Modular QoS Configuration Guide for Cisco NCS 6000 Series Routers, IOS XR Release 7.0.x

First Published: 2019-08-01

Last Modified: 2019-07-11

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Preface

Note This product has reached end-of-life status. For more information, see the End-of-Life and End-of-Sale Notices.

This guide describes the Cisco IOS XR QoS configurations. The preface for the *Modular QoSConfiguration Guide for Cisco NCS 6000 Series Routers* contains these sections:

- Changes to this Document, on page ix
- Communications, Services, and Additional Information, on page ix

Changes to this Document

This table lists the changes made to this document since it was first published.

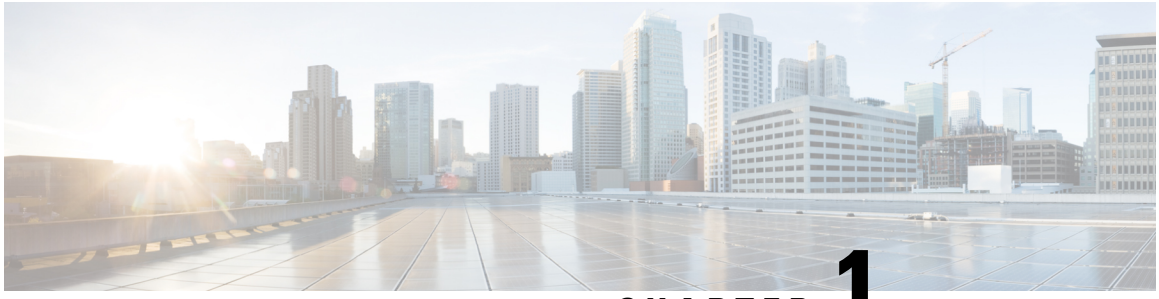
Date	Summary
August 2019	Initial release of this document.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.



CHAPTER 1

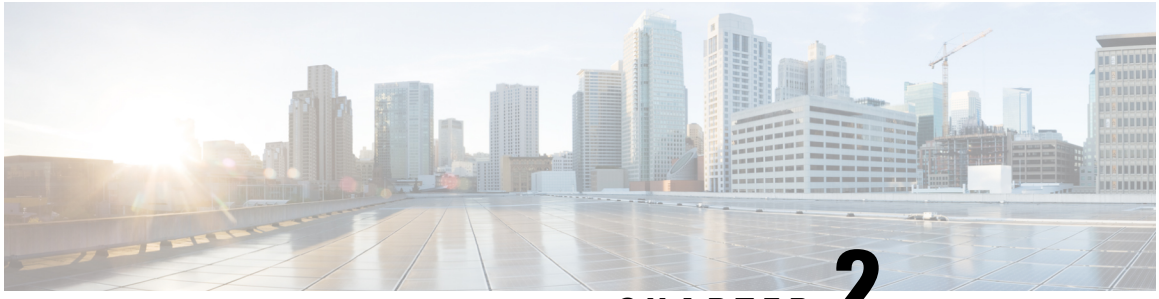
New and Changed QoS Features

New and Changed QoS Features

Table 1: QoS Features Added or Modified in IOS XR Release 7.0.x

Feature	Description	Changed in Release	Where Documented
Inter-SDR Collapsed Forwarding	The inter-SDR (secure domain routers) collapsed forwarding means that the cross-SDR interconnect (CSI) interface is used for forwarding packets. With the Ingress QoS Classify/Remark request on CSI interface feature, you can now remark the packet headers based on the policy maps attached to the CSI interface.	Release 7.0.1	Inter-SDR Collapsed Forwarding, on page 56

New and Changed QoS Features



CHAPTER 2

Modular QoS Overview

Quality of Service (QoS) is the technique of prioritizing traffic flows and providing preferential forwarding for higher-priority packets. The fundamental reason for implementing QoS in your network is to provide better service for certain traffic flows. A traffic flow can be defined as a combination of source and destination addresses, source and destination socket numbers, and the session identifier. A traffic flow can more broadly be described as a packet moving from an incoming interface that is destined for transmission to an outgoing interface. The traffic flow must be identified, classified, and prioritized on all routers and passed along the data forwarding path throughout the network to achieve end-to-end QoS delivery. The terms *traffic flow* and *packet* are used interchangeably throughout this module.

To implement QoS on a network requires the configuration of QoS features that provide better and more predictable network service by supporting bandwidth allocation, improving loss characteristics, avoiding and managing network congestion, metering network traffic, or setting traffic flow priorities across the network.

This module contains overview information about modular QoS features within a service provider network.

Information About Modular Quality of Service Overview

Before configuring modular QoS on your network, you must understand these concepts:

Benefits of Cisco IOS XR QoS Features

The Cisco IOS XR QoS features enable networks to control and predictably service a variety of networked applications and traffic types. Implementing Cisco IOS XR QoS in your network promotes these benefits:

- Control over resources. You have control over which resources (bandwidth, equipment, wide-area facilities, and so on) are being used. For example, you can limit bandwidth consumed over a backbone link by FTP transfers or give priority to an important database access.
- Tailored services. If you are an Internet Service Provider (ISP), the control and visibility provided by QoS enables you to offer carefully tailored grades of service differentiation to your customers.
- Coexistence of mission-critical applications. Cisco IOS XR QoS features ensure:
  - That bandwidth and minimum delays required by time-sensitive multimedia and voice applications are available.

- That your WAN is used efficiently by mission-critical applications that are most important to your business.
- That bandwidth and minimum delays required by time-sensitive multimedia and voice applications are available.
- That other applications using the link get their fair service without interfering with mission-critical traffic.

QoS Techniques

QoS on Cisco IOS XR software relies on these techniques to provide for end-to-end QoS delivery across a heterogeneous network:

- Packet classification
- Congestion management
- Congestion avoidance

Before implementing the QoS features for these techniques, you should identify and evaluate the traffic characteristics of your network because not all techniques are appropriate for your network environment.

Packet Classification

Packet classification techniques identify the traffic flow, and provide the capability to partition network traffic into multiple priority levels or classes of service. After is , can be

Identification of a traffic flow can be performed by using several methods within a single routeraccess control lists (ACLs), protocol match, IP precedence, IP differentiated service code point (DSCP), .

Marking of a traffic flow is performed by:

- Setting IP Precedence or DSCP bits in the IP Type of Service (ToS) byte.
- Setting EXP bits within the imposed or the topmost Multiprotocol Label Switching (MPLS) label.
- Setting qos-group and discard-class bits.

Marking can be carried out:

- Unconditionally—As part of the class-action.
- Conditionally—As part of a policer-action.
- Combination of conditionally and unconditionally.

For detailed conceptual and configuration information about packet marking, see the Configuring Modular Quality of Service Packet Classification on Cisco IOS XR Software module for unconditional marking, and Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Softwaremodule for conditional marking.

Congestion Management

Congestion management techniques control congestion after it has occurred. One way that network elements handle an overflow of arriving traffic is to use a queuing algorithm to sort the traffic, then determine some servicing method of prioritizing it onto an output link.

Cisco IOS XR software implements the low-latency Queuing (LLQ) feature, which brings strict priority queuing (PQ) to the Modified Deficit Round Robin (MDRR) scheduling mechanism. LLQ with strict PQ allows delay-sensitive data such as voice, to be dequeued and sent before packets in other queues are dequeued.

Cisco IOS XR software includes traffic policing capabilities available on a per-class basis as well as class-based shaping.

The traffic policing feature limits the input or output transmission rate of a class of traffic based on user-defined criteria, and can mark packets by setting values such as IP Precedence, QoS group, or DSCP value.

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

Cisco IOS XR software supports a class-based traffic shaping method through a CLI mechanism in which parameters are applied per class.

For detailed conceptual and configuration information about congestion management, see the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Softwaremodule.

Congestion Avoidance

Congestion avoidance techniques monitor network traffic flows in an effort to anticipate and avoid congestion at common network and internetwork bottlenecks before problems occur. These techniques are designed to provide preferential treatment for traffic (such as a video stream) that has been classified as real-time critical under congestion situations while concurrently maximizing network throughput and capacity utilization and minimizing packet loss and delay. Cisco IOS XR software supports the Random Early Detection (RED), Weighted RED (WRED), and tail drop QoS congestion avoidance features.

For detailed conceptual and configuration information about congestion avoidance techniques, see the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.

Differentiated Service Model for Cisco IOS XR Software

Cisco IOS XR software supports a differentiated service that is a multiple-service model that can satisfy different QoS requirements. However, unlike in the integrated service model, an application using differentiated service does not explicitly signal the router before sending data.

For differentiated service, the network tries to deliver a particular kind of service based on the QoS specified by each packet. This specification can occur in different ways, for example, using the IP Precedence bit settings in IP packets or source and destination addresses. The network uses the QoS specification to classify, mark, shape, and police traffic, and to perform intelligent queuing.

The differentiated service model is used for several mission-critical applications and for providing end-to-end QoS. Typically, this service model is appropriate for aggregate flows because it performs a relatively coarse level of traffic classification.

Additional Cisco IOS XR QoS Supported Features

These sections describe the additional features that play an important role in the implementation of QoS on Cisco IOS XR software.

Modular QoS Command-Line Interface

In Cisco IOS XR software, QoS features are enabled through the Modular QoS command-line interface (MQC) feature. The *MQC* is a command-line interface (CLI) structure that allows you to create policies and attach these policies to interfaces. A traffic policy contains a traffic class and one or more QoS features. A traffic class is used to classify traffic, whereas the QoS features in the traffic policy determine how to treat the classified traffic. One of the main goals of MQC is to provide a platform-independent interface for configuring QoS across Cisco platforms.

For detailed conceptual and configuration information about the MQC feature, see the Configuring Modular Quality of Service Packet Classification on Cisco IOS XR Software module.

Fabric QoS

The fabricq queue selection mechanism is known as *Fabric QoS*. These ports are defined:

- High-priority port for internal control traffic and classified high-priority traffic

Each port has a queue associated with every physical interface on the line card. The queues map to a physical egress interface and are assigned when the interface is created. The associated quanta for each of the queues are derived from the bandwidth of the physical or logical interfaces in relative terms to the other interfaces present on that line card or PLIM.

Where to Go Next

To configure the packet classification features that involve identification and marking of traffic flows, see the Configuring Modular Quality of Service Packet Classification module in this guide.

To configure the queuing, scheduling, policing, and shaping features, see the Configuring Modular Quality of Service Congestion Management module in this guide.

To configure the WRED and RED features, see the Configuring Modular QoS Congestion Avoidance module in this guide.

To configure fabric QoS, see the Configuring Fabric Quality of Service Policies and Classes on Cisco IOS XR Software module.

Additional References

The following sections provide references related to implementing QoS.

Related Documents

QoS Commands	*Modular Quality of Service Command Reference for Cisco N Series Routers*

User groups and task IDs	*"Configuring AAA Services on Cisco IOS XR Software"* r *System Security Configuration Guide for Cisco NCS 6000 S*

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

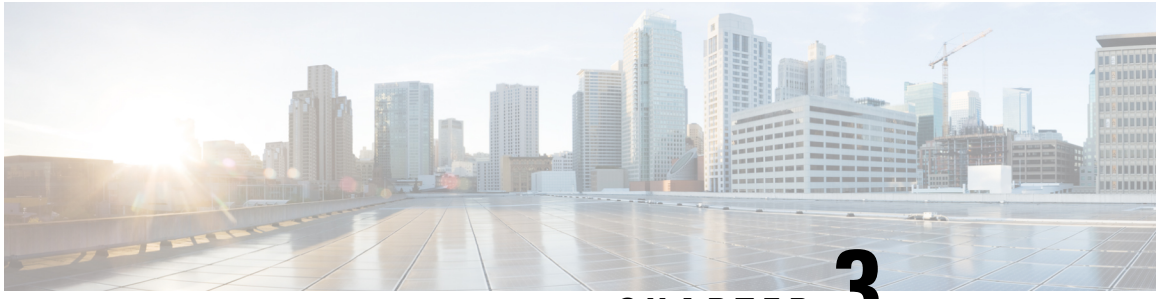
MIBs	MIBs Link
	To locate and download MIBs using Cisco IOS XR softw Cisco MIB Locator found at the following URL and choo under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.sht

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



CHAPTER 3

Configuring Modular QoS Congestion Avoidance

Congestion avoidance techniques monitor traffic flow in an effort to anticipate and avoid congestion at common network bottlenecks. Avoidance techniques are implemented before congestion occurs as compared with congestion management techniques that control congestion after it has occurred.

