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

Feature History for Configuring Modular QoS Congestion Management on Cisco IOS XR Software

Release	Modification
Release 5.0.0	This feature was introduced.
Release 5.2.5	QoS Multipriority Queues feature was introduced.

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

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.

Note In general, the quantum size for a queue should not be smaller than the maximum transmission unit (MTU) of the interface to ensure that the scheduler always serves at least one packet from each nonempty queue.

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.

High-Priority Propagation

High-priority traffic under all ports is serviced before any low-priority traffic. This means that the scope of priority assignment at the queue level is global. This is referred to as high-priority propagation, which improves low-latency treatment for high-priority traffic, such as real-time voice and video traffic.

Priority is supported only at the queue level, or lowest-level policy map. Priority assignment at the parent level for an egress interface policy is not supported.

Egress Queuing

- Egress Queuing Only:

  The smallest step size supported is 8 kbps for 10 gigabit interfaces and 64 kbps for 100 gigabit interfaces for queues and groups. Step size increases with the rate value. Rounding error does not exceed 0.4 per cent or 8 kbps, whichever is higher.

Multi-Level Priority Queues

The Multi-Level Priority Queue (MPQ) feature allows you to configure multiple priority queues for multiple traffic classes by specifying a different priority level for each of the traffic classes in a single service policy map. You can configure multiple service policy maps per device. Having multiple priority queue enables the device to place delay-sensitive traffic on the outbound link before delay-insensitive traffic. As a result, high-priority traffic receives the lowest latency possible on the device.

While the oversubscription of priority traffic is allowed, an equal treatment of classes having the same priority level is not guaranteed. During oversubscription, priority level is strictly followed for classes with different priority levels.

Egress Minimum Bandwidth

- Minimum bandwidth of a parent class must be equal to or greater than the sum of the police rates of the high priority classes in the hierarchy. If the configured value does not meet these requirements, the minimum group bandwidth is automatically increased to satisfy the requirements. Minimum bandwidth of a parent class must be equal to or greater than the sum of the police rates of the high priority classes in the hierarchy
- Oversubscription of minimum bandwidth is permitted, except for the topmost level. In the event of oversubscription, the actual minimum bandwidth that a queue receives is proportional to its configured value.

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.

Layer-All Accounting

The Cisco NCS 6008 router uses Layer 2 accounting by default. To configure Layer-all accounting, use the **service-policy** command with **account nolayer2** keyword option. This command turns off layer 2 QoS-specific accounting and enables Layer 3 QoS accounting. Configuring Layer-all accounting on Ethernet interfaces results in 20 bytes of Layer 1 overhead in addition to the Layer 2 overhead..

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 entering the interface with traffic policing 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.

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

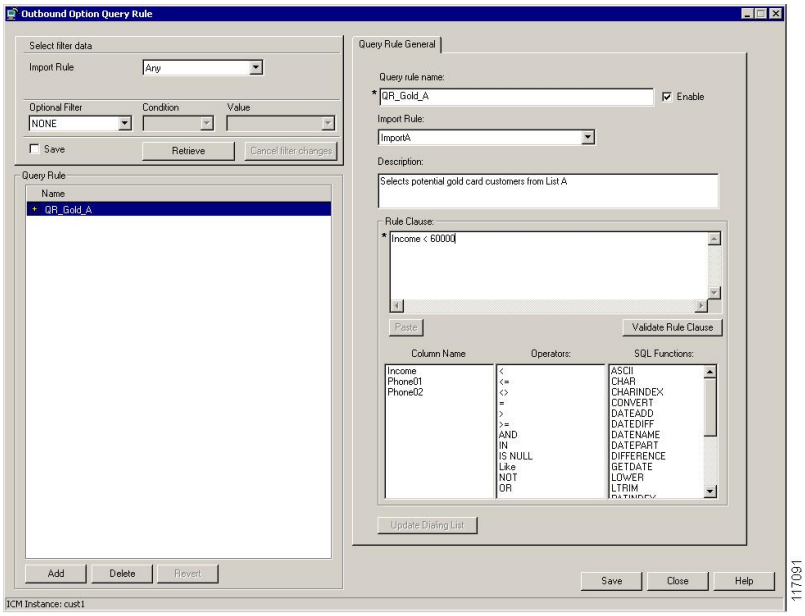
This section describes the single-rate mechanism.

