Configuring Modular QoS Congestion Management

Congestion management controls congestion after it has occurred on a network. Congestion is managed on Cisco IOS XR software by using packet queuing methods and by shaping the packet flow through use of traffic regulation mechanisms.

The types of traffic regulation mechanisms supported are:

- Traffic shaping:
  - Modified Deficit Round Robin (MDRR)
  - Low-latency queuing (LLQ) with strict priority queuing (PQ)
- Traffic policing:
  - Color blind
  - Color-aware (ingress direction)

Line Card, SIP, and SPA Support

This table lists the features that are supported on the ASR 9000 Ethernet Line Cards and SIP 700 for the ASR 9000.

Feature	ASR 9000 Ethernet Line Cards	SIP 700 for the ASR 9000
Congestion Management Using DEI	yes	yes
Guaranteed and Remaining Bandwidth	yes	yes
Low-Latency Queueing with Strict Priority Queueing	yes	yes
Traffic Policing	yes	yes
Traffic Shaping	yes	yes

Note Ingress queueing is not supported on the A9K-24X10GE-1G-SE line card.

Feature History for Configuring Modular QoS Congestion Management on Cisco ASR 9000 Series Router

Release	Modification
Release 3.7.2	The Congestion Avoidance feature was introduced on ASR 9000 Ethernet Line Cards.. The Guaranteed and Remaining Bandwidth, Low-Latency Queueing with Strict Priority Queueing, Traffic Policing, and Traffic Shaping features were introduced on ASR 9000 Ethernet Line Cards.
Release 3.9.0	The Guaranteed and Remaining Bandwidth, Low-Latency Queueing with Strict Priority Queueing, Traffic Policing, and Traffic Shaping features were supported on the SIP 700 for the ASR 9000.
Release 4.0.0	The Congestion Management Using DEI feature was introduced on ASR 9000 Ethernet Line Cards.
Release 4.0.1	The **police rate** command was updated to include packet-based specifications of policing rates and burst sizes.
Release 4.1.0	The 2-rate 3-color policer feature was added, including the **conform-color** and **exceed-color** commands. This feature is applicable to the SIP 700 line cards, ingress side.
Release 4.2.1	The Configured Accounting and QoS for IPv6ACLs features were added.
Release 5.3.2	The existing egress priority levels are enhanced from P1, P2 and P3 to P1, P2, P3, P4, P5, P6 and P7. For all the releases prior to 5.3.2, system supports Priority levels P1, P2 and P3 only.
Release 6.0.1	Traffic Policing on Layer 2 ATM Interfaces

Prerequisites for Configuring QoS Congestion Management

These prerequisites are required for configuring QoS congestion management on your network:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- You must be familiar with Cisco IOS XR QoS configuration tasks and concepts.

Information About Configuring Congestion Management

Congestion Management Overview

Congestion management features allow you to control congestion by determining the order in which a traffic flow (or packets) is sent out an interface based on priorities assigned to packets. Congestion management entails the creation of queues, assignment of packets to those queues based on the classification of the packet, and scheduling of the packets in a queue for transmission. The congestion management features in Cisco IOS XR software allow you to specify creation of a different number of queues, affording greater or lesser degree of differentiation of traffic, and to specify the order in which that traffic is sent.

During periods with light traffic flow, that is, when no congestion exists, packets are sent out the interface as soon as they arrive. During periods of transmit congestion at the outgoing interface, packets arrive faster than the interface can send them. If you use congestion management features, packets accumulating at an interface are queued until the interface is free to send them; they are then scheduled for transmission according to their assigned priority and the queuing method configured for the interface. The router determines the order of packet transmission by controlling which packets are placed in which queue and how queues are serviced with respect to each other.

In addition to queuing methods, QoS congestion management mechanisms, such as policers and shapers, are needed to ensure that a packet adheres to a contract and service. Both policing and shaping mechanisms use the traffic descriptor for a packet.

Policers and shapers usually identify traffic descriptor violations in an identical manner through the token bucket mechanism, but they differ in the way they respond to violations. A policer typically drops traffic flow; whereas, a shaper delays excess traffic flow using a buffer, or queuing mechanism, to hold the traffic for transmission at a later time.

Traffic shaping and policing can work in tandem. For example, a good traffic shaping scheme should make it easy for nodes inside the network to detect abnormal flows.

Modified Deficit Round Robin

MDRR is a class-based composite scheduling mechanism that allows for queueing of up to eight traffic classes. It operates in the same manner as class-based weighted fair queueing (CBWFQ) and allows definition of traffic classes based on customer match criteria (such as access lists); however, MDRR does not use the weighted fair queuing algorithm.

When MDRR is configured in the queuing strategy, nonempty queues are served one after the other. Each time a queue is served, a fixed amount of data is dequeued. The algorithm then services the next queue. When a queue is served, MDDR keeps track of the number of bytes of data that were dequeued in excess of the configured value. In the next pass, when the queue is served again, less data is dequeued to compensate for the excess data that was served previously. As a result, the average amount of data dequeued per queue is close to the configured value. In addition, MDRR allows for a strict priority queue for delay-sensitive traffic.

Each queue within MDRR is defined by two variables:

- Quantum value—Average number of bytes served in each round.
- Deficit counter—Number of bytes a queue has sent in each round. The counter is initialized to the quantum value.

Packets in a queue are served as long as the deficit counter is greater than zero. Each packet served decreases the deficit counter by a value equal to its length in bytes. A queue can no longer be served after the deficit counter becomes zero or negative. In each new round, the deficit counter for each nonempty queue is incremented by its quantum value.

Low-Latency Queueing with Strict Priority Queueing

The LLQ feature brings strict priority queuing (PQ) to the MDRR scheduling mechanism. PQ in strict priority mode ensures that one type of traffic is sent, possibly at the expense of all others. For PQ, a low-priority queue can be detrimentally affected, and, in the worst case, never allowed to send its packets if a limited amount of bandwidth is available or the transmission rate of critical traffic is high.

Strict PQ allows delay-sensitive data, such as voice, to be dequeued and sent before packets in other queues are dequeued.

LLQ enables the use of a single, strict priority queue within MDRR at the class level, allowing you to direct traffic belonging to a class. To rank class traffic to the strict priority queue, you specify the named class within a policy map and then configure the **priority** command for the class. (Classes to which the **priority** command is applied are considered priority classes.) Within a policy map, you can give one or more classes priority status. When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is enqueued to the same, single, strict priority queue.

Through use of the **priority** command, you can assign a strict PQ to any of the valid match criteria used to specify traffic. These methods of specifying traffic for a class include matching on access lists, protocols, IP precedence, and IP differentiated service code point (DSCP) values. Moreover, within an access list you can specify that traffic matches are allowed based on the DSCP value that is set using the first six bits of the IP type of service (ToS) byte in the IP header.

Overhead Accounting

Traffic shapers and policers use packet traffic descriptors to ensure adherence to the service level agreement in QoS. However, when traffic flows from one hop to another in a network, headers added or removed at interim hops affect the packet bytes being accounted for by QoS at each hop. When your end-user network measures the packet bytes to ensure they receive the payload as agreed, these additional header bytes cause a discrepancy.

QoS overhead accounting provides the flexibility to operators to decide which header bytes can be excluded by the traffic shaper and policer and which can be included, depending on the end user’s requirements and device capabilities, to meet the committed payload in units of bytes.

For example, if the QoS commitment includes the additional header bytes, the overhead accounting feature allows your router to account for this overhead and reduces the traffic policing and shaping rates accordingly. This is also called a **positive accounting overhead**.

If however, the committed rate doesn’t include the additional bytes, overhead accounting allows your router to adjust the core stream traffic such that the traffic policing and shaping rates are increased. This is also called a **negative accounting overhead**.

To summarize, QoS overhead accounting enables the router to account for packet overhead when shaping and policing traffic to a specific rate. This accounting ensures that the router runs QoS features on the actual bandwidth that the subscriber traffic consumes.

Any interface that supports QoS policies supports overhead accounting.

Note You can enable user overhead accounting using the optional configuration of **accounting user-defined <*overhead size in bytes*>** while attaching the service policy on the egress interface.

Prerequisites and Restrictions

- Overhead accounting for ingress shaping is not supported.
- Overhead accounting is not reflected in any QoS counters (classification, policing, or queuing). In other words, when you run show policy-map statistics, the results do not include the overhead bytes but display the actual packet sizes.
- Dynamic changing of accounting overhead after application of the policy on the interface is not supported.
- You must remove the service policy from the interface and apply it back with the required overhead value. You can, however, remove and reapply the service policy in a single configuration commit.

