set ge-2/0/9 unit 0 family inet address 10.10.1.1/30
   user@PE2# set ge-2/0/9 unit 0 family mpls
   user@PE2# set ge-2/1/7 gigether-options 802.3ad ae0
   user@PE2# set ge-2/2/7 gigether-options 802.3ad ae0
   user@PE2# set ge-2/3/7 gigether-options 802.3ad ae0
   user@PE2# set ae0 unit 0 family inet address 10.11.11.2/30
   user@PE2# set ae0 unit 0 family iso
   user@PE2# set ae0 unit 0 family mpls
   user@PE2# set ae1 unit 0 family inet address 10.1.1.1/30
   user@PE2# set ae1 unit 0 family mpls
   ```

2. Configure Device PE2 routing protocols and MPLS.

   ```
   [edit protocols]
   user@PE2# set rsvp interface all
   user@PE2# set rsvp interface fxp0.0 disable
   user@PE2# set mpls label-switched-path PE2-to-PE3 from 10.255.19.77
   user@PE2# set mpls label-switched-path PE2-to-PE3 to 10.255.19.79
   user@PE2# set mpls label-switched-path PE2-to-PE1 to 10.255.71.214
   user@PE2# set mpls interface all
   user@PE2# set mpls interface fxp0.0 disable
   ```

```
user@PE2# set bgp group int type internal
user@PE2# set bgp group int local-address 10.255.19.77
user@PE2# set bgp group int family inet any
user@PE2# set bgp group int family l2vpn signaling
user@PE2# set bgp group int neighbor 10.255.71.214
user@PE2# set bgp group int neighbor 10.255.19.79
user@PE2# set ospf traffic-engineering
user@PE2# set ospf area 0.0.0.0 interface lo0.0
user@PE2# set ospf area 0.0.0.0 interface ge-2/0/0.0
user@PE2# set ospf area 0.0.0.0 interface ge-2/0/1.0
user@PE2# set ospf area 0.0.0.0 interface ge-2/0/2.0
user@PE2# set ospf area 0.0.0.0 interface ae0.0
user@PE2# set ospf area 0.0.0.0 interface ae1.0
user@PE2# set ospf area 0.0.0.0 interface all
user@PE2# set ospf area 0.0.0.0 interface fxp0.0 disable
user@PE2# set ldp interface all
user@PE2# set ldp interface fxp0.0 disable
```

3. Configure VPLS on Device PE2.

```
[edit routing-instances vpls]
user@PE2# set instance-type vpls
user@PE2# set interface ge-2/0/4.0
user@PE2# set route-distinguisher 65000:1
user@PE2# set vrf-target target:65000:1
user@PE2# set protocols vpls site-range 3
user@PE2# set protocols vpls no-tunnel-services
user@PE2# set protocols vpls site 2 site-identifier 2
user@PE2# set protocols vpls site 2 interface ge-2/0/4.0
user@PE2# set protocols vpls vpls-id 100
user@PE2# set protocols vpls bum-hashing
```

Results

From configuration mode, confirm your configuration by issuing the `show forwarding-options`, `show interfaces`, `show protocols`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

Device PE1

```
user@PE1# show forwarding-options
hash-key {
    family multiservice {
        source-mac;
        destination-mac;
        payload {
            ip {
                layer-3;
            }
        }
    }
}
```

```
user@PE1# show interfaces
ge-0/0/6 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/1/6 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/2/2 {
    encapsulation ethernet-vpls;
    unit 0 {
        family vpls;
    }
}
ge-0/2/3 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/2/6 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/3/0 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/3/1 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/3/6 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-1/0/6 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-1/2/6 {
    unit 0 {
        family inet {
            address 10.1.1.2/30;
        }
    }
}
ae0 {
    unit 0 {
        family inet {
            address 10.11.11.1/30;
        }
        family iso;
        family mpls;
    }
}
```

```
user@PE1# show protocols
mpls {
    interface all;
    interface fxp0.0 {
        disable;
    }
    label-switched-path PE1-to-PE2 {
        to 10.255.19.77;
    }
    label-switched-path PE1-to-PE3 {
        to 10.255.19.79;
    }
}
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    group int {
        type internal;
        local-address 10.255.71.214;
        family inet {
            any;
        }
        family l2vpn {
            signaling;
        }
        neighbor 10.255.19.77;
        neighbor 10.255.19.79;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
```

```
user@PE1# show policy-options
policy-statement exp-to-fwd {
    term a {
        from community grn-com;
        then {
            install-nexthop lsp PE1-to-PE2;
            accept;
        }
    }
}
community grn-com members target:65000:1;
```

```
user@PE1# show routing-instances
vpls {
    instance-type vpls;
    interface ge-0/2/2.0;
    route-distinguisher 65000:1;
    vrf-target target:65000:1;
    protocols {
        vpls {
            site-range 3;
            no-tunnel-services;
            site asia {
                site-identifier 1;
                interface ge-0/2/2.0;
            }
            vpls-id 100;
            bum-hashing;
        }
    }
}
```

Device PE2

```
user@PE2# show interfaces
ge-0/0/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/1/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/2/3 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-0/2/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-2/0/0 {
    gigether-options {
        802.3ad ae1;
    }
}
ge-2/0/1 {
    gigether-options {
        802.3ad ae1;
    }
}
ge-2/0/2 {
    gigether-options {
        802.3ad ae1;
    }
}
ge-2/0/4 {
    encapsulation ethernet-vpls;
    unit 0 {
        family vpls;
    }
}
ge-2/0/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-2/0/9 {
    unit 0 {
        family inet {
            address 10.10.1.1/30;
        }
        family mpls;
    }
}
ge-2/1/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-2/2/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-2/3/7 {
    gigether-options {
        802.3ad ae0;
    }
}
ae0 {
    unit 0 {
        family inet {
            address 10.11.11.2/30;
        }
        family iso;
        family mpls;
    }
}
ae1 {
    unit 0 {
        family inet {
            address 10.1.1.1/30;
        }
        family mpls;
    }
}
```

```
user@PE2# show protocols
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
mpls {
    label-switched-path PE2-to-PE3 {
        from 10.255.19.77;
        to 10.255.19.79;
    }
    label-switched-path PE2-to-PE1 {
        to 10.255.71.214;
    }
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    group int {
        type internal;
        local-address 10.255.19.77;
        family inet {
            any;
        }
        family l2vpn {
            signaling;
        }
        neighbor 10.255.71.214;
        neighbor 10.255.19.79;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface lo0.0;
        interface ge-2/0/0.0;
        interface ge-2/0/1.0;
        interface ge-2/0/2.0;
        interface ae0.0;
        interface ae1.0;
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

```
user@PE2# show routing-instances
vpls {
    instance-type vpls;
    interface ge-2/0/4.0;
    route-distinguisher 65000:1;
    vrf-target target:65000:1;
    protocols {
        vpls {
            site-range 3;
            no-tunnel-services;
            site 2 {
                site-identifier 2;
                interface ge-2/0/4.0;
            }
            vpls-id 100;
            bum-hashing;
        }
    }
}
```

Verification

You can monitor the operation of the routing instance by running the `show interfaces ae1.0 extensive` and `monitor interface traffic` commands.

For troubleshooting, you can configure tracing operations for all of the protocols.

RELATED DOCUMENTATION

Configuring Point-to-Multipoint LSPs for an MBGP MVPN

Understanding Wildcards to Configure Selective Point-to-Multipoint LSPs for an MBGP MVPN

show interfaces (Aggregated Ethernet)

Load Balancing VPLS Non-Unicast Traffic Across Member Links of an Aggregate Interface | **147**

6

CHAPTER

Configuring Other Forwarding Options

IN THIS CHAPTER

Configuring Routers, Switches, and Interfaces as DHCP and BOOTP Relay Agents

You can configure the router, switch, or interface to act as a Dynamic Host Configuration Protocol (DHCP) and Bootstrap Protocol (BOOTP) relay agent. This means that a locally attached host can issue a DHCP or BOOTP request as a broadcast message. If the router, switch, or interface sees this broadcast message, it relays the message to a specified DHCP or BOOTP server.

You should configure the router, switch, or interface to be a DHCP and BOOTP relay agent if you have locally attached hosts and a distant DHCP or BOOTP server. For MX Series routers connected via IRB, see the note below to prevent BOOTP reply packets from being dropped.

