

Chapter 33

CoS Configuration Guidelines

To configure CoS properties, you can include the following statements at the [edit class-of-service] hierarchy level of the configuration:

```
class-of-service {
  classifiers {
    type classifier-name {
      import (classifier-name | default);
      forwarding-class class-name {
        loss-priority (low | high) code-points [ alias | bits ];
      }
    }
  }
  code-point-aliases {
    (dscp | exp | ieee-802.1 | inet-precedence) {
      alias-name bits;
    }
  }
  drop-profiles {
    profile-name {
      fill-level percentage drop-probability percentage;
      interpolate {
        drop-probability value;
        fill-level value;
      }
    }
  }
  fabric {
    scheduler-map {
      priority (low | high) scheduler scheduler-name;
    }
  }
  forwarding-classes {
    queue queue-number class-name priority (low | high);
  }
  forwarding-policy {
    next-hop-map map-name {
      forwarding-class class-name {
        next-hop [ next-hop-name ];
        lsp-next-hop [ lsp-regular-expression ];
      }
    }
  }
}
```

```

class class-name {
  classification-override {
    forwarding-class class-name;
  }
}
}
interfaces
  interface-name {
    scheduler-map map-name;
    unit logical-unit-number {
      bandwidth rate;
      classifiers {
        (dscp | exp | ieee-802.1 | inet-precedence) (classifier-name | default);
      }
      forwarding-class class-name;
      per-unit-scheduler;
      rewrite-rules {
        dscp (rewrite-name | default);
        exp (rewrite-name | default) protocol protocol-types;
        exp-push-push-push default;
        exp-swap-push-push default;
        ieee-802.1 default;
        inet-precedence (rewrite-name | default);
      }
      scheduler-map map-name;
    }
  }
}
rewrite-rules {
  (dscp | exp | inet-precedence) rewrite-name {
    import (rewrite-name | default);
    forwarding-class class-name {
      loss-priority (low | high) code-point (alias | bits);
    }
  }
}
scheduler-maps {
  map-name {
    forwarding-class class-name scheduler scheduler-name;
  }
}
schedulers {
  scheduler-name {
    buffer-size (percent percentage | remainder | temporal microseconds);
    drop-profile-map loss-priority (low | high) protocol (non-tcp | tcp | any)
      drop-profile profile-name;
    priority (low | high | strict-high);
    transmit-rate (rate | percent percentage | remainder | exact);
  }
}
}

```

The following RFCs define the standards supported by certain aspects of the CoS software:

RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*

RFC 2598, *An Expedited Forwarding PHB* (see also draft-ietf-diffserv-rfc2598bis-01.txt)

RFC 2597, *Assured Forwarding PHB Group*

RFC 2983, *Diffserv and Tunnels*, is not supported.

The JUNOS software supports only two loss priorities and, by default, supports only one assured forwarding (AF) class, although you can configure more at the expense of other class types.

This chapter includes the following sections:

Hardware Capabilities and Limitations on page 588

Define Code-Point Aliases on page 589

Configure Forwarding Classes on page 592

Classify Packets by Behavior Aggregate on page 594

Configure Scheduling Policy Maps on page 596

Configure RED Drop Profiles on page 601

Rewrite Packet Header Information on page 602

Configure CoS-Based Forwarding on page 610

Example: Configure Class of Service on page 617

Hardware Capabilities and Limitations

Juniper Networks T-series platforms and M-series platforms with an enhanced FPC have more CoS capabilities than the M-series platforms that employ the earlier FPC model. Table 33 lists the differences between the FPC and the enhanced FPC.

Table 33: CoS Hardware Capabilities and Limitations

Feature	M-series FPC	M-series Enhanced FPC	T-series FPC	Comments
Classifiers				
Limit per FPC or PIC	1	8	64	For M-series FPC, the one-classifier limit includes the default IP precedence classifier. If you create a new classifier and apply it to an interface, the new classifier does not override the default classifier for other interfaces on the same FPC. The general rule for classifier replacement is that the first classifier associated with a logical interface is the one that is used. The default classifier can be replaced only when a single interface is associated with the default classifier. For M-series Enhanced FPC, four classifiers are shared between DSCP and IP and four are shared between MPLS EXP and IEEE 802.1p.
dscp	no	yes	yes	
ieee-802.1p	no	yes	yes	
inet-precedence	yes	yes	yes	For M-series FPC and T-series FPC, the loss priority cannot be arbitrarily decoded from the code point.
mpls-exp	yes	yes	yes	For M-series FPC, only the default MPLS EXP classifier is supported; the default MPLS EXP classifier takes the EXP bits 1 and 2 as the output queue number.
Rewrite Markers				
Limit per FPC or PIC	none	none	64	
dscp	no	yes	yes	For T-series FPC, you must decode the loss priority using the firewall filter before you can use loss priority to select the rewrite code point.
ieee-802.1	no	yes	yes	For M-series Enhanced FPC and T-series FPC, fixed rewrite loss priority determines the value for bit 0; queue number (forwarding class) determines bits 1 and 2.
inet-precedence	yes	yes	yes	For T-series FPC, you must decode the loss priority using the firewall filter before you can use loss priority to select the rewrite code point.
mpls-exp	yes	yes	yes	For T-series FPC, you must decode the loss priority using the firewall filter before you can use loss priority to select the rewrite code point. For M-series FPC, fixed rewrite loss priority determines the value for bit 0; queue number (forwarding class) determines bits 1 and 2.
Queuing				
Priority	no	yes	yes	
Drop Profiles				

Feature	M-series FPC	M-series Enhanced FPC	T-series FPC	Comments
Limit per FPC or PIC	2	16	32	
Per queue	no	yes	yes	
Per loss priority	yes	yes	yes	
Per TCP bit	no	yes	yes	

Define Code-Point Aliases

A code-point alias is a name you assign to a set of DiffServ code-point (DSCP) bits. When you configure classes and define classifiers, you can refer to the code points by these alias names. You can configure user-defined classifiers in terms of alias names. If the value of an alias changes, it alters the behavior of any classifier that references that alias.

Table 34 shows the default mappings between the bit values and standard aliases. For example, it is widely accepted that the alias for DSCP 101110 is ef (expedited forwarding).