Congestion avoidance is achieved through packet dropping. Cisco IOS XR software supports these quality of service (QoS) congestion avoidance techniques that drop packets:

- Random early detection (RED)
- Weighted random early detection (WRED)
- Tail drop

The module describes the concepts and tasks related to these congestion avoidance techniques.

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

Release	Modification
Release 5.0.0	This feature was introduced.

Prerequisites for Configuring Modular QoS Congestion Avoidance

This prerequisite is required for configuring QoS congestion avoidance 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.

Information About Configuring Modular QoS Congestion Avoidance

Random Early Detection and TCP

The Random Early Detection (RED) congestion avoidance technique takes advantage of the congestion control mechanism of TCP. By randomly dropping packets prior to periods of high congestion, RED tells the packet source to decrease its transmission rate. Assuming the packet source is using TCP, it decreases its transmission rate until all packets reach their destination, indicating that the congestion is cleared. You can use RED as a way to cause TCP to slow transmission of packets. TCP not only pauses, but it also restarts quickly and adapts its transmission rate to the rate that the network can support.

RED distributes losses in time and maintains normally low queue depth while absorbing traffic bursts. When enabled on an interface, RED begins dropping packets when congestion occurs at a rate you select during configuration.

Average Queue Size for WRED

The router automatically determines the parameters to use in the WRED calculations. The average queue size is based on the previous average and current size of the queue. The formula is:

average = (old_average * (1-2 -x)) + (current_queue_size * 2 -x)

where *x* is the exponential weight factor.

For high values of *x*, the previous average becomes more important. A large factor smooths out the peaks and lows in queue length. The average queue size is unlikely to change very quickly, avoiding a drastic change in size. The WRED process is slow to start dropping packets, but it may continue dropping packets for a time after the actual queue size has fallen below the minimum threshold. The slow-moving average accommodates temporary bursts in traffic.

Note The exponential weight factor, *x*, is fixed and is not user configurable.

Note If the value of *x* gets too high, WRED does not react to congestion. Packets are sent or dropped as if WRED were not in effect.

For low values of *x*, the average queue size closely tracks the current queue size. The resulting average may fluctuate with changes in the traffic levels. In this case, the WRED process responds quickly to long queues. Once the queue falls below the minimum threshold, the process stops dropping packets.

If the value of *x* gets too low, WRED overreacts to temporary traffic bursts and drops traffic unnecessarily.

Tail Drop and the FIFO Queue

Tail drop is a congestion avoidance technique that drops packets when an output queue is full until congestion is eliminated. Tail drop treats all traffic flow equally and does not differentiate between classes of service. It manages the packets that are unclassified, placed into a first-in, first-out (FIFO) queue, and forwarded at a rate determined by the available underlying link bandwidth.

See the "Default Traffic Class" section of the "Configuring Modular Quality of Service Packet Classification on Cisco IOS XR Software" module.

Configuring Random Early Detection

This configuration task is similar to that used for WRED except that the **random-detect precedence** command is not configured and the **random-detect** command with the **default** keyword must be used to enable RED.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-map-name*
3. **class** *class-name*
4. **random-detect** {**cos** *value* | **default** | **discard-class** *value* | **dscp** *value* | **exp** *value* | **precedence** *value* | *min-threshold* [*units*] *max-threshold* [*units*] }
5. **bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value*
6. **shape average** {**percent** *percentage* | *value* [*units*]}
7. **exit**
8. **exit**
9. **interface** *type interface-path-id*
10. **service-policy** {**input** | **output**} *policy-map*
11. Use the **commit** or **end** command.

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-map-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 and enters the policy map class configuration mode.

	Command or Action	Purpose
Step 4	**random-detect** {**cos** *value* | **default** | **discard-class** *value* | **dscp** *value* | **exp** *value* | **precedence** *value* | *min-threshold* [*units*] *max-threshold* [*units*] } **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# random-detect default`	Enables RED with default minimum and maximum thresholds.

Step 5	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value*
	Example:
	`RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth percent 30`
	or
	`RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`
		(Optional) Specifies the bandwidth allocated for a class belonging to a policy map.
		or
		(Optional) Specifies how to allocate leftover bandwidth to various classes.

Step 6	**shape average** {**percent** *percentage* | *value* [*units*]} **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# shape average percent 50`	(Optional) Shapes traffic to the specified bit rate or a percentage of the available bandwidth.
Step 7	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to XR Config mode.
Step 9	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundGigE 0/2/0/0`	Enters the configuration mode and configures an interface.
Step 10	**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.

	Command or Action	Purpose
Step 11	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 mode, without committing the configuration changes.

Configuring Weighted Random Early Detection

WRED drops packets selectively based on IP precedence. Edge routers assign IP precedences to packets as they enter the network. WRED uses these precedences to determine how to treat different types of traffic.

When a packet arrives, the following actions occur:

- The average queue size is calculated.
- If the average queue size is less than the minimum queue threshold, the arriving packet is queued.
- If the average queue size is between the minimum queue threshold for that type of traffic and the maximum threshold for the interface, the packet is either dropped or queued, depending on the packet drop probability for that type of traffic.
- If the average queue size is greater than the maximum threshold, the packet is dropped.

Restrictions

- You cannot configure WRED in a class that has been set for priority queueing (PQ).
- You cannot use the **random-detect** command in a class configured with the **priority** command.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **random-detect dscp** *dscp-value min-threshold* [*units*] *max-threshold* [*units*]
5. **bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value]*
6. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}
7. **bandwidth remaining percent** *value*
8. **shape average** {**percent** *percentage* | *value* [*units*]}
9. **queue-limit** *value* [*units*]
10. **exit**
11. **interface** *type interface-path-id*

12. **service-policy** {**input** | **output**} *policy-map*
13. Use the **commit** or **end** command.

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.
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	**random-detect dscp** *dscp-value min-threshold* [*units*] *max-threshold* [*units*] **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# random-detect dscp af11 1000000 bytes 2000000 bytes`	Modifies the minimum and maximum packet thresholds for the DSCP value. • Enables RED. • *dscp-value*—Number from 0 to 63 that sets the DSCP value. Reserved keywords can be specified instead of numeric values. • *min-threshold*—Minimum threshold in the specified units. The value range of this argument is from 0 to 1073741823. When the average queue length reaches the minimum threshold, WRED randomly drops some packets with the specified DSCP value. • *max-threshold*—Maximum threshold in the specified units. The value range of this argument is from 0 to 1073741823. When the average queue length exceeds the maximum threshold, WRED drops all packets with the specified DSCP value. • *units*—Units of the threshold value. This can be **bytes**, **gbytes**, **kbytes**, **mbytes**, **ms** (milliseconds), **packets**, or **us** (microseconds). The default is **packets**. • This example shows that for packets with DSCP AF11, the WRED minimum threshold is 1,000,000 bytes and maximum threshold is 2,000,000 bytes.
Step 5	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value]*	(Optional) Specifies the bandwidth allocated for a class belonging to a policy map.

	Command or Action	Purpose

	Example:
	`RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth percent 30`
	or
	`RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`
		or
		(Optional) Specifies how to allocate leftover bandwidth to various classes.

Step 6	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth percent 30`	(Optional) Specifies the bandwidth allocated for a class belonging to a policy map. This example guarantees 30 percent of the interface bandwidth to class class1.
Step 7	**bandwidth remaining percent** *value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	(Optional) Specifies how to allocate leftover bandwidth to various classes. • The remaining bandwidth of 70 percent is shared by all configured classes. • In this example, class class1 receives 20 percent of the 70 percent.
Step 8	**shape average** {**percent** *percentage* | *value* [*units*]} **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# shape average percent 50`	(Optional) Shapes traffic to the specified bit rate or a percentage of the available bandwidth.
Step 9	**queue-limit** *value* [*units*] **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# queue-limit 50 ms`	(Optional) Changes queue-limit to fine-tune the amount of buffers available for each queue. The default queue-limit is 100 ms of the service rate for a non-priority class and 10ms of the service rate for a priority class. **Note** Even though this command is optional, it is recommended that you use it to fine-tune the queue limit, instead of relying on your system default settings. If the queue limit is too large, the buffer consumption goes up, resulting in delays. On the other hand, too small a queue limit may result in extra drops while allowing for faster rate adaption.
Step 10	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to XR Config mode.

	Command or Action	Purpose
Step 11	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundGigE 0/2/0/0`	Enters the configuration mode and configures an interface.
Step 12	**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. • Ingress policies are not valid; the **bandwidth** and **bandwidth remaining** commands cannot be applied to ingress policies.
Step 13	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.

Configuring Tail Drop

Packets satisfying the match criteria for a class accumulate in the queue reserved for the class until they are serviced. The **queue-limit** command is used to define the maximum threshold for a class. When the maximum threshold is reached, enqueued packets to the class queue result in tail drop (packet drop).

The **queue-limit** value uses the guaranteed service rate (GSR) of the queue as the reference value for the **queue_bandwidth**. If the class has bandwidth percent associated with it, the **queue-limit** is set to a proportion of the bandwidth reserved for that class.

If the GSR for a queue is zero, use the following to compute the default **queue-limit**:

- 1 percent of the interface bandwidth for queues in a nonhierarchical policy.
- 1 percent of parent maximum reference rate for hierarchical policy.

  The parent maximum reference rate is the minimum of parent shape, policer maximum rate, and the interface bandwidth.

The default **queue-limit** is set as follows:

- **For priority class**

default queue limit (in bytes) = (<10> ms * queue_bandwidth kbps) / 8

- **For non-priority class**

  default queue limit (in bytes) = (<100> ms * queue_bandwidth kbps) / 8

Note You can configure a maximum queue threshold up to 1GB, which translates to 80ms of buffering for 100Gbps queue.

Restrictions

- When configuring the **queue-limit** command in a class, you must configure one of the following commands: **priority**, **shape average**, **bandwidth**, or **bandwidth remaining**, except for the default class.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **queue-limit** *value* [*units*]
5. **priority** [**level** *priority-level*]
6. **police rate percent** *percentage*
7. **class** *class-name*
8. **bandwidth** {*bandwidth* [*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.

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 also 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	**queue-limit** *value* [*units*] **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# queue-limit 1000000 bytes`	Specifies or modifies the maximum the queue can hold for a class policy configured in a policy map. The default value of the *units* argument is **packets**. In this example, when the queue limit reaches 1,000,000 bytes, enqueued packets to the class queue are dropped.
Step 5	**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.
Step 6	**police rate percent** *percentage* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# police rate percent 30`	Configures traffic policing.
Step 7	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class2`	Specifies the name of the class whose policy you want to create or change. In this example, class2 is configured.
Step 8	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth percent 30`	(Optional) Specifies the bandwidth allocated for a class belonging to a policy map. This example guarantees 30 percent of the interface bandwidth to class class2.
Step 9	**bandwidth remaining percent** *value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	(Optional) Specifies how to allocate leftover bandwidth to various classes. This example allocates 20 percent of the leftover interface bandwidth to class class2.
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:**	Returns the router to XR Config mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-pmap)# exit`	
Step 12	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/0`	Enters the 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.

