

Chapter 6

Classifying Packets Based on Various Packet Header Fields

A multifield (MF) classifier is a method of classifying traffic flows. Devices that sit at the edge of a network usually classify packets according to codings that are located in multiple packet header fields. MF classification is normally performed at the network edge because of the general lack of DiffServ Code Point (DSCP) or IP precedence support in end-user applications.

In an edge router, an MF classifier provides the filtering functionality that scans through a variety of packet fields to determine the forwarding class for a packet. Typically, a classifier performs matching operations on the selected fields against a configured value.

Unlike a behavior aggregate (BA), which classifies packets based on class-of-service (CoS) bits in the packet header, an MF classifier can examine multiple fields in the packet header—for example, the source and destination address of the packet, and the source and destination port numbers of the packet. An MF classifier typically matches one or more of the six packet header fields: destination address, source address, IP protocol, source port, destination port, and DSCP. MF classifiers are used when a simple BA classifier is insufficient to classify a packet.

If you configure both a BA classifier and an MF classifier, BA classification is performed first; then MF classification is performed. If they conflict, any BA classification result is overridden by the MF classifier.



NOTE: For a specified interface, you can configure both an MF classifier and a BA classifier without conflicts. Because the classifiers are always applied in sequential order, the BA classifier followed by the MF classifier, any BA classification result is overridden by an MF classifier if they conflict.

In the JUNOS software, you configure an MF classifier with a firewall filter and its associated match conditions. This enables you to use any filter match criteria to locate packets that require classification. From a CoS perspective, MF classifiers (or firewall filter rules) provide the following services:

- Classify packets to a forwarding class and loss priority. The forwarding class determines the output queue. The loss priority is used by schedulers in conjunction with the random early discard (RED) algorithm to control packet discard during periods of congestion.
- Police traffic to a specific bandwidth and burst size. Packets exceeding the policer limits can be discarded, or can be assigned to a different forwarding class, to a different loss priority, or to both.

To activate an MF classifier, you must configure it on a logical interface. There is no restriction on the number of MF classifiers you can configure.

To configure MF classifiers, you can include the following statements at the `[edit firewall]` hierarchy level of the configuration:

```
[edit firewall]
family family-name {
  filter filter-name {
    term term-name {
      from {
        match-conditions;
      }
      then {
        dscp 0;
        forwarding-class class-name;
        loss-priority (high | low);
      }
    }
  }
}
simple-filter filter-name {
  term term-name {
    from {
      match-conditions;
    }
    then {
      forwarding-class class-name;
      loss-priority (high | low | medium);
    }
  }
}
}
```

The `[edit firewall]` configuration statements are discussed in detail in the *JUNOS Policy Framework Configuration Guide*.

This chapter includes examples showing how to use multifield classifiers to classify packets based on destination address, and to classify packets according to whether the traffic is voice over IP (VoIP), best effort, or network control. These examples are shown in the following sections:

- Example: Classifying Packets Based on a Destination Address on page 67
- Example: Configuring and Confirming a Complex MF Filter on page 68
- Example: Writing Different DSCP and EXP Values in MPLS-Tagged IP Packets on page 71
- Example: Configuring a Simple Filter on page 72
- Configuring a Logical Bandwidth Policer on page 74
- Configuring a Logical Bandwidth Policer on page 74

Example: Classifying Packets Based on a Destination Address

Configure an MF classifier that ensures that all IPv4 packets destined for the 10.10.10.0/24 network are placed into the `platinum` forwarding class. This assignment occurs regardless of the received CoS bit values in the packet. Apply this filter to the inbound interface `so-1/2/2.0`.