To configure the router or switch to act as a DHCP and BOOTP relay agent, include the `bootp` statement at the `[edit forwarding-options helpers]` hierarchy level:

```
[edit forwarding-options helpers]
bootp {
    client-response-ttl number;
    description text-description;
    interface (interface-name | interface-group) {
        client-response-ttl number;
        description text-description;
        maximum-hop-count number;
        minimum-wait-time seconds;
        no-listen;
        server address {
            logical-system logical-system-name <routing-instance [ <default> routing-instance-
names ]>;
            routing-instance [ <default> routing-instance-names ];
        }
    }
    maximum-hop-count number;
    minimum-wait-time seconds;
    relay-agent-option;
    server server-identifier {
        <logical-system logical-system-name>
<routing-instance [ routing-instance-names ]>;
    }
}
```

To set the description of the BOOTP service, DHCP service, or interface, include the `description` statement.

To set a logical interface or a group of logical interfaces with a specific DHCP relay or BOOTP configuration, include the `interface` statement.

To set the routing instance of the server to forward, include the `routing-instance` statement. You can include as many routing instances as necessary in the same statement.

To stop packets from being forwarded on a logical interface, a group of logical interfaces, or the router or switch, include the `no-listen` statement.

To set the maximum allowed number in the hops field of the BOOTP header, include the `maximum-hop-count` statement. Headers that have a larger number in the hops field are not forwarded. If you omit the `maximum-hop-count` statement, the default value is four hops.

To set the minimum allowed number of seconds in the **secs** field of the BOOTP header, include the `minimum-wait-time` statement. Headers that have a smaller number in the **secs** field are not forwarded. The default value for the minimum wait time is zero (0).

To set the IP address that specify the DHCP or BOOTP server for the router, switch, or interface, include the `server` statement. You can include multiple `server` statements.

To set an IP time-to-live (TTL) value for DHCP response packets sent to a DHCP client, include the `client-response-ttl` statement.

To use the DHCP relay agent option in relayed BOOTP/DHCP messages, include the `relay-agent-option` statement. This option is primarily useful for enabling DHCP forwarding between different VRF routing instances. This option is documented in RFC 3046, *DHCP Relay Agent Information Option*.

You can also configure an individual logical interface to be a DHCP and BOOTP relay agent if you have locally attached hosts and a remote DHCP or BOOTP server connected to one of the router's or switch's interfaces. For more information, see the Junos OS Administration Library for Routing Devices.

The following example demonstrates a BOOTP relay agent configuration.

```
user@host# show forwarding-options
helpers {
    bootp {
        description "dhcp relay agent global parameters";
        server 192.168.55.44;
        server 172.16.0.3 routing-instance c3;
        maximum-hop-count 10;
        minimum-wait-time 8;
        interface {
            fe-1/3/0 {
                description "use this info for this interface";
                server 10.10.10.10;
                server 192.168.14.14;
                maximum-hop-count 11;
                minimum-wait-time 3;
            }
            fe-1/3/1 {
                no-listen;  ###ignore DHCPDISCOVER messages on this interface
            }
            all {
                description "globals apply to all other interfaces";
            }
        }
    }
}
```

BEST PRACTICE: To use `bootp helper` on a MX Series router (MX80, MX240, MX480 and MX960) connected via IRB, you may need to take steps to ensure that DHCP discover packets (the bootp reply) are sent to clients and received as expected. Otherwise, **bootp** replies may be dropped because the DHCP client is clearing the broadcast bit in the discover packet, or because the DHCP server is stripping **option-82** flags from the **offer**.

This happens when the IRB interface is a layer 3 (logical) interface associated with a bridge domain that has multiple layer 2 (physical) interfaces associated with it. In such cases, if the **offer** from the DHCP server is unicast and doesn't include an ingress interface identifying the physical interface on which the **discovery** packet was received, the MX router won't be able to determine an interface for sending out **offers**.

1. Enable `broadcast` on the IRB interface to flood **discovery** frames from all physical interfaces in the bridge domain. For example,

   ```
   user@host# edit forwarding-options helpers bootp interface irb.o
       broadcast;
           server 202.67.4.1;
       }
   ```

   or,

2. Enable `relay-agent-option` on the bootp helper. For example,

```
user@host# edit forwarding-options helpers bootp
    relay-agent-option;
        server 202.67.4.1;
    }
```

3. Configure the IRB interface connected to the DHCP server so it echoes **option-82** flags back to the router. This will ensure that the **option-82** string, which identifies the interface used by the router, is preserved.

Configuring DNS and TFTP Packet Forwarding

IN THIS SECTION

You can configure the router or switch to support Domain Name System (DNS) and Trivial File Transfer Protocol (TFTP) packet forwarding for IPv4 traffic, which allows clients to send DNS or TFTP requests to the router or switch. The responding DNS or TFTP server recognizes the client address and sends a response directly to that address. By default, the router or switch ignores DNS and TFTP request packets.

To enable DNS or TFTP packet forwarding, include the `helpers` statement at the `[edit forwarding-options]` hierarchy level:

```
[edit forwarding-options]
helpers {
    domain {
        description text-description;
        interface interface-name {
            description text-description;
            no-listen;
            server [ addresses {
                logical-system logical-system-name;
                routing-instance instance-name;
            }
        }
    }
    tftp {
        description text-description;
        interface interface-name {
            description text-description;
            no-listen;
            server address;
            server logical-system name < [ routing-instance routing-instance-names ] >;
            server < [ routing-instance routing-instance-names ] >;
        }
    }
}
```

To set domain packet forwarding, include the `domain` statement.

To set the description of the DNS or TFTP service, include the `description` statement.

To set TFTP packet forwarding, include the `tftp` statement.

To set a DNS or TFTP server (with an IPv4 address), include the `server` statement. Use one address for either a global configuration or for each interface.

To set the routing instance of the server to forward, include the `routing-instance` statement. You can include as many routing instances as necessary in the same statement.

To disable recognition of DNS or TFTP requests on one or more interfaces, include the `no-listen` statement. If you do not specify at least one interface with this statement, the forwarding service is global to all interfaces on the router or switch.

The following sections discuss the following:

Tracing BOOTP, DNS, and TFTP Forwarding Operations

BOOTP, DNS, and TFTP forwarding tracing operations track all BOOTP, DNS, and TFTP operations and record them in a log file. The logged error descriptions provide detailed information to help you solve problems faster.

By default, nothing is traced. If you include the `traceoptions` statement at the `[edit forwarding-options helpers]` hierarchy level, the default tracing behavior is the following:

- Important events are logged in a file called **fud** located in the **/var/log** directory.

- When the file **fud** reaches 128 kilobytes (KB), it is renamed **fud.0**, then **fud.1**, and so on, until there are 3 trace files. Then the oldest trace file (**fud.2**) is overwritten. (For more information about how log files are created, see the System Log Explorer.)
- Log files can be accessed only by the user who configures the tracing operation.

You cannot change the directory (**/var/log**) in which trace files are located. However, you can customize the other trace file settings by including the following statements at the `[edit forwarding-options helpers]` hierarchy level:

```
[edit forwarding-options helpers]
traceoptions {
    file filename <files number> <match regular-expression> <size size> <world-readable | no-
world-readable>;
    flag {
        address;
        all;
        config;
        domain;
        ifdb;
        io;
        main;
        port;
        rtsock;
        tftp;
        trace;
        ui;
        util;
    }
    level severity-level;
    no-remote-trace;
}
```

These statements are described in the following sections:

Configuring the Log Filename

By default, the name of the file that records trace output is **fud**. You can specify a different name by including the `file filename` statement at the `[edit forwarding-options helpers traceoptions]` hierarchy level:

```
[edit forwarding-options helpers traceoptions]
file filename;
```

Configuring the Number and Size of Log Files

By default, when the trace file reaches 128 kilobytes (KB) in size, it is renamed ***filename*.0**, then ***filename*.1**, and so on, until there are three trace files. Then the oldest trace file (***filename*.2**) is overwritten.

You can configure the limits on the number and size of trace files by including the following statements at the `[edit forwarding-options helpers traceoptions]` hierarchy level:

```
[edit forwarding-options helpers traceoptions]
file files number size size;
```

For example, set the maximum file size to 2 MB, and the maximum number of files to 20. When the file that receives the output of the tracing operation (***filename***) reaches 2 MB, ***filename*** is renamed ***filename*.0**, and a new file called ***filename*** is created. When the new ***filename*** reaches 2 MB, ***filename*.0** is renamed ***filename*.1** and ***filename*** is renamed ***filename*.0**. This process repeats until there are 20 trace files. Then the oldest file (***filename*.19**) is overwritten by the newest file (***filename*.0**).