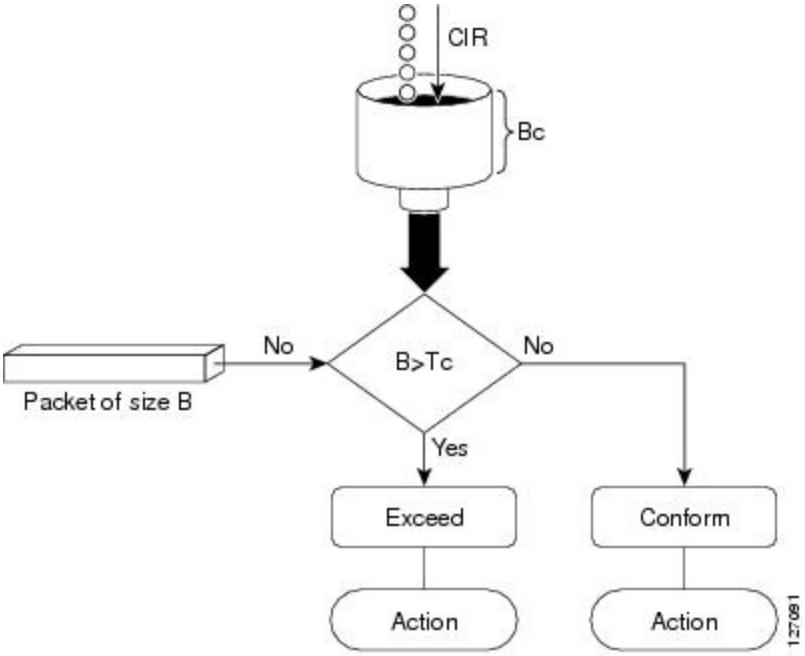
Single-Rate Policer

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 1: Marking Packets and Assigning Actions





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.

Policing on the Cisco NCS 6008 router

- The maximum number of policer buckets supported per slice is 8000 per direction (ingress or egress).

- Smallest granularity supported is 8 kbps (for rates up to 8 Mbps). The step size is higher for higher rates but is never greater than 0.2% of the rate value. For very high ratios of PIR/CIR the rounding error can be greater than 0.2%.
- The maximum permitted burst size is 2 MB for rates up to 131 Mbps, and 100 ms for higher rates.
- Burst granularity
  - For rates that are less than or equal to 131 Mbps, burst granularity varies from 128 bytes to 16,384 bytes in proportion to the burst value. The worst case rounding error is 1.6%.
  - For rates greater than 131 Mbps, the granularity is 1 ms (with the corresponding rate as reference).

Multiple Action Set

To support multiple action sets, the following combinations are supported of conform and exceed actions:

- set-mpls-exp-imp, set-clp

At least two set actions for each policer action can be configured by using the **conform-action** command, the **exceed-action** command, or the **violate-action** command within a class map for IP, MPLS data paths.

Note If partial multiple set actions are used, hierarchical policing is not supported.

This table lists the conditional policer ingress markings for IP, MPLS data paths that are applicable.

Table 1: Conditional Policer Ingress Markings for IP, MPLS

Layer 3 IP Packets	Layer 3 MPLS Packets
DSCP or precedence	MPLS experimental imposition
MPLS experimental imposition	discard-class
discard-class	qos-group
qos-group	—

Note
- Both DSCP and precedence packets are mutually exclusive.
- Both tunnel DSCP and tunnel packets markings are mutually exclusive.

This table lists the conditional egress policer markings for IP, MPLS data paths that are applicable.

Table 2: Conditional Egress Policer Markings for IP, MPLS

Layer 3 IP Packets	Layer 3 MPLS Packets
DSCP or precedence	MPLS experimental topmost
discard-class	discard-class

Note

- Both DSCP and precedence packets are mutually exclusive.

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.

Policer Granularity and Shaper Granularity

The Policer Granularity and Shaper Granularity features allow you to override the default policer and shaper granularity values.

The police rate you set should be a multiple of the policer granularity. For example, if the police rate is set to 72 kbps but the default policer granularity is 64 kbps, the effective police rate is 64 kbps. To get an actual police rate of 72 kbps, configure the policer granularity to 8 kbps. Because 72 is a multiple of 8, the police rate will be exactly 72 kbps.