To verify your configuration, issue the `show interfaces filters` command.

```

firewall {
  family inet {
    filter set-FC-to-platinum {
      term match-a-single-route {
        from {
          destination-address {
            10.10.10.0/24;
          }
        }
        then {
          forwarding-class platinum;
          accept;
        }
      }
      term accept-all {
        then accept;
      }
    }
  }
}
interfaces {
  so-1/2/2 {
    unit 0 {
      family inet {
        filter {
          input set-FC-to-platinum;
        }
      }
    }
  }
}

```

```

    }
  }
}

```

Example: Configuring and Confirming a Complex MF Filter

In this example, SIP signaling (VoIP) messages use TCP/UDP, port 5060, and RTP media channels use UDP with port assignments from 16,384 through 32,767.

Configure the following:

- Classify VoIP traffic as EF.
- Classify network control traffic as NC.
- Classify all remaining traffic with IP precedence 0 as BE.
- Police BE traffic to 1 Mbps with excess data marked with PLP high.

The firewall filter called `classify` matches on the transport protocol and ports identified in the incoming packets and classifies packets into the forwarding classes specified by your criteria.

Classifying SIP Signaling Messages

The first term, `sip`, classifies SIP signaling messages. The `port` statement matches any source port or destination port (or both) that is coded to 5060.

```

firewall {
  family inet {
    filter classify {
      interface-specific;
      term sip {
        from {
          protocol [ udp tcp ];
          port 5060;
        }
        then {
          forwarding-class expedited-forwarding;
          accept;
        }
      }
    }
  }
}

```

Classifying VoIP Channels That Use UDP

The second term, `rtp`, classifies VoIP media channels that use UDP-based transport.

```

term rtp {
  from {
    protocol udp;
    port 16384-32767;
  }
  then {
    forwarding-class expedited-forwarding;
    accept;
  }
}

```

Configuring the Policer The policer's burst tolerance is set to the recommended value for a low-speed interface, which is ten times the interface MTU. For a high-speed interface, the recommended burst size is the transmit rate of the interface times 3 to 5 milliseconds.

```

    policer be-policer {
      if-exceeding {
        bandwidth-limit 1m;
        burst-size-limit 15k;
      }
      then loss-priority high;
    }
  }
}

```

Policing All Remaining Traffic The third term, `be`, ensures that all remaining traffic is policed according to a bandwidth restriction.

```

    term be {
      then policer be-policer;
    }
  }
}

```

Confirming Default Classification The `be` term does not include a `forwarding-class` action modifier. Furthermore, there is no explicit treatment of network control (NC) traffic provided in the `classify` filter. You can configure explicit classification of NC traffic and all remaining IP traffic, but you do not need to, because the default IP precedence classifier correctly classifies the remaining traffic. To confirm, display the default classifiers in effect on the interface by issuing the `show class-of-service interface interface-name` command. The display confirms that the `ipprec-compatibility` classifier is in effect by default.

```

user@host> show class-of-service interface fe-0/0/2
Physical interface: so-0/2/3, Index: 135
Queues supported: 8, Queues in use: 4
Scheduler map: <default>, Index: 2032638653

Logical interface: fe-0/0/1.0, Index: 68
Shaping rate: 32000

```

| Object | Name | Type | Index |
|---------------|-----------------------------|------|-------|
| Scheduler-map | <default> | | 27 |
| Rewrite | exp-default | exp | 21 |
| Classifier | exp-default | exp | 5 |
| Classifier | ipprec-compatibility | ip | 8 |

Displaying Default Classifier Mappings

To view the default classifier mappings, issue the `show class-of-service classifier name name` command. The highlighted output confirms that traffic with IP precedence setting of 0 is correctly classified as BE, and NC traffic, with precedence values of 6 or 7, is properly classified as NC.

```

user@host> show class-of-service classifier name ipprec-compatibility
Classifier: ipprec-compatibility, Code point type: inet-precedence, Index:
12
Code point      Forwarding class      Loss priority
000           best-effort         low
001            best-effort          high
010            best-effort          low
011            best-effort          high
100            best-effort          low
101            best-effort          high
110           network-control    low
111           network-control    high