Table 34: Default DSCP Mappings

DiffServ Code Designator	Mapping
DSCP Code Points	
ef	101110
af11	001010
af12	001100
af13	001110
af21	010010
af22	010100
af23	010110
af31	011010
af32	011100
af33	011110
af41	100010
af42	100100
af43	100110
be	000000
cs1	001000
cs2	010000
cs3	011000
cs4	100000
cs5	101000
nc1/cs6	110000
nc2/cs7	111000

DiffServ Code Designator	Mapping
MPLS EXP Code Points	
be	000
be1	001
ef	010
ef1	011
af11	100
af12	101
nc1/cs6	110
nc2/cs7	111
IEEE 802.1 Code Points	
be	000
be1	001
ef	010
ef1	011
af11	100
af12	101
nc1/cs6	110
nc2/cs7	111
Legacy IP Precedence Code Points	
be	000
be1	001
ef	010
ef1	011
af11	100
af12	101
nc1/cs6	110
nc2/cs7	111

You use code-point aliases to do the following:

- Define an alias for bits that currently have no alias

- Define multiple aliases for the same bits

- Redefine an alias name to mean a different set of bits than the default

To define a code-point alias, include the `code-point-aliases` statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
code-point-aliases {
  (dscp | exp | ieee-802.1 | inet-precedence) {
    alias-name bits;
  }
}
```

For example, you might set up the following configuration:

```
[edit class-of-service]
code-point-aliases {
  dscp {
    my1 110001;
    my2 101110;
    be 000001;
    cs7 110000;
  }
}
```

The sample configuration produces this mapping:

```
user@host>show class-of-service code-point-aliases dscp
Code point type: dscp
Alias      Bit pattern
ef/my2    101110
af11      001010
af12      001100
af13      001110
af21      010010
af22      010100
af23      010110
af31      011010
af32      011100
af33      011110
af41      100010
af42      100100
af43      100110
be        000001
cs1       001000
cs2       010000
cs3       011000
cs4       100000
cs5       101000
nc1/cs6/cs7 110000
nc2       111000
my1       110001
```

The following notes explain certain results in the mapping:

my1 110001:

110001 was not mapped to anything before, and my1 is a new alias.

Nothing in the default mapping table is changed by this statement.

my2 101110:

101110 is now mapped to my2 as well as ef.

be 000001:

be is now mapped to 000001.

The old value of be, 000000, is not associated with any alias. Packets with this DSCP value are now classified to the default forwarding class.

cs7 110000:

cs7 is now mapped to 110000, as well as nc1 and cs6.

The old value of cs7, 111000, is still mapped to nc2.

Configure Forwarding Classes

Forwarding classes replace output queues from the previous CoS configuration command set. You assign each forwarding class to an internal queue number by including the forwarding-classes statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
forwarding-classes {
  queue queue-number class-name priority (high | low);
}
```



Note

You cannot commit a configuration that assigns the same forwarding class to two different queues.

Table 35 shows the four forwarding classes defined by default.

Table 35: Default Forwarding Classes

Queue	Forwarding Class Name
queue 0	best-effort
queue 1	expedited-forwarding
queue 2	assured-forwarding
queue 3	network-control

The following rules govern queue assignment:

If classifiers fail to classify a packet, the packet always receives the default classification to the class associated with queue 0.

The number of queues is dependent on the hardware plugged into the chassis. CoS configurations are inherently contingent on the number of queues on the system. Only two classes, best-effort and network-control, are actually referenced in the default configuration. The default configuration works on any platform.

CoS configurations that specify more queues than the platform can support are not accepted. The commit fails with a detailed message that states the total number of queues available.

All default CoS configuration is based on queue number. The name of the forwarding class that shows up when the default configuration is displayed is the forwarding class currently associated with that queue.

This is the default configuration for forwarding-classes:

```
[edit class-of-service]
forwarding-classes {
  queue 0 best-effort;
  queue 1 expedited-forwarding;
  queue 2 assured-forwarding;
  queue 3 network-control;
}
```

If you reassign the forwarding-class names, the best-effort forwarding-class name appears in the locations in the configuration previously occupied by network-control as follows:

```
forwarding-classes {
  queue 0 network-control;
  queue 1 assured-forwarding;
  queue 2 expedited-forwarding;
  queue 3 best-effort;
}
```

All the default rules of classification and scheduling that applied to queue 3 still apply. Queue 3 is simply now renamed best-effort.

In the current default configuration:

Only IP precedence classifiers are associated with interfaces.

The only classes designated are best-effort and network-control.

Schedulers are not defined for the expedited-forwarding or assured-forwarding classes.

You must make a conscious effort to classify packets to the expedited-forwarding or assured-forwarding class and define schedulers for these classes.

Override Fabric Priority Queuing

For T-series platforms only, you can override automatic fabric priority queuing. For egress interfaces, fabric priority queuing matches the queue priority you assign at the [edit class-of-service schedulers *scheduler-name*] hierarchy level. High-priority egress traffic is automatically assigned to high-priority fabric queues. Likewise, low-priority egress traffic is automatically assigned to low-priority fabric queues.

You can override the default fabric priority queuing of egress traffic by including the priority statement at the [edit class-of-service forwarding-classes queue *queue-number class-name*] hierarchy level:

```
[edit class-of-service forwarding-classes queue queue-number class-name]
priority (low | high);
```

For information about associating a scheduler with a fabric priority, see “Associate a Scheduler with a Fabric Priority” on page 600.

Classify Packets by Behavior Aggregate

The simplest way to classify a packet is to use behavior aggregate classification. The DSCP or IP precedence bits of the IP header convey the behavior aggregate class information. The information might also be found in the MPLS EXP bits or IEEE 802.1p CoS bits.

Table 36 shows the default system classification scheme for the well-known DSCPs.

Table 36: Default Behavior Aggregate Classification

DSCP	Forwarding Class	PLP
ef	expedited-forwarding	low
af11	assured-forwarding	low
af12	assured forwarding	high
af13	assured forwarding	high
af21	best-effort	low
af22	best-effort	low
af23	best-effort	low
af31	best-effort	low
af32	best-effort	low
af33	best-effort	low
af41	best-effort	low
af42	best-effort	low
af43	best-effort	low
be	best-effort	low
cs1	best-effort	low
cs2	best-effort	low
cs3	best-effort	low
cs4	best-effort	low

DSCP	Forwarding Class	PLP
cs5	best-effort	low
nc1/cs6	network-control	low
nc2/cs7	network control	low
other	best-effort	low

All af classes other than af1X are mapped to best-effort, since RFC 2597 prohibits a node from aggregating classes. In effect, mapping to best-effort implies that the node does not support that class.