The number of files can be from 2 through 1000 files. The file size of each file can be from 10 KB through 1 gigabyte (GB).

Configuring Access to the Log File

By default, log files can be accessed only by the user who configures the tracing operation.

To specify that any user can read all log files, include the **world-readable** option with the `file` statement at the `[edit forwarding-options helpers traceoptions]` hierarchy level:

```
[edit forwarding-options helpers traceoptions]
file world-readable;
```

To explicitly set the default behavior, include the **no-world-readable** option with the `file` statement at the `[edit forwarding-options helpers traceoptions]` hierarchy level:

```
[edit forwarding-options helpers traceoptions]
file no-world-readable;
```

Configuring a Regular Expression for Lines to Be Logged

By default, the trace operation output includes all lines relevant to the logged events.

You can refine the output by including the **match** option with the `file` statement at the `[edit forwarding-options helpers traceoptions]` hierarchy level and specifying a regular expression (regex) to be matched:

```
[edit forwarding-options helpers traceoptions]
file filename match regular-expression;
```

Example: Configuring DNS Packet Forwarding

Enable DNS packet request forwarding to all interfaces on a router except **t1-1/1/2** and **t1-1/1/3**:

```
[edit forwarding-options helpers]
dns {
    server 10.10.10.30;
    interface {
        t1-1/1/2 {
            no-listen;
            server 10.10.10.9;
        }
        t1-1/1/3 {
            no-listen;
            server 10.10.10.4;
        }
    }
}
```

Configuring Port-based LAN Broadcast Packet Forwarding

You can enable a router or switch to forward LAN broadcast traffic on only custom UDP ports to specified servers by configuring *port helpers* with the `[edit forwarding-options helpers] port` configuration statement. Port helpers are also referred to as port forwarding or UDP broadcast packet forwarding services. When you configure a port helper, the router or switch listens for incoming UDP traffic for the configured port with destination Layer 2 MAC and Layer 3 IP broadcast addresses, and forwards the packets as unicast traffic to a configured server.

Port helpers forward the traffic for configured ports transparently, without considering the application layer protocols in the packets being forwarded. However, you cannot configure a port helper to forward traffic for standard ports used by services such as BOOTP, DNS and TFTP. These services have their own explicit packet forwarding helper configuration options (see *helpers* and "Configuring DNS and TFTP Packet Forwarding" on page 171).

You can configure port helpers to listen for and forward broadcast traffic for a configured port using any of the following scopes:

- Global scope—Forward incoming broadcast traffic on the port to a configured destination server IP address.

  Configure a global port helper using only the `server` configuration option, without specifying a particular interface. The port helper listens for incoming traffic on any interfaces to forward to the configured server. For example:

  ```
  set forwarding-options helpers port 1300 server 10.20.30.40
  ```

- VLAN-specific scope—Forward incoming broadcast traffic on the port from a configured VLAN to a configured destination server IP address.

  Configure a VLAN-specific port helper using the `interface` statement with an IRB interface name for a VLAN, and the `server` statement. The port helper listens for incoming traffic from interfaces in the VLAN to forward to the configured server. For example:

  ```
  set forwarding-options helpers port 1064 interface irb.100 server 192.0.2.50
  ```

- Interface-specific scope—Forward incoming broadcast traffic on the port from a configured Layer 3 interface to a configured destination server IP address.

Configure an interface-specific port helper using the `interface` statement with a Layer 3 interface name, and the `server` statement. The port helper listens for incoming traffic only from the configured interface to forward to the configured server. For example:

```
set forwarding-options helpers port 1064 interface ge-0/0/3 server 192.0.2.50
```

For any scope, optionally use the `description` statement to label or describe the configured forwarding service.

Previously, you can configure only one destination server for a given port number. Presently, you can configure forwarding traffic to multiple servers for a given port in any port helper scope. To configure forwarding the traffic on a specified port to multiple destination servers, include multiple configuration items for the port and each server (or interface and server). For example, in the global scope:

```
set forwarding-options helpers port 1300 server 10.20.30.4
set forwarding-options helpers port 1300 server 10.20.30.5
set forwarding-options helpers port 1300 server 10.20.30.6
```

To temporarily disable listening on a configured port from a configured interface, include the `no-listen` option with the configured item, as follows:

```
set forwarding-options helpers port port-number interface interface-name server address no-
listen
```

To remove a configured port helper service from a router or switch, delete the configured `port` number item, as follows:

```
delete forwarding-options helpers port <port-number>
```

If multiple servers are configured for a particular port, to remove any or all such forwarding services, you must delete each configured port and server item individually. For example:

```
delete forwarding-options helpers port 1300 server 10.20.30.4
delete forwarding-options helpers port 1300 server 10.20.30.5
delete forwarding-options helpers port 1300 server 10.20.30.6
```

RELATED DOCUMENTATION

port (Packet Forwarding)

server (DNS, Port, and TFTP Service)

interface (DNS, Port, and TFTP Packet Forwarding or Relay Agent)

Preventing DHCP Spoofing on MX Series 5G Universal Routing Platforms

A problem that sometimes occurs with DHCP is *DHCP spoofing*. in which an untrusted client floods a network with DHCP messages. Often these attacks utilize source IP address spoofing to conceal the true source of the attack.

DHCP snooping helps prevent DHCP spoofing by copying DHCP messages to the control plane and using the information in the packets to create anti-spoofing filters. The anti-spoofing filters bind a client's MAC address to its DHCP-assigned IP address and use this information to filter spoofed DHCP messages. In a typical topology, a carrier edge router (in this function also referred to as the broadband services router [BSR]) connects the DHCP server and the MX Series router (or broadband services aggregator [BSA]) performing the snooping. The MX Series router connects to the client and the BSR.

DHCP snooping works as follows in the network topology mentioned above:

1. The client sends a DHCP discover message to obtain an IP address from the DHCP server.
2. The BSA intercepts the message and might add option 82 information specifying the slot, port, VPI/VCI, and so on.
3. The BSA then sends the DHCP discover message to the BSR, which converts it to a unicast packet and sends it to the DHCP server.
4. The DHCP server looks up the client's MAC address and option 82 information in its database. A valid client is assigned an IP address, which is returned to the client using a DHCP offer message. Both the BSR and BSA send this message upstream to the client.
5. The client examines the DHCP offer, and if it is acceptable, issues a DHCP request message that is sent to the DHCP server through the BSA and BSR.
6. The DHCP server confirms that the IP address is still available. If it is, the DHCP server updates its local tables and sends a DHCP ACK message to the client.
7. The BSR receives the DHCP ACK message and passes the message to the BSA.

8. The BSA creates an anti-spoofing filter by binding the IP address in the ACK message to the MAC address of the client. After this point, any DHCP messages from this IP address that are not bound to the client's MAC address are dropped.

9. The BSA sends the ACK message to the client so that the process of assigning a IP address can be completed.

You configure DHCP snooping by including within a DHCP group the appropriate interfaces of the BSA:

```
[edit routing-instances routing-instance-name bridge-domains bridge-domain-name forwarding-options
dhcp-relay group group-name]
interface interface-name;
```

In a VPLS environment, DHCP requests are forwarded over pseudowires. You can configure DHCP snooping over VPLS at the `[edit routing-instances` *`routing-instance-name`*`]` hierarchy level.

DHCP snooping works on a per learning bridge basis in bridge domains. Each learning domain must have an upstream interface configured. This interface acts as the flood port for DHCP requests coming from the client side. DHCP requests are be forwarded across learning domains in a bridge domain. You can configure DHCP snooping on bridge domains at the `[edit routing-instances` *`routing-instance-name`* `bridge-domains` *`bridge-domain-name`*`]` hierarchy level.

RELATED DOCUMENTATION

Preventing DHCP Spoofing

Understanding the Hyper Mode Feature on Enhanced MPCs for MX Series Routers and EX9200 Switches

Enhanced MPCs can be configured to support increased packet processing rates. Enhanced MPCs include these models: MPC3E, MPC4E, MPC5E, MPC6E, MPC7E-MRATE, MPC7E-10G, MX2K-MPC8E, and MX2K-MPC9E.

Starting with Junos OS Release 18.2R1, MPC JNP10K-LC2101 can be configured to support increased packet processing rates. A higher rate of processing of data packets results in the optimization of the lifetime of a data packet. Optimization of the data packet lifetime enables the network device (a router or a switch) to provide better performance and throughput.

To enable the device to support increased packet processing rates, you must configure the hyper mode feature. After configuring the hyper mode feature, you must reboot the device for the changes to take effect.