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.

Shaper granularity can only be configured in the egress direction. The shape rate you set, using the **shape average** command, should be a multiple of the shaper granularity. For example, if the shape rate is set to 320 kbps but the shaper granularity is configured to 256 kbps, the effective shape rate is 512 kbps, that is a multiple of 256 kbps. To get an actual shape rate of 320 kbps, configure the shaper granularity to 64 kbps. Because 320 is a multiple of 64, the shape rate will be exactly 320 kbps.

Policer and shaper granularity values, whether default or configured, apply to the SPA Interface Processor (SIP) and to all shared port adapters (SPAs) that are installed in the SIP.

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/RP0/CPU0:router# configure`	Enters XR Config mode.

	Command or Action	Purpose
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/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/RP0/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/RP0/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/RP0/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/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 7	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class2`	Specifies the name of a different class whose policy you want to create or change.
Step 8	**bandwidth** {*rate* [*units*] | **percent** *value*} **Example:** `RP/0/RP0/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/RP0/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.

	Command or Action	Purpose
Step 10	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 11	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to XR Config mode.
Step 12	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/0`	Enters interface configuration mode and configures an interface.
Step 13	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RP0/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.
Step 15	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface HundredGigE 0/7/0/0`	(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.

Restrictions

- 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.
- The **shape average**,**bandwidth**, and **random-detect**commands cannot be configured in the same class with the **priority** command.

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/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/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/RP0/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.

	Command or Action	Purpose
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/RP0/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/RP0/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/RP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map class configuration mode.
Step 7	**priority**[**level** *priority_level*] **Example:** `RP/0/RP0/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/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 9	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to XR Config mode.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/0`	Enters interface configuration mode, and configures an interface.
Step 11	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RP0/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.

	Command or Action	Purpose
		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/RP0/CPU0:router# show policy-map interface HundredGigE 0/7/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Multi-Level Priority Queues

Before configuring multi-level priority queues in a policy map, the traffic classes, class maps, and policy maps must exist.

SUMMARY STEPS

1. **enable**
2. **Configure**
3. policy map policy name
4. class class-name
5. priority level level

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	
Step 2	**Configure**	
Step 3	policy map policy name **Example:** `RP/0/0/CPU0:ios(config)#policy-map Premium`	Creates or modifies a policy map and enters policy-map configuration mode. • Enter the name of a previously configured traffic class.
Step 4	class class-name **Example:** `RP/0/0/CPU0:ios(config-pmap)# class business`	Specifies a traffic class and enters policy-map class configuration mode. • Enter the name of a previously configured traffic class.

	Command or Action	Purpose
Step 5	priority level level **Example:** `RP/0/0/CPU0:ios(config-pmap-c)#priority level 2`	Assigns priority to a traffic class at the priority level specified.

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 outgoing interfaces is done at the Layer 2 level and includes the Layer 2 header in the rate calculation. Shaping performed on incoming interfaces is done at the Layer 3 level and does not include 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.

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/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/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.

	Command or Action	Purpose
Step 3	**class** *class-name* **Example:** `RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to XR Config mode.
Step 7	Specifies the name of the class whose policy you want to create or change.**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/0`	Enters interface configuration mode and configures an interface.
Step 8	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RP0/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.

	Command or Action	Purpose
Step 10	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface HundredGigE 0/7/0/0`	(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.

set cos is not allowed as an ingress policer action.

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*] | **percent** *percentage*]
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/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/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.

	Command or Action	Purpose
Step 3	**class** *class-name* **Example:** `RP/0/RP0/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*] | **percent** *percentage*] **Example:** `RP/0/RP0/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/RP0/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: **cos** *value*—Sets the class of service value. Range is 0 to 7. **discard-class** *value*—Sets the discard class on IP Version 4 (IPv4) or Multiprotocol Label Switching (MPLS) packets. Range is 0 to 7. **dscp** —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** —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/RP0/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:**	Returns the router to policy map class configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-pmap-c-police)# exit`	
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 9	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to XR Config mode.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/0`	Enters configuration mode and configures an interface.
Step 11	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RP0/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/RP0/CPU0:router# show policy-map interface HundredGigE 0/7/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Policing (2R3C)