To define new classifiers for all code-point types, include the classifiers statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
classifiers {
  (dscp | exp | ieee-802.1 | inet-precedence) classifier-name {
    import [classifier-name | default];
    forwarding-class class-name {
      loss-priority (low | high) code-points [alias | bits];
    }
  }
}
```

A classifier takes a specified bit pattern as either the literal pattern or as a defined alias and attempts to match it to the type of packet arriving on the interface. If the information in the packet's header matches the specified pattern, the packet is sent to the appropriate queue, defined by the forwarding class associated with the classifier.

You can use any table, including the default, in the definition of a new classifier by including the import statement. The imported classifier is used as a template and is not modified. Whenever you commit a configuration that assigns a new *class-name* and loss-priority value to a code-point alias or set of bits, it replaces that entry in the imported classifier template. As a result, you must explicitly specify every code point in every designation that requires modification.



Note

If an interface is mounted on an M-series FPC, you can apply to the interface the default exp classifier only. If an interface is mounted on an enhanced FPC, you can define and apply to it a new exp classifier.

You can apply the classification map to a logical interface by including the classifiers statement in the following configuration:

```
[edit class-of-service]
interfaces {
  interface-name {
    unit logical-unit-number {
      classifiers (dscp | exp | ieee-802.1 | inet-precedence) (classifier-name | default);
    }
  }
}
```

You can use interface wildcards for *interface-name* and *logical-unit-number*.

The dscp classifier classifies all incoming IPv4 packets, while the exp classifier handles MPLS packet classification.



Note

You cannot mix L2 and L3 classification on an interface. For the purposes of this configuration, MPLS is considered L3 classification.

Configure Scheduling Policy Maps

You use scheduling policy maps to configure the forwarding classes that represent packet queues and associate them with physical interfaces.

A *scheduler* configuration block specifies the buffer size, bandwidth, and priority for a queue. It also specifies the RED drop profile for packets that fall within specification and out of specification. To configure schedulers, include the schedulers statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
schedulers {
  scheduler-name {
    buffer-size (percent percentage | remainder | temporal microseconds);
    drop-profile-map loss-priority (low | high) protocol (non-tcp | tcp | any)
      drop-profile profile-name;
    priority (low | high | strict-high);
    transmit-rate (rate | percent percentage | remainder | exact);
  }
}
```

Once you define a scheduler, you can include it in a *scheduler map* that is used to map a specified forwarding class to a scheduler configuration:

```
[edit class-of-service]
scheduler-maps {
  map-name {
    forwarding-class class-name scheduler scheduler-name;
  }
}
```

When you have defined the *map-name*, you can associate it with an output interface:

```
[edit class-of-service]
interfaces {
  interface-name {
    scheduler-map map-name;
  }
}
```

Interface wildcards are supported.

You must configure each forwarding class in turn. The following default scheduler is provided with the installation. These settings are not visible in the output of the show class-of-service command; rather, they are implicit.

```
[edit class-of-service]
schedulers {
  network-control {
    transmit-rate percent 5;
    buffer-size percent 5;
    priority low;
    drop-profile-map loss-priority any protocol any;
    drop-profile terminal;
  }
  best-effort {
    transmit-rate percent 95;
    buffer-size percent 95;
    priority low;
    drop-profile-map loss-priority any protocol any;
    drop-profile terminal;
  }
}
drop-profiles {
  terminal {
    fill-level 100 drop-probability 100;
  }
}
```

Associate a Scheduler with a DLCI or VLAN on a Channelized QPP Interface

For Channelized OC-12 QPP, Channelized STM-1 QPP, Channelized T3 QPP, Channelized E1 QPP, and Gigabit Ethernet QPP interfaces with Frame Relay or VLAN encapsulation, you can associate a scheduler map name with a logical interface. To activate transmission scheduling on a logical interface, include the per-unit-scheduler statement at the [edit interfaces *interface-name*] hierarchy level, and the scheduler-map statement at the [edit class-of-service interfaces *interface-name* unit *logical-unit-number*] hierarchy level:

```
[edit interfaces interface-name]
per-unit-scheduler;

[edit class-of-service interfaces interface-name unit logical-unit-number]
scheduler-map map-name;
```

For each port, the maximum number of scheduled DLCIs or VLANs is 383.

For channelized QPP interfaces, the number of schedulers you can apply varies by channel level. Table 37 shows the number of schedulers you can apply at each channel level.

Table 37: Scheduler Limitations for Channelized QPP Interfaces

Channelized PICs with QPP	Number of DLCIs per level
Channelized OC-12	63 for T3, OC-3, SONET, and Channelized OC-12 interfaces
Channelized T3	63 for T3 interfaces
Channelized STM-1	63 for STM-1 interfaces and 15 for E1 interfaces
Channelized E1	15 for E1 interfaces

**Note**

Logical QPP interface scheduling is not supported for Channelized T1 and NxDS-0 interfaces.

You can associate up to four forwarding classes per physical interface.

To specify the amount of bandwidth allocated to the logical interface, you must also include the bandwidth statement at the [edit class-of-service interfaces *interface-name* unit *logical-unit-number*] hierarchy level:

```
[edit class-of-service interfaces interface-name unit logical-unit-number]
bandwidth rate;
```

You can specify a peak bandwidth rate in bps, either as a complete decimal number or as a decimal number followed by the abbreviation k (1000), m (1,000,000), or g (1,000,000,000). The range is 1000 through 32,000,000,000 bps.

If you do not include the bandwidth statement in the configuration, the logical interface might not be able to transmit traffic unless surplus bandwidth is available on the physical interface. The sum of the bandwidth you allocate to all of the logical interfaces on a physical interface should not exceed the bandwidth of the physical interface.

Example: Associate a Scheduler with a Logical QPP Interface

Associate one scheduler with logical interface unit 0, and another with logical interface unit 1. The scheduler sched-map-logical-0 is associated with t3-1/0/0.0, and sched-map-logical-1 is associated with t3-1/0/0.1. The logical interface units 0 and 1 are allocated transmission bandwidths of 10 Mbit per second and 20 Mbit per second, respectively.

The allocated bandwidth is shared among the individual forwarding classes in the scheduler map. Although these schedulers are configured on a single physical interface, they are independent from each other. Traffic on one logical interface unit does not affect the transmission priority, bandwidth allocation, or drop behavior on the other logical interface unit.

```
[edit interfaces]
t3-1/0/0:1 {
  encapsulation frame-relay;
  per-unit-scheduler;
}
```

```

[edit class-of-service]
interfaces {
  t3-1/0/0:1 {
    unit 0 {
      dlc1 1022;
      scheduler-map sched-map-logical-0;
      bandwidth 10m;
    }
    unit 1 {
      dlc1 1021;
      scheduler-map sched-map-logical-1;
      bandwidth 20m;
    }
  }
}

scheduler-maps {
  sched-map-logical-0 {
    forwarding-class best-effort scheduler sched-best-effort-0;
    forwarding-class assured-forwarding scheduler sched-bronze-0;
    forwarding-class expedited-forwarding scheduler sched-silver-0;
    forwarding-class network-control scheduler sched-gold-0;
  }
  sched-map-logical-1 {
    forwarding-class best-effort scheduler sched-best-effort-1;
    forwarding-class assured-forwarding scheduler sched-bronze-1;
    forwarding-class expedited-forwarding scheduler sched-silver-1;
    forwarding-class network-control scheduler sched-gold-1;
  }
}

schedulers {
  sched-best-effort-0 {
    transmit-rate 4m;
  }
  sched-bronze-0 {
    transmit-rate 3m;
  }
  sched-silver-0 {
    transmit-rate 2m;
  }
  sched-gold-0 {
    transmit-rate 1m;
  }
  sched-best-effort-1 {
    transmit-rate 8m;
  }
  sched-bronze-1 {
    transmit-rate 6m;
  }
  sched-silver-1 {
    transmit-rate 4m;
  }
  sched-gold-1 {
    transmit-rate 2m;
  }
}

```

Associate a Scheduler with a Fabric Priority

On T-series platforms only, you can associate a scheduler with a class of traffic that has a specific priority while transiting the fabric. Traffic transiting the fabric can have two priority values: low or high. To associate a scheduler with a fabric priority, include the priority and scheduler statements at the [edit class-of-service fabric scheduler-map] hierarchy level:

```
[edit class-of-service fabric scheduler-map]
priority (low | high) scheduler scheduler-name;
```

For fabric CoS configuration, schedulers are restricted to transmit rates and drop profiles. Only the percent and remainder options are supported for transmit rates.