When you configure the hyper mode feature on the device, the configured mode changes from normal mode to hyper mode. However, because the configuration does not take effect until you reboot the device the current mode of the device remains as normal mode. The current mode changes from normal mode to hyper mode after you reboot the device. If the hyper mode feature is not configured, thedevice processes data packets in normal mode.

NOTE:

- You can enable the hyper mode feature only if the network-service mode on the device is configured as either `enhanced-ip` or `enhanced-ethernet`.
- Hyper-mode is the default forwarding mode on the SCBE3-MX. If your deployment does not need hyper-mode, disable hyper-mode using the `set forwarding-options no-hyper-mode` cli command before installing the Routing Engine into the SCBE3-MX. See also: SCBE3-MX Description

Table 6 on page 180 displays the values of the current and configured mode based on the hyper mode configuration and system reboot.

Table 6: Current Mode and Configured Mode Values Based on Hyper mode Configuration

Action	Current Mode	Configured Mode
Hyper mode is configured but the device is not rebooted.	Normal mode	Hyper mode
Hyper mode is configured and device is rebooted.	Hyper mode	Hyper mode
Hyper mode configuration is removed and device is not rebooted.	Hyper mode	Normal mode
Hyper mode configuration is removed and device is rebooted.	Normal mode	Normal mode

When you configure hyper mode, the following features are not supported:

- Creation of Virtual Chassis
- Forwarding class accounting (enhanced mode)
- Interoperability with legacy DPCs, including MS-DPCs. The MPC in hyper mode accepts and transmits data packets only from other existing MPCs.
- Interoperability with non-Ethernet MICs and non-Ethernet Interfaces such as channelized interfaces, multilink interfaces,
- Junos Fusion
- Junos Node Slicing
- Padding of Ethernet frames with VLAN is not supported in Junos OS releases prior to 19.2R1.
- Precision Time Protocol
- PBB-EVPN.
- Sending Internet Control Message Protocol (ICMP) redirect messages is not supported in Junos OS releases prior to 19.2R1. In versions prior to 19.2R1, ICMP redirects are disabled by default and cannot be re-enabled in hyper mode.
- Termination or tunneling of all subscriber-based services.
- Collecting interface family statistics for IPv4 and IPv6 is not supported in Junos OS releases prior to 19.2R1. The interface family statistics are collected by using the `show interfaces statistics detail` *`interface-name`* command.

After you configure the hyper mode feature and reboot the device, existing MPCs that do not support the hyper mode feature, such as MPC1, MPC2, and MPC3, power on in normal mode. Also, when you have installed MICs and PICs on MPCs that are in normal mode when the hyper mode feature is enabled, those MICs and PICs do not power on. Following is a list of the MICs and PICs that do not power on:

- Channelized E1/T1 Circuit Emulation MIC
- Channelized E1/T1 Circuit Emulation MIC (H)
- *Channelized OC3/STM1 (Multi-Rate) Circuit Emulation MIC with SFP*
- Channelized OC3/STM1 (Multi-Rate) Circuit Emulation MIC with SFP (H)
- Channelized SONET/SDH OC3/STM1 (Multi-Rate) MICs with SFP
- DS3/E3 MIC
- SONET/SDH OC3/STM1 (Multi-Rate) MICs with SFP

- SONET/SDH OC192/STM64 MIC with XFP
- Channelized OC48/STM16 Enhanced IQ (IQE) PIC with SFP

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
18.2R1	Starting with Junos OS Release 18.2R1, MPC JNP10K-LC2101 can be configured to support increased packet processing rates.
15.1	Starting with Junos OS Release 15.1, enhanced MPCs can be configured to support increased packet processing rates.

RELATED DOCUMENTATION

Configuring Hyper Mode on Enhanced MPCs to Speed Up Packet Processing | **182**

Unsupported Features and CLI Commands When Hyper Mode Is Enabled | **184**

show forwarding-options hyper-mode

hyper-mode (forwarding-options)

Configuring Hyper Mode on Enhanced MPCs to Speed Up Packet Processing

Enhanced MPCs can be configured to support increased packet processing rates. Enhanced MPCs include these models: MPC3E, MPC4E, MPC5E, MPC6E,MPC7E-MRATE, MPC7E-10G, MX2K-MPC8E, and MX2K-MPC9E.

JNP10K-LC2101 MPC can be configured to support increased packet processing rates.A higher rate of processing of data packets results in the optimization of the lifetime of a data packet. Optimization of the data packet lifetime enables the network device (a router or a switch) to provide better performance and throughput.

To configure the device to support increased packet processing rates, you must configure the hyper mode feature. After configuring the hyper mode feature, you must reboot the device for the changes to take effect. If the hyper mode feature is not configured, the device processes data packets in normal mode.

NOTE: You can enable the hyper mode feature only if the network-service mode on the device is configured as either `enhanced-ip` or `enhanced-ethernet`.

To configure hyper mode on enhanced MPCs to speed up packet processing:

1. Configure hyper mode by including the `forwarding-options hyper-mode` statement at the [edit] hierarchy level.

   ```
   [edit]
   user@host# set forwarding-options hyper-mode
   ```

2. After configuring hyper mode, commit the configuration.

   ```
   [edit]
   user@host# commit
   ```

   NOTE: After configuring hyper mode and committing the configuration, the configured mode changes to `hyper-mode` but the current mode remains as `normal mode`. The device displays the following warning message after you commit the configuration: `[edit forwarding-options] 'hyper-mode' WARNING: forwarding-options hyper-mode configuration changed. A system reboot is mandatory. Please reboot the system NOW. Continuing without a reboot might result in unexpected system behavior. commit complete`

   When hyper-mode configuration is enabled on supported devices and reboot is performed, the backup routing engine might not come back up in correct mode.

   To recover the backup RE, perform `commit force`, to get the same configuration as the master routing engine. However, to get the enhanced IP configuration on the backup routing engine for the network services, after syncing the script, perform reboot using the command `request routing-engine login other-routing-engine`. Now both the routing engines will come up in enhanced IP mode.

3. Reboot the device for the configuration to take effect.

   ```
   user@host> request system reboot
   ```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
18.2R1	Starting with Junos OS Release 18.2R1, JNP10K-LC2101 MPC can be configured to support increased packet processing rates.
15.1	Starting with Junos OS Release 15.1, enhanced MPCs can be configured to support increased packet processing rates.

RELATED DOCUMENTATION

Understanding the Hyper Mode Feature on Enhanced MPCs for MX Series Routers and EX9200 Switches | **179**

Unsupported Features and CLI Commands When Hyper Mode Is Enabled | **184**

show forwarding-options hyper-mode

hyper-mode (forwarding-options)

Unsupported Features and CLI Commands When Hyper Mode Is Enabled

Table 7 on page 184 lists the features and corresponding CLI commands that are not supported when the hyper mode feature is enabled. Also, the table lists the error messages displayed when you use the unsupported commands.

Table 7: Unsupported Features and CLI Commands When Hyper Mode Is Enabled

Features	Commands	Error Message
Virtual Chassis	`set virtual-chassis preprovisioned`	`To configure virtual-chassis, 'forwarding-options hyper-mode' should not be configured`
	`set virtual-chassis member` *`member-id`* `role` *`role`* `serial-number` *`ser_num`*	

Table 7: Unsupported Features and CLI Commands When Hyper Mode Is Enabled *(Continued)*

Features	Commands	Error Message
	`set virtual-chassis no-split-detection`	
ICMP Redirects Supported in Junos OS Release 19.2R1 and later releases.	`set system no-redirects`	Supported in Junos OS Release 19.2R1 and later releases. In Junos OS versions prior to 19.2R1, the following error message was displayed: `To configure system no-redirects, 'forwarding-options hyper-mode' should not be configured`
	`set system no-redirects-ipv6`	Supported in Junos OS Release 19.2R1 and later releases. In Junos OS versions prior to 19.2R1, the following error message was displayed: `To configure system no-redirects-ipv6, 'forwarding-options hyper-mode' should not be configured`
	`set interfaces` *`interface-name`* `unit` *`logical-unit-number`* `family` *`family`* `no-redirects`	Supported in Junos OS Release 19.2R1 and later releases. In Junos OS versions prior to 19.2R1, the following error message was displayed: `To configure family inet no-redirects, 'forwarding-options hyper-mode' should not be configured`
PPPoE	`set protocols pppoe service-name-tables` *`table-name`*	`To configure pppoe, 'forwarding-options hyper-mode' should not be configured`
	`set dynamic-profiles` *`profile-name`* `interfaces` *`interface-name`* `unit` *`unit`* `family pppoe`	`To configure family pppoe, 'forwarding-options hyper-mode' should not be configured`