Configuring for Overhead Accounting

To configure overhead accounting, you must:

1. Create a policy map and configure QoS actions for that map.
2. Configure overhead accounting and attach the map to an interface.

```
/* create QoS policy */
Router#configure terminal
Router(config)#policy-map policer
Router(config-pmap)#class class-default
Router(config-pmap-c)#police rate percent 10
Router(config-pmap-c-police)#commit

/* configure account overhead value while attaching the QoS policy to interface */
Router(config)#int hundredGigE 0/0/0/2
Router(config-if)#service-policy input policer account user-defined 12
Router(config-if)#commit
Router(config-if)#root
Router(config)#end
```

Running Configuration

```
Router#sh run int hundredGigE 0/0/0/2
interface HundredGigE0/0/0/2
service-policy input policer account user-defined 12
!
```

Verification

```
Router#sh qos-ea int hundredGigE 0/0/0/2 input detail
Interface: HundredGigE0_0_0_2 input policy: policer
Total number of classes:   1
Total number of UBRL classes:  0
Total number of CAC classes:   0
------------------------------------------------------
Policy name: policer
Hierarchical depth 1
Interface type HundredGigE
Interface rate 100000000 kbps
Port Shaper rate 0 kbps
Interface handle 0x00000680
```

```
ul_ifh 0x00000000, ul_id  0x00000080
uidb index 0x0003
qos_ifh 0x10200020800003
Local port 16, NP 0
Policy map id 0x6018, format 8, uidb index 0x0003
------------------------------------------------------
Index 0 Level 0 Class name class-default service_id 0x0 Policy name policer
Node flags: LEAF DEFAULT DEFAULT-ALL
 Stats flags: Policer type 1 Max category 0
 Node Config:
 Police Color aware 0 Type 1 CIR/CBS/PIR/PBS: 10000000kbps/125000000B/0kbps/0B
 Node Result: Class-based stats:Stat ID 0x00000310
 Queue: N/A Stat ID(Commit): 0x00000000
 Stat ID(Drop Curve 0): 0x00000000
 Stat ID(Drop Curve 1): 0x00000000
 Stat ID(Drop Curve 2): 0x00000000
 Stat ID(Drop Curve 3): 0x00000000
 Police ID
(Token/Conform/Exceed/Violate/Parent-Drop):0x00000008/0x00000310/0x00000311/0x00000312/0x00000313


Hash key (host format):
0x55b44c6b93dd: 02 00 00 00 00 14 00 14
UIDB Index: 0x0000 (0)
Class ID  : 0x0000 (0)
SW SOURCE-PORT: 0x14 (20)

HW NP-PORT: 0x14 (20)

hash result (host format):
0x55b44c6b939b: 02 00 33 00 00 00 00 00
0x55b44c6b93a3: 00 00 00 00 08 00 00 00
0x55b44c6b93ab: 00 00 00 00 00 00 00 00
0x55b44c6b93b3: 10 03 00 00 00 00 00 00
0x55b44c6b93bb: 00 00 00 00 00 00 00 00
0x55b44c6b93c3: 00 00 00 00 00 00 00 00
0x55b44c6b93cb: 00 00 00 00 00 00 00 00
0x55b44c6b93d3: 00 00 00 00 00 00 00 00
0x55b44c6b93db: 0000
FID0                : 0x00000000 (0)
FID1                : 0x00000000 (0)
Token Bucket        : 0x000008 (8)
Parent Token Bucket : 0x000000 (0)
Stats ID            : 0x000310 (784)
Parent Stats ID     : 0x000000 (0)
COS                 : 0x000003 (3)
L1 Overhead Size    : 0x00000C (12)
L1 Overhead Negative: 0x000000 (0)


hash result (ucode format):
0300000000001400400000000000000800310a88000000c0000000000000000000000000000000000000000000000000000000
hash_tbl_id                       =          0 (0x0)
uidb_index                        =          3 (0x3)
class_id                          =          0 (0x0)
np_port                           =          0 (0x0)
sat_icl_bundle_port               =          0 (0x0)
ctrl_egress                       =          0 (0x0)
ctrl_ingress                      =          0 (0x0)
RESULT:
  q_action_valid                  =          1 (0x1)
  turbo                           =          0 (0x0)
  l1_oh_neg                       =          0 (0x0)
  match                           =          1 (0x1)
```

```
  queue_stats_offset               =           4 (0x4)
  c_stat                           =     8389392 (0x800310)
  p_stat                           =     8388608 (0x800000)
  val                              =          12 (0xc)
  c_tb                             =           0 (0x0)
  p_tb                             =           0 (0x0)
  gp_tb                            =           0 (0x0)
  queue_en                         =           0 (0x0)
  profile_table_ptr                =           0 (0x0)
queue_id 0
  offset_table_ptr                 =           0 (0x0)
  offset_array_0                   =           0 (0x0)
                                   =  0 0 0 0 0 0 0 0
  offset_array_1                   =           0 (0x0)
                                   =  0 0 0 0 0 0 0 0

tcam result (host format):
0x55b44c6b9374: 80 41 00 03 00 00 00 00
0x55b44c6b937c: 4A FF 00 00 FF 00 00 FF
0x55b44c6b9384: 00 00 40 FF 00 00 FF 00
0x55b44c6b938c: 00 FF 00 00 00 00 00 00
0x55b44c6b9394: 00000000000000
```

The following example shows how to **configure a negative overhead accounting value**:

```
Router#conf
Router(config)#int hundredGigE 0/0/0/2
Router(config-if)#service-policy input policer account user-defined -12
Router(config-if)#commit
```

The following example shows **how to verify the negative overhead accounting value** you configured in the preceding example:

```
Router#sh qos-ea int hundredGigE 0/0/0/2 input detail
Interface: HundredGigE0_0_0_2 input policy: policer
RP/0/RSP0/CPU0:Router#sh qos-ea int hundredGigE 0/0/0/2 input detail
Interface: HundredGigE0_0_0_2 input policy: policer
Total number of classes:   1
Total number of UBRL classes:  0
Total number of CAC classes:   0
------------------------------------------------------
Policy name: policer
Hierarchical depth 1
Interface type HundredGigE
Interface rate 100000000 kbps
Port Shaper rate 0 kbps
Interface handle 0x00000680
ul_ifh 0x00000000, ul_id  0x00000080
uidb index 0x0003
qos_ifh 0x10200020800003
Local port 16, NP 0
Policy map id 0x601C, format 8, uidb index 0x0003
------------------------------------------------------
Index 0 Level 0 Class name class-default service_id 0x0 Policy name policer
Node flags: LEAF DEFAULT DEFAULT-ALL
 Stats flags: Policer type 1 Max category 0
 Node Config:
 Police Color aware 0 Type 1 CIR/CBS/PIR/PBS: 10000000kbps/125000000B/0kbps/0B
 Node Result: Class-based stats:Stat ID 0x00000310
 Queue: N/A Stat ID(Commit): 0x00000000
 Stat ID(Drop Curve 0): 0x00000000
 Stat ID(Drop Curve 1): 0x00000000
 Stat ID(Drop Curve 2): 0x00000000
 Stat ID(Drop Curve 3): 0x00000000
 Police ID
```

```
(Token/Conform/Exceed/Violate/Parent-Drop):0x00000010/0x00000310/0x00000311/0x00000312/0x00000313


Hash key (host format):
0x55c9fa43d3dd: 02 00 00 00 00 14 00 14
UIDB Index: 0x0000 (0)
Class ID  : 0x0000 (0)
SW SOURCE-PORT: 0x14 (20)

HW NP-PORT: 0x14 (20)

hash result (host format):
0x55c9fa43d39b: 82 00 33 00 00 00 00 00
0x55c9fa43d3a3: 00 00 00 00 10 00 00 00
0x55c9fa43d3ab: 00 00 00 00 00 00 00 00
0x55c9fa43d3b3: 10 03 00 00 00 00 00 00
0x55c9fa43d3bb: 00 00 00 00 00 00 00 00
0x55c9fa43d3c3: 00 00 00 00 00 00 00 00
0x55c9fa43d3cb: 00 00 00 00 00 00 00 00
0x55c9fa43d3d3: 00 00 00 00 00 00 00 00
0x55c9fa43d3db: 0000
FID0                : 0x00000000 (0)
FID1                : 0x00000000 (0)
Token Bucket        : 0x000010 (16)
Parent Token Bucket : 0x000000 (0)
Stats ID            : 0x000310 (784)
Parent Stats ID     : 0x000000 (0)
COS                 : 0x000003 (3)
L1 Overhead Size    : 0x00000C (12)
L1 Overhead Negative: 0x000001 (1)


hash result (ucode format):
0300000000001400400000000000000800310b88000000c0000000000000000000000000000000000000000000000000000000
hash_tbl_id                         =           0 (0x0)
uidb_index                          =           3 (0x3)
class_id                            =           0 (0x0)
np_port                             =           0 (0x0)
sat_icl_bundle_port                 =           0 (0x0)
ctrl_egress                         =           0 (0x0)
ctrl_ingress                        =           0 (0x0)
RESULT:
RP/0/RSP0/CPU0:Router#
```