Additional References

These sections provide references related to implementing QoS congestion avoidance.

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

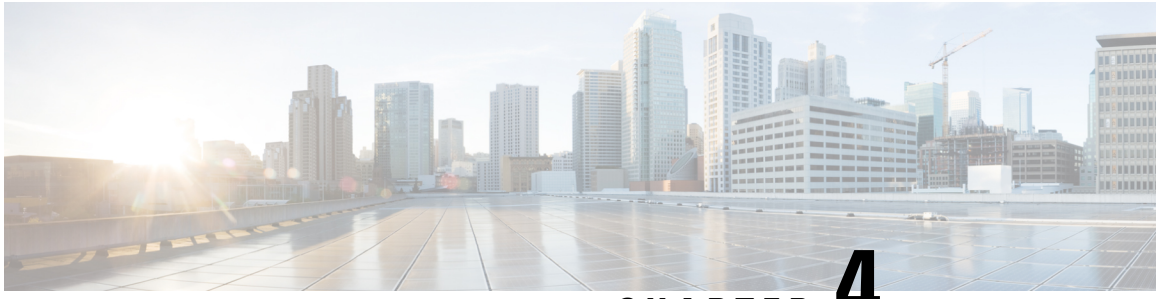
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, Cisco MIB Locator found at the following URL and choose a under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

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



CHAPTER 4

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

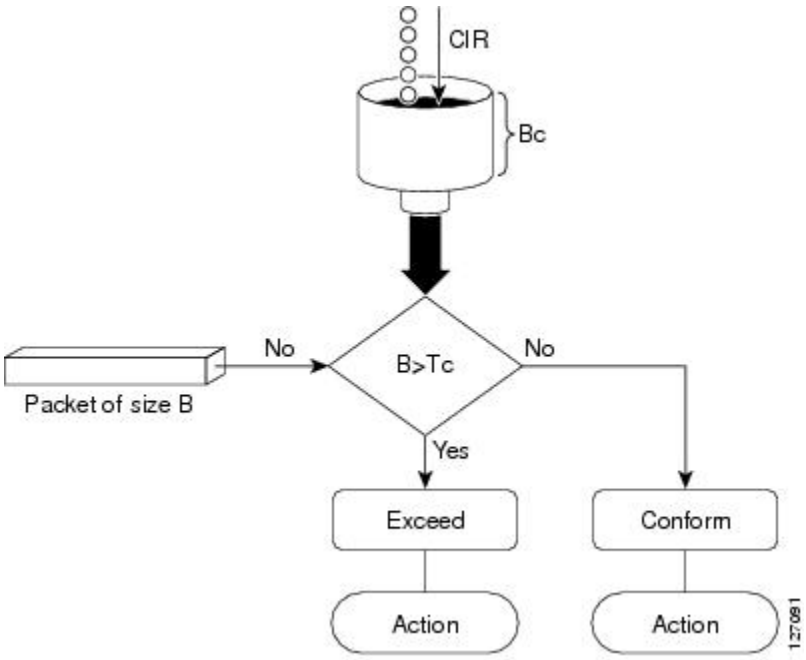
This section describes the single-rate mechanism.

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

- 1Tbs line card Police rates can be configured in the range of 8 Kbps - 134 Gbps for a slice. 2Tbs line card Police rates can be configured in the range of 32 Kbps - 340 Gbps for a slice.
- 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 2: 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 3: 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.
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:**	Specifies how to allocate leftover bandwidth to various classes.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	**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.
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:**	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-if)# service-policy output policy1`	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.
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.

	Command or Action	Purpose
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. **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.
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.
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:**	Enters interface configuration mode and configures an interface.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/0`	
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.
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.
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.

	Command or Action	Purpose
		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:** `RP/0/RP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
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.

	Command or Action	Purpose
		• **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.

	Command or Action	Purpose
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.
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.

	Command or Action	Purpose
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.
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 Cisc Series Routers*
User groups and task IDs	*"Configuring AAA Services on Cisco IOS XR Software"* r *System Security Configuration Guide for Cisco NCS 6000 S*

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

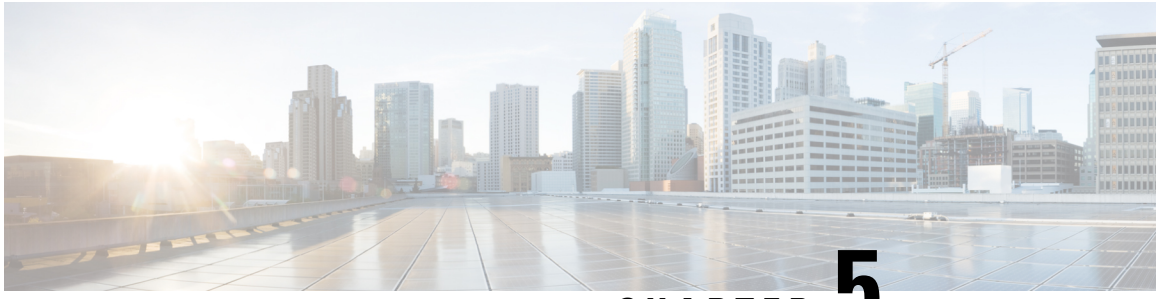
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, Cisco MIB Locator found at the following URL and choose a under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

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.htm



CHAPTER 5

Configuring Modular QoS Service Packet Classification

Packet classification identifies and marks traffic flows that require congestion management or congestion avoidance on a data path. The Modular Quality of Service (QoS) command-line interface (MQC) is used to define the traffic flows that should be classified, where each traffic flow is called a class of service, or class. Subsequently, a traffic policy is created and applied to a class. All traffic not identified by defined classes falls into the category of a default class.

This module provides the conceptual and configuration information for QoS packet classification.

Feature History for Configuring Modular QoS Packet Classification on Cisco IOS XR Software

Release	Modification
Release 5.0.0	This feature was introduced.

Prerequisites for Configuring Modular QoS Packet Classification

These prerequisites are required for configuring modular QoS packet classification 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 Modular QoS Packet Classification

Packet Classification Overview

Packet classification involves categorizing a packet within a specific group (or class) and assigning it a traffic descriptor to make it accessible for QoS handling on the network. The traffic descriptor contains information about the forwarding treatment (quality of service) that the packet should receive. Using packet classification, you can partition network traffic into multiple priority levels or classes of service. The source agrees to adhere to the contracted terms and the network promises a quality of service. Traffic policers and traffic shapers use the traffic descriptor of a packet to ensure adherence to the contract.

Traffic policers and traffic shapers rely on packet classification features, such as IP precedence, to select packets (or traffic flows) traversing a router or interface for different types of QoS service. For example, by using the three precedence bits in the type of service (ToS) field of the IP packet header, you can categorize packets into a limited set of up to eight traffic classes. After you classify packets, you can use other QoS features to assign the appropriate traffic handling policies including congestion management, bandwidth allocation, and delay bounds for each traffic class.

Methods of classification may consist of the logical combination of any fields in the packet header, where a packet header may be a Layer 2, a Layer 3, or a Layer 4 header; or classification based on the incoming or outgoing physical or virtual interface.

Traffic Class Elements

The purpose of a traffic class is to classify traffic on your router. Use the **class-map** command to define a traffic class.

A traffic class contains three major elements: a name, a series of **match** commands, and, if more than one **match** command exists in the traffic class, an instruction on how to evaluate these **match** commands. The traffic class is named in the **class-map** command. For example, if you use the word *cisco* with the **class-map** command, the traffic class would be named *cisco*.

Note From Release 5.2.1, the **class-map** command supports both **match-any** and **match-all** keywords.

The **match** commands are used to specify various criteria for classifying packets. Packets are checked to determine whether they match the criteria specified in the **match** commands. If a packet matches the specified criteria, that packet is considered a member of the class and is forwarded according to the QoS specifications set in the traffic policy. Packets that fail to meet any of the matching criteria are classified as members of the default traffic class. See the Default Traffic Class.

The instruction on how to evaluate these **match** commands needs to be specified if more than one match criterion exists in the traffic class. The evaluation instruction is specified with the **class-map [match-any] [match-all]** command. If the **match-any** option is specified as the evaluation instruction, the traffic being evaluated by the traffic class must match at least one of the specified criteria. If the **match-all** option is specified, the traffic must match all of the match criteria.

Note Users can provide multiple values for a match type in a single line of configuration; that is, if the first value does not meet the match criteria, then the next value indicated in the match statement is considered for classification.

This table lists the traffic class match criteria supported on the router.

Note Unless otherwise indicated, the match criteria for Layer 3 physical interfaces applies to bundle interfaces.

Table 4: Supported Traffic Class Match Criteria

Match Criteria	Layer 3 Ingress			Layer 3 Egress		
	Phy	SIf	P-SIf	Phy	SIf	P-S
prec	Y	Y	Y	Y	Y	Y
dscp	Y	Y	Y	Y	Y	Y
vlan	Y	N	Y	Y	N	Y
CoS	Y	Y9	Y9	N	N	N
qos-group	Y	Y	Y	Y	Y	Y
discard-class	N	N	N	Y	Y	Y
EXP	Y	Y	Y	Y	Y	Y
protocol	Y	Y	Y	Y	Y	Y
access-group	Y	Y	Y	Y	Y	Y
vpls known	N	N	N	N	N	N
vpls unknown	N	N	N	N	N	N
vpls multicast	N	N	N	N	N	N
match atm-clp	N	N	N	N	N	N

Note
- The match qos-group command is supported on ingress (Layer 3) for QPPB only.
- PAC stands for port attachment circuit.
- CAC stands for customer attachment circuit.
- p-C stands for physical interface with underlying CACs.
- Phy stands for physical interface, SIIf stands for subinterface, and P-SIf stands for physical interface with underlying subinterfaces.
- The match atm-clp is the only match criteria supported on ATM interfaces.

The traffic class configuration task is described in the Creating a Traffic Class.

Traffic Policy Elements

The purpose of a traffic policy is to configure the QoS features that should be associated with the traffic that has been classified in a user-specified traffic class or classes. The **policy-map** command is used to create a traffic policy. A traffic policy contains three elements: a name, a traffic class (specified with the **class** command), and the QoS policies. The name of a traffic policy is specified in the policy map MQC (for example, the **policy-map** *policy1* command creates a traffic policy named *policy1*). The traffic class that is used to classify traffic to the specified traffic policy is defined in class map configuration mode. After choosing the traffic class that is used to classify traffic to the traffic policy, the user can enter the QoS features to apply to the classified traffic.

The MQC does not necessarily require that users associate only one traffic class to one traffic policy. When packets match to more than one match criterion, as many as 1024 traffic classes can be associated to a single traffic policy. The 1024 class maps include the default class and the classes of the child policies, if any.

The order in which classes are configured in a policy map is important. The match rules of the classes are programmed into the TCAM in the order in which the classes are specified in a policy map. Therefore, if a packet can possibly match multiple classes, only the first matching class is returned and the corresponding policy is applied.

The function of these commands is described more thoroughly in the *Modular Quality of Service Command Reference for Cisco NCS 6000 Series RoutersModular Quality of Service Command Reference*.

The traffic policy configuration task is described in Creating a Traffic Policy.

Limitation

Fragmented IPv4 packets are subjected to egress QoS policies only on the main interface and not on sub-interfaces. The fragmented IPv4 packets are subjected to the Local Packet Transport Services (LPTS) policer. IPv4 packets are fragmented when the egress interface MTU is smaller than the packet size.