Table 7: Unsupported Features and CLI Commands When Hyper Mode Is Enabled *(Continued)*

Features	Commands	Error Message
	`set dynamic-profiles` *`profile-name`* `interfaces demux0 unit` *`unit`* `family pppoe`	`To configure family pppoe, 'forwarding-options hyper-mode' should not be configured`
L2TP	`set access tunnel-profile` *`profile-name`* `tunnel` *`tunnel-id`* `tunnel-type l2tp`	`To configure l2tp, 'forwarding-options hyper-mode' should not be configured`
	`set services l2tp`	`To configure services l2tp, 'forwarding-options hyper-mode' should not be configured`
Forwarding class accounting (enhanced mode)	For more information and CLIs, see CoS-Based Interface Counters for IPv4 or IPv6 Aggregate on Layer 2 and forwarding-class-accounting.	--
Junos Node Slicing	For CLIs, see Setting Up Junos Node Slicing.	--
Junos Fusion	For CLIs, see Junos Fusion Provider Edge User Guide and Junos Fusion Enterprise User Guide.	--
Provider Backbone Bridging (PBB) and Ethernet VPN (EVPN)	For CLIs, see EVPN User Guide.	--
Precision Time Protocol	For more information, see Configuring Precision Time Protocol	—

RELATED DOCUMENTATION

Configuring Hyper Mode on Enhanced MPCs to Speed Up Packet Processing | **182**

Understanding the Hyper Mode Feature on Enhanced MPCs for MX Series Routers and EX9200 Switches | **179** *show forwarding-options hyper-mode*

hyper-mode (forwarding-options)

Custom Profiles for Hardware Resources

SUMMARY

This topic provides an overview of custom profiles for hardware resources

IN THIS SECTION

hw-profile

Using hw-db-profile you can statically reserve hardware resources. But hw-db-profile cannot be used for all hw resources because there are resources that need to be managed independently. You can use the custom profile infrastructure instead to customize hardware resource allocation as per application requirements. Using the custom profile infrastructure allows you to create application-specific dynamic hardware resource allocations.

In the following configuration stanza, the hardware profile (`hw-profile`) is the top level configuration statement under which you specify the currently active custom hardware profile that is applied on the system. `hw-profile-name` is the name of the currently active custom hardware profile that is applied to the system. Upon successful commit, the system restarts the PFE. Note that you can create multiple custom hardware profiles but only use one as the currently active custom hardware profile using this CLI.

```
system {
packet-forwarding-options {
    hw-profile {
        <hw-profile-name>
    }
  }
}
```

Custom hardware profile

As discussed in the preceding section, the custom hardware profile (`hw-profile-name`) is user-defined and holds details for custom profiles created for hardware resources such as counters.

In the following configuration stanza, `<hw-profile-name>` is the name of the custom hardware profile that holds details for custom profiles created for counters (`counter-profile <counter-profile>`) hardware resource.

```
system {
packet-forwarding-options {
    custom-profiles {
        hw-profiles <hw-profile-name> {
            counter-profile <counter-profile>
        }
    }
  }
}
```

In the following configuration stanza, custom profiles are created for counters (hardware resources). Custom profiles created for counters are called counter profiles. In Junos OS release 25.3 only counter profiles are supported.

```
system {
packet-forwarding-options {
    custom-profiles {
        counter-profiles <counter-profile-name> {
            app {
                tcam-ingress {
                    counter-4k <num-of-counter-engines>
                    counter-8k <num-of-counter-engines>
                    counter-16k <num-of-counter-engines>
                    external-cntr <num-of-entries>
                }
                voq {
                    counter-4k <num-of-counter-engines>
                    counter-8k <num-of-counter-engines>
                    counter-16k <num-of-counter-engines>
                }
                ifl-ingress {
                    counter-4k <num-of-counter-engines>
                    counter-8k <num-of-counter-engines>
                    counter-16k <num-of-counter-engines>
                }
                .....
            }
        }
    }
  }
}
```

So the sequence of steps to follow is:

- Create custom counter profiles. See counter-profiles
- Create custom hardware profiles. Each custom hardware profile can be defined with one to many distinct custom profiles. See hw-profiles.
- Apply one custom hardware profile as the active hardware profile. See hw-profile.

Custom counter profile

A custom counter profile provides a mechanism to allocate counter engines per application. The allocations are made based on the scale requirements of applications. Counter resources are provided through various counter engines. There are three types of counter engines based on the number of counters they support. 4k counter engine support 4K counters. Similarly, 8K and 16 counter engine supports 8K and 16K counters. Number of counter engines varies depending on platform. See counter-profiles.

In the following configuration stanza, `<counter-profile-name>` is the name of the custom counter profile. Under `app`, `tcam-ingress`, `voq`, and `ifl-ingress`, are the names of the applications that have been allocated counter hardware resources.

```
system {
packet-forwarding-options {
    custom-profiles {
        counter-profiles <counter-profile-name> {
            app {
                tcam-ingress {
                    counter-4k <num-of-counters>
                    counter-8k <num-of-counters>
                    counter-16k <num-of-counters>
                    external-cntr <num-of-entries>
                }
                voq {
                    counter-4k <num-of-counters>
                    counter-8k <num-of-counters>
                    counter-16k <num-of-counters>
                }
                ifl-ingress {
                    counter-4k <num-of-counters>
                    counter-8k <num-of-counters>
                    counter-16k <num-of-counters>
                }

            }
        }
    }
  }
}
```

Example

The following example demonstrates setting counter resources for applications. Two custom counter profiles are created. Two custom hardware profiles are created for each custom counter profile. One custom hardware profile is set as the active hardware profile

The steps are as follows:

Create two custom counter profiles `cntr1` and `cntr2`

```
set system packet-forwarding-options custom-profiles counter-profiles cntr1 app tcam-ingress
counter-8k 3
set system packet-forwarding-options custom-profiles counter-profiles cntr1 app voq counter-16k 2
set system packet-forwarding-options custom-profiles counter-profiles cntr1 app ifl-ingress
counter-8k 2
set system packet-forwarding-options custom-profiles counter-profiles cntr1 app perf-mon-ingress
counter-4k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr1 app policer-ingress
counter-16k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr1 app tcam-egress
counter-4k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr1 app ifl-egress
counter-8k 2
set system packet-forwarding-options custom-profiles counter-profiles cntr1 app perf-mon-egress
counter-4k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr1 app policer-egress
counter-16k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app tcam-ingress
counter-8k 3
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app ifl-ingress
counter-4k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app ifl-ingress
counter-8k 2
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app ifl-ingress
counter-16k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app perf-mon-ingress
counter-16k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app policer-ingress
counter-16k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app tcam-egress
counter-4k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app ifl-egress
counter-4k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app ifl-egress
counter-8k 2
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app ifl-egress
counter-16k 1
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app storm-control
counter-4k 1
```

```
set system packet-forwarding-options custom-profiles counter-profiles cntr2 app mcast counter-4k
1
```

Create two custom hardware profiles each for the two custom counter profiles

```

set system packet-forwarding-options custom-profiles hw-profiles hw1 counter-profile cntr1
set system packet-forwarding-options custom-profiles hw-profiles hw2 counter-profile cntr2
```

Set one custom hardware profile as the active hardware profile

```

set system packet-forwarding-options hw-profile hw1
```

Considerations in Counter Profile Management

Default behaviour

- If no custom counter profile is defined, the platform uses the default statically allocated counter profile. This allocation will be same as existing allocation prior to the 25.3 release.
- Once a custom profile is defined, only the application mentioned in custom counter profile will have counter allocation as configured by user.

VOQ Statistics - Reserved Counters

- ACX7100 / ACX7509 platforms have one 8K counter engine internally reserved for Port VOQ statistics.
- Other platforms reserve one 4K counter engine by default.
- If a custom counter profile is configured without explicit VOQ allocation, the reserved engine supports only Port VOQ. For HQoS statistics, a custom counter profile with app `voq` needs to be configured.

TCAM and Policer Engine

- It is mandatory to configure the TCAM and Policer applications (tcam-ingress, tcam-egress, policer-ingress, and policer-egress) when defining a custom counter profile in the 25.3 release till it is available for user to configure. TCAM and policer app cannot be configured with values other than their default profile values.