To **modify an overhead accounting value,** you must:

1. Remove the existing QoS policy and re-add it.
2. Configure the new overhead accounting value.

```
Router#conf
Router(config)#int hundredGigE 0/0/0/2
Router(config-if)#no service-policy input policer
Router(config-if)#service-policy input policer account user-defined -20
Router(config-if)#commit
Router#sh run int hundredGigE 0/0/0/2
interface HundredGigE0/0/0/2
service-policy input policer account user-defined -20
!
```

To **verify the modified overhead accounting value**:

```
Router#sh qos-ea int hundredGigE 0/0/0/2 input detail
Interface: HundredGigE0_0_0_2 input policy: policer
Total number of classes:   1
Total number of UBRL classes:  0
Total number of CAC classes:   0
------------------------------------------------------
Policy name: policer
Hierarchical depth 1
Interface type HundredGigE
Interface rate 100000000 kbps
Port Shaper rate 0 kbps
Interface handle 0x00000680
ul_ifh 0x00000000, ul_id  0x00000080
uidb index 0x0003
qos_ifh 0x10200020800003
Local port 16, NP 0
Policy map id 0x6020, format 8, uidb index 0x0003
------------------------------------------------------
Index 0 Level 0 Class name class-default service_id 0x0 Policy name policer
Node flags: LEAF DEFAULT DEFAULT-ALL
 Stats flags: Policer type 1 Max category 0
 Node Config:
 Police Color aware 0 Type 1 CIR/CBS/PIR/PBS: 10000000kbps/125000000B/0kbps/0B
 Node Result: Class-based stats:Stat ID 0x00000314
 Queue: N/A Stat ID(Commit): 0x00000000
 Stat ID(Drop Curve 0): 0x00000000
 Stat ID(Drop Curve 1): 0x00000000
 Stat ID(Drop Curve 2): 0x00000000
 Stat ID(Drop Curve 3): 0x00000000
 Police ID
(Token/Conform/Exceed/Violate/Parent-Drop):0x00000018/0x00000314/0x00000315/0x00000316/0x00000317


Hash key (host format):
0x557e9a5ad3dd: 02 00 00 00 00 14 00 14
UIDB Index: 0x0000 (0)
Class ID  : 0x0000 (0)
SW SOURCE-PORT: 0x14 (20)

HW NP-PORT: 0x14 (20)

hash result (host format):
0x557e9a5ad39b: 82 00 53 00 00 00 00 00
0x557e9a5ad3a3: 00 00 00 00 18 00 00 00
0x557e9a5ad3ab: 00 00 00 00 00 00 00 00
0x557e9a5ad3b3: 14 03 00 00 00 00 00 00
0x557e9a5ad3bb: 00 00 00 00 00 00 00 00
0x557e9a5ad3c3: 00 00 00 00 00 00 00 00
0x557e9a5ad3cb: 00 00 00 00 00 00 00 00
0x557e9a5ad3d3: 00 00 00 00 00 00 00 00
0x557e9a5ad3db: 0000
FID0                : 0x00000000 (0)
FID1                : 0x00000000 (0)
Token Bucket        : 0x000018 (24)
Parent Token Bucket : 0x000000 (0)
Stats ID            : 0x000314 (788)
Parent Stats ID     : 0x000000 (0)
COS                 : 0x000003 (3)
L1 Overhead Size    : 0x000014 (20)
L1 Overhead Negative: 0x000001 (1)


hash result (ucode format):
0300000000001400400000000000000000800314b880000014000000000000000000000000000000000000000000000000000000000000
```

```
hash_tbl_id                     =           0 (0x0)
uidb_index                      =           3 (0x3)
class_id                        =           0 (0x0)
np_port                         =           0 (0x0)
sat_icl_bundle_port             =           0 (0x0)
ctrl_egress                     =           0 (0x0)
ctrl_ingress                    =           0 (0x0)
```

Positive Accounting Use Case

If QoS commitment includes Preamble, Frame Delimiter & Interframe Gap and has the following configuration:

```
service-policy input <foo> account user-defined +20
```

For QoS purposes, your router treats this packet as a packet of size = Actual Packet size + 20. Hence, the effective policing and shaping is *reduced* to match the downstream interface.

Negative Accounting Use Case

If QoS commitment to ASR9000 does not include VLAN header information, and has the following configuration:

```
service-policy input <foo> account user-defined -4
```

For QoS purposes, your router treats this packet as a packet of size = Actual Packet size – 4. Hence, the effective policing and shaping is *increased* to match the downstream interface.

Associated Commands

service-policy (overhead accounting)

Traffic Shaping

Traffic shaping allows you to control the traffic flow exiting an interface to match its transmission to the speed of the remote target interface and ensure that the traffic conforms to policies contracted for it. Traffic adhering to a particular profile can be shaped to meet downstream requirements, thereby eliminating bottlenecks in topologies with data-rate mismatches.

To match the rate of transmission of data from the source to the target interface, you can limit the transfer of data to one of the following:

- A specific configured rate
- A derived rate based on the level of congestion

The rate of transfer depends on these three components that constitute the token bucket: burst size, mean rate, and time (measurement) interval. The mean rate is equal to the burst size divided by the interval.

When traffic shaping is enabled, the bit rate of the interface does not exceed the mean rate over any integral multiple of the interval. In other words, during every interval, a maximum of burst size can be sent. Within the interval, however, the bit rate may be faster than the mean rate at any given time.

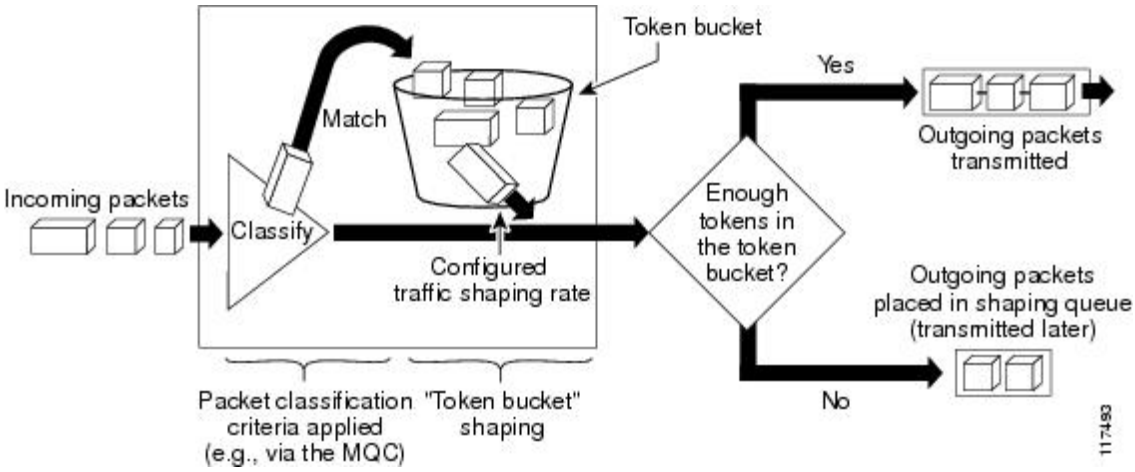
When the peak burst size equals 0, the interface sends no more than the burst size every interval, achieving an average rate no higher than the mean rate. However, when the peak burst size is greater than 0, the interface can send as many as the burst size plus peak burst bits in a burst, if in a previous time period the maximum amount was not sent. Whenever less than the burst size is sent during an interval, the remaining number of bits, up to the peak burst size, can be used to send more than the burst size in a later interval.

Regulation of Traffic with the Shaping Mechanism

When incoming packets arrive at an interface, the packets are classified using a classification technique, such as an access control list (ACL) or the setting of the IP Precedence bits through the Modular QoS CLI (MQC). If the packet matches the specified classification, the traffic-shaping mechanism continues. Otherwise, no further action is taken.

This figure illustrates how a traffic shaping mechanism regulates traffic flow.

Figure 1: How a Traffic Shaping Mechanism Regulates Traffic



Packets matching the specified criteria are placed in the token bucket. The maximum size of the token bucket is the confirm burst (Bc) size plus the Be size. The token bucket is filled at a constant rate of Bc worth of tokens at every Tc. This is the configured traffic shaping rate.

If the traffic shaping mechanism is active (that is, packets exceeding the configured traffic shaping rate already exist in a transmission queue) at every Tc, the traffic shaper checks to see if the transmission queue contains enough packets to send (that is, up to either Bc [or Bc plus Be] worth of traffic).

If the traffic shaper is not active (that is, there are no packets exceeding the configured traffic shaping rate in the transmission queue), the traffic shaper checks the number of tokens in the token bucket. One of the following occurs:

- If there are enough tokens in the token bucket, the packet is sent (transmitted).
- If there are not enough tokens in the token bucket, the packet is placed in a shaping queue for transmission at a later time.