Default Traffic Class

Unclassified traffic (traffic that does not meet the match criteria specified in the traffic classes) is treated as belonging to the default traffic class.

If the user does not configure a default class, packets are still treated as members of the default class. However, by default, the default class has no enabled features. Therefore, packets belonging to a default class with no configured features have no QoS functionality. These packets are then placed into a first in, first out (FIFO) queue and forwarded at a rate determined by the available underlying link bandwidth. This FIFO queue is managed by a congestion avoidance technique called tail drop.

For further information about congestion avoidance techniques, such as tail drop, see Configuring Modular QoS Congestion Avoidance on Cisco IOS XR Software module.

Class-based Unconditional Packet Marking Feature and Benefits

The Class-based, Unconditional Packet Marking feature provides users with a means for efficient packet marking by which the users can differentiate packets based on the designated markings.

The Class-based, Unconditional Packet Marking feature allows users to perform these tasks:

- Mark packets by setting the IP precedence bits or the IP differentiated services code point (DSCP) in the IP ToS byte.
- Mark Multiprotocol Label Switching (MPLS) packets by setting the EXP bits within the imposed or topmost label.
- Mark packets by setting the value of the *qos-group* argument.
- Mark packets by setting the value of the *discard-class* argument.

Note When the router receives multicast traffic from a Multicast Label Distribution Protocol (MLDP) solution, the MPLS label from the received packet is not dispositioned at the ingress line-card. Instead, the label is removed at the egress line-card. As a result, you cannot mark the IP header for incoming multicast traffic in an MLDP scenario. This means that such packets will not be marked with a Differentiated Services Code Point (DSCP) or precedence value. This is expected behavior for the line cards listed below and is applicable for unconditional marking and for packet marking as policer action (also known as conditional marking):

- ASR 9000 Ethernet Line Cards
- Cisco ASR 9000 High Density 100GE Ethernet line cards

Unconditional packet marking allows you to partition your network into multiple priority levels or classes of service, as follows:

- Use QoS unconditional packet marking to set the IP precedence or IP DSCP values for packets entering the network. Routers within your network can then use the newly marked IP precedence values to determine how the traffic should be treated.

  For example, weighted random early detection (WRED), a congestion avoidance technique, uses IP precedence values to determine the probability that a packet is dropped. In addition, low-latency queuing (LLQ) can then be configured to put all packets of that mark into the priority queue.
- Use QoS unconditional packet marking to assign MPLS packets to a QoS group. The router uses the QoS group to determine how to prioritize packets for transmission. To set the QoS group identifier on MPLS packets, use the **set qos-group** command in policy map class configuration mode.

The configuration task is described in the Configuring Class-based Unconditional Packet Marking.

This table shows the supported class-based unconditional packet marking operations.

Note Unless otherwise indicated, the class-based unconditional packet marking for Layer 3 physical interfaces applies to bundle interfaces.

Table 5: Supported Class-based Unconditional Packet Marking Operations

Marking Operation	Layer 3 Ingress			Layer 3 Egress		
	Phy	SIf	P-SIf	Phy	SIf	P-Sif
prec	Y	Y	Y	Y	Y	Y

Marking Operation	Layer 3 Ingress			Layer 3 Egress		
dscp	Y	Y	Y	Y	Y	Y
CoS	N	N	N	Y	Y	Y
qos-group	Y	Y	Y	N	N	N
discard-class	Y	Y	Y	N	N	N
EXP, imposition	Y	Y	Y	N	N	N
EXP, topmost	Y	Y	Y	Y	Y	Y

Note

- PAC stands for port attachment circuit.
- CAC stands for customer attachment circuit.
- Phy stands for physical interface, SIIf stands for subinterface, and P-SIf stands for physical interface with underlying subinterfaces.
- p-C stands for physical interface with underlying CACs.
- The atm-clp is the only command supported on ATM subinterfaces.

Inter-SDR Collapsed Forwarding

The inter-SDR (secure domain routers) collapsed forwarding means that the cross-SDR interconnect (CSI) interface is used for forwarding packets. With the Ingress QoS Classify/Remark request on CSI interface feature, you can now remark the packet headers based on the policy maps attached to the CSI interface.

When the packets arrive at the ingress of an SDR through the CSI interface, the packets are remarked according to the ingress policy map defined for that CSI interface.

Using this remarking feature, you can differentially treat the packets according to the remarked fields in the egress path.

Restrictions

- The CSI interface currently supports only unconditional marking.
- Only ingress remarking is supported in collapsed forwarding.
- Only the unconditional remarking feature is supported. Other QoS features are not supported over CSI.

Unconditional Multiple Action Set

The Unconditional Multiple Action Set feature allows you to mark packets with unconditional multiple action sets through a class map. These unconditional markings are supported.

Unconditional Ingress Markings

Both the discard-class and qos-group packets are marked independent of other markings. In addition to the discard-class and qos-group, at the maximum, two set actions are supported in each of the data paths (IP, MPLS, and Layer 2).

In a hierarchical policy, markings for a parent and its child classes in the same hierarchy cannot exceed more than two actions, which is different from the discard-class and qos-group packets.

If the same type of marking is configured in both parent and child, it is considered as a single marking as child marking overrides the parent marking.

This table lists the unconditional QoS ingress markings that are supported for IP, MPLS, or Layer 2 data paths.

Note Unless otherwise indicated, the unconditional QoS ingress marking for Layer 3 physical interfaces applies to bundle interfaces.

Common Packets for IP, MPLS, or Layer 2	Layer 3 IP Packets	Layer 3 MPLS Packets
discard-class	dscp or precedence	MPLS experimental imposition
qos-group	—	MPLS experimental topmost
—	MPLS experimental imposition	—
—	—	—
—	—	—

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

Unconditional Egress Markings

The cos packets are marked as independent from other markings. In addition to the cos packets, one more set actions are supported in the IP and MPLS data paths. In a hierarchical policy, markings for a parent and its child classes in the same hierarchy cannot exceed more than two, which is different from the cos packets.

If the same type of marking is used for both a parent and child, the marking type is considered as a single marking as the marking at child level overrides the marking at parent level.

This table lists the unconditional QoS egress markings that are supported for IP, MPLS, or Layer 2 data paths.

Note Unless otherwise indicated, the unconditional QoS egress marking for Layer 3 physical interfaces applies to bundle interfaces.

Common Packets for IP, MPLS, or Layer 2	Layer 3 IP Packets	Layer 3 MPLS Packets
cos	dscp or precedence MPLS Experimental Imposition	MPLS experimental topmost MPLS Experimental Imposition

Note

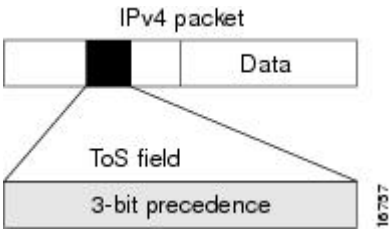
- Both DSCP and precedence packets are mutually exclusive.

Note For a list of supported conditional marking operations, see Table 3 in the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.

Specification of the CoS for a Packet with IP Precedence

Use of IP precedence allows you to specify the CoS for a packet. You use the three precedence bits in the ToS field of the IP version 4 (IPv4) header for this purpose. This figure shows the ToS field.

Figure 2: IPv4 Packet Type of Service Field



Using the ToS bits, you can define up to eight classes of service. Other features configured throughout the network can then use these bits to determine how to treat the packet in regard to the ToS to grant it. These other QoS features can assign appropriate traffic-handling policies, including congestion management strategy and bandwidth allocation. For example, although IP precedence is not a queuing feature, queuing features, such as LLQ, can use the IP precedence setting of the packet to prioritize traffic.

By setting precedence levels on incoming traffic and using them in combination with the Cisco IOS XR QoS queuing features, you can create differentiated service.

So that each subsequent network element can provide service based on the determined policy, IP precedence is usually deployed as close to the edge of the network or administrative domain as possible. You can think of IP precedence as an edge function that allows core, or backbone, QoS features, such as WRED, to forward traffic based on CoS. IP precedence can also be set in the host or network client, but this setting can be overridden by policy within the network.

The configuration task is described in the Configuring Class-based Unconditional Packet Marking.

IP Precedence Bits Used to Classify Packets

Use the three IP precedence bits in the ToS field of the IP header to specify the CoS assignment for each packet. As mentioned earlier, you can partition traffic into a maximum of eight classes and then use policy maps to define network policies in terms of congestion handling and bandwidth allocation for each class.

For historical reasons, each precedence corresponds to a name. These names are defined in RFC 791. This table lists the numbers and their corresponding names, from least to most important.

Table 6: IP Precedence Values

Number	Name
0	routine
1	priority
2	immediate
3	flash
4	flash-override
5	critical
6	internet
7	network

The IP precedence feature allows you considerable flexibility for precedence assignment. That is, you can define your own classification mechanism. For example, you might want to assign precedence based on application or access router.

Note IP precedence bit settings 6 and 7 are reserved for network control information, such as routing updates.

IP Precedence Value Settings

By default, Cisco IOS XR software leaves the IP precedence value untouched. This preserves the precedence value set in the header and allows all internal network devices to provide service based on the IP precedence setting. This policy follows the standard approach stipulating that network traffic should be sorted into various types of service at the edge of the network and that those types of service should be implemented in the core of the network. Routers in the core of the network can then use the precedence bits to determine the order of transmission, the likelihood of packet drop, and so on.

Because traffic coming into your network can have the precedence set by outside devices, we recommend that you reset the precedence for all traffic entering your network. By controlling IP precedence settings, you prohibit users that have already set the IP precedence from acquiring better service for their traffic simply by setting a high precedence for all of their packets.

The class-based unconditional packet marking, LLQ, and WRED features can use the IP precedence bits.

You can use these features to set the IP precedence in packets:

- Class-based unconditional packet marking. See Configuring Class-based Unconditional Packet Marking.

IP Precedence Compared to IP DSCP Marking

IP precedence and DSCP markings are used to decide how packets should be treated in WRED.

The IP DSCP value is the first six bits in the ToS byte, and the IP precedence value is the first three bits in the ToS byte. The IP precedence value is actually part of the IP DSCP value. Therefore, both values cannot be set simultaneously. If both values are set simultaneously, the packet is marked with the IP DSCP value.

If you need to mark packets in your network and all your devices support IP DSCP marking, use the IP DSCP marking to mark your packets because the IP DSCP markings provide more unconditional packet marking options. If marking by IP DSCP is undesirable, however, or if you are unsure if the devices in your network support IP DSCP values, use the IP precedence value to mark your packets. The IP precedence value is likely to be supported by all devices in the network.

You can set up to 8 different IP precedence markings and 64 different IP DSCP markings.

QoS Policy Propagation Using Border Gateway Protocol

Packet classification identifies and marks traffic flows that require congestion management or congestion avoidance on a data path. Quality-of-service Policy Propagation Using Border Gateway Protocol (QPPB) allows you to classify packets by Qos Group ID, based on access lists (ACLs), Border Gateway Protocol (BGP) community lists, BGP autonomous system (AS) paths, Source Prefix address, or Destination Prefix address. After a packet has been classified, you can use other QoS features such as policing and weighted random early detection (WRED) to specify and enforce policies to fit your business model.

QoS Policy Propagation Using BGP (QPPB) allows you to map BGP prefixes and attributes to Cisco Express Forwarding (CEF) parameters that can be used to enforce traffic policing. QPPB allows BGP policy set in one location of the network to be propagated using BGP to other parts of the network, where appropriate QoS policies can be created.

QPPB supports both the IPv4 and IPv6 address-families.