```

Applying the Classifier Apply the classify classifier to the fe-0/0/2 interface:

```

interfaces {
  fe-0/0/2 {
    unit 0 {
      family inet {
        filter {
          input classify;
        }
        address 10.12.0.13/30;
      }
    }
  }
}

```

Confirming MF Classification

To confirm that your MF classifier is working correctly, you can monitor the queue counters for the router's egress interface used when forwarding traffic received from the peer. Displaying the queue counters for the ingress interface (fe-0/0/2) does not allow you to check your ingress classification, because queuing generally occurs only at egress in the JUNOS software. (Ingress queuing is supported on Gigabit Ethernet IQ2 PICs only, as discussed in "Shaping Input and Output Traffic on Ethernet IQ2 Interfaces" on page 199.)

1. To determine which egress interface is used for the traffic, use the `traceroute` command.
2. After you identify the egress interface, clear its associated queue counters by issuing the `clear interfaces statistics interface-name` command.
3. Confirm the default forwarding class-to-queue number assignment. This allows you to predict which queues are used by the VoIP, NC, and other traffic. To do this, issue the `show class-of-service forwarding-class` command.
4. Display the queue counts on the interface by issuing the `show interfaces queue` command.

Example: Writing Different DSCP and EXP Values in MPLS-Tagged IP Packets

On M320 and T-series platforms, you can selectively set the DSCP field of MPLS-tagged IPv4 and IPv6 packets to 000000. In the same packets, you can set the MPLS EXP field according to a configured rewrite table, which is based on the forwarding classes that you set in incoming packets using a BA or MF classifier.

Queue selection is based on the forwarding classes you assign in scheduler maps. This means that you can direct traffic to a single output queue, regardless of whether the DSCP field is unchanged or rewritten to 000000. To do this, you must configure an MF classifier that matches selected packets and modifies them with the `dscp 0` action.

Selective marking of DSCP fields to 0, without affecting output queue assignment, can be useful. For example, suppose you need to use the MPLS EXP value to configure CoS applications for core provider routers. At the penultimate egress provider edge (PE) router where the MPLS labels are removed, the CoS bits need to be provided by another value, such as DSCP code points. This case illustrates why it is useful to mark both the DSCP and MPLS EXP fields in the packet. Furthermore, it is useful to be able to mark the two fields differently, because the CoS rules of the core provider router might differ from the CoS rules of the egress penultimate router. At egress, as always, you can use a rewrite table to rewrite the MPLS EXP values corresponding to the forwarding classes that you need to set.

In the following example, term 1 of the MF classifier matches packets with DSCP 001100 code points coming from a certain VRF, rewrites the bits to DSCP 000000, and sets the forwarding class to `best-effort`. In term 2, the classifier matches packets with DSCP 010110 code points and sets the forwarding class to `best-effort`. Because term 2 does not include the `dscp 0` action modifier, the DSCP 010110 bits remain unchanged. Because the classifier sets the forwarding class for both code points to `best-effort`, both traffic types are directed to the same output queue.

```

firewall {
  family inet {
    filter vrf-rewrite {
      term 1 {
        from {
          dscp 001100;
        }
        then {
          dscp 0;
          forwarding-class best-effort;
        }
      }
      term 2 {
        from {
          dscp 010110;
        }
        then {
          forwarding-class best-effort;
        }
      }
    }
  }
}

```

Applying the MF Classifier Apply the filter to an input interface corresponding to the VRF:

```

interfaces {
  so-0/1/0 {
    unit 0 {
      family inet {
        filter input vrf-rewrite;
      }
    }
  }
}

```



NOTE: The `dscp 0` action is supported in both input and output filters. You can use this action for non-MPLS packets as well as for IPv4 and IPv6 packets entering an MPLS network. All IPv4 and IPv6 firewall filter match conditions are supported with the `dscp 0` action.