You might set up the following configuration:

```
[edit class-of-service]
schedulers {
  fab-be-scheduler {
    drop-profile-map loss-priority low protocol any drop-profile fab-be-profile;
    drop-profile-map loss-priority high protocol any drop-profile fab-be-profile;
  }
  fab-ef-scheduler {
    drop-profile-map loss-priority low protocol any drop-profile fab-ef-profile;
    drop-profile-map loss-priority high protocol any drop-profile fab-ef-profile;
  }
}
drop-profiles {
  fab-ef-profile {
    fill-level 100 drop-probability 100;
    fill-level 95 drop-probability 50;
  }
  fab-be-profile {
    fill-level 100 drop-probability 100;
    fill-level 85 drop-probability 50;
  }
}
fabric {
  scheduler-map {
    priority low scheduler fab-be-scheduler;
    priority high scheduler fab-ef-scheduler;
  }
}
```



Note

For a scheduler that you associate with a fabric priority, you cannot include the buffer-size, transmit-rate, and priority statements at the [edit class-of-service schedulers *scheduler-name*] hierarchy level.


For information about associating a forwarding class with a fabric priority, see “Override Fabric Priority Queuing” on page 594.

Configure RED Drop Profiles

RED drop profiles are associated with the forwarding classes and loss priorities from the scheduler-map you configured on the interface. To configure the drop profiles themselves, include the drop-profiles statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
drop-profiles {
  profile-name {
    fill-level percentage drop-probability percentage;
    interpolate {
      fill-level value;
      drop-probability value;
    }
  }
}
```

In this configuration, you include either the interpolate statement and its options, or the fill-level and drop-probability *percentage* values. These two alternatives enable you to configure either each drop probability at up to 64 fill-level/drop-probability paired values, or a profile represented as a series of line segments.

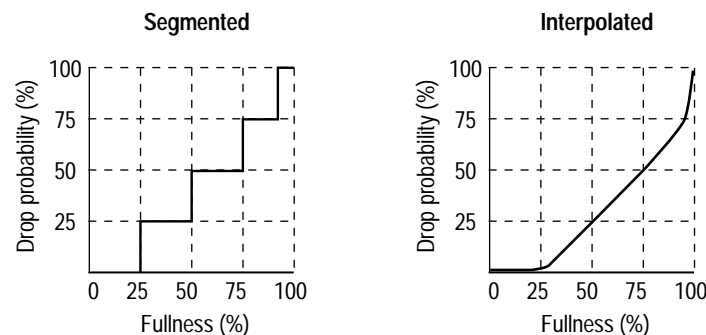


Note If you configure the interpolate statement, you can specify more than 64 pairs, but the system generates only 64 discrete entries.

The line segments are defined in terms of the following graphical model: in the first quadrant, the x axis represents the fill level and the y axis represents the drop probability. The initial line segment spans from the origin (0,0) to the point ($\langle l1 \rangle$, $\langle p1 \rangle$); a second line runs from ($\langle l1 \rangle$, $\langle p1 \rangle$) to ($\langle l2 \rangle$, $\langle p2 \rangle$) and so forth, until a final line segment connects (100, 100). The system automatically constructs a drop profile containing 64 fill levels at drop probabilities that approximate the calculated line segments.

Figure 31 shows sample line graphs comparing use of the segment percentages (on the left) and interpolated values (on the right):

Figure 31: Segmented and Interpolated Drop Profiles



1704

Packet Loss Priority

The system supports two packet loss priority (PLP) designations, low and high.

The packet loss priority is used to determine the RED drop profile when queuing a packet. You can set it by configuring a classifier or policer.

Rewrite Packet Header Information

You can rewrite the packet header bits because the logical interface transmits the packet along with the forwarding-class and PLP information associated with the packet. If the packet's header field is blank, the forwarding class and PLP information specified in the rule are entered, or if the forwarding class and PLP information are already present, they are replaced with the new values. The traffic is then classified accordingly. The rewrite-rules configurations define the mappings.

Table 38 shows the default mappings.

Table 38: Default Packet Header Rewrite Mappings

Map To DSCP/EXP/IEEE/IP	Map From Forwarding Class	PLP Value
ef	expedited-forwarding	low
ef	expedited-forwarding	high
af11	assured-forwarding	low
af12 (DSCP/EXP)	assured-forwarding	high
be	best-effort	low
be	best-effort	high
nc1/cs6	network-control	low
nc2/cs7	network-control	high

See “Define Code-Point Aliases” on page 589 for the default bit definitions of DSCP, EXP, and IEEE code points.

To configure a rewrite-rules mapping and associate it with the appropriate forwarding class and code-point alias or bit set, include the rewrite-rules statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
rewrite-rules {
  (dscp | exp | inet-precedence) rewrite-name {
    import (rewrite-name | default);
    forwarding-class class-name {
      loss-priority (low | high) code-point alias | bits;
    }
  }
}
```

To assign the rewrite-rules configuration to the output logical interface, include the following configuration:

```
[edit class-of-service]
interfaces {
  interface-name {
    unit logical-unit-number {
      rewrite-rules {
        dscp (rewrite-name | default);
        exp (rewrite-name | default) protocol protocol-types;
        exp-push-push-push default;
        exp-swap-push-push default;
        ieee-802.1 default;
        inet-precedence (rewrite-name | default);
      }
    }
  }
}
```

You can include interface wildcards for *interface-name* and *logical-unit-number*. You can also include Layer 2 and Layer 3 rewrite information in the same configuration.

The following sections explain several rewrite rule applications you can configure:

Rewrite EXP Bits on a Particular Node on page 603

Rewrite MPLS and IPv4 Packet Headers on page 605

Rewrite the EXP Bits of All Three Labels of an Outgoing Packet on page 607

Rewrite IEEE 802.1p Packet Headers with MPLS EXP Value on page 609

Rewrite EXP Bits on a Particular Node

To configure a custom table to rewrite the EXP bits, also known as CoS bits, on a particular node, the classifier table and the rewrite table must specify exactly the same code points.