QPPB allows you to classify packets based on:

- Access lists.
- BGP community lists. You can use community lists to create groups of communities to use in a match clause of a route policy. As with access lists, you can create a series of community lists.
- BGP autonomous system paths. You can filter routing updates by specifying an access list on both incoming and outbound updates, based on the BGP autonomous system path.
- Source Prefix address. You can classify a set of prefixes coming from the address of a BGP neighbor(s).
- Destination Prefix address. You can classify a set of BGP prefixes.

Classification can be based on the source or destination address of the traffic. BGP and CEF must be enabled for the QPPB feature to be supported.

Hierarchical Ingress Policing

The Hierarchical Ingress Policing feature is an MQC-based solution that supports hierarchical policing on ingress interfaces. This feature allows enforcement of service level agreements (SLA) while applying the classification sub-model for different QoS classes on the inbound interface. The hierarchical ingress policing provides support at three levels:

- Parent level
- Child level

Hierarchical policing allows policing of individual traffic classes as well as on a collection of traffic classes. This is useful in a situation where you want the collective police rate to be less than the sum of individual police (or maximum) rates.

In a child policy, the reference used for percentage police rates is the net maximum rate of the parent class. Net maximum is the lower of the configured shape and police rates in a class. The maximum police rate is that rate for which the action is to drop traffic. If the percentage or peak rate keywords do not have associated drop actions, then police rates do not influence the net maximum rate of a class.

In-Place Policy Modification

The In-Place Policy Modification feature allows you to modify a QoS policy even when the QoS policy is attached to one or more interfaces. When you modify the QoS policy attached to one or more interfaces, the QoS policy is automatically modified on all the interfaces to which the QoS policy is attached. A modified policy is subject to the same checks that a new policy is subject to when it is bound to an interface

However, if the policy modification fails on any one of the interfaces, an automatic rollback is initiated to ensure that the earlier policy is effective on all the interfaces. After successfully modifying a policy, the modifications take effect on all the interfaces to which the policy is attached.

The configuration session is blocked until the policy modification is successful on all the relevant interfaces. In case of a policy modification failure, the configuration session is blocked until the rollback is completed on all relevant interfaces.

Note You cannot resume the configuration on the routers until the configuration session is unblocked.

When a QoS policy attached to an interface is modified, QoS is first disabled on the interface, hardware is reprogrammed for the modified policy, and QoS is reenabled on the interface. For a short period of time, no QoS policy is active on the interface. In addition, the QoS statistics for the policy that is attached to an interface is lost (reset to 0) when the policy is modified.

Recommendations for Using In-Place Policy Modification

For a short period of time while a QoS policy is being modified, no QoS policy is active on the interface. In the unlikely event that the QoS policy modification and rollback both fail, the interface is left without a QoS policy.

For these reasons, it is best to modify QoS policies that affect the fewest number of interfaces at a time. Use the **show policy-map targets** command to identify the number of interfaces that will be affected during policy map modification.

Bidirectional Forwarding Detection Echo Packet Prioritization

If no QoS policy is attached to the interface on which BFD echo packets are received and switched back, then the BFD echo packets are marked as vital packets (when received) and are sent to the high priority queue in the ingressq ASIC reserved for transit control traffic.

If ingress QOS policy is present on the interface on which BFD echo packets are received and switched back, then the BFD echo packets are marked as vital packets (when received) and all QOS actions of the matching class except for taildrop and WRED are performed on the packets. The packets are then sent to the high priority

queue in the ingressq ASIC reserved for transit control traffic, overriding the queue selected by the ingress QOS policy.

In the egress direction, the BFD echo packets are treated like other vital packets (locally originated control packets) and are sent to the high priority queue of the interface in the egressq ASIC.

How to Configure Modular QoS Packet Classification

Creating a Traffic Class

To create a traffic class containing match criteria, use the **class-map** command to specify the traffic class name, and then use the following **match** commands in class-map configuration mode, as needed.

Note Users can provide multiple values for a match type in a single line of configuration; that is, if the first value does not meet the match criteria, then the next value indicated in the match statement is considered for classification.

For conceptual information, see the Traffic Class Elements.

Restrictions

- All **match** commands specified in this configuration task are considered optional, but you must configure at least one match criterion for a class.
- For the **match access-group** command, QoS classification based on the packet length or TTL (time to live) field in the IPv4 and IPv6 headers is not supported.
- For the **match access-group** command, when an ACL list is used within a class-map, the deny action of the ACL is ignored and the traffic is classified based on the specified ACL match parameters.

  An empty ACL (contains no rules, only remarks) within a QoS policy-map permits all traffic by default, and the implicit deny condition doesn't work with an empty ACL. Within a QoS policy map, the corresponding class-map matches all traffic not yet matched by the preceding traffic classes. In such a case, any explicit deny rule in the ACL leads to configuration commit failure.
- The **match discard-class** command is not supported on the Asynchronous Transfer Mode (ATM) interfaces.
- When QoS policy-maps use ACLs to classify traffic, ACEs of ACLs consume some amount of TCAM memory of the line card. Each QoS policy-map for ASR9000 supports up to a maximum of 3072 TCAM IPv4 entries. If you cross the limit, IOS XR fails to apply this policy-map with the insufficient memory available error. If you encounter this error, decrease the number of ACEs in ACLs for the policy-map. This error typically appears when using nested policy-maps, where ACEs in ACLs on different levels are multiplied.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match [not] access-group** [**ipv4**| **ipv6**] *access-group-name*

4. **match** [**not**] **cos** [*cos-value*] [*cos-value0 ... cos-value7*]
5. **match destination-address mac** *destination-mac-address*
6. **match source-address mac** *source-mac-address*
7. **match** [**not**] **discard-class** *discard-class-value* [*discard-class-value1 ... discard-class-value6*]
8. **match** [**not**] **dscp** [**ipv4** | **ipv6**] *dscp-value* [*dscp-value ... dscp-value*]
9. **match** [**not**] **mpls experimental topmost** *exp-value* [*exp-value1 ... exp-value7*]
10. **match** [**not**] **precedence** [**ipv4** | **ipv6**] *precedence-value* [*precedence-value1 ... precedence-value6*]
11. **match** [**not**] **protocol** *protocol-value* [*protocol-value1 ... protocol-value7*]
12. **match** [**not**] **qos-group** [*qos-group-value1 ... qos-group-value8*]
13. **match vlan [inner]** *vlanid* [vlanid1 ... vlanid7]
14. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map class201`	Creates a class map to be used for matching packets to the class whose name you specify and enters the class map configuration mode. If you specify **match-any**, one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. This is the default. If you specify **match-all**, the traffic must match all the match criteria.
Step 3	**match [not] access-group** [**ipv4**| **ipv6**] *access-group-name* **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match access-group ipv4 map1`	(Optional) Configures the match criteria for a class map based on the specified access control list (ACL) name.
Step 4	**match** [**not**] **cos** [*cos-value*] [*cos-value0 ... cos-value7*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match cos 5`	(Optional) Specifies a *cos-value* in a class map to match packets. The *cos-value* arguments are specified as an integer from 0 to 7.
Step 5	**match destination-address mac** *destination-mac-address* **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match destination-address mac 00.00.00`	(Optional) Configures the match criteria for a class map based on the specified destination MAC address.
Step 6	**match source-address mac** *source-mac-address* **Example:**	(Optional) Configures the match criteria for a class map based on the specified source MAC address.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cmap)# match source-address mac 00.00.00`	
Step 7	**match** [**not**] **discard-class** *discard-class-value* [*discard-class-value1 ... discard-class-value6*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match discard-class 5`	(Optional) Specifies a *discard-class-value* in a class map to match packets. The *discard-class-value* argument is specified as an integer from 0 to 7. The **match discard-class** command is supported only for an egress policy. The **match discard-class** command is not supported on the Asynchronous Transfer Mode (ATM) interfaces.
Step 8	**match** [**not**] **dscp** [**ipv4** | **ipv6**] *dscp-value* [*dscp-value ... dscp-value*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match dscp ipv4 15`	(Optional) Identifies a specific DSCP value as a match criterion. • Value range is from 0 to 63. • Reserved keywords can be specified instead of numeric values. • Up to eight values or ranges con be used per match statement.
Step 9	**match** [**not**] **mpls experimental topmost** *exp-value* [*exp-value1 ... exp-value7*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match mpls experimental topmost 3`	(Optional) Configures a class map so that the three-bit experimental field in the topmost Multiprotocol Label Switching (MPLS) labels are examined for experimental (EXP) field values. The value range is from 0 to 7.
Step 10	**match** [**not**] **precedence** [**ipv4** | **ipv6**] *precedence-value* [*precedence-value1 ... precedence-value6*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match precedence ipv4 5`	(Optional) Identifies IP precedence values as match criteria. • Value range is from 0 to 7. • Reserved keywords can be specified instead of numeric values.
Step 11	**match** [**not**] **protocol** *protocol-value* [*protocol-value1 ... protocol-value7*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match protocol igmp`	(Optional) Configures the match criteria for a class map on the basis of the specified protocol.
Step 12	**match** [**not**] **qos-group** [*qos-group-value1 ... qos-group-value8*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match qos-group 1 2 3 4 5 6 7 8`	(Optional) Specifies service (QoS) group values in a class map to match packets. • *qos-group-value* identifier argument is specified as the exact value or range of values from 0 to 63. • Up to eight values (separated by spaces) can be entered in one match statement.

	Command or Action	Purpose
		• **match qos-group** command is supported only for an egress policy.
Step 13	**match vlan [inner]** *vlanid* [vlanid1 ... vlanid7] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match vlan vlanid vlanid1`	(Optional) Specifies a VLAN ID or range of VLAN IDs in a class map to match packets. • *vlanid* is specified as an exact value or range of values from 1 to 4094. • Total number of supported VLAN values or ranges is 8.
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.

Creating a Traffic Policy

To create a traffic policy, use the **policy-map** command to specify the traffic policy name.

The traffic class is associated with the traffic policy when the **class** command is used. The **class** command must be issued after you enter the policy map configuration mode. After entering the **class** command, the router is automatically in policy map class configuration mode, which is where the QoS policies for the traffic policy are defined.

These class-actions are supported:

- bandwidth—Configures the bandwidth for the class. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.
- police—Police traffic. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.
- priority—Assigns priority to the class. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.
- queue-limit—Configures queue-limit (tail drop threshold) for the class. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.
- random-detect—Enables Random Early Detection. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.
- service-policy—Configures a child service policy.

- set—Configures marking for this class. See the Class-based Unconditional Packet Marking Feature and Benefits.
- shape—Configures shaping for the class. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.

For additional commands that can be entered as match criteria, see the *Modular Quality of Service Command Reference*Modular Quality of Service Command Reference for Cisco NCS 6000 Series Routers .

For conceptual information, see Traffic Policy Elements.

A maximum of 1024 classes (including Level 1, Level 2, and Level 3 hierarchical classes as well as implicit default classes) can be applied to one policy map.

For a hierarchical policy (with Level 1, Level 2, and Level 3 hierarchical classes) applied on a subinterface, the only allowed Level 1 class in the policy is the *class-default* class.

SUMMARY STEPS

1. **configure**
2. **policy-map** [**type qos**] *policy-name*
3. **class** class-name
4. **set precedence** [**tunnel**] *precedence-value*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**policy-map** [**type qos**] *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	**set precedence** [**tunnel**] *precedence-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set precedence 3`	Sets the precedence value in the IP header. **Note** • A policy configured with the set precedence tunnel command or the set dscp tunnel command can be applied on any Layer 3 interface in the ingress direction.

	Command or Action	Purpose
Step 5	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.

Attaching a Traffic Policy to an Interface

After the traffic class and traffic policy are created, you must use the service-policy interface configuration command to attach a traffic policy to an interface, and to specify the direction in which the policy should be applied (either on packets coming into the interface or packets leaving the interface).