Traffic Policing

In general, traffic policing allows you to control the maximum rate of traffic sent or received on an interface and to partition a network into multiple priority levels or class of service (CoS).

Traffic policing manages the maximum rate of traffic through a token bucket algorithm. The token bucket algorithm uses user-configured values to determine the maximum rate of traffic allowed on an interface at a given moment in time. The token bucket algorithm is affected by all traffic entering or leaving the interface (depending on where the traffic policy with traffic policing is configured) and is useful in managing network bandwidth in cases where several large packets are sent in the same traffic stream.

Traffic policing is often configured on interfaces at the edge of a network to limit the rate of traffic entering or leaving the network. In the most common traffic policing configurations, traffic that conforms to the CIR is sent and traffic that exceeds is sent with a decreased priority or is dropped. Users can change these configuration options to suit their network needs. Traffic policing also provides a certain amount of bandwidth management by allowing you to set the burst size (Bc) for the committed information rate (CIR). When the peak information rate (PIR) is supported, a second token bucket is enforced and then the traffic policer is called a two-rate policer.

Regulation of Traffic with the Policing Mechanism

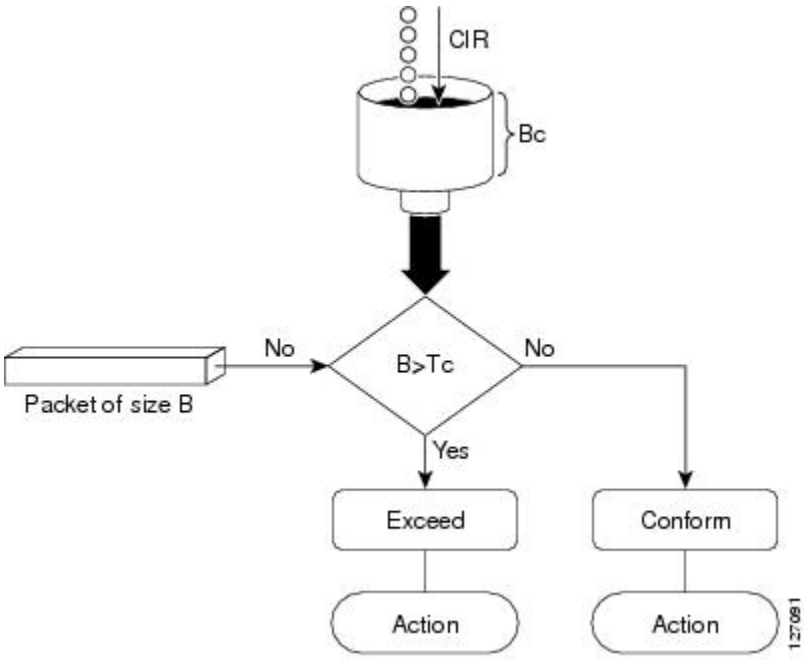
This section describes the single-rate and two-rate policing mechanisms.

Single-Rate Policer

A single-rate, two-action policer provides one token bucket with two actions for each packet: a conform action and an exceed action.

This figure illustrates how a single-rate token bucket policer marks packets as either conforming or exceeding a CIR, and assigns an action.

Figure 2: Marking Packets and Assigning Actions—Single-Rate Policer



The time interval between token updates (Tc) to the token bucket is updated at the CIR value each time a packet arrives at the traffic policer. The Tc token bucket can contain up to the Bc value, which can be a certain number of bytes or a period of time. If a packet of size B is greater than the Tc token bucket, then the packet exceeds the CIR value and a configured action is performed. If a packet of size B is less than the Tc token bucket, then the packet conforms and a different configured action is performed.

Two-Rate Policer

The two-rate policer manages the maximum rate of traffic by using two token buckets: the committed token bucket and the peak token bucket. The dual-token bucket algorithm uses user-configured values to determine the maximum rate of traffic allowed on a queue at a given moment. In this way, the two-rate policer can meter traffic at two independent rates: the committed information rate (CIR) and the peak information rate (PIR).

The committed token bucket can hold bytes up to the size of the committed burst (bc) before overflowing. This token bucket holds the tokens that determine whether a packet conforms to or exceeds the CIR as the following describes:

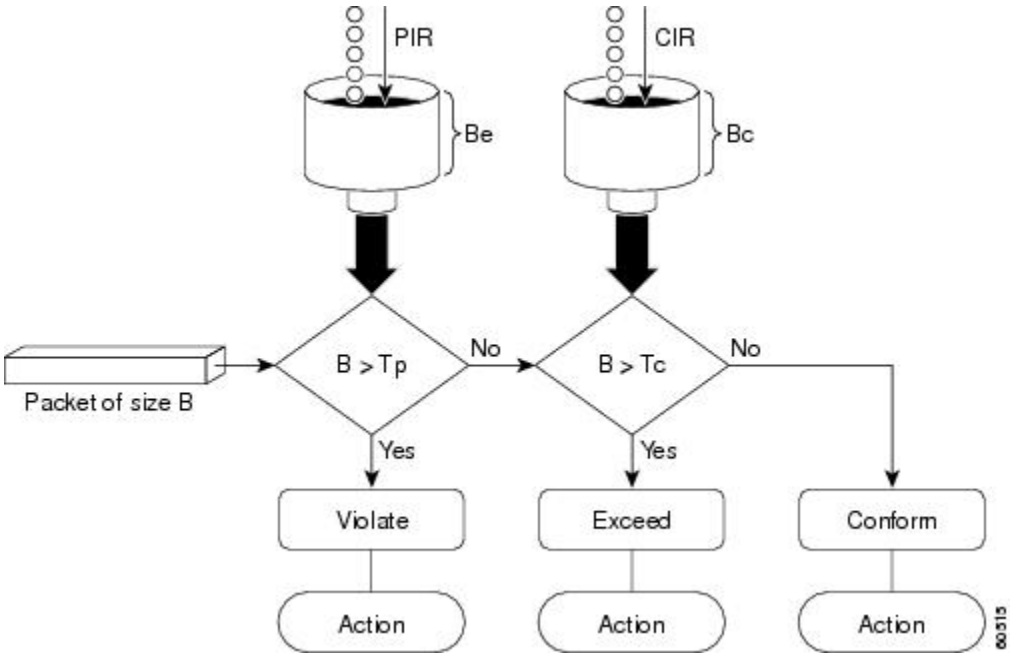
- A traffic stream is conforming when the average number of bytes over time does not cause the committed token bucket to overflow. When this occurs, the token bucket algorithm marks the traffic stream green.
- A traffic stream is exceeding when it causes the committed token bucket to overflow into the peak token bucket. When this occurs, the token bucket algorithm marks the traffic stream yellow. The peak token bucket is filled as long as the traffic exceeds the police rate.

The peak token bucket can hold bytes up to the size of the peak burst (be) before overflowing. This token bucket holds the tokens that determine whether a packet violates the PIR. A traffic stream is violating when it causes the peak token bucket to overflow. When this occurs, the token bucket algorithm marks the traffic stream red.

The dual-token bucket algorithm provides users with three actions for each packet—a conform action, an exceed action, and an optional violate action. Traffic entering a queue with the two-rate policer configured is placed into one of these categories. Within these three categories, users can decide packet treatments. For instance, packets that conform can be configured to be sent; packets that exceed can be configured to be sent with a decreased priority; and packets that violate can be configured to be dropped.

This figure shows how the two-rate policer marks a packet and assigns a corresponding action to the packet.

Figure 3: Marking Packets and Assigning Actions—2-Rate Policer



For example, if a data stream with a rate of 250 kbps arrives at the two-rate policer, and the CIR is 100 kbps and the PIR is 200 kbps, the policer marks the packet in the following way:

- 100 kbps conforms to the rate
- 100 kbps exceeds the rate
- 50 kbps violates the rate

The router updates the tokens for both the committed and peak token buckets in the following way:

- The router updates the committed token bucket at the CIR value each time a packet arrives at the interface. The committed token bucket can contain up to the committed burst (bc) value.
- The router updates the peak token bucket at the PIR value each time a packet arrives at the interface. The peak token bucket can contain up to the peak burst (be) value.
- When an arriving packet conforms to the CIR, the router takes the conform action on the packet and decrements both the committed and peak token buckets by the number of bytes of the packet.
- When an arriving packet exceeds the CIR, the router takes the exceed action on the packet, decrements the committed token bucket by the number of bytes of the packet, and decrements the peak token bucket by the number of overflow bytes of the packet.
- When an arriving packet exceeds the PIR, the router takes the violate action on the packet, but does not decrement the peak token bucket.

Committed Bursts and Excess Bursts

Unlike a traffic shaper, a traffic policer does not buffer excess packets and transmit them later. Instead, the policer executes a “send or do not send” policy without buffering. During periods of congestion, proper configuration of the excess burst parameter enables the policer to drop packets less aggressively. Therefore, it is important to understand how policing uses the committed (normal) and excess burst values to ensure the router reaches the configured committed information rate (CIR).