- The configured values for these applications must match the platform's default allocations exactly.
- Any deviation from the default counter enignes (e.g., setting tcam-ingress counter-8k to 2 instead of the default 3) will result in a validation error during commit.
- Counter engines are shared between counters and policers. Applications using policers must explicitly reserve the required number of counter engines to avoid resource conflicts.

Multicast Route Statistics

- Starting from release 25.3, Multicast statistics are supported through the Counter Profile Infrastructure, enabling resource flexibility and application-specific allocation.
- To ensure backward compatibility, the legacy CLI: `set system packet-forwarding-options mcast stats-enable` is retained (hidden) in applicable releases. This allows support for older configurations while transitioning to the new model.
- Validations are introduced to prevent duplicate counter engine reservations for multicast, avoiding configuration conflicts between the legacy CLI and the new profile-based approach.
- It is recommended that users explicitly configure multicast statistics using the custom counter profile, and remove any legacy CLI configuration to ensure clarity and avoid unintended behavior.

External Counters (external-cntr) in OP2 (ACX7332)

- In ACX7332, TCAM uses counter engine in OP2 for IFF/FTF based filters. However, for policers and other uses, TCAM continues to use the internal counter engine. Hence allocation needs to be done accordingly.
- `external-cntr` is allocated based on number of counter entries. Only `tcam-ingress` application can reserve the `external-cntr` entries and is set to 262142 entries.

Configuration guidelines

The following table summarizes the configuration recommendations for counter engines for each application. It is recommended that users configure the number of counter engines to match the number of application scale.

Table 8: Configuration Guidelines for Applications Using Custom Profiles

Application	Configuration Guidelines
storm-control	Storm-control relies on the counter engine to implement policing functionality. Therefore, configuring the counter engine beyond the application scale is not necessary.
VOQ	HQoS scale with statistics support can be increased based on available free counter engine resources. The increase in HQoS scale is not directly proportional to counter engine availability, as it also depends on other hardware resources." Counter engines are assigned at the system level for HQoS
tcam-ingress	Not applicable as counter engines cannot be modified.
perf-mon-ingress	TWAMP and inline sFlow functionality utilize the counter engine for timestamping and sequencing purposes respectively. Therefore, configuring the counter engine beyond the application scale is not necessary.
ifl-ingress	When IFLs are symmetrically distributed across multiple PFEs, a smaller number of counter engine entries may suffice— For example, with 16K IFLs split evenly between two PFEs (8K IFLs per PFE), 8K counter engine entries might be sufficient. This is because the counter engine itself is also distributed symmetrically across PFE. However, if the same total number of IFLs (e.g., 16K IFLs) is configured on a single PFE, it is recommended to provision the full amount of counter engine entries (e.g., 16K) at the system level to ensure statistics can be collected for all interfaces. Above explanation holds good for physical IFL but not for logical IFLs.

Table 8: Configuration Guidelines for Applications Using Custom Profiles *(Continued)*

Application	Configuration Guidelines
policer-ingress	Not Applicable, as counter engines cannot be modified
tcam-egress	Not Applicable, as counter engines cannot be modified
perf-mon-egress	Same as perf-mon-ingress for inline Sflow application.
ifl-egress	Same as ifl-ingress
policer-egress	Not Applicable, as counter engines cannot be modified
multicast	Counter engines are assigned at the system level for multicast routes. The availability of statistics for multicast routes is also dependent on other HW resources (FEC-3 Hierarchy)

- In platforms such as ACX7100-32C, and ACX7100-48L it is not possible to directly copy the default counter profile configuration into a custom profile due to internal reservation of certain apps. As a result, users must manually adjust the counter engines in the custom profile to match the behavior or scale of the default profile, if needed.

  For example, the default counter profile below is the counter engine allocation.

```
show system packet-forwarding-options hw-profile counter-profile
Active Hw-Profile       : default-hw-profile
Active Counter-Profile  : default-counter-profile
Counter Profile Status  : SUCCESS
-----------------------------------------------------------------------------------------
Application-Name   |   4K    |   8K    |   16K   |  External  |  Total    |    Status

                   | Engines | Engines | Engines |            |   (K)     |
-----------------------------------------------------------------------------------------
STORM_CONTROL      |   1     |   0     |   0     |  0         |   4       |  SUCCESS
VOQ                |   1     |   0     |   2     |  0         |   36      |  SUCCESS
TCAM_INGRESS       |   0     |   3     |   0     |  0         |   24      |  SUCCESS
```

```
IFL_INGRESS        |  0     |  2     |  0     | 0          |  16     |  SUCCESS
PERF_MON_INGRESS   |  1     |  0     |  0     | 0          |  4      |  SUCCESS
POLICER_INGRESS    |  0     |  0     |  1     | 0          |  16     |  SUCCESS
IFL_EGRESS         |  0     |  2     |  0     | 0          |  16     |  SUCCESS
TCAM_EGRESS        |  1     |  0     |  0     | 0          |  4      |  SUCCESS
PERF_MON_EGRESS    |  1     |  0     |  0     | 0          |  4      |  SUCCESS
POLICER_EGRESS     |  0     |  0     |  1     | 0          |  16     |  SUCCESS
```

Same scale as default counter profile can be achieved via below by adjusting the counter engines. (VOQ internally 8K is reserved. Internal reservation can be checked in `show cntr-app-map`).

```


show system packet-forwarding-options hw-profile counter-profile
Active Hw-Profile       :     hw1
Active Counter-Profile  :   cntr1
Counter Profile Status  : SUCCESS
---------------------------------------------------------------------------------------
Application-Name   |  4K    |  8K    |  16K   | External  |  Total   |    Status

                   | Engines | Engines | Engines |          |   (K)    |
---------------------------------------------------------------------------------------
STORM_CONTROL      |  1     |  0     |  0     | 0          |  4      |  SUCCESS
VOQ                |  1     |  3     |  0     | 0          |  28     |  SUCCESS    8k
internally reserved (28+8=36K)
TCAM_INGRESS       |  0     |  3     |  0     | 0          |  24     |  SUCCESS
IFL_INGRESS        |  0     |  0     |  1     | 0          |  16     |  SUCCESS
PERF_MON_INGRESS   |  1     |  0     |  0     | 0          |  4      |  SUCCESS
POLICER_INGRESS    |  0     |  0     |  1     | 0          |  16     |  SUCCESS
IFL_EGRESS         |  0     |  0     |  1     | 0          |  16     |  SUCCESS
TCAM_EGRESS        |  1     |  0     |  0     | 0          |  4      |  SUCCESS
PERF_MON_EGRESS    |  1     |  0     |  0     | 0          |  4      |  SUCCESS
POLICER_EGRESS     |  0     |  0     |  1     | 0          |  16     |  SUCCESS
---------------------------------------------------------------------------------------
Total-Allocated    |  4     |  6     |  4     | 0          |
Total-Free         |  2     |  0     |  1     | 0          |
Total-Engines      |  6     |  6     |  5     | 0          |
----------------------------------------------------------------------------------------
```

- For ACX7024, default counter profile allocation is as below.

```
show system packet-forwarding-options hw-profile counter-profile
Active Hw-Profile       : default-hw-profile
Active Counter-Profile  : default-counter-profile
Counter Profile Status  : SUCCESS
-----------------------------------------------------------------------------------
Application-Name   |   4K    |   8K    |   16K   |  External  |  Total    |    Status

                   | Engines | Engines | Engines |            |   (K)     |
-----------------------------------------------------------------------------------
STORM_CONTROL      |  1      |  0      |  0      | 0          |  4        |  SUCCESS
VOQ                |  0      |  0      |  1      | 0          |  16       |  SUCCESS
TCAM_INGRESS       |  0      |  0      |  1      | 0          |  16       |  SUCCESS
IFL_INGRESS        |  0      |  1      |  0      | 0          |  8        |  SUCCESS
PERF_MON_INGRESS   |  1      |  0      |  0      | 0          |  4        |  SUCCESS
POLICER_INGRESS    |  0      |  0      |  1      | 0          |  16       |  SUCCESS
IFL_EGRESS         |  0      |  1      |  0      | 0          |  8        |  SUCCESS
TCAM_EGRESS        |  1      |  0      |  0      | 0          |  4        |  SUCCESS
PERF_MON_EGRESS    |  1      |  0      |  0      | 0          |  4        |  SUCCESS
POLICER_EGRESS     |  0      |  0      |  1      | 0          |  16       |  SUCCESS
```