For additional commands that can be entered in policy map class configuration mode, see the Modular Quality of Service Command ReferenceModular Quality of Service Command Reference for Cisco NCS 6000 Series Routers.

Prerequisites

A traffic class and traffic policy must be created before attaching a traffic policy to an interface.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **service-policy** {**input** | **output**} *policy-map*
4. Use the **commit** or **end** command.
5. **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	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/1`	Configures an interface and enters the interface configuration mode.

	Command or Action	Purpose
Step 3	**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 4	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 5	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface HundredGigE 0/7/0/1`	(Optional) Displays statistics for the policy on the specified interface.

Configuring Class-based Unconditional Packet Marking

This configuration task explains how to configure the following class-based, unconditional packet marking features on your router:

- IP precedence value
- IP DSCP value
- QoS group value (ingress only)
- CoS value (egress only)
- MPLS experimental value
- Discard class (ingress only)

Note IPv4 and IPv6 QoS actions applied to MPLS tagged packets are not supported. The configuration is accepted, but no action is taken.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **set precedence**
5. **set dscp**
6. **set qos-group** *qos-group-value*
7. **set cos** *cos-value*
8. **set mpls experimental** {**imposition** | **topmost**} *exp-value*
9. **set discard-class** *discard-class-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.
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 and enters the policy class map configuration mode.
Step 4	**set precedence** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set precedence 1`	Sets the precedence value in the IP header.
Step 5	**set dscp** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set dscp 5`	Marks a packet by setting the DSCP in the ToS byte.
Step 6	**set qos-group** *qos-group-value*	Sets the QoS group identifiers on IPv4 or MPLS packets.

	Command or Action	Purpose
	Example: `RP/0/RP0/CPU0:router(config-pmap-c)# set qos-group 31`	The **set qos-group** command is supported only on an ingress policy.
Step 7	**set cos** *cos-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set cos 7`	Sets the specific IEEE 802.1Q Layer 2 CoS value of an outgoing packet. Values are from 0 to7. Sets the Layer 2 CoS value of an outgoing packet. • This command should be used by a router if a user wants to mark a packet that is being sent to a switch. Switches can leverage Layer 2 header information, including a CoS value marking. • Packets entering an interface cannot be set with a CoS value.
Step 8	**set mpls experimental** {**imposition** | **topmost**} *exp-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set mpls experimental imposition 3`	Sets the experimental value of the MPLS packet top-most or imposition labels. **Note** The **imposition** keyword can be used only in service policies that are attached in the ingress policy.
Step 9	**set discard-class** *discard-class-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set discard-class 3`	Sets the discard class on IP Version 4 (IPv4) or Multiprotocol Label Switching (MPLS) packets. **Note** This command can be used only in service policies that are attached in the ingress policy.
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 0/7/0/1`	Configures an interface and enters the interface configuration mode.
Step 13	**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. In this example, the traffic policy evaluates all traffic leaving that interface.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-if)# service-policy output policy1`	
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 mode, 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 0/7/0/1`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring QoS Policy Propagation Using Border Gateway Protocol

This section explains how to configure Policy Propagation Using Border Gateway Protocol (BGP) on a router based on BGP community lists, BGP autonomous system paths, access lists, source prefix address, or destination prefix address.

Policy Propagation Using BGP Configuration Task List

Policy propagation using BGP allows you to classify packets by QoS group ID, based on BGP community lists, BGP autonomous system paths, access lists, source prefix address and destination prefix address. After a packet has been classified, you can use other quality-of-service features such as weighted random early detection (WRED) to specify and enforce policies to fit your business model.

Overview of Tasks

To configure Policy Propagation Using BGP, perform these basic tasks:

- Configure BGP and Cisco Express Forwarding (CEF). To configure BGP, see Routing Command Reference for Cisco NCS 6000 Series Routers . To configure CEF, see IP Addresses and Services Command Reference for Cisco NCS 6000 Series Routers.
- Configure a BGP community list or access list.
- Define the route policy. Set the QoS group ID, based on the BGP community list, BGP autonomous system path, access list, source prefix address or destination prefix address.
- Apply the route policy to BGP.

- Configure QPPB on the desired interfaces .
- Configure and enable a QoS Policy to use the above classification (QoS group ID). To configure committed access rate (CAR), WRED and tail drop, see the Configuring Modular QoS Congestion Avoidance on Cisco IOS XR Software module.

Defining the Route Policy

This task defines the route policy used to classify BGP prefixes with QoS group ID.

Prerequisites

Configure the BGP community list, or access list, for use in the route policy.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **set qos-group** *qos-group-value*
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**route-policy** *name* **Example:** `RP/0/RP0/CPU0:router(config)# route-policy r1`	Enters route policy configuration mode and specifies the name of the route policy to be configured.
Step 3	**set qos-group** *qos-group-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set qos-group 30`	Sets the QoS group identifiers. The set qos-group command is supported only on an ingress policy.
Step 4	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.

Applying the Route Policy to BGP

This task applies the route policy to BGP.

Prerequisites

Configure BGP and CEF.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6**} *address-family-modifier*
4. **table-policy** *policy-name*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode.
Step 3	**address-family** { **ipv4** | **ipv6**} *address-family-modifier* **Example:** `RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Enters address-family configuration mode, allowing you to configure an address family.
Step 4	**table-policy** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config-bgp-af) # table-policy qppb a1`	Configures the routing policy for installation of routes to RIB.
Step 5	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.

Configuring QPPB on the Desired Interfaces

This task applies QPPB to a specified interface. The traffic begins to be classified, based on matching prefixes in the route policy. The source or destination IP address of the traffic can be used to match the route policy.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **ipv4 | ipv6 bgp policy propagation input**{|**qos-group**} {**destination**[{*destination*|*source*}] {**source**[{*destination*|*source*}]
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/1`	Enters interface configuration mode and associates one or more interfaces to the VRF.
Step 3	**ipv4 | ipv6 bgp policy propagation input**{|**qos-group**} {**destination**[{*destination*|*source*}] {**source**[{*destination*|*source*}] **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 bgp policy propagation input qos-group destination`	Enables QPPB on an interface
Step 4	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 mode, without committing the configuration changes.

QPPB Scenario

Consider a scenario where in traffic is moving from Network1 to Network2 through (a single) router port1 and port2. If QPPB is enabled on port1, then

- for qos on ingress: attach an ingress policy on the interface port1.
- for qos on egress: attach an egress policy on interface port2.

Configuring Hierarchical Ingress Policing

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **service-policy** *policy-name*
5. **police rate percent** *percentage*
6. **conform-action** *action*
7. **exceed-action** *action*
8. Use the **commit** or **end** command.

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 parent`	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 class-default`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**service-policy** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# service-policy child`	Specifies the service-policy as a QoS policy within a policy map.

	Command or Action	Purpose
Step 5	**police rate percent** *percentage* **Example:** `RP/0/RP0/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/RP0/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** that transmits the packets.
Step 7	**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. The allowed action is **drop** that drops the packet.
Step 8	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.

Overview of Multiple QoS Policy Support

In Cisco Common Classification Policy Language (C3PL), the order of precedence of a class in a policy is based on the position of the class in the policy, that is, the class-map configuration which appears first in a policy-map has higher precedence. Also, the actions to be performed by the classified traffic are defined inline rather than using action templates. As a result of these two characteristics, aggregated actions cannot be applied to traffic that matches different classes.

In order to overcome this limitation, the “Multiple QoS Policy Support” feature is introduced. This feature enables the users to apply aggregated actions to various classes of traffic and apply multiple QoS policies on an interface.

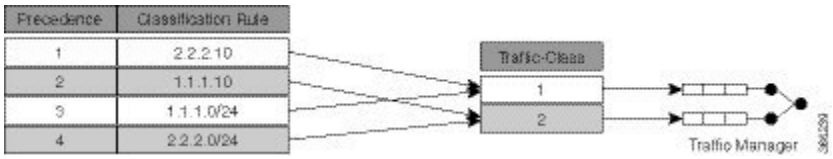
Use Case — Multiple QoS Policy Support

Consider a scenario where:

- The classification rules must be applied at different precedence levels.

- Each classification rule must be associated with non-queueing actions (that is, policing/marking).
- Multiple classification rules at different precedence levels must be mapped to a traffic-class.
- Each traffic-class or a group of traffic-classes must be associated with a single queue.

The figure below provides a detailed explanation of the above explained scenario—



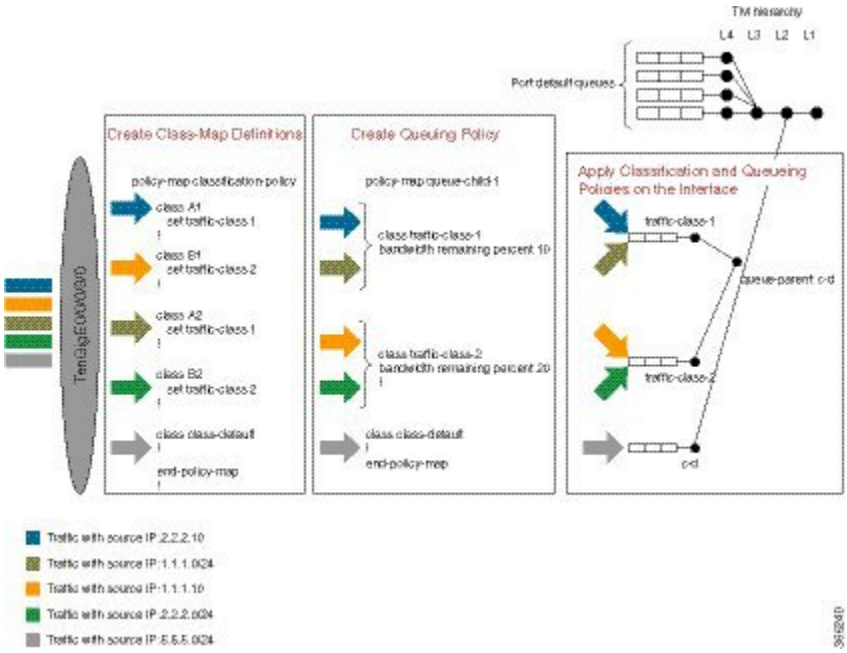
In this example, if the traffic packet matches 2.2.2.10 or 1.1.1.0/24, then the traffic packet is forwarded to the queue that is associated with traffic-class 1. And if the traffic packet matches 1.1.1.10 or 2.2.2.0/24, then the traffic packet is forwarded to the queue that is associated with traffic class 2.

With the existing Modular Quality of Service, we have the following limitations in order to achieve the above mentioned requirement—

1. Packets are matched in the order of precedence that is defined based on the position of the class-maps. There is no way to explicitly specify precedence for a class-map.
2. A queuing action under a class-map in a policy-map, creates a queue for that class.
3. Queues cannot be shared across class-maps.

These limitations can be overcome by separating classification from queuing. By doing this, it is possible to reorder the class-map from higher precedence to lower precedence and also share queues with multiple class-maps.

The example below depicts the implementation—



In this example, 4 classes A1, A2, B1, and B2 are created. Later, classification policies and queueing policies for these classes (A1, A2, B1, and B2) are created. After this, both the classification and queuing policies are applied to the interface. The detailed configuration steps are explained in the following section.