SUMMARY STEPS

1. **configure**
2. **class-map** [**match-all**][**match-any**] *class-map-name*
3. **match [not] precedence [ipv4 | ipv6]***precedence_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-action** *action*
8. **exceed-action** *action*
9. **exit**
10. **exit**
11. **exit**
12. **interface** *type interface-path-id*
13. **service-policy** *policy-map*
14. Use the **commit** or **end** command.
15. **show policy-map interface** *type interface-path-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**class-map** [**match-all**][**match-any**] *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map match-all match-not-frde`	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] precedence [ipv4 | ipv6]***precedence_value* **Example:** `RP/0/RP0/CPU0:router(config)# match not precedence ipv4 5`	Specifies a precedence value that is used as the match criteria against which packets are checked to determine if they belong to the class specified by the class map.
Step 4	**policy-map** *policy-name* **Example:** `RP/0/RP0/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.

	Command or Action	Purpose
Step 5	**class** *class-name* **Example:** `RP/0/RP0/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/RP0/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-action** *action* **Example:** `RP/0/RP0/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: **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. • **transmit**—Transmits the packets.
Step 8	**exceed-action** *action* **Example:** `RP/0/RP0/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 9	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
Step 10	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.

	Command or Action	Purpose
Step 11	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to XR Config mode.
Step 12	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/0`	Enters configuration mode and configures an interface.
Step 13	**service-policy** *policy-map* **Example:** `RP/0/RP0/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 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.
Step 15	**show policy-map interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router# show policy-map interface HundredGigE 0/7/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuration Examples for Configuring Congestion Management

Traffic Shaping for an Input Interface: Example

This example shows how to configure a policy map on an input interface:

```
policy-map p2
 class voip
```

```
 shape average percent 20

!
interface bundle-pos 1
 service-policy input p2
 commit
RP/0/RP0/CPU0:Jun 8 16:55:11.819 : config[65546]: %MGBL-LIBTARCFG-6-COMMIT : Configuration
 committed by user 'cisco'. Use 'show configuration commit changes 1000006140' to view the
 changes.
```

This example show the display output for the policy-map configuration.

```
RP/0/RP0/CPU0:router#show policy-map interface bundle-e 1 in
Sat Feb 28 07:20:03.269 UTC

Bundle-Ether1 input: ingress

Class prec-1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           545449301/52363132896          5215354
    Transmitted         :           189729675/18214048800          1811636
    Total Dropped       :           355719626/34149084096          3403718
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :           189729675/18214048800          1811636
    Policed(exceed)     :           355719626/34149084096          3403718
    Policed(violate)    :                   0/0                    0
    Policed and dropped :           355719626/34149084096
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         :                   0/0                    0
    Total Dropped       :                   0/0                    0
```

Configuring Multi-Level Priority Queues: Example

The following examples show how to setup multi-level priority queues:

Class Maps

```
RP/0/RSP0/CPU0:R81#conf t
Wed Jul 15 16:52:53.003 PDT
RP/0/RSP0/CPU0:R81(config)#class-map match-any c_1
RP/0/RSP0/CPU0:R81(config-cmap)# match precedence 0
RP/0/RSP0/CPU0:R81(config-cmap)# end-class-map
RP/0/RSP0/CPU0:R81(config)#!
RP/0/RSP0/CPU0:R81(config)#class-map match-any c_2
RP/0/RSP0/CPU0:R81(config-cmap)# match precedence 1
RP/0/RSP0/CPU0:R81(config-cmap)# end-class-map
RP/0/RSP0/CPU0:R81(config)#!
RP/0/RSP0/CPU0:R81(config)#class-map match-any c_3
RP/0/RSP0/CPU0:R81(config-cmap)# match precedence 2
RP/0/RSP0/CPU0:R81(config-cmap)# end-class-map
RP/0/RSP0/CPU0:R81(config)#!
RP/0/RSP0/CPU0:R81(config)#class-map match-any c_4
RP/0/RSP0/CPU0:R81(config-cmap)# match precedence 3
RP/0/RSP0/CPU0:R81(config-cmap)# end-class-map
RP/0/RSP0/CPU0:R81(config)#!
RP/0/RSP0/CPU0:R81(config)#class-map match-any c_5
RP/0/RSP0/CPU0:R81(config-cmap)# match precedence 4
RP/0/RSP0/CPU0:R81(config-cmap)# end-class-map
RP/0/RSP0/CPU0:R81(config)#!
```

```
RP/0/RSP0/CPU0:R81(config)#class-map match-any c_6
RP/0/RSP0/CPU0:R81(config-cmap)# match precedence 5
RP/0/RSP0/CPU0:R81(config-cmap)# end-class-map
RP/0/RSP0/CPU0:R81(config)#!
RP/0/RSP0/CPU0:R81(config)#class-map match-any c_7
RP/0/RSP0/CPU0:R81(config-cmap)# match precedence 6
RP/0/RSP0/CPU0:R81(config-cmap)# end-class-map
RP/0/RSP0/CPU0:R81(config)#!
RP/0/RSP0/CPU0:R81(config)#class-map match-any c_8
RP/0/RSP0/CPU0:R81(config-cmap)# match precedence 7
RP/0/RSP0/CPU0:R81(config-cmap)# end-class-map
RP/0/RSP0/CPU0:R81(config)#!
RP/0/RSP0/CPU0:R81(config)#class-map match-any c_9
RP/0/RSP0/CPU0:R81(config-cmap)# match dscp af11
RP/0/RSP0/CPU0:R81(config-cmap)# end-class-map
RP/0/RSP0/CPU0:R81(config)#!
RP/0/RSP0/CPU0:R81(config)#class-map match-any c_10
RP/0/RSP0/CPU0:R81(config-cmap)# match dscp af12
RP/0/RSP0/CPU0:R81(config-cmap)# end-class-map
RP/0/RSP0/CPU0:R81(config)#!
RP/0/RSP0/CPU0:R81(config)#commit
```

Ingress Policy

```
RP/0/RSP0/CPU0:R81#conf t
RP/0/RSP0/CPU0:R81(config)#policy-map Ingress_Test
RP/0/RSP0/CPU0:R81(config-pmap)# class c_1
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 2
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set qos-group 0
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_2
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set qos-group 1
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 2
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_3
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 2
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set qos-group 2
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_4
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 2
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set qos-group 3
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_5
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 1
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set qos-group 4
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_6
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 1
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set qos-group 5
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_7
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 1
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set qos-group 6
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_8
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set qos-group 7
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 1
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# end-policy-map
RP/0/RSP0/CPU0:R81(config)#!
RP/0/RSP0/CPU0:R81(config)#commit
```

Egress Policy

```
RP/0/RSP0/CPU0:R81#conf t
RP/0/RSP0/CPU0:R81(config)#policy-map Egress_Test
RP/0/RSP0/CPU0:R81(config-pmap)# class c_1
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 8
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set precedence 0
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)#class c_2
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 7
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set precedence 1
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_3
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 6
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set precedence 2
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_4
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 5
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set precedence 3
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_5
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 4
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set precedence 4
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_6
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 3
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set precedence 5
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_7
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 2
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set precedence 6
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_8
RP/0/RSP0/CPU0:R81(config-pmap-c)#  set precedence 7
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 1
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)#  class class-default
RP/0/RSP0/CPU0:R81(config-pmap-c)#  bandwidth remaining percent 100
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# end-policy-map
RP/0/RSP0/CPU0:R81(config)#commit
```

Fabric Policy

```
RP/0/RSP0/CPU0:R81#conf t
RP/0/RSP0/CPU0:R81(config-pmap-c)#policy-map fabqos
RP/0/RSP0/CPU0:R81(config-pmap)# class c_1
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 2
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_2
RP/0/RSP0/CPU0:R81(config-pmap-c)#  priority level 1
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class c_3
RP/0/RSP0/CPU0:R81(config-pmap-c)#  bandwidth remaining percent 50
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# class class-default
RP/0/RSP0/CPU0:R81(config-pmap-c)# !
RP/0/RSP0/CPU0:R81(config-pmap-c)# end-policy-map
RP/0/RSP0/CPU0:R81(config)#commit
```