In addition, the least significant bit of the code point itself must represent the PLP value. For example, code point 000 must be associated with PLP low, 001 must be associated with PLP high, and so forth.

Example: Rewrite EXP Bits on a Particular Node

Configure a custom table to rewrite the EXP bits on a particular node:

```
[edit class-of-service]
classifiers {
  exp exp-class {
    forwarding-class be {
      loss-priority low code-points 000;
      loss-priority high code-points 001;
    }
    forwarding-class af {
      loss-priority low code-points 010;
      loss-priority high code-points 011;
    }
    forwarding-class ef {
      loss-priority low code-points 100;
      loss-priority high code-points 101;
    }
    forwarding-class nc {
      loss-priority low code-points 110;
      loss-priority high code-points 111;
    }
  }
}
rewrite-rules {
  exp exp-rw {
    forwarding-class be {
      loss-priority low code-point 000;
      loss-priority high code-point 001;
    }
    forwarding-class af {
      loss-priority low code-point 010;
      loss-priority high code-point 011;
    }
    forwarding-class ef {
      loss-priority low code-point 100;
      loss-priority high code-point 101;
    }
    forwarding-class nc {
      loss-priority low code-point 110;
      loss-priority high code-point 111;
    }
  }
}
```

Rewrite MPLS and IPv4 Packet Headers

You can apply a rewrite rule to MPLS and IPv4 packet headers simultaneously. This allows you to initialize MPLS EXP and IP precedence bits at LSP ingress. You can configure different rewrite rules depending on whether the traffic is VPN or non-VPN.

The default MPLS EXP rewrite table contents are shown in Table 39.

Table 39: Default MPLS EXP Rewrite Table

Forwarding Class	Loss Priority	Code Point
best-effort	low	000
best-effort	high	001
expedited-forwarding	low	010
expedited-forwarding	high	011
assured-forwarding	low	100
assured-forwarding	high	101
network-control	low	110
network-control	high	111

To override the default MPLS EXP rewrite table and rewrite MPLS and IPv4 packet headers simultaneously, include the protocol statement at the [edit class-of-service interfaces *interface-name* unit *logical-unit-number* rewrite-rules exp *rewrite-rule-name*] hierarchy level:

```
[edit class-of-service interfaces interface-name unit logical-unit-number rewrite-rules exp
rewrite-rule-name]
protocol types;
```

The protocol statement defines the types of MPLS packets and packet headers to which the specified rewrite rule is applied. The MPLS packet can be a standard MPLS packet or an MPLS packet with an IPv4 payload. Specify the type of MPLS packet using the following options:

mpls-any—Applies the rewrite rule to MPLS packets and writes the code point value to MPLS headers.

mpls-inet-both—Applies the rewrite rule to VPN MPLS packets with IPv4 payloads. On T-series platforms, writes the code point value to the MPLS and IPv4 headers. On M-series routers, causes all ingress MPLS LSP packets with IPv4 payloads to be initialized with 000 code points for IP precedence and MPLS EXP values.

mpls-inet-both-non-vpn—Applies the rewrite rule to non-VPN MPLS packets with IPv4 payloads. On T-series platforms, writes the code point value to the MPLS and IPv4 headers. On M-series routers, causes all ingress MPLS LSP packets with IPv4 payloads to be initialized with 000 code points for IP precedence and MPLS EXP values.

An alternative to overwriting the default with a rewrite-rules mapping is to configure the default packet header rewrite mappings, as shown in Table 38 on page 602.

Example: Rewrite MPLS and IPv4 Packet Headers

On a T-series platform, configure rewrite tables and apply them in various ways to achieve the following results:

For interface so-3/1/0, the three EXP rewrite tables are applied to packets, depending on the protocol of the payload:

IPv4 packets (VPN) that enter the LSPs on interface so-3/1/0 are initialized with values from rewrite table exp-inet-table. An identical three-bit value is written into the IP precedence and MPLS EXP bit fields.

IPv4 packets (non-VPN) that enter the LSPs on interface so-3/1/0 are initialized with values from rewrite table rule-non-vpn. An identical three-bit value is written into the IP precedence and MPLS EXP bit fields.

Non-IPv4 packets that enter the LSPs on interface so-3/1/0 are initialized with values from rewrite table rule1, and written into the MPLS EXP header field only. The statement exp rule1 has the same result as exp rule1 protocol mpls-any.

For interface so-3/1/0, IPv4 packets transmitted over a non-LSP layer are initialized with values from IP precedence rewrite table rule2.

For interface so-3/1/1, IPv4 packets that enter the LSPs are initialized with values from EXP rewrite table exp-inet-table. An identical 3-bit value is written into the IP precedence and MPLS EXP bit fields.

For interface so-3/1/1, MPLS packets other than IPv4 Layer 3 types are also initialized with values from table exp-inet-table. For VPN MPLS packets with IPv4 payloads, the code point value is written to MPLS and IPv4 headers. For VPN MPLS packets without IPv4 payloads, the code point value is written to MPLS headers only.

```
[edit class-of-service]
rewrite-rules {
  exp exp-inet-table {
    forwarding-class best-effort {
      loss-priority low code-point 000;
      loss-priority high code-point 001;
    }
    forwarding-class assured-forwarding {
      loss-priority low code-point 010;
      loss-priority high code-point 011;
    }
    forwarding-class expedited-forwarding {
      loss-priority low code-point 111;
      loss-priority high code-point 110;
    }
    forwarding-class network-control {
      loss-priority low code-point 100;
      loss-priority high code-point 101;
    }
  }
  exp rule1 {
    ...
  }
}
```

```

inet-precedence rule2 {
    ...
}
}
exp rule_non_vpn {
    ...
}

interfaces {
  so-3/1/0 {
    unit 0 {
      rewrite-rules {
        exp rule1;
        inet-precedence rule2;
        exp exp-inet-table protocol mpls-inet-both;# for all VPN traffic
        exp rule_non_vpn protocol mpls-inet-both-non-vpn;# for all non-VPN traffic
      }
    }
  }
  so-3/1/1 {
    unit 0 {
      rewrite-rules {
        exp exp-inet-table protocol [mpls-any mpls-inet-both];
      }
    }
  }
}

```

Rewrite the EXP Bits of All Three Labels of an Outgoing Packet

In interprovider, carrier-of-carrier, and complex traffic engineering scenarios, it is sometimes necessary to push three labels on the next hop, using a swap-push-push or triple-push operation.

By default, on M-series routers, the top MPLS EXP label of an outgoing packet is not rewritten when you configure swap-push-push and triple-push operations. On M-series routers, you can rewrite the EXP bits of all three labels of an outgoing packet, thereby maintaining CoS of an incoming MPLS or non-MPLS packet.