Burst parameters are based on a generic buffering rule for routers, which recommends that you configure buffering to be equal to the round-trip time bit-rate to accommodate the outstanding TCP windows of all connections in times of congestion.

The following sections describe committed bursts and excess bursts, and the recommended formula for calculating each of them:

- Committed Bursts
- Excess Bursts
- Deciding if Packets Conform or Exceed the Committed Rate

Committed Bursts

The committed burst (bc) parameter of the police command implements the first, conforming (green) token bucket that the router uses to meter traffic. The bc parameter sets the size of this token bucket. Initially, the token bucket is full and the token count is equal to the committed burst size (CBS). Thereafter, the meter updates the token counts the number of times per second indicated by the committed information rate (CIR).

The following describes how the meter uses the conforming token bucket to send packets:

- If sufficient tokens are in the conforming token bucket when a packet arrives, the meter marks the packet green and decrements the conforming token count by the number of bytes of the packet.
- If there are insufficient tokens available in the conforming token bucket, the meter allows the traffic flow to borrow the tokens needed to send the packet. The meter checks the exceeding token bucket for the number of bytes of the packet. If the exceeding token bucket has a sufficient number of tokens available, the meter marks the packet:

  Green and decrements the conforming token count down to the minimum value of 0.

  Yellow, borrows the remaining tokens needed from the exceeding token bucket, and decrements the exceeding token count by the number of tokens borrowed down to the minimum value of 0.
- If an insufficient number of tokens is available, the meter marks the packet red and does not decrement either of the conforming or exceeding token counts.

Note When the meter marks a packet with a specific color, there must be a sufficient number of tokens of that color to accommodate the entire packet. Therefore, the volume of green packets is never smaller than the committed information rate (CIR) and committed burst size (CBS). Tokens of a given color are always used on packets of that color.

The default committed burst size is the greater of 2 milliseconds of bytes at the police rate or the network maximum transmission unit (MTU).

Committed Burst Calculation

To calculate committed burst, use the following formula:

bc = CIR bps * (1 byte) / (8 bits) * 1.5 seconds

Note 1.5 seconds is the typical round-trip time.

For example, if the committed information rate is 512000 bps, then using the committed burst formula, the committed burst is 96000 bytes.

bc = 512000 * 1/8 * 1.5

bc = 64000 * 1.5 = 96000

Note When the be value equals 0, we recommend that you set the egress bc value to be greater than or equal to the ingress bc value plus 1. Otherwise, packet loss can occur. For example: be = 0 egress bc >= ingress bc + 1

Excess Bursts

The excess burst (be) parameter of the police command implements the second, exceeding (yellow) token bucket that the router uses to meter traffic. The exceeding token bucket is initially full and the token count is equal to the excess burst size (EBS). Thereafter, the meter updates the token counts the number of times per second indicated by the committed information rate (CIR).

The following describes how the meter uses the exceeding token bucket to send packets:

- When the first token bucket (the conforming bucket) meets the committed burst size (CBS), the meter allows the traffic flow to borrow the tokens needed from the exceeding token bucket. The meter marks the packet yellow and then decrements the exceeding token bucket by the number of bytes of the packet.
- If the exceeding token bucket does not have the required tokens to borrow, the meter marks the packet red and does not decrement the conforming or the exceeding token bucket. Instead, the meter performs the exceed-action configured in the police command (for example, the policer drops the packets).

Excess Burst Calculation

To calculate excess burst, use the following formula:

be = 2 * committed burst

For example, if you configure a committed burst of 4000 bytes, then using the excess burst formula, the excess burst is 8000 bytes.

be = 2 * 4000 = 8000

The default excess burst size is 0.

Deciding if Packets Conform or Exceed the Committed Rate

Policing uses normal or committed burst (bc) and excess burst (be) values to ensure that the configured committed information rate (CIR) is reached. Policing decides if a packet conforms or exceeds the CIR based on the burst values you configure. Several factors can influence the policer's decision, such as the following:

- Low burst values—If you configure burst values too low, the achieved rate might be much lower than the configured rate.
- Temporary bursts—These bursts can have a strong adverse impact on throughput of Transmission Control Protocol (TCP) traffic.

It is important that you set the burst values high enough to ensure good throughput. If your router drops packets and reports an exceeded rate even though the conformed rate is less than the configured CIR, use the show interface command to monitor the current burst, determine whether the displayed value is consistently close to the committed burst (bc) and excess burst (be) values, and if the actual rates (the committed rate and exceeded rate) are close to the configured committed rate. If not, the burst values might be too low. Try reconfiguring the burst rates using the suggested calculations in the Committed Burst Calculation and the Excess Burst Calculation.

Two-Rate Three-Color (2R3C) Policer

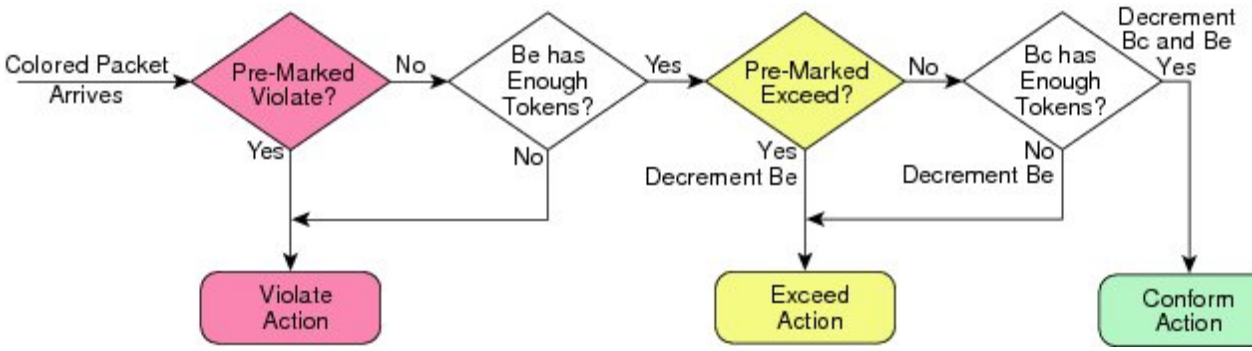
For the SIP 700 card, a two-rate, three-color (2R3C) policer is supported on policy maps for ingress Layer 2 interfaces. The policer reads a preexisting marking—the frame-relay discard-eligibility (FRDE) bit in the packet header—that was set by a policer on a previous network node. By default the FRDE bit is set to 0. At the receiving node, the system uses this bit to determine the appropriate color-aware policing action for the packet:

- To classify the FRDE bit value 0 as conform color, create a conform-color class-map for frde=0 packets. This causes packets to be classified as color green, and the system applies the conform action.
- To classify the FRDE bit value 1 as exceed color, create an exceed-color class-map for frde=1 packets. This causes packets to be classified as color yellow, and the system applies the exceed action.

Note Color-aware policing is not supported for hierarchical QoS.

The 2R3C policing process is shown in this figure.

Figure 4: 2R3C Policing Process Flowchart



Note When ingress QoS policy is applied for 9000v the counters are organized in logical pairs. If 3-color policer is applied, only two counters will be considered:

- Green and Non-Green (Yellow + Red)
- Red and Non-Red (Green + Yellow)

Hierarchical Policing

The Hierarchical Policing feature is an MQC-based solution that supports hierarchical policing on both the ingress and egress interfaces on Cisco ASR 9000 Series Router.

This feature allows enforcement of service level agreements (SLA) while applying the classification submodel for different QoS classes on the inbound interface.

Hiearchical policing provides support at two levels:

- Parent level
- Child level

Multiple Action Set

Packet Marking Through the IP Precedence Value, IP DSCP Value, and the MPLS Experimental Value Setting

In addition to rate-limiting, traffic policing allows you to independently mark (or classify) the packet according to whether the packet conforms or violates a specified rate. Packet marking also allows you to partition your network into multiple priority levels or CoS. Packet marking as a policer action is conditional marking.

Use the traffic policer to set the IP precedence value, IP DSCP value, or Multiprotocol Label Switching (MPLS) experimental value for packets that enter the network. Then networking devices within your network can use this setting to determine how the traffic should be treated. For example, the Weighted Random Early Detection (WRED) feature uses the IP precedence value to determine the probability that a packet is dropped.

If you want to mark traffic but do not want to use traffic policing, see the "Class-based, Unconditional Packet Marking Examples" section to learn how to perform packet classification.

Note Marking IP fields on an MPLS-enabled interface results in non-operation on that particular interface.

Traffic Policing on Layer 2 ATM Interfaces

Traffic policing is supported on the Layer 2 ATM interfaces in the ingress imposition path. The OAM cells are policed along with the user cells unless the QoS policy is explicitly configured to exclude the OAM cells from being policed.