Traffic Policing for a Bundled Interface: Example

This example shows how to configure a policy map for a bundled interface. Note that for bundled interfaces, policing can be configured only as a percentage and not a specific rate per second:

```
policy-map p2
 class voip
 police rate 23425
!
interface bundle-pos 1
 service-policy input p2
 commit
RP/0/RP0/CPU0:Jun  8 16:51:36.623 : qos_ma[286]: %QOS-QOS_RC_QOSMGR-3-RC_BUNDLE_BW_NOPCT :
 Absolute bw specified for bundle interfaces, use percentage values instead
RP/0/RP0/CPU0:Jun  8 16:51:36.624 : qos_ma[286]: %QOS-QOS-3-MSG_SEND_FAIL : Failed to send
 message to feature rc while adding class. Error code - Invalid argument

% Failed to commit one or more configuration items during an atomic operation, no changes
have been made. Please use 'show configuration failed' to view the errors
!
```

An error occurred after the attempted commit of an invalid configuration.

```
!
!
policy-map p2
 class voip
 police rate percent 20
 commit
RP/0/RP0/CPU0:Jun  8 16:51:51.679 : config[65546]: %MGBL-LIBTARCFG-6-COMMIT : Configuration
 committed by user 'cisco'.   Use 'show configuration commit changes 1000006135' to view
the changes.
 exit
exit
interface bundle-pos 1
 service-policy input p2
 commit
RP/0/RP0/CPU0:Jun  8 16:52:02.650 : config[65546]: %MGBL-LIBTARCFG-6-COMMIT : Configuration
 committed by user 'cisco'.   Use 'show configuration commit changes 1000006136' to view
the changes.
```

This example shows the display output for the successful policy map configuration in which policing was configured as a percentage:

```
RP/0/RP0/CPU0:router#show policy-map interface bundle-e 1 in
Sat Feb 28 07:20:03.269 UTC

Bundle-Ether1 input: ingress

Class prec-1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           545449301/52363132896          5215354
    Transmitted         :           189729675/18214048800          1811636
    Total Dropped       :           355719626/34149084096          3403718
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :           189729675/18214048800          1811636
    Policed(exceed)     :           355719626/34149084096          3403718
    Policed(violate)    :                   0/0                    0
    Policed and dropped :           355719626/34149084096
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
```

```
    Matched             :                   0/0                    0
    Transmitted         :                   0/0                    0
    Total Dropped       :                   0/0                    0
```

Multiple Action Set: Examples

These examples show how to configure multiple action sets for both conditional and unconditional markings in both the ingress and egress directions:

Conditional Policer Markings in the Ingress Direction: Example

This example shows how to configure conditional policer markings in the ingress direction:

```
configure
 policy-map p1
  class c1
   police rate percent 30 peak-rate percent 50
    conform-action set precedence 2
    conform-action set mpls experimental imposition 3
    conform-action set mpls experimental topmost 4
    exceed-action set precedence 4
    exceed-action set mpls experimental imposition 5
    exceed-action set mpls experimental topmost 6
    violate-action set discard-class 3
    violate-action set qos-group 4
  !
 !
  class class-default
 !
 end-policy-map
!
end
```

If policy map p1 is applied as an ingress policy, the following action sets are applied:

- By using the **conform-action** command, IP packets are marked with the precedence value of 2 and the MPLS experimental value for the imposition label is set to 3; whereas, MPLS packets are marked with the MPLS experimental value for the imposition label that is set to 3 and the topmost label is set to 4.
- By using the **exceed-action** command, IP packets are marked with the precedence value of 4 and the MPLS experimental value for the imposition label is set to 5; whereas, MPLS packets are marked with the MPLS experimental value for the imposition label that is set to 5 and the topmost label is set to 6.
- By using the **violate-action** command, IP packets are marked with the discard class value of 3 and the QoS group value of 4; whereas, MPLS packets are marked with the discard class value of 3 and the QoS group value of 4.

Unconditional Quality-of-Service Markings in the Ingress Direction: Examples

These examples show how to configure unconditional QoS markings in the ingress direction.

Example One

```
configure
 policy-map p4
  class c1
```

```
    set discard-class 2
    set qos-group 4
    set precedence 5
    set mpls experimental imposition 3
    set mpls experimental topmost 4
    !
   class class-default
   !
  end-policy-map
!
```

If policy map p4 is applied as an ingress policy, the following sets are applied:

- IP packets are marked with the precedence value of 5 by using the **set precedence** command. The MPLS experimental value for the imposition label is marked by using the **set mpls experimental** command.
- MPLS packets are marked with MPLS experimental value of the imposition label is set to 3 and topmost label is set to 4 by using the **set mpls experimental** command.