To do this on incoming MPLS packets, include the `exp-swap-push-push` default statement at the [edit class-of-service interfaces *interface-name* unit *logical-unit-number* rewrite-rules] hierarchy level:

```
[edit class-of-service interfaces interface-name unit logical-unit-number rewrite-rules]
exp-swap-push-push default;
```

To do this on incoming non-MPLS packets, include the `exp-push-push-push` default statement at the [edit class-of-service interfaces *interface-name* unit *logical-unit-number* rewrite-rules] hierarchy level:

```
[edit class-of-service interfaces interface-name unit logical-unit-number rewrite-rules]
exp-push-push-push default;
```

These configurations apply the default MPLS EXP rewrite table, as shown in Table 39 on page 605. You can configure these operations and also override the default MPLS EXP rewrite table with a custom table. For more information about writing and applying a custom rewrite table, see “Rewrite Packet Header Information” on page 602.

Example: Rewrite the EXP Bits of All Three Labels of an Outgoing Packet

Configure a swap-push-push operation, and override the default rewrite table with a custom table:

```
[edit class-of-service]
forwarding-classes {
  queue 0 be;
  queue 1 ef;
  queue 2 af;
  queue 3 nc;
}
interfaces {
  so-1/1/3 {
    unit 0 {
      rewrite-rules {
        exp exp_rew;    #Apply custom rewrite table
        exp-swap-push-push default;
      }
    }
  }
}
rewrite-rules {
  exp exp_rew {
    forwarding-class be {
      loss-priority low code-point 000;
      loss-priority high code-point 100;
    }
    forwarding-class ef {
      loss-priority low code-point 001;
      loss-priority high code-point 101;
    }
    forwarding-class af {
      loss-priority low code-point 010;
      loss-priority high code-point 110;
    }
    forwarding-class nc {
      loss-priority low code-point 011;
      loss-priority high code-point 111;
    }
  }
}
```

Rewrite IEEE 802.1p Packet Headers with MPLS EXP Value

For Ethernet interfaces installed on a T-series platform with a peer connection to an M-series router or a T-series platform, you can rewrite both MPLS EXP and IEEE 802.1p bits to a configured value. This allows you to pass the configured value to the Layer 2 VLAN path.

To rewrite both the MPLS EXP and IEEE 802.1p bits, you must include EXP and IEEE 802.1p rewrite rules in the interface configuration. To configure EXP and IEEE 802.1p rewrite rules, include the rewrite-rules statement at the [edit class-of-service interfaces *interface-name* unit *logical-unit-number*] hierarchy level, specifying the exp and ieee-802.1 options:

```
[edit class-of-service interfaces interface-name unit logical-unit-number]
rewrite-rules {
  exp rewrite-rule-name;
  ieee-802.1 default;
}
```

When you combine these two rewrite rules, only the EXP rewrite table is used for rewriting packet headers. If you do not configure a VLAN on the interface, only the EXP rewriting is in effect. If you do not configure an LSP on the interface or if the MPLS EXP rewrite rule mapping is removed, the IEEE 802.1p default rewrite rules mapping takes effect.



Note

You can also combine other rewrite rules. IP, DSCP, and MPLS EXP are associated with Layer 3 packet headers, and IEEE 802.1p is associated with the Layer 2 packet header.

If you combine IEEE 802.1p and IP rewrite rules, the Layer 3 packets and Layer 2 headers are rewritten with the IP rewrite rule.

If you combine IEEE 802.1p and DSCP rewrite rules, three bits of the Layer 2 header and six bits of the Layer 3 packet header are rewritten with the DSCP rewrite rule.

The following example shows how to configure an EXP rewrite rule and apply it to both MPLS EXP and IEEE 802.1p bits:

```
[edit class-of-service]
rewrite-rules {
  exp exp-ieee-table {
    forwarding-class best-effort {
      loss-priority low code-point 000;
      loss-priority high code-point 001;
    }
    forwarding-class assured-forwarding {
      loss-priority low code-point 010;
      loss-priority high code-point 011;
    }
    forwarding-class expedited-forwarding {
      loss-priority low code-point 111;
      loss-priority high code-point 110;
    }
  }
}
```

```

        forwarding-class network-control {
            loss-priority low code-point 100;
            loss-priority high code-point 101;
        }
    }
}
interfaces {
    so-3/1/0 {
        unit 0 {
            rewrite-rules {
                exp exp-ieee-table;
                ieee-802.1 default;
            }
        }
    }
}
}

```

Configure CoS-Based Forwarding

CoS-based forwarding (CBF) enables you to control next-hop selection based on a packet's class of service and, in particular, the value of the IP packet's precedence bits.

For example, you might want to specify a particular interface or next hop to carry high-priority traffic while all best-effort traffic takes some other path. When a routing protocol discovers equal cost paths, it can pick a path at random or load-share across the paths either through hash selection or round robin. CBF allows path selection based on class.

You can apply CBF only to a defined set of routes. Therefore you must configure a policy statement as in the following example:

```

[edit]
policy-options {
    policy-statement my-cos-forwarding {
        from {
            route-filter filter-name;
        }
        then {
            cos-next-hop-map map-name;
        }
    }
}

```

This configuration specifies that routes matching the route filter will be subject to the CoS next-hop mapping specified by *map-name*. For more information about configuring policy statements, see the *JUNOS Internet Software Configuration Guide: Policy Framework*.

To specify a CoS next-hop map, include the forwarding-policy statement at the [edit class-of-service] hierarchy level:

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

When you configure CBF with OSPF as the IGP, you must specify the next hop as an interface name or next-hop alias, not as an IP address. For an example configuration, see “Example: Configure CoS-Based Forwarding” on page 612.

You cannot configure packet classification for IPv6. Only IP precedence classifiers are supported. For an example configuration, see “Configure CoS-Based Forwarding for IPv6” on page 615.

The JUNOS software applies the CoS next-hop map to the set of next hops previously defined; the next hops themselves can be located across any outgoing interfaces on the router. For example, the following configuration associates a set of forwarding classes and next-hop identifiers:

```
[edit class-of-service forwarding-policy]
next-hop-map map1 {
  forwarding-class expedited-forwarding {
    next-hop next-hop1;
    next-hop next-hop2;
  }
  forwarding-class best-effort {
    next-hop next-hop3;
    lsp-next-hop lsp-next-hop4;
  }
}
```

In this example, next-hop N is either an IP address or an egress interface for some next hop, and lsp-next-hop4 is a regular expression corresponding to any next hop with that label. Q1 through Q N are a set of forwarding classes that map to the specific next hop. That is, when a packet is switched with Q1 through Q N , it will be forwarded out the interface associated with the associated next hop.

This configuration has the following implications:

A single forwarding class can map to multiple standard next hops or LSP next hops. This implies that load sharing is done across standard next hops or LSP next hops servicing the same class value. To make this work properly, the JUNOS software creates a list of the equal-cost next hops and forwards packets according to standard load-sharing rules for that forwarding class.