Configuring Multiple QoS Policy Support

In brief, configuring Multiple QoS policy support involves the following steps—

1. Configure Class Map—In this procedure, the traffic classes are defined.

```
/*Defining ACLs for Traffic Filtering*/
ipv4 access-list acl-a1
 10 permit ipv4 host 2.2.2.10 any
ipv4 access-list acl-b1
 10 permit ipv4 host 1.1.1.10 any
ipv4 access-list acl-a2
 10 permit ipv4 1.1.1.0/24 any
ipv4 access-list acl-b2
 10 permit ipv4 2.2.2.0/24 any
!
/*Creating Class Maps*/
class-map match-any A1
 match access-group ipv4 acl-a1
class-map match-any B1
 match access-group ipv4 acl-b1
class-map match-any A2
 match access-group ipv4 acl-a2
class-map match-any B2
 match access-group ipv4 acl-b2

class-map match-any traffic-class-1
 match traffic-class 1
class-map match-any traffic-class-2
 match traffic-class 2
```

2. Configure Policy—In this procedure, the classification and the queuing policies are created.

```
/*Creating Classification Policy*/
policy-map classification-policy
class A1
  set traffic-class 1
 class B1
  set traffic-class 2
class A2
  set traffic-class 1
 class B2
  set traffic-class 2
class class-default

!
/*Creating Queuing Policy*/
policy-map queue-parent
 class class-default
  service-policy queue-child
  shape average 50 mbps
policy-map queue-child
 class traffic-class-1
  bandwidth remaining percent 10
 class traffic-class-2
  bandwidth remaining percent 20
 !
 class class-default
```

```
 !
 end-policy-map
```

3. Apply Multiple Services on an Interface—In this procedure, the classification and queuing policies are applied on the interface.

```
/*Applying Policies on an Interface*/
Interface TenGigE0/0/0/3/0
service-policy output classification-policy
service-policy output queue-parent
```

To summarize, two policies (classification and queuing policies) are applied in the Egress direction. The classification policy executes first and classifies traffic at different precedence levels and marks the traffic-class field. The queuing policy executes second, matches on the traffic-class field to select the queue. For traffic matching in different classification precedence to share the same queue, mark the traffic-class field with the same value.

Verification

The **show qos interface** *interface-name* **output** command displays:

- per class per output policy QoS configuration values
- queuing policy followed by the classification policy
- traffic-classes matched by each class in queuing-policy

```
Router#show qos interface TenGigE 0/0/0/3/0 output
Interface: TenGigE0/0/0/3/0 output
Bandwidth configured: 50000 kbps Bandwidth programed: 50000 kbps
ANCP user configured: 0 kbps ANCP programed in HW: 0 kbps
Port Shaper programed in HW: 50000 kbps
Policy: queue-parent Total number of classes: 4
----------------------------------------------------------------------
Level: 0 Policy: queue-parent Class: class-default
Matches: traffic-classes : { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
 63,} and no traffic-class
QueueID: N/A
Shape CIR : NONE
Shape PIR Profile : 8 (Grid) Scale: 134 PIR: 49920 kbps  PBS: 624000 bytes
WFQ Profile: 3/9 Committed Weight: 10 Excess Weight: 10
Bandwidth: 0 kbps, BW sum for Level 0: 0 kbps, Excess Ratio: 1
----------------------------------------------------------------------
Level: 1 Policy: queue-child Class: traffic-class-1
Matches: traffic-classes : { 1}
Parent Policy: queue-parent Class: class-default
QueueID: 1040402 (Priority Normal)
Queue Limit: 66 kbytes Abs-Index: 19 Template: 0 Curve: 0
Shape CIR Profile: INVALID
WFQ Profile: 3/19 Committed Weight: 20 Excess Weight: 20
Bandwidth: 0 kbps, BW sum for Level 1: 0 kbps, Excess Ratio: 10
----------------------------------------------------------------------
Level: 1 Policy: queue-child Class: traffic-class-2
Matches: traffic-classes : {2}
Parent Policy: queue-parent Class: class-default
QueueID: 1040403 (Priority Normal)
Queue Limit: 126 kbytes Abs-Index: 29 Template: 0 Curve: 0
Shape CIR Profile: INVALID
WFQ Profile: 3/39 Committed Weight: 40 Excess Weight: 40
Bandwidth: 0 kbps, BW sum for Level 1: 0 kbps, Excess Ratio: 20
```

```
-----------------------------------------------------------------------
Level: 1 Policy: queue-child Class: class-default
Matches: traffic-classes : { 0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,}
and no traffic-class
Parent Policy: queue-parent Class: class-default
QueueID: 1040404 (Priority Normal)
Queue Limit: 446 kbytes Abs-Index: 52 Template: 0 Curve: 0
Shape CIR Profile: INVALID
WFQ Profile: 3/98 Committed Weight: 139 Excess Weight: 139
Bandwidth: 0 kbps, BW sum for Level 1: 0 kbps, Excess Ratio: 70
-----------------------------------------------------------------------
Interface: TenGigE0/0/0/3/0 output
Bandwidth configured: 10000000 kbps Bandwidth programed: 10000000 kbps
ANCP user configured: 0 kbps ANCP programed in HW: 0 kbps
Port Shaper programed in HW: 0 kbps
Policy: classification-policy Total number of classes: 5
-----------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: A1
Set traffic-class : 1
QueueID: 0 (Port Default)
Policer Profile: 59 (Single)
Conform: 100000 kbps (100 mbps) Burst: 1250000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
-----------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: B1
Set traffic-class : 2
QueueID: 0 (Port Default)
Policer Profile: 60 (Single)
Conform: 200000 kbps (200 mbps) Burst: 2500000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
-----------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: A2
Set traffic-class : 1
QueueID: 0 (Port Default)
Policer Profile: 61 (Single)
Conform: 300000 kbps (300 mbps) Burst: 3750000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
-----------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: B2
Set traffic-class : 2
QueueID: 0 (Port Default)
Policer Profile: 62 (Single)
Conform: 400000 kbps (400 mbps) Burst: 5000000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
-----------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: class-default
QueueID: 0 (Port Default)
-----------------------------------------------------------------------
```

Restrictions for Multiple QoS Policy Support

Policy Classification Restrictions

- Classification policy must always be executed before the queuing policy. Also, queuing actions are not supported within a classification policy.
- Classification policy supports unconditional set traffic-class actions. The valid values for **set traffic-class** are 0 – 63.
- In a conditional policer action, **set traffic-class** action is not supported.
- At least one **set traffic-class** action must be present for a policy to be considered a classification policy in the multi policy context.
- Only two additional packet fields can be unconditionally set along with **set traffic-class**.
- Class-maps in a classification policy cannot be used to match on traffic-class.
- Only one **set traffic-class** action is permitted in a hierarchy (either parent or child).
- Flow aware and shared policers are not supported.
- In a three-level policy, **set traffic-class** action is permitted only at the lowest two-levels.
- In a policer action, conditional **set traffic-class** is not supported.

Queuing Policy Restrictions

- Queuing policy can only classify on **traffic-class** field.
  - Valid values for **match traffic-class** are 0-63.
  - Class-maps can match up to 8 discreet traffic-class values or traffic-class ranges.
- At least one class-map with **match traffic-class** must be present for a policy to be considered a queuing policy in the multiple qos policy support feature.
- Class-map with match **not** traffic-class is not supported.
- Non-queuing actions like policer and set are not supported.
- Since policer is not supported in queuing policy, when priority level 1 queue is used, the service rate computed for lower priority queues is very low (with priority 1 utilizing all the bandwidth, the bandwidth remaining for lower priority queues is very low). Due to the same reason, **minimum bandwidth** is also not be supported with priority level 1. However, **bandwidth remaining ratio** may be used instead of **minimum bandwidth**. Since the **default queue-limit** and **time based queue-limit** configurations use service-rate to calculate **queue-limit** in bytes, it is recommended to explicitly configure queue-limit in bytes when using priority 1 queue.

Applying Multiple Services on an Interface Restrictions

- Applying multiple polices is supported only when one policy is a classification policy and the other policy is a queuing policy.

- Applying multiple polices (not more than 2 policies) is supported only in the egress direction. Applying more than 1 policy in the ingress direction is not supported.
- Applying multiple policies is supported only on the following interfaces:
  - Main-interface
  - Sub-interface
  - Bundle interface
  - Bundle sub-interface
- Applying Multi policies is not supported on the following interfaces:
  - PWHE
  - GRE
  - BVI
  - Satellite interfaces
- Multi policies are only supported on Cisco ASR 9000 High Density 100GE Ethernet line cards, Cisco ASR 9000 Enhanced Ethernet line cards, and Cisco ASR 9000 Ethernet line cards.
- The same classification policy cannot be applied with different queuing policies on a different interface of the same line card.
- Classification policy and queuing policy cannot be applied with any of the following feature options
  - account
  - service-fragment-parent
  - subscriber-parent

Policy Combinations

The different policy combinations are displayed in the below table:

Policies Already Applied on the Interface			**Policies that are yet to be Applied on the Interface**			**Accepted**
Regular Policy (no set/match traffic-class)	**Classification Policy**	**Queuing Policy**	**Regular Policy (no set/match traffic-class)**	**Classification Policy**	**Queuing Policy**	
Yes	No	No	Any combination			No

Policies Already Applied on the Interface			Policies that are yet to be Applied on the Interface			Accepted
No	Yes	No	No	No	No	No
			No	No	No	No
			Yes	No	No	No
			No	No	Yes	Yes
			No	Yes	Yes	No
			No	Yes	Yes	No
No	No	Yes	No	Yes	No	Yes
			Yes	Yes	No	No
			No	No	Yes	No
			No	Yes	Yes	No
			No	Yes	Yes	No
No	Yes	Yes	Any combination			No

Note To change a policy to a different policy of the same type you must first remove the existing policy and then apply the new policy.

Multi Policy and Interface Hierarchy

Multi Policy and Interface Hierarchy is displayed in the below table:

Main/Bundle Interface			Sub/Bundle Sub Interface			Comments
Regular Policy (no set/match traffic-class)	**Classification Policy**	**Queuing Policy**	**Regular Policy**	**Classification Policy**	**Queuing Policy**	

Main/Bundle Interface			Sub/Bundle Sub Interface	Comments
Non Port Shaper Policy	No	No	No policy allowed on child interfaces	The policy is enabled and is inherited by all the child interfaces. The same policy executes on the main interface and all its child interface traffic.
No	Yes	No	No policy allowed on child interfaces	Policy is disabled
No	Yes	No	No policy allowed on child interfaces	Policy is disabled
No	Yes	Yes	No policy allowed on child interfaces	Both policies are enabled and are inherited by all the child interfaces. The classification policy is executed first, followed by the queuing policy on the main interface and all its child interface traffic.

Main/Bundle Interface			Sub/Bundle Sub Interface			Comments
Port Shaper Policy	No	No	Yes	No	No	Main interface policy enabled. Sub interface policy is enabled and uses the port shaper rate as the reference bandwidth. If port shaper is applied after sub interface policy, then the applied sub interface policy will be updated with the new reference bandwidth. If the port shaper rate is lower than any sub interface policy rate, then the port shaper policy is rejected.
			No	Yes	No	Main interface policy enabled. Sub interface policy is disabled
			No	No	Yes	Main interface policy enabled. Sub interface policy is disabled

Main/Bundle Interface			Sub/Bundle Sub Interface			Comments
			No	Yes	Yes	Main interface policy enabled. Both the sub interface policies are enabled and both the policies use the port shaper rate as the reference bandwidth. If port shaper is applied after sub interface policies, then both the applied sub interface policies will be updated with the new reference bandwidth. If the port shaper rate is lower than any sub interface policy rate, then the port shaper policy is rejected.

Main/Bundle Interface	Sub/Bundle Sub Interface			Comments
Non port shaper policy not allowed on main interface	Yes	No	No	Policy is enabled
	No	Yes	No	Policy is disabled
	No	No	Yes	Policy is disabled
	No	Yes	Yes	Both policies are enabled and the classification policy is executed first followed by the queuing policy.

Statistics

Users can retrieve and verify the classification and queuing policy statistics per interface (per direction) in a multi-policy configuration, using the show policy-map interface interface-name output pmap-name command.