Note Policing is supported for the virtual circuit (VC), and the virtual path (VP) modes. However, policing is not supported for the port mode on the Layer 2 ATM interfaces.

Different match criteria can be used in the policy map with class-default matching all the traffic including the OAM cells.

Policing is performed on the ATM Adaptation Layer type 0 (AAL0) cells but translates to ATM Adaptation Layer type 5 (AAL5) packets as described below:

- AAL5 packet conforms, if all the cells in the packet conform to peak cell rate (PCR) and sustainable cell rate (SCR) buckets.
- AAL5 packet exceeds, if at least one cell does not conform to the SCR bucket.
- AAL5 packet violates, if at least one cell does not conform to the PCR bucket.

The following policer options are supported:

- Rate in cells per second, and percent
- Peak rate in cells per second, and percent
- Delay tolerance in microseconds
- Maximum burst size in cells

The following policer actions are supported on the Layer 2 ATM interfaces in the ingress direction:

- **transmit**
- **drop**
- **set mpls exp imposition** <*exp*> (AToM only)
- **set qos-group** <*qos-group*> (AToM and local switching)

- **set discard-class** <*discard-class*> (AToM and local switching)
- **set atm-clp** (Exceed action only, AToM and local switching)
- **drop** (Violate action)

Multiple policing action is supported on the Layer 2 ATM interfaces using the **set mpls exp imposition** and **set atm-clp** combination.

Restrictions

The following list shows non-supported configuration for traffic policing on a layer 2 ATM interface.

- Applying hierarchical policy maps.
- Configuring service policy on a physical interface.
- Policing of egress traffic.
- Configuring multiple police classes on a policy.
- Configuring conform or violate actions.
- Only **match atm clp** command is supported.

Traffic Policing on a Layer 2 ATM interface: Example

The following example illustrates a sample configuration of traffic policing on a layer 2 ATM interface.

```
policy-map atm
 class class-default
  police rate percent 10
  !
 !
 end-policy-map
!
interface ATM0/1/0/0.1 l2transport
 pvc 10/100
encapsulation aal0
service-policy input atm
```

Explicit Congestion Notification

In mobile networks, a Base Station Controller (BSC) does not have the knowledge if a particular cell site is being overwhelmed by traffic on a particular link, as it sits behind the ASR9000 series router and it will continue to send traffic even if there is acute congestion on the link. So, once the cell site marks the traffic with the (Explicit Congestion Notification) ECN bits and sends it to the BSC, the BSC will mark the affected session from the congested site with the ECN bit flagged towards the ASR9000 series router.

ECN is an extension to WRED (Weighted Random Early Detection). ECN will mark packets instead of dropping them when the average queue length exceeds a specific threshold value. When configured, ECN helps routers and end hosts to understand that the network is congested and slow down sending packets. However If the number of packets in the queue is above the maximum threshold, packets are dropped based on the drop probability. This is the identical treatment a packet receives when WRED is enabled without ECN configured on the router.

Limitations

- ECN is supported only on ASR 9000 SIP-700 linecards.

For more information on the ECN feature, please refer the *Modular QoS Configuration Guide for Cisco ASR 9000 Series Routers*

Implementing ECN

Implementing ECN requires an ECN-specific field that has two bits—the ECN-capable Transport (ECT) bit and the CE (Congestion Experienced) bit—in the IP header. The ECT bit and the CE bit can be used to make four ECN field combinations of 00 to 11. The first number is the ECT bit and the second number is the CE bit.

ECN Bit Setting

ECT Bit	CE Bit	Combination Indicates
0	0	Not-ECN-capable.
0	1	Endpoints of the transport protocol are ECN-capable.
1	0	Endpoints of the transport protocol are ECN-capable.
1	1	Congestion experienced.

The ECN field combination 00 indicates that a packet is not using ECN. The ECN field combinations 01 and 10—called ECT(1) and ECT(0), respectively—are set by the data sender to indicate that the endpoints of the transport protocol are ECN-capable. Routers treat these two field combinations identically. Data senders can use either one or both of these two combinations. The ECN field combination 11 indicates congestion to the endpoints. Packets arriving a full queue of a router will be dropped.

Packet Handling when ECN is enabled

When the number of packets in the queue is below the minimum threshold, packets are transmitted. This happens whether or not ECN is enabled, and this treatment is identical to the treatment a packet receives when WRED only is being used on the network. If the number of packets in the queue is above the maximum threshold, packets are dropped based on the drop probability. This is the identical treatment a packet receives when WRED is enabled without ECN configured on the router. Three different scenarios arise if the number of packets in the queue is between the minimum threshold and the maximum threshold:

- If the ECN field on the packet indicates that the endpoints are ECN-capable (that is, the ECT bit is set to 1 and the CE bit is set to 0, or the ECT bit is set to 0 and the CE bit is set to 1)—and the WRED algorithm determines that the packet should have been dropped based on the drop probability—the ECT and CE bits for the packet are changed to 1, and the packet is transmitted. This happens because ECN is enabled and the packet gets marked instead of dropped.
- If the ECN field on the packet indicates that neither endpoint is ECN-capable (that is, the ECT bit is set to 0 and the CE bit is set to 0), the packet may be dropped based on the WRED drop probability. This is the identical treatment that a packet receives when WRED is enabled without ECN configured on the router.
- If the ECN field on the packet indicates that the network is experiencing congestion (that is, both the ECT bit and the CE bit are set to 1), the packet is transmitted. No further marking is required.

QoS for IPv6 ACLs

The Modular Weapon-X line cards support classification of IPv6 properties based on Source IP, Destination IP, Source Port, Destination Port, Protocol, TOS, Hop Limit, and ACL-based classification.

The supported interfaces are indicated below.

Supported Interface	Ethernet Linecard	Enhanced Ethernet Linecard
L3 main interface	yes	yes
L3 sub-interface	yes	yes
L3 bundle-interface/ sub-interface	yes	yes
L2 main interface	no	yes
L2 sub-interface	no	yes
L2 bundle-interface/ sub-interface	no	yes

Policer Granularity and Shaper Granularity

Policer granularity can be configured in the ingress and egress directions. The policer granularity is specified as a permissible percentage variation between the user-configured policer rate, and the hardware programmed policer rate.

Policers applied in either the ingress or egress direction can have any configured rate. However, different line card generations have different granularity as to what rates can be programmed in the hardware. Because of this, a desired rate configured in the policy map may get rounded down to the nearest granularity increment.

Ethernet line cards support a granularity of 64 kbps increments. Hence, if you specify a police rate on Ethernet line cards that is not a multiple of 64, the police rate is rounded down to the nearest 64 kbps increment.

Enhanced Ethernet line cards support a granularity of 8 kbps, so a configured rate is rounded down to the nearest 8 kbps increment.

For all generations of linecards, the minimum police rate is 64 kbps.

To verify the programmed rate of the hardware, run the **show qos interface <interface> <direction>** command. For example:

```
RP/0/RSP0/CPU0:A9K-BNG#show qos interface gigabitEthernet 0/0/0/1 input
Tue Dec 19 16:45:58.260 EDT
….
------------------------------------------------------------------------
Level: 0 Policy: telnet Class: 3play-voip
QueueID: 162 (Port Default)
Policer Profile: 62 (Single)
Conform: 96 kbps(100 kbps) Burst: 1600 bytes (0 Default)
```

Here, the programmed rate is displayed outside the parentheses while the configured rate is displayed within parentheses.

Congestion Management Using DEI

You can manage congestion based on the Drop Eligible Indicator (DEI) bit that is present in 802.1ad frames and 802.1ah frames. Random early detection based on the DEI value is supported on 802.1ad packets for:

- Layer 2 subinterfaces
- Layer 2 main interfaces
- Layer 3 main interfaces
- Ingress and egress

Note If there are any marking actions in the policy, the marked values are used for doing WRED.

How to Configure QoS Congestion Management

Configuring Guaranteed and Remaining Bandwidths

The **bandwidth** command allows you to specify the minimum guaranteed bandwidth to be allocated for a specific class of traffic. MDRR is implemented as the scheduling algorithm.

The **bandwidth remaining** command specifies a weight for the class to the MDRR. The MDRR algorithm derives the weight for each class from the bandwidth remaining value allocated to the class. If you do not configure the **bandwidth remaining** command for any class, the leftover bandwidth is allocated equally to all classes for which **bandwidth remaining** is not explicitly specified.

Guaranteed Service rate of a queue is defined as the bandwidth the queue receives when all the queues are congested. It is defined as:

Guaranteed Service Rate = minimum bandwidth + excess share of the queue

Restrictions

The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead.