For custom profile configuration available 4k counter engine is less than default, as 1 4k is internally reserved to VOQ. Hence need to adjust either `PERF_MON_INGRESS or PERF_MON_EGRESS` app to 8k as show below.

```
show system packet-forwarding-options hw-profile counter-profile
Active Hw-Profile       :     hw1
Active Counter-Profile  :   cntr1
Counter Profile Status  : SUCCESS

-----------------------------------------------------------------------------------
Application-Name   |   4K    |   8K    |   16K   |  External  |  Total    |    Status

                   | Engines | Engines | Engines |            |   (K)     |
-----------------------------------------------------------------------------------
STORM_CONTROL      |  1      |  0      |  0      | 0          |  4        |  SUCCESS
VOQ                |  0      |  0      |  1      | 0          |  16       |  SUCCESS   >>>
16+4 (20K available,adjust as needed)
```

```
TCAM_INGRESS       |   0    |   0    |   1    |  0         |   16    |  SUCCESS
IFL_INGRESS        |   0    |   1    |   0    |  0         |   8     |  SUCCESS
PERF_MON_INGRESS   |   0    |   1    |   0    |  0         |   8     |  SUCCESS   >>>
Moved to 8k
POLICER_INGRESS    |   0    |   0    |   1    |  0         |   16    |  SUCCESS
IFL_EGRESS         |   0    |   1    |   0    |  0         |   8     |  SUCCESS
TCAM_EGRESS        |   1    |   0    |   0    |  0         |   4     |  SUCCESS
PERF_MON_EGRESS    |   1    |   0    |   0    |  0         |   4     |  SUCCESS
POLICER_EGRESS     |   0    |   0    |   1    |  0         |   16    |  SUCCESS
-----------------------------------------------------------------------------------------
Total-Allocated    |   3    |   3    |   4    |  0         |
Total-Free         |   0    |   1    |   0    |  0         |
Total-Engines      |   3    |   4    |   4    |  0         |
```

Validation Examples

The following are a few validation examples where the system performs a check before committing the configuration statement.

Ifl symmetric validation for ingress and egress

```

user@host# set system packet-forwarding-options custom-profiles counter-profiles cntr1 app ifl-
ingress counter-4k 7

[edit]
user@host# commit
[edit system packet-forwarding-options custom-profiles counter-profiles cntr1]
  'app'
    Number of ifl ingress and egress counters should be same. Current configuration has ingress:
7 egress: 1
error: commit failed: (validation hook evaluation failed)
```

Number of configured engines exceeds supported engines

```

root@acx7100-w-pfe-17# set system packet-forwarding-options custom-profiles counter-profiles
cntr1 app perf-mon-ingress counter-4k 2
root@acx7100-w-pfe-17# commit
```

```
[edit system packet-forwarding-options custom-profiles counter-profiles cntr1]
  'app'
    '4k' counter engine exceeds limit. Only '5' 4k counter engine can be configured across
application. Current configured is 6
error: commit failed: (validation hook evaluation failed)
```

App specific validation- Inline Sflow is configured, however counter engines required for sflow is not configured

```
[edit system packet-forwarding-options custom-profiles hw-profiles hw1 counter-profile]
  'counter-profile cntr1'
    perf-mon-ingress and perf-mon-egress counter engine required for sflow is not configured.
error: commit failed: (validation hook evaluation failed)
```

Mandatory configuration of TCAM and policer app in the current release

```
root@acx7100-w-pfe-17# set system packet-forwarding-options custom-profiles counter-profiles
cntr2 app ifl-ingress counter-4k 1
[edit system packet-forwarding-options custom-profiles counter-profiles cntr2]
  'app'
    Error: None of the required applications (policer-ingress, policer-egress, tcam-ingress,
tcam-egress) are configured with counters. Default values for these applications are:
    policer-ingress: {'4k': 0, '8k': 0, '16k': 1, 'ext': 0}
    policer-egress: {'4k': 0, '8k': 0, '16k': 1, 'ext': 0}
    tcam-ingress: {'4k': 0, '8k': 3, '16k': 0, 'ext': 0}
    tcam-egress: {'4k': 1, '8k': 0, '16k': 0, 'ext': 0}
error: commit failed: (validation hook evaluation failed)
TCAM and policer app cannot be configured other than default profile values.
```

Configure the same value as the default profile for TCAM app

```
root@acx7100-w-pfe-17# set system packet-forwarding-options custom-profiles counter-profiles
cntr1 app tcam-ingress counter-8k 2
[edit]
root@acx7100-w-pfe-17# commit
[edit system packet-forwarding-options custom-profiles counter-profiles cntr1]
  'app'
```

```
    Error: Need to configure the same value as the default profile for app 'tcam-ingress'
counter-8k. Default value is 3
```

Handling upgrade and downgrade scenarios

You must delete the hardware profile, all custom hardware profiles, and all custom counter profiles to downgrade to a Junos OS version that has no support for the custom profile infrastructure. To upgrade to a Junos OS version where custom profile is supported, follow the regular Junos OS upgrade procedure, and then start using the custom profile infrastructure.

Because prior to 25.3 release, multicast route statistics was supported using the CLI `set system packet-forwarding-options mcast stats-enable`, the system handles multicast route statistics for upgrade and downgrade procedures in the manner summarized in the following table.

Table 9: Multicast Route statistics support during upgrade or downgrade procedures

Scenario	Expected behavior
Upgrade to image with custom profile support - when multicast statistics is not configured on the device.	The upgrade will be successful. User can configure custom profile with `multicast` application, if required.

Table 9: Multicast Route statistics support during upgrade or downgrade procedures *(Continued)*

Scenario	Expected behavior

Upgrade to image with custom profile support - when multicast statistics is configured on the device.	• The multicast statistics CLI command `set system packet-forwarding-options mcast stats-enable` will be hidden to user.
	
	• Counter-engine will be statically reserved for `multicast` application.
	
	• If any multicast statistics configuration has been made with `set system packet-forwarding-options mcast stats-enable` in the old Junos image, this configuration has to be deleted. Then a new custom profile has to be created for the `multicast` application using the new CLIs in the custom profile infrastructure.
	
	• Until if any previous multicast configuration is removed, the system will not allow to configure custom profile for `multicast` application using the new custom profile infrastructure CLIs.
	
	The sequence of steps are as below:
	
	  • Delete the previous CLI configuration
	
	  `delete system packet-forwarding-options mcast stats-enable`
	
	  • Define custom counter profile for the `multicast` application. The following is an example of such a configuration.
	
	  `set system packet-forwarding-options custom-profiles counter-profiles cntr1 app multicast counter-16k 1`
	
	  • Define a custom hardware profile with this custom counter profile

Table 9: Multicast Route statistics support during upgrade or downgrade procedures *(Continued)*

Scenario	Expected behavior

	`set system packet-forwarding-options custom-profiles hw-profiles hw1 counter-profile cntr1`
	
	• Apply this custom hardware profile as the active hardware profile
	
	`set system packet-forwarding-options hw-profile hw1`

Downgrade/ rollback to older Junos OS image, where counter profile infrastructure is not supported, and custom profile configurations for `multicast` application.	Will fail the downgrade. To successfully downgrade to an older Junos OS version, the custom profile infrastructure CLI configurations for `multicast` application need to be deleted. After downgrading to an older Junos image where custom profile infrastructure is not supported, user can configure multicast statistics support using the old multicast statistics CLI - `set system packet-forwarding-options mcast stats-enable`
Downgrade/ rollback to older Junos OS image, where counter profile infrastructure is not supported, and multicast statistic configuration has already been set using `set system packet-forwarding-options mcast stats-enable`	Downgrade to the older image will be successful and multicast counter-engine would be reserved after the image downgrade.

Troubleshooting - hw-profile

- Use show commands `show system packet-forwarding-options hw-profile` and `show system packet-forwarding-options hw-profile counter-profile` to display any failure in counter profile and hardware profile activation.
- You can also use PFE commands - `show evo-pfemand cntr-info` and `show evo-pfemand cntr-app-map` to diagnose the failure in PFE. `show evo-pfemand cntr-info` will display the counter engines allocated to

each feature. This should match the number of counter engines configured by user. `show evo-pfemand cntr-app-map` will display whether counter configuration is success or failure for each feature in PFE and the reason for failure.