  For both IP and MPLS packets, the discard class value is set by using the **set discard-class** command. The QoS group is set by using the **set qos-group** command.

Example Two

```
configure
 policy-map p5
  class c1
   set discard-class 2
   set qos-group 4
   set precedence 5
   set dscp tunnel 3
   set mpls experimental topmost 4
   !
  class class-default
  !
 end-policy-map
!
```

If policy map p5 is applied as an ingress policy, the following sets are applied:

- IP packets are marked with the precedence value by using the **set precedence** command. If the packets are sent out of MDT tunnel interface, they are marked with the DSCP value in the tunnel header by using the **set dscp** command.
- MPLS packets are marked with the MPLS experimental value for the topmost label by using the **set mpls experimental** command.

Example Three

```
configure
 policy-map hp
  class prec123
   service-policy child
   set discard-class 4
   set qos-group 4
   set precedence 3
   set dscp tunnel 2
   !
  class class-default
```

```
  !
 end-policy-map
!

configure
 policy-map child
  class prec1
   set discard-class 3
   set qos-group 3
   set precedence 2
   set dscp tunnel 4
   !
  class class-default
  !
 end-policy-map
!
```

If policy map hp (hierarchical policy) is applied as an ingress policy, the following sets are applied:

- IP packets with the precedence value set to 1 are marked with discard class value set to 3 by using the **set discard-class** command, qos-group value set to 3 by using the **set qos-group** command, and the precedence value set to 2 by using the **set precedence** command. If the packets are sent out of the MDT tunnel interface, they are marked with the DSVP value of 4 in the tunnel header by using the **set dscp** command.
- IP packets with precedence values of 2 and 3 are marked with discard class value set to 4, qos-group value set to 4, precedence value set to 3, and the dscp tunnel set to 2.

Conditional Policer Markings in the Egress Direction: Example

This example shows how to configure conditional policer markings in the egress direction:

```
configure
 policy-map p3
  class c1
  police rate percent 30 peak-rate percent 50
   conform-action set precedence 2
   conform-action set cos 3
   conform-action set mpls experimental topmost 3
   exceed-action set precedence 4
   exceed-action set cos 4
   exceed-action set mpls experimental topmost 4
   violate-action set discard-class 3
   violate-action set cos 5
   !
  !
 class class-default
 !
 end-policy-map
!
```

If policy map p3 is applied as an egress policy, the following action sets are applied:

- By using the **conform-action** command, IP packets are marked with the precedence value of 2 and the CoS value of 3; whereas, MPLS packets are marked with the MPLS experimental value of the topmost label that is set to 3 and the CoS value of 3.
- By using the **exceed-action** command, IP packets are marked with the precedence value of 4 and the CoS value of 4; whereas, MPLS packets are marked with the MPLS experimental value of the topmost label that is set to 4 and the CoS value of 4.

- By using the **violate-action** command, IP packets are marked with the discard class value of 3 and the CoS value of 5; whereas, MPLS packets are marked with the discard class value of 3 and the CoS value of 5.

Unconditional Quality-of-Service Markings in the Egress Direction: Example

This example shows how to configure the unconditional QoS markings in the egress direction:

```
configure
 policy-map p6
  class c1
   set cos 2
   set precedence 5
   set mpls experimental topmost 4
   !
  class class-default
  !
 end-policy-map
!
```

If policy map p6 is applied as an egress policy, the following sets are applied:

- IP packets are marked with the CoS value of 2 from the **set cos** command and the precedence value of 5 from the **set precedence** command.
- MPLS packets are marked with CoS and the MPLS experimental value for the topmost label.

Additional References

These sections provide references related to implementing QoS congestion management.

Related Documents

QoS Commands	*Modular Quality of Service Command Reference for Cisco NCS 6000 Series Routers*
User groups and task IDs	*"Configuring AAA Services on Cisco IOS XR Software"* module of *System Security Configuration Guide for Cisco NCS 6000 Series Routers*

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, use the Cisco MIB Locator found at the following URL and choose a platform 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/c/r/ciscolive/global/on-demand-library.html#/