If a forwarding class configuration includes LSP next hops and standard next hops, the LSP next hops are preferred over the standard next hops. In the preceding example, if both next-hop3 and lsp-next-hop4 are valid next hops for a route to which map1 is applied, the forwarding table includes entry lsp-next-hop4 only.

If next-hop-map does not specify all possible forwarding classes, the default forwarding class is selected as the default. If the default forwarding class is not specified in the next-hop map, a default is designated randomly. The default forwarding class is the class associated with queue 0.

For LSP next hops, the JUNOS software uses UNIX regex(3)-style regular expressions. For example, if the following labels exist: lsp, lsp1, lsp2, lsp3, the statement lsp-next-hop lsp matches lsp, lsp1, lsp2, and lsp3. If you do not desire this behavior, you must use the anchor characters lsp-next-hop "^lsp\$", which match lsp only.

The final step is to apply the route filter to routes exported to the forwarding engine. This is shown in the following example:

```
routing-options {
  forwarding-table {
    export my-cos-forwarding;
  }
}
```

This configuration instructs the routing process to insert routes to the forwarding engine matching my-cos-forwarding with the associated next-hop CBF rules.

The following algorithm is used when you apply a configuration to a route:

If the route is a single next-hop route, all traffic will go to that route; that is, no CBF will take effect.

For each next hop, associate the proper forwarding class. If a next hop appears in the route but not in the cos-next-hop map, it will not appear in the forwarding table entry.

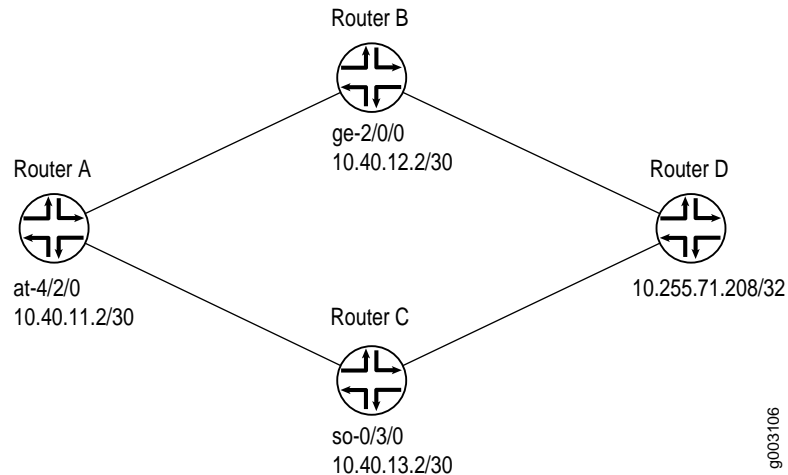
The default forwarding class is used if all forwarding classes are not specified in the next-hop map. If the default is not specified, one is chosen randomly.

Example: Configure CoS-Based Forwarding

Router A has two routes to destination 10.255.71.208 on Router D. One route goes through Router B, and the other goes through Router C, as shown in Figure 32.

Configure Router A with CBF to select Router B for queue 0 and queue 2, and Router C for queue 1 and queue 3.

Figure 32: Sample CoS-Based Forwarding



When you configure CBF with OSPF as the IGP, you must specify the next hop as an interface name, not as an IP address. The next hops in this example are specified as ge-2/0/0.0 and so-0/3/0.0.

```

[edit class-of-service]
forwarding-policy {
  next-hop-map my_cbf {
    forwarding-class be {
      next-hop ge-2/0/0.0;
    }
    forwarding-class ef {
      next-hop so-0/3/0.0;
    }
    forwarding-class af {
      next-hop ge-2/0/0.0;
    }
    forwarding-class nc {
      next-hop so-0/3/0.0;
    }
  }
}
classifiers {
  inet-precedence inet {
    forwarding-class be {
      loss-priority low code-points [ 000 100 ];
    }
    forwarding-class ef {
      loss-priority low code-points [ 001 101 ];
    }
    forwarding-class af {
      loss-priority low code-points [ 010 110 ];
    }
    forwarding-class nc {
      loss-priority low code-points [ 011 111 ];
    }
  }
}
}

```

```

forwarding-classes {
  queue 0 be;
  queue 1 ef;
  queue 2 af;
  queue 3 nc;
}
interfaces {
  at-4/2/0 {
    unit 0 {
      classifiers {
        inet-precedence inet;
      }
    }
  }
}

[edit policy-options]
policy-statement cbf {
  from {
    route-filter 10.255.71.208/32 exact;
  }
  then cos-next-hop-map my_cbf;
}

[edit routing-options]
graceful-restart;
forwarding-table {
  export cbf;
}

[edit interfaces]
traceoptions {
  file trace-intf size 5m world-readable;
  flag all;
}
so-0/3/0 {
  unit 0 {
    family inet {
      address 10.40.13.1/30;
    }
    family iso;
    family mpls;
  }
}
ge-2/0/0 {
  unit 0 {
    family inet {
      address 10.40.12.1/30;
    }
    family iso;
    family mpls;
  }
}
at-4/2/0 {
  atm-options {
    vpi 1 {
      maximum-vcs 1200;
    }
  }
}

```

```

unit 0 {
  vci 1.100;
  family inet {
    address 10.40.11.2/30;
  }
  family iso;
  family mpls;
}
}

```

Configure CoS-Based Forwarding for IPv6 Configure CBF next-hop maps and CBF LSP next-hop maps for IPv6 addresses. The following example shows a CBF next-hop map for IPv6 addresses.

You can configure a next-hop map with both IPv4 and IPv6 addresses, or you can configure separate next-hop maps for IPv4 and IPv6 addresses and include the `from family (inet | inet6)` statements at the `[edit policy-options policy-options policy-statement policy-name term term-name]` hierarchy level to ensure that only next-hop maps of a specified protocol are applied to a specified route.

If you do not configure separate next-hop maps and include the `from family (inet | inet6)` statements in the configuration, when a route uses two next hops (whether IPv4, IPv6, interface, or LSP next hop) in at least two of the specified forwarding classes, CBF is used for the route; otherwise, the CBF policy is ignored.

1. Define the CBF next-hop map:

```

[edit class-of-service]
forwarding-policy {
  next-hop-map cbf-map {
    forwarding-class best-effort {
      next-hop [ ::192.168.139.38 192.168.139.38 ];
    }
    forwarding-class expedited-forwarding {
      next-hop [ ::192.168.140.5 192.168.140.5 ];
    }
    forwarding-class assured-forwarding {
      next-hop [ ::192.168.145.5 192.168.145.5 ];
    }
    forwarding-class network-control {
      next-hop [ ::192.168.141.2 192.168.141.2 ];
    }
  }
}

```

2. Define the CBF forwarding policy:

```

[edit policy-options]
policy-statement ls {
  then cos-next-hop-map cbf-map;
}

```

3. Export the CBF forwarding policy:

```

[edit routing-options]
forwarding-table {
  export ls;
}

```

Override the Input Classification

For IPv4 or IPv6 packets, you can override the incoming classification, assigning them to the same forwarding class based on their input interface, input precedence bits, or destination address. You do so by defining a policy class when configuring CoS properties and referencing this class when configuring a routing policy.