The **bandwidth** command is supported only on policies configured on outgoing interfaces.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **bandwidth** {*rate* [*units*]| **percent** *value*}
5. **bandwidth remaining percent** *value*
6. **exit**
7. **class** *class-name*
8. **bandwidth** {*rate* [*units*] | **percent** *value*}
9. **bandwidth remaining percent** *value*
10. **exit**
11. **exit**
12. **interface** *type interface-path-id*

13. **service-policy** {**input** | **output**} *policy-map*
14. Use the **commit** or **end** command.
15. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.
Step 4	**bandwidth** {*rate* [*units*]| **percent** *value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 50`	Specifies the bandwidth allocated for a class belonging to a policy map and enters the policy map class configuration mode. In this example, class class1 is guaranteed 50 percent of the interface bandwidth.
Step 5	**bandwidth remaining percent** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	Specifies how to allocate leftover bandwidth to various classes. **Note** The remaining bandwidth of 40 percent is shared by class class1 and class2 (see Steps 8 and 9) in a 20:80 ratio: class class1 receives 20 percent of the 40 percent, and class class2 receives 80 percent of the 40 percent.
Step 6	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 7	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class2`	Specifies the name of a different class whose policy you want to create or change.

	Command or Action	Purpose
Step 8	**bandwidth** {*rate* [*units*] | **percent** *value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 10`	Specifies the bandwidth allocated for a class belonging to a policy map. In this example, class class2 is guaranteed 10 percent of the interface bandwidth.
Step 9	**bandwidth remaining percent** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 80`	Specifies how to allocate leftover bandwidth to various classes. **Note** The remaining bandwidth of 40 percent is shared by class class1 and class2 (see Steps 8 and 9) in a 20:80 ratio: class class1 receives 20 percent of the 40 percent, and class class2 receives 80 percent of the 40 percent.
Step 10	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 12	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface POS 0/2/0/0`	Enters interface configuration mode and configures an interface.
Step 13	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 14	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

	Command or Action	Purpose
Step 15	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface POS 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Guaranteed Bandwidth

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **bandwidth** {*rate* [*units*]| **percent** *percentage-value*}
5. **exit**
6. **class** *class-name*
7. **bandwidth** {*rate* [*units*]| **percent** *percentage-value*}
8. **exit**
9. **class** *class-name*
10. **bandwidth** {*rate* [*units*]| **percent** *percentage-value*}
11. **exit**
12. **exit**
13. **interface** *type interface-path-id*
14. **service-policy** {**input** | **output**} *policy-map*
15. **end** or **commit**
16. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.

	Command or Action	Purpose
Step 4	**bandwidth** {*rate* [*units*]| **percent** *percentage-value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 40`	Enters policy map class configuration mode. • Specifies the bandwidth allocated for a class belonging to a policy map. • In this example, class class1 is guaranteed 40 percent of the interface bandwidth.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 6	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class2`	Specifies the name of the class whose policy you want to create or change.
Step 7	**bandwidth** {*rate* [*units*]| **percent** *percentage-value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 40`	Enters policy map class configuration mode. • Specifies the bandwidth allocated for a class belonging to a policy map. • In this example, class class2 is guaranteed 40 percent of the interface bandwidth.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 9	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Specifies the name of the class whose policy you want to create or change.
Step 10	**bandwidth** {*rate* [*units*]| **percent** *percentage-value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 20`	Enters policy map class configuration mode. • Specifies the bandwidth allocated for a class belonging to a policy map. • In this example, class class-default is guaranteed 20 percent of the interface bandwidth.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.

	Command or Action	Purpose
Step 12	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 13	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/2/0/0`	Enters interface configuration mode and configures an interface.
Step 14	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. • In this example, the traffic policy evaluates all traffic leaving that interface.

Step 15	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-if)# end`
	or
	`RP/0/RSP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 16	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Bandwidth Remaining

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **bandwidth remaining percent** *percentage-value*
5. **exit**
6. **class** *class-name*
7. **bandwidth remaining percent** *percentage-value*
8. **exit**
9. **class** *class-name*
10. **bandwidth remaining percent** *percentage-value*
11. **exit**
12. **exit**
13. **interface** *type interface-path-id*
14. **service-policy** {**input** | **output**} *policy-map*
15. **end** or **commit**
16. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.
Step 4	**bandwidth remaining percent** *percentage-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 40`	Specifies how to allocate leftover bandwidth for class class1.
Step 5	**exit** **Example:**	Returns the router to policy map configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	
Step 6	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class2`	Specifies the name of the class whose policy you want to create or change.
Step 7	**bandwidth remaining percent** *percentage-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 40`	Specifies how to allocate leftover bandwidth for class class2.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 9	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Specifies the name of the class whose policy you want to create or change.
Step 10	**bandwidth remaining percent** *percentage-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	Specifies how to allocate leftover bandwidth for class class-default.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 12	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 13	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/2/0/0`	Enters interface configuration mode and configures an interface.
Step 14	**service-policy** {**input** | **output**} *policy-map* **Example:**	Attaches a policy map to an input or output interface to be used as the service policy for that interface.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	• In this example, the traffic policy evaluates all traffic leaving that interface.

Step 15	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-if)# end`
	or
	`RP/0/RSP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 16	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Low-Latency Queueing with Strict Priority Queueing

The **priority** command configures LLQ with strict priority queuing (PQ) that allows delay-sensitive data such as voice to be dequeued and sent before packets in other queues are dequeued. When a class is marked as high priority using the **priority** command, you must configure a policer to limit the priority traffic. This configuration ensures that the priority traffic does not constrain all the other traffic on the line card, which protects low priority traffic from limitations. Use the **police** command to explicitly configure the policer.

Note Eight levels of priorities are supported: priority level 1, priority level 2, priority level 3, priority level 4, priority level 5, priority level 6, priority level 7 and the priority level normal. If no priority level is configured, the default is priority level normal.

Note The output of the show policy-map interface command is inconsistent for priority level 1, priority level 2, priority level 3 that is configured on the bundle interface, when the interface has either of these combinations:

- 3rd generation of ASR 9000 LC with SE and TR versions
- 4th generation of ASR 9000 LC with SE and 3rd generation of ASR 9000 LC with TR versions

Additionally, for priority level 3 configuration the show policy-map interface command output is inconsistent when the bundle interface has a combination of 3rd generation ASR 9000 LC with TR and 4th generation of ASR 9000 LC with TR versions.

In order to display consistent show policy-map interface, use **show policy-map interface bundle-eth** <*number*> **[input | output] member** {*interface type interface-path-id*} command.

Restrictions

- Unused priority queues cannot be used for a different priority level.
- The eight priority levels can be configured only on egress of main physical interface or main bundle interface.
- Eight priority levels work on Cisco ASR 9000 High Density 100GE Ethernet line cards only.
- The policy-map with eight priorities must have only one queuing class at the parent level of the priority class.
- If the policy-map has a parent class, the parent class cannot have bandwidth configured.
- Within a policy map, you can give one or more classes priority status. When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is queued to the same single priority queue.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [*burst-units*]] [**peak-burst** *peak-burst* [*burst-units*]] [**peak-rate** *value* [*units*]]
5. **exceed-action** *action*
6. **exit**
7. **priority**[**level** *priority_level*]
8. **exit**
9. **exit**
10. **interface** *type interface-path-id*
11. **service-policy** {**input** | **output**} *policy-map*
12. Use the **commit** or **end** command.
13. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map voice`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class voice`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [*burst-units*]] [**peak-burst** *peak-burst* [*burst-units*]] [**peak-rate** *value* [*units*]] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 250`	Configures traffic policing and enters policy map police configuration mode. In this example, the low-latency queue is restricted to 250 kbps to protect low-priority traffic from starvation and to release bandwidth.
Step 5	**exceed-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action drop`	Configures the action to take on packets that exceed the rate limit.
Step 6	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map class configuration mode.
Step 7	**priority**[**level** *priority_level*] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# priority level 1`	Specifies priority to a class of traffic belonging to a policy map. If no priority level is configured, the default is priority 1.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.

	Command or Action	Purpose
Step 9	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to Global Configuration mode.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet`	Enters interface configuration mode, and configures an interface.
Step 11	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 12	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 13	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Shaping

Traffic shaping allows you to control the traffic exiting an interface to match its transmission to the speed of the remote target interface and ensure that the traffic conforms to policies contracted for it.

Shaping performed on incoming and outgoing interfaces is done at the Layer 2 level and includes the Layer 2 header in the rate calculation.

Restrictions

- The bandwidth, priority and shape average commands should not be configured together in the same class.