Show command outputs when hw-profile is not configured - default hw profile

```
user@router# run show system packet-forwarding-options hw-profile
Active Hw-Profile  : default-hw-profile
Profile Status     : SUCCESS

-----------------------------------------------------------
Profile-Name    |        App              |    Status
-----------------------------------------------------------
 default-counter-profile |       COUNTER    |   SUCCESS
```

```
user@router# run show system packet-forwarding-options hw-profile counter-profile
Active Hw-Profile       : default-hw-profile
Active Counter-Profile  : default-counter-profile
Counter Profile Status  : SUCCESS
------------------------------------------------------------------------------------
Application-Name   |   4K    |   8K    |   16K   |  External  |  Total   |    Status
                   | Engines | Engines | Engines |            |   (K)    |
------------------------------------------------------------------------------------
STORM_CONTROL      |   1     |  0      |  0      |  0         |   4      |  SUCCESS
VOQ                |   1     |  0      |  2      |  0         |   36     |  SUCCESS
TCAM_INGRESS       |   0     |  3      |  0      |  0         |   24     |  SUCCESS
IFL_INGRESS        |   0     |  2      |  0      |  0         |   16     |  SUCCESS
PERF_MON_INGRESS   |   1     |  0      |  0      |  0         |   4      |  SUCCESS
POLICER_INGRESS    |   0     |  0      |  1      |  0         |   16     |  SUCCESS
IFL_EGRESS         |   0     |  2      |  0      |  0         |   16     |  SUCCESS
TCAM_EGRESS        |   1     |  0      |  0      |  0         |   4      |  SUCCESS
PERF_MON_EGRESS    |   1     |  0      |  0      |  0         |   4      |  SUCCESS
POLICER_EGRESS     |   0     |  0      |  1      |  0         |   16     |  SUCCESS
```

Show command outputs when hw-profile is configured

```
user@router# run show system packet-forwarding-options hw-profile
Active Hw-Profile  :     hw1
Profile Status     : SUCCESS
-------------------------------------------------------------
Profile-Name    |        App              |    Status
-------------------------------------------------------------
 cntr1          |       COUNTER           |   SUCCESS
```

```
user@router# run show system packet-forwarding-options hw-profile counter-profile
Active Hw-Profile       :     hw1
Active Counter-Profile  :   cntr1
Counter Profile Status  : SUCCESS

-------------------------------------------------------------------------------------
Application-Name   |   4K    |   8K    |   16K   |  External  |  Total    |    Status
                   | Engines | Engines | Engines |            |   (K)     |
-------------------------------------------------------------------------------------
TCAM_INGRESS       |   0     |   3     |   0     |  0         |   24      |  SUCCESS
POLICER_INGRESS    |   0     |   0     |   1     |  0         |   16      |  SUCCESS
TCAM_EGRESS        |   1     |   0     |   0     |  0         |   4       |  SUCCESS
POLICER_EGRESS     |   0     |   0     |   1     |  0         |   16      |  SUCCESS
-------------------------------------------------------------------------------------
Total-Allocated    |   1     |   3     |   2     |  0         |
Total-Free         |   4     |   4     |   3     |  0         |
Total-Engines      |   5     |   7     |   5     |  0         |
--------------------------------------------------------------------------------------
```

PFE debug commands - default counter profile

```
user@router:pfe> show evo-pfemand cntr-app-map
Counter Profile Name: default-counter-profile
Counter App Map Info:

    Appname      Counter4k     Counter8k     Counter16k     CounterExt     Status
================================================================================
STORM_CONTROL        1          0            0              0        SUCCESS
VOQ                  1          0            2              0        SUCCESS
TCAM_INGRESS         0          3            0              0        SUCCESS
IFL_INGRESS          0          2            0              0        SUCCESS
PERF_MON_INGRESS     1          0            0              0        SUCCESS
POLICER_INGRESS      0          0            1              0        SUCCESS
IFL_EGRESS           0          2            0              0        SUCCESS
TCAM_EGRESS          1          0            0              0        SUCCESS
PERF_MON_EGRESS      1          0            0              0        SUCCESS
POLICER_EGRESS       0          0            1              0        SUCCESS
================================================================================
Total-used           5          7            4              0
================================================================================
Total-Usr-Available  5          7            5             NA
================================================================================
Total-BCM-Available  8          8            6             NA
================================================================================
```

```


user@router:pfe> show evo-pfemand cntr-info
InitDone:1
Total Num of Engines:22
Bitmap: num_units=1 size=22 utilized=72.7273% Used=16 peak utilization=16% lastUnitIdx=22
mask=8000000000000000 BM_UNIT_SIZE=64HWM Ts=1752158450972060177%
BM[    0]= XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX0011111111011100011111 = 0xff71f
EngineType   StartIdx    EndIdx
===============================
 4kEngine:     0           7
 8kEngine:     8           15
 16kEngine:    16          21

EngineId  AppName
==========================
       0  Storm Control
       1  Voq Stats
       2  FW StatsEgr
       3  Oamp Egress Stats
       4  Oamp Stats
       8  FW StatsIng
       9  FW StatsIng
      10  FW StatsIng
```

```
        12  Ingress Lif Stats
        13  Ingress Lif Stats
        14  Egress Lif Stats
        15  Egress Lif Stats
        16  Voq Stats
        17  Voq Stats
        18  FW PolicerIng
        19  FW PolicerEgr
```

Troubleshooting - custom counter profile

Debug Commands- Counter profile configured

```
user@router:pfe> show evo-pfemand cntr-app-map
Counter Profile Name: cntr1
Counter App Map Info:
    Appname      Counter4k    Counter8k    Counter16k    CounterExt    Status
================================================================================
TCAM_INGRESS          0          3            0              0        SUCCESS
POLICER_INGRESS       0          0            1              0        SUCCESS
TCAM_EGRESS           1          0            0              0        SUCCESS
POLICER_EGRESS        0          0            1              0        SUCCESS

Reserved Counter App Map Info:

VOQ                   1          0            0              0        SUCCESS
================================================================================
Total-used            2          3            2              0
================================================================================
Total-Usr-Available   5          7            5             NA
================================================================================
Total-BCM-Available   8          8            6             NA
================================================================================
```

```
user@router:pfe> show evo-pfemand cntr-info
InitDone:1
```

```
Total Num of Engines:22
Bitmap: num_units=1 size=22 utilized=31.8182% Used=7 peak utilization=7% lastUnitIdx=22
mask=8000000000000000
BM_UNIT_SIZE=64HWM Ts=1748943680386874258%
BM[    0]= XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX0000110000011100000011 = 0x30703
EngineType   StartIdx    EndIdx
===============================
 4kEngine:     0           7
 8kEngine:     8           15
 16kEngine:    16          21

EngineId  AppName
==========================
       0  FW StatsEgr
       1  Voq Stats
       8  FW StatsIng
       9  FW StatsIng
      10  FW StatsIng
      16  FW PolicerIng
      17  FW PolicerEgr
```

Caveats

- Policers can only be allocated with 16K engines
- Certain counters are reserved for internal use and are not available to users.
- In ACX7332, TCAM uses counter engine in OP2 for IFF/FTF based filters. However, for policers and other uses, TCAM still continue to internal counter engine. Hence allocation needs to be done accordingly.
- `Ifl-ingress` and `ifl-egress` counter engines must be symmetrically allocated.
- Prior to 25.3 release, Multicast counter engines are shared with LSP/IP tunnel counter engines and only one of them can be enabled using respective CLI. Hence the counter-engine reserved by LSP/IP applications will continue to be reserved. When multicast stats is enabled via old CLI (`set system packet-forwarding-options mcast stats-enable`), it would continue to statically reserve the counter-engine shared with LSP/Transit. If multicast stats is enabled via counter-profile, it will use from the resources available to the custom counter-profile.
- Port VOQ uses internally reserved counters. Once the user-configured counter engines for HQoS are exhausted, they may overlap with the remaining entries of the Port VOQ counters.

Limitations

- The behavior will be non-deterministic if the application scales beyond the number of custom profile counter engines allocated. For example, if IFL scale is 8k, but only one 4k counter engine is configured, then it cannot be guaranteed which IFL statistics will be displayed.
- BNG DB profile is not supported from hw-db-profile in cases where hw-db-profile and custom hardware profile co-exist on a system.
- Because evo-pfemand is restarted, when a CLI successfully committed, existing stats are cleared.

7

CHAPTER

Configuration Statements and Operational Commands

IN THIS CHAPTER

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Read this guide to learn about the syntax and options that make up the statements and commands. Also understand the contexts in which you'll use these CLI elements in your network configurations and operations.

- Junos CLI Reference

Click the links to access Junos OS and Junos OS Evolved configuration statement and command summary topics.

- Configuration Statements
- Operational Commands