When you override the classification of incoming packets, any mappings you configured for associated precedence bits or incoming interfaces to output transmission queues are ignored. Also, if the packet loss priority (PLP) bit was set in the packet by the incoming interface, the PLP bit is cleared.

To override the input packet classification, do the following:

1. Define the policy class by including the class statement at the [edit class-of-service policy] hierarchy level:

```
[edit class-of-service]
forwarding-policy {
  class class-name {
    classification-override {
      forwarding-class class-name;
    }
  }
}
```

class-name is a name that identifies the class.

2. Associate the policy class with a routing policy by including it in a policy-statement statement at the [edit policy-options] hierarchy level. Specify the destination prefixes in the route-filter statement and the CoS policy class name in the then statement.

```
[edit policy-options]
policy-statement policy-name {
  term term-name {
    from {
      route-filter destination-prefix match-type <class class-name>;
    }
    then class class-name;
  }
}
```

3. Apply the policy by including the export statement at the [edit routing-option] hierarchy level:

```
[edit routing-options]
forwarding-table {
  export policy-name;
}
```

Example: Configure Class of Service

The following example includes classifiers, rewrite markers, and schedulers to configure a class of service policy.

1. Define a classifier that matches IP traffic arriving on the interface. The affected IP traffic has IP precedence bits with patterns matching those defined by aliases A or B. The loss priority of the matching packets is set to low, and the forwarding class is mapped to best effort (queue 0):

```
[edit]
class-of-service {
  classifiers {
    inet-precedence normal-traffic {
      forwarding-class best-effort {
        loss-priority low code-points [my1 my2];
      }
    }
  }
}
```

Following are the code-point alias and forwarding-class mappings referenced in the normal-traffic classifier:

```
[edit]
class-of-service {
  code-point-aliases {
    inet-precedence {
      my1 000;
      my2 001;
      ...
    }
  }
}

[edit]
class-of-service {
  forwarding-classes {
    queue 0 best-effort;
    queue 1 expedited-forwarding;
  }
}
```

2. Use rewrite markers to redefine the bit pattern of outgoing packets. Assign the new bit pattern based on specified forwarding classes, regardless of the loss priority of the packets:

```
[edit]
class-of-service {
  rewrite-rules {
    inet-precedence clear-prec {
      forwarding-class best-effort {
        loss-priority low code-point 000;
        loss-priority high code-point 000;
      }
      forwarding-class expedited-forwarding {
        loss-priority low code-point 100;
        loss-priority high code-point 100;
      }
    }
  }
}
```

3. Configure a scheduler map associating forwarding classes with schedulers and drop-profiles:

```
[edit]
class-of-service {
  scheduler-maps {
    one {
      forwarding-class expedited-forwarding scheduler special;
      forwarding-class best-effort scheduler normal;
    }
  }
}
```

Schedulers establish how to handle the traffic within the output queue for transmission onto the wire. Following is the scheduler referenced in scheduler map one:

```
[edit]
class-of-service {
  schedulers {
    special {
      transmit-rate percent 30;
      priority high;
    }
    normal {
      transmit-rate percent 70;
      priority low;
    }
  }
}
```

4. Apply the normal-traffic classifier to all SONET/SDH interfaces and all logical interfaces of SONET/SDH interfaces; apply the clear-prec rewrite marker to all Gigabit Ethernet interfaces and all logical interfaces of Gigabit Ethernet interfaces; and apply the one scheduler map to all interfaces:

```
[edit]
class-of-service {
  interfaces {
    so-0/0/0 {
      scheduler-map one;
      unit 0 {
        classifiers {
          inet-precedence normal-traffic;
        }
      }
    }
    so-0/0/1 {
      scheduler-map one;
      unit 1 {
        classifiers {
          inet-precedence normal-traffic;
        }
      }
    }
  }
  ge-1/0/0 {
    scheduler-map one;
    unit 0 {
      rewrite-rules {
        inet-precedence clear-prec;
      }
    }
    unit 1 {
      rewrite-rules {
        inet-precedence clear-prec;
      }
    }
  }
  ge-1/0/1 {
    scheduler-map one;
    unit 0 {
      rewrite-rules {
        inet-precedence clear-prec;
      }
    }
    unit 1 {
      rewrite-rules {
        inet-precedence clear-prec;
      }
    }
  }
}
```

Following is the complete configuration:

```
[edit class-of-service]
classifiers {
  inet-precedence normal-traffic {
    forwarding-class best-effort {
      loss-priority low code-points [my1 my2];
    }
  }
}
code-point-aliases {
  inet-precedence {
    my1 000;
    my2 001;
    cs1 010;
    cs2 011;
    cs3 100;
    cs4 101;
    cs5 111;
    cs6 111;
  }
}
drop-profiles {
  high-priority {
    fill-level 20 drop-probability 100;
  }
  low-priority {
    fill-level 90 drop-probability 95;
  }
  big-queue {
    fill-level 100 drop-probability 100;
  }
}
forwarding-classes {
  queue 0 best-effort;
  queue 1 expedited-forwarding;
}
interfaces {
  so-0/0/0 {
    scheduler-map one;
    unit 0 {
      classifiers {
        inet-precedence normal-traffic;
      }
    }
  }
  so-0/0/1 {
    scheduler-map one;
    unit 1 {
      classifiers {
        inet-precedence normal-traffic;
      }
    }
  }
}
```

```

ge-1/0/0 {
  scheduler-map one;
  unit 0 {
    rewrite-rules {
      inet-precedence clear-prec;
    }
  }
  unit 1 {
    rewrite-rules {
      inet-precedence clear-prec;
    }
  }
}
ge-1/0/1 {
  scheduler-map one;
  unit 0 {
    rewrite-rules {
      inet-precedence clear-prec;
    }
  }
  unit 1 {
    rewrite-rules {
      inet-precedence clear-prec;
    }
  }
}
}
rewrite-rules {
  inet-precedence clear-prec {
    forwarding-class best-effort {
      loss-priority low code-point 000;
      loss-priority high code-point 000;
    }
    forwarding-class expedited-forwarding {
      loss-priority low code-point 100;
      loss-priority high code-point 100;
    }
  }
}
}
scheduler-maps {
  one {
    forwarding-class expedited-forwarding scheduler special;
    forwarding-class best-effort scheduler normal;
  }
}
schedulers {
  special {
    transmit-rate percent 30;
    priority high;
  }
  normal {
    transmit-rate percent 70;
    priority low;
  }
}
}

```