- A flat port-level shaper requires a child policy with 100% bandwidth explicitly allocated to the class-default.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **shape average** {**percent** *value* | *rate* [*units*]}
5. **exit**
6. **exit**
7. Specifies the name of the class whose policy you want to create or change.**interface** *type interface-path-id*
8. **service-policy** {**input** | **output**} *policy-map*
9. Use the **commit** or **end** command.
10. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**shape average** {**percent** *value*	*rate* [*units*]} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# shape average percent 50`	Shapes traffic to the indicated bit rate according to average rate shaping in the specified units or as a percentage of the bandwidth.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 6	**exit** **Example:**	Returns the router to global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pmap)# exit`	
Step 7	Specifies the name of the class whose policy you want to create or change.**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet`	Enters interface configuration mode and configures an interface.
Step 8	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 10	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Policing (Two-Rate Color-Blind)

Traffic policing allows you to control the maximum rate of traffic sent or received on an interface. This section provides the procedure for configuring two-rate color-blind traffic policing.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]]
5. **conform-action** *action*

6. **exceed-action** *action*
7. **exit**
8. **exit**
9. **exit**
10. **interface** *type interface-path-id*
11. **service-policy** {**input** | **output**} *policy-map*
12. Use the **commit** or **end** command.
13. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 250000`	Configures traffic policing and enters policy map police configuration mode. The traffic policing feature works with a token bucket algorithm.
Step 5	**conform-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-action set mpls experimental topmost 3`	Configures the action to take on packets that conform to the rate limit. The *action* argument is specified by one of these keywords: • **drop**—Drops the packet. • **set**—Has these keywords and arguments: **discard-class** *value*—Sets the discard class value. Range is 0 to 7. **dscp** —Sets the differentiated services code point (DSCP) value and sends the packet.

	Command or Action	Purpose
		mpls experimental {**topmost** | **imposition**} *value*—Sets the experimental (EXP) value of the Multiprotocol Label Switching (MPLS) packet topmost label or imposed label. Range is 0 to 7. **precedence** —Sets the IP precedence and sends the packet. **qos-group**—Sets the QoS group value. Range is from 0 to 511. • **transmit**—Transmits the packets.
Step 6	**exceed-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action set mpls experimental topmost 4`	Configures the action to take on packets that exceed the rate limit. The *action* argument is specified by one of the keywords specified in Step 5 .
Step 7	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 9	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet`	Enters configuration mode and configures an interface.
Step 11	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 12	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions:

	Command or Action	Purpose
		• **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 13	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Policing (2R3C)

This section provides the procedure for configuring two-rate three-color traffic policing. It is applicable to SIP 700 line cards on the ingress side only.

SUMMARY STEPS

1. **configure**
2. **class-map** [**match-all**][**match-any**] *class-map-name*
3. **match [not] fr-de***fr-de-bit-value*
4. **policy-map** *policy-name*
5. **class** *class-name*
6. **police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]]
7. **conform-color** *class-map-name*
8. **exceed-color** *class-map-name*
9. **conform-action** *action*
10. **exceed-action** *action*
11. **exit**
12. **exit**
13. **exit**
14. **interface** *type interface-path-id*
15. **service-policy** *policy-map*
16. Use the **commit** or **end** command.
17. **show policy-map interface** *type interface-path-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map** [**match-all**][**match-any**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-all match-not-frde`	(Use with SIP 700 line card, ingress only) Creates or modifies a class map that can be attached to one or more interfaces to specify a matching policy and enters the class map configuration mode.
Step 3	**match [not] fr-de***fr-de-bit-value* **Example:** `RP/0/RSP0/CPU0:router(config)# match not  fr-de 1`	(Use with SIP 700 line card, ingress only) Specifies the matching condition: • Match *not* fr-de 1 is typically used to specify a conform-color packet. • Match fr-de 1 is typically used to specify an exceed-color packet.
Step 4	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 5	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 6	**police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 768000 burst 288000 peak-rate 1536000 peak-burst 576000`	Configures traffic policing and enters policy map police configuration mode. The traffic policing feature works with a token bucket algorithm.
Step 7	**conform-color** *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-color match-not-frde`	(Use with SIP 700 line card, ingress only) Configures the class-map name to assign to conform-color packets.
Step 8	**exceed-color** *class-map-name*	(Use with SIP 700 line card, ingress only)

	Command or Action	Purpose
	Example: `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-color match-frde`	Configures the class-map name to assign to exceed-color packets.
Step 9	**conform-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-action set mpls experimental topmost 3`	Configures the action to take on packets that conform to the rate limit. The *action* argument is specified by one of these keywords: • **drop**—Drops the packet. • **set**—Has these keywords and arguments: **discard-class** *value*—Sets the discard class value. Range is 0 to 7. **dscp** *value*—Sets the differentiated services code point (DSCP) value and sends the packet. **mpls experimental** {**topmost** | **imposition**} *value*—Sets the experimental (EXP) value of the Multiprotocol Label Switching (MPLS) packet topmost label or imposed label. Range is 0 to 7. **precedence** *precedence*—Sets the IP precedence and sends the packet. **qos-group**—Sets the QoS group value. Range is 0 to 63. • **transmit**—Transmits the packets.
Step 10	**exceed-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action set mpls experimental topmost 4`	Configures the action to take on packets that exceed the rate limit. The *action* argument is specified by one of the keywords specified in Step 5.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
Step 12	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 13	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.

	Command or Action	Purpose
Step 14	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface pos 0/5/0/0`	Enters configuration mode and configures an interface.
Step 15	**service-policy** *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy policy1`	Attaches a policy map to an input interface to be used as the service policy for that interface.
Step 16	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 17	**show policy-map interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface POS 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Hierarchical Policing

Hierarchical policing provides support at two levels:

- Parent level
- Child level

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **service-policy** *policy-map-name*
5. **police rate percent** *percentage*
6. **conform-action** *action*
7. **exceed-action** *action*

8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
Step 4	**service-policy** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy child`	Attaches a policy map to an input or output interface to be used as the service policy for that interface.
Step 5	**police rate percent** *percentage* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate percent 50`	Configures traffic policing and enters policy map police configuration mode.
Step 6	**conform-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-action transmit`	Configures the action to take on packets that conform to the rate limit. The allowed action is: **transmit**—Transmits the packets.
Step 7	**exceed-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action drop`	Configures the action to take on packets that exceed the rate limit. The allowed action is: **drop**—Drops the packet.
Step 8	**end** or **commit** **Example:** `RP/0/RSP0/CPU0:router(config-if)# end` or	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-if)# commit`	Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring ECN

ECN helps routers and end hosts to understand that the network is congested and slow down the rate at which packets are transmitted.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **bandwidth** [*percent* | *value*]
5. **random-detect** { **default** | **discard-class** | **dscp** | **precedence** }
6. **random-detect ecn**
7. **exit**
8. **exit**
9. **end** or **commit**
10. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

	Command or Action	Purpose
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
Step 4	**bandwidth** [*percent* | *value*] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth 100`	Specifies or modifies the bandwidth allocated for a class in a specific policy-map. **Note** ECN can be configured with any queuing action, such as , bandwidth, shaping, etc.
Step 5	**random-detect** { **default** | **discard-class** | **dscp** | **precedence** } **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# random-detect dscp 1 1000 packets 2000 packets`	Configures the WRED profile. WRED profile entry is required to apply ECN for a particular class.
Step 6	**random-detect ecn** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# random-detect ecn`	Enables ECN.
Step 7	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.

Step 9	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-if)# end`
	or
	`RP/0/RSP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

	Command or Action	Purpose
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.
Step 10	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet 0/2/0/0`	(Optional) Displays statistics for all classes configured for all service policies on the specified interface. If ECN is enabled, displays ECN marking information for the specified interface.

Configuration Examples for Configuring Congestion Management

Service Fragment Configurations: Example

This example shows the service-fragment premium being created.

```
policy-map tsqos-port-policy
    class class-default
        shape 500 mbps
    class dscp1
        shape 1 Gbps
        service-fragment premium
        end-policy
      exit
```

This example shows the service-fragment premium being referred (at the sub-interface):

```
policy-map tsqos-subif-policy-premium
    class class-default
        fragment premium
        shape 20 mbps
        bandwidth remaining ratio 20
        service-policy subif-child
        end-policy
       exit
```

ECN: Example

The following example shows how to run the **random-detect ecn** command to configure ECN:

```
config
policy-map p1
class c1
bandwidth 100
random-detect dscp 1 1000 packets 2000 packets
random-detect ecn
```

```
exit
exit
commit
```

Hierarchical Policing: Example

Additional References

These sections provide references related to implementing QoS congestion management.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco ASR 9000 Series Aggregation Services Router Getting Guide*
QoS commands	*Cisco ASR 9000 Series Aggregation Services Router Modular of Service Command Reference*
User groups and task IDs	"Configuring AAA Services on Cisco ASR 9000 Series Route of Cisco Cisco ASR 9000 Series Aggregation Services Route Security Configuration Guide

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, Cisco MIB Locator found at the following URL and choose a under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html
For information about fabric scheduling, virtual output queuing (VOQ), and more, search for “voq” on community.cisco.com.	community.cisco.com
For information about session id BRKSPG-2904 and BRKARC-2003, search on Cisco Live on-demand library.	https://www.ciscolive.com/on-demand/on-demand-library