The **show policy-map interface all**, **show policy-map interface** *interface-name*, and **show policy-map interface** *interface-name* output displays statistics for all the policies in the each direction on an interface.

Classification Policy

- Statistics counters are allocated for every leaf class and updated for every packet match – match counters.
- Statistics counters are allocated for each policer used in the policy and updated during policing operation.
- There are no queue counters.

Queuing policy

- Each queue has a transmit and drop statistics counter associated with it which is updated for every queuing operation.
- There is a separate drop counter for each WRED color/curve in a queuing class.
- No match counters are allocated for a class. Instead, match counters is derived by adding the queue transmit statistics and all the queue drop statistics.

Example: Egress Policy Classification Statistics

```
Router# show policy-map interface TenGigE 0/0/0/3/9.1 output pmap-name classification

TenGigE0/0/0/3/9.1 output: classification
Class A1
Classification statistics          (packets/bytes)     (rate - kbps)
```

```
    Matched             :            83714645/83714645000          100006
    Transmitted         : N/A
    Total Dropped       : N/A
Class B1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            83714645/83714645000          100006
    Transmitted         : N/A
    Total Dropped       : N/A
Class A2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            83714645/83714645000          100006
    Transmitted         : N/A
    Total Dropped       : N/A
Class B2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            83714645/83714645000          100006
    Transmitted         : N/A
    Total Dropped       : N/A
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         : N/A
    Total Dropped       : N/A
```

Example: Egress Queuing Policy Statistics

```
Router# show policy-map interface TenGigE 0/0/0/3/9.1 output pmap-name queueing

TenGigE0/0/0/3/9.1 output: queueing
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           534226989/534226989000         400067
    Transmitted         :           355884870/355884870000         280381
    Total Dropped       :           106961210/106961210000         119726
  Policy queueing-child Class traffic-class-1
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :           178155114/178155114000         200014
      Transmitted         :           178155114/178155114000         200014
      Total Dropped       :                   0/0                    0
    Queueing statistics
      Queue ID                            : 647264
      High watermark                      : N/A
      Inst-queue-len  (packets)           : 0
      Avg-queue-len                       : N/A
      Taildropped(packets/bytes)          : 0/0
      Queue(conform)      :           178155114/178155114000         200014
      Queue(exceed)       :                   0/0                    0
      RED random drops(packets/bytes)     : 0/0
  Policy queueing-child Class traffic-class-2
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :           178098546/178098546000         200111
      Transmitted         :            71137336/71137336000          80385
      Total Dropped       :           106961210/106961210000         119726
    Queueing statistics
      Queue ID                            : 647265
      High watermark                      : N/A
      Inst-queue-len  (packets)           : 1620
      Avg-queue-len                       : N/A
      Taildropped(packets/bytes)          : 106961210/106961210000
      Queue(conform)      :            71137336/71137336000          80385
      Queue(exceed)       :                   0/0                    0
      RED random drops(packets/bytes)     : 0/0
Policy egress-queueing-child Class class-default
    Classification statistics          (packets/bytes)     (rate - kbps)
```

```
  Matched             :                  0/0                    0
  Transmitted         :                  0/0                    0
  Total Dropped       :                  0/0                    0
Queueing statistics
  Queue ID                            : 647266
  High watermark                      : N/A
  Inst-queue-len  (packets)           : 0
  Avg-queue-len                       : N/A
  Taildropped(packets/bytes)          : 0/0
  Queue(conform)      :                  0/0                    0
  Queue(exceed)       :                  0/0                    0
  RED random drops(packets/bytes)     : 0/0
```

Restrictions for Statistics

- The clear counters all is not supported for multi policy.
- The match statistics in a queuing policy are derived from the queue statistics. Therefore, there is no match statistics available for classes, which do not have a dedicated queue. Statistics for packets matching such classes (with no dedicated queue) shows up in the match statistics in the corresponding queuing class.
- Per classification class queue transmit and drop statistics are not available; only aggregated queue transmit and drop statistics are available.

Policy Modification

Modifying a policy when it is already applied on the interface, which is referred to as “In-place modification” is supported for both classification policy and queuing policy.

When a classification policy (or an ACL used in a classification policy) is modified, the previously applied classification policy and the corresponding queueing policy are removed from all interfaces. Then, the modified version of the classification policy is applied and the configured queuing policy is reapplied on all interfaces. If there is an error on any interface when applying the modified version of the classification policy, then all changes are reverted. That is, the modified version is removed from all interfaces on which it was applied and the previous (original, unmodified) version of both policies are reapplied on all interfaces. The modification attempt is terminated.

This modification process is the same for any modifications of the queuing policy. The previously applied queuing policy is removed and the modified version is applied (along with a reapplication of the corresponding classification policy.) In cases of error, the modification attempt is terminated and the previous versions of both policies are reapplied on all interfaces.

Since both classification and queuing polices are removed and then reapplied when either policy is modified, statistic counters in both policies is reset after a successful or failed modification.

Policy Modification Restrictions

- When a classification policy is applied on an interface, any modification, which changes it to a non-classification policy, for example, removing all set traffic-class actions or adding a class that matches on traffic-class, is rejected.

  So, in order to modify a classification policy to a non-classification policy, users must first remove the policy from all the interfaces and then modify.
- When a queuing policy is applied on an interface, any modification, which changes it to a non-queuing policy, for example, removing all classes that match on traffic-class, or adding a non-queuing action

(police or set), is rejected. So, in order to modify a queuing policy to a non-queuing policy, users must first remove the policy from all the interfaces and then modify.

Supported Features by Multi Policies

The following table displays the features supported and not supported by Multi policies—

Feature	Multi Policy- Classification	Multi Policy- Queuing
Classification	Except traffic-class field, all other fields that are currently supported	Only on traffic-class field
Unconditional Marking	Traffic-class and all other fields that are currently supported	No
1R2C	Yes	
1R3C		
2R3C		
Policer/Conditional Marking	Except traffic-class field, all other fields that are currently supported	
Grand Parent Policer	Yes	
Color Aware Policer		
Conform Aware Policer		
Shared Policer	No	
Flow Aware Policer	No	
Priority	No	
Shape		
Bandwidth		
Bandwidth Remaining		Yes
WRED		Supported but no WRED classification on traffic-class
Statistics	Match counters, policer exceed/conform/violate counters	Match, queue transmit, queue drop, WRED drop counters
Rate Calculation	Match and policer statistics	Match and queue statistics
SPI	No	No
Port Shaper	Yes	Yes
Policy Inheritance	Yes	Yes

Configuration Examples for Configuring Modular QoS Packet Classification

Traffic Classes Defined: Example

In this example, two traffic classes are created and their match criteria are defined. For the first traffic class called class1, ACL 101 is used as the match criterion. For the second traffic class called class2, ACL 102 is used as the match criterion. Packets are checked against the contents of these ACLs to determine if they belong to the class.

```
class-map class1
  match access-group ipv4 101
  exit
!
class-map class2
  match access-group ipv4 102
  exit
```

Use the **not** keyword with the **match** command to perform a match based on the values of a field that are not specified. The following example includes all packets in the class qos_example with a DSCP value other than 4, 8, or 10.

```
class-map match-any qos_example
  match not dscp 4 8 10
!
end
```

Traffic Policy Created: Example

In this example, a traffic policy called policy1 is defined to contain policy specifications for the two classes—class1 and class2. The match criteria for these classes were defined in the Traffic Classes Defined: Example.

For class1, the policy includes a bandwidth allocation request and a maximum byte limit for the queue reserved for the class. For class2, the policy specifies only a bandwidth allocation request.

```
policy-map policy1
  class class1
    bandwidth 3000 kbps
    queue-limit 1000 packets
!
  class class2
    bandwidth 2000 kbps
!
  class class-default
!
end-policy-map
!
end
```

Traffic Policy Attached to an Interface: Example

This example shows how to attach an existing traffic policy to an interface (see the Traffic Classes Defined: Example). After you define a traffic policy with the policy-map command, you can attach it to one or more interfaces to specify the traffic policy for those interfaces by using the **service-policy** command in interface configuration mode. Although you can assign the same traffic policy to multiple interfaces, each interface can have only one traffic policy attached at the input and only one traffic policy attached at the output.

```
interface HundredGigE 0/7/0/1
  service-policy output policy1
  exit
!
```

Default Traffic Class Configuration: Example

This example shows how to configure a traffic policy for the default class of the traffic policy called policy1. The default class is named class-default, consists of all other traffic, and is being shaped at 60 percent of the interface bandwidth.

```
policy-map policy1
  class class-default
    shape average percent 60
```

class-map match-any Command Configuration: Example

This example illustrates how packets are evaluated when multiple match criteria exist. Only one match criterion must be met for the packet in the **class-map match-any** command to be classified as a member of the traffic class (a logical OR operator). In the example, protocol IP OR QoS group 4 OR access group 101 have to be successful match criteria:

```
class-map match-any class1
  match protocol ipv4
  match qos-group 4
  match access-group ipv4 101
```

In the traffic class called class1, the match criteria are evaluated consecutively until a successful match criterion is located. The packet is first evaluated to determine whether IPv4 protocol can be used as a match criterion. If IPv4 protocol can be used as a match criterion, the packet is matched to traffic class class1. If IP protocol is not a successful match criterion, then QoS group 4 is evaluated as a match criterion. Each matching criterion is evaluated to see if the packet matches that criterion. Once a successful match occurs, the packet is classified as a member of traffic class class1. If the packet matches at least one of the specified criteria, the packet is classified as a member of the traffic class.

Note The **match qos-group** command is supported only on an egress policy and on an ingress policy for QoS Policy Propagation using BGP (QPPB)-based policies.

Traffic Policy as a QoS Policy (Hierarchical Traffic Policies) Configuration: Examples

A traffic policy can be nested within a QoS policy when the **service-policy** command is used in policy map class configuration mode. A traffic policy that contains a nested traffic policy is called a hierarchical traffic policy.

Hierarchical traffic policies can be attached to all supported interfaces for this Cisco IOS XR software release, such as the OC-192 and 10-Gigabit Ethernet interfaces.

Two-Level Hierarchical Traffic Policy Configuration: Example

A two-level hierarchical traffic policy contains a child and a parent policy. The child policy is the previously defined traffic policy that is being associated with the new traffic policy through the use of the **service-policy** command. The new traffic policy using the pre-existing traffic policy is the parent policy. In the example in this section, the traffic policy called child is the child policy, and the traffic policy called parent is the parent policy.

In this example, the child policy is responsible for prioritizing traffic, and the parent policy is responsible for shaping traffic. In this configuration, the parent policy allows packets to be sent from the interface, and the child policy determines the order in which the packets are sent.

```
policy-map child
 class mpls
  priority level 1
  police rate 100 mbps burst 10 ms
  !
 !
 class class-default
 !
 end-policy-map
!
policy-map parent
 class class-default
  service-policy child
  shape average 1000 mbps
 !
 end-policy-map
!
```

Class-based Unconditional Packet Marking: Examples

These are typical class-based unconditional packet marking examples:

IP Precedence Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a previously defined class map called *class1* through the use of the **class** command, and then the service policy is attached to the output HundredGigE interface 0/7/0/1. The IP precedence bit in the ToS byte is set to 1:

```
policy-map policy1
  class class1
    set precedence 1
!
```

```
interface HundredGigE 0/7/0/1
  service-policy output policy1
```

IP DSCP Marking Configuration: Example

In this example, a service policy called policy1 is created. This service policy is associated to a previously defined class map through the use of the **class** command. In this example, it is assumed that a class map called class1 was previously configured and new class map called class2 is created.

In this example, the IP DSCP value in the ToS byte is set to 5:

```