The following limitations apply:

- You can use an MF classifier to rewrite DSCP fields to value 0 only. Other values are not supported.
- If a packet matches a filter that has the `dscp 0` action, then the outgoing DSCP value of the packet is 0, even if the packet matches a rewrite rule, and the rewrite rule is configured to mark the packet to a non-zero value. The `dscp 0` action overrides any other rewrite rule actions configured on the router.
- Although you can use the `dscp 0` action on an input filter, the output filter and other classifiers do not see the packet as being marked `dscp 0`. Instead, they classify the packet based on its original incoming DSCP value. The DSCP value of the packet is set to 0 after all other classification actions have completed on the packet.

Example: Configuring a Simple Filter

Configure a simple filter. Simple filters are recommended for metropolitan Ethernet applications. They are supported on Gigabit Ethernet intelligent queuing 2 (IQ2) interfaces only. Unlike normal filters, simple filters are for IPv4 traffic only and have the following restrictions:

- The `next term` action is not supported.
- Qualifiers, such as the `except` and `protocol-except` statements, are not supported.
- Noncontiguous masks are not supported.
- Multiple source addresses and destination addresses in a single term are not supported.

- Output filters are not supported. You can apply a simple filter to ingress traffic only.
- Explicitly configurable terminating actions, such as `accept`, `reject`, and `discard`, are not supported. Simple filters always accept packets.

```

firewall {
  family inet {
    simple-filter filter1 {
      term 1 {
        from {
          source-address {
            1.1.1.1/32;
          }
          protocol {
            tcp;
          }
        }
        then loss-priority low;
      }
      term 2 {
        from {
          source-address {
            4.0.0.0/8;
          }
          source-port {
            http;
          }
        }
        then loss-priority high;
      }
      term 3 {
        from {
          destination-address {
            6.6.6.6/32;
          }
        }
        then {
          loss-priority low;
          forwarding-class best-effort;
        }
      }
    }
  }
}
interfaces {
  ge-0/0/1 {
    unit 0 {
      family inet {
        scheduler-filter {
          input filter1;
        }
        address 10.1.2.3/30;
      }
    }
  }
}

```

Configuring a Logical Bandwidth Policer

Logical bandwidth policers are used to apply the same shaping rate limit on logical interfaces as on physical interfaces. The feature is supported only for interface filters and policers and on IQ PICs.

To apply a policer to a logical interface, include the `logical-bandwidth-policer` statement in the policer at the `[edit firewall policer]` hierarchy level.

```
[edit firewall policer policer-name]
  logical-bandwidth-policer;
```

The policer must be applied at the logical interfaces level and reference a valid `shaping-rate` statement at the class-of-service hierarchy level.

Example: Configuring a Logical Bandwidth Policer

This example applies a logical bandwidth policer rate to two logical interfaces on interface `ge-0/2/7`. The policed rate on unit 0 is 2 Mbps (50 percent of 4 Mbps) and the policed rate on unit 1 is 1 Mbps (50 percent of 2 Mbps).

```
[edit firewall]
policer Logical_Policer {
  logical-bandwidth-policer; # This applies the policer to logical interfaces
  if-exceeding {
    bandwidth-percent 50; # This applies 50 percent to the shaping-rate
    burst-size-limit 125k
  }
  then discard;
}

[edit class-of-service]
interfaces {
  ge-0/2/7 {
    unit 0 {
      shaping-rate 4m # This establishes the rate to be policed on unit 0
    }
    unit 1 {
      shaping-rate 2m # This establishes the rate to be policed on unit 1
    }
  }
}

[edit interfaces ge-0/2/7]
per-unit-scheduler;
vlan-tagging;
unit 0 {
  vlan-id 100;
  family inet {
    policer {
      input Logical_Policer;
      output Logical_Policer;
    }
    address 172.1.1.1/30;
  }
}
```

```
unit 1 {  
    vlan-id 200;  
    family inet {  
        policer {  
            input Logical_Policer;  
            output Logical_Policer;  
        }  
        address 172.2.1.1/30;  
    }  
}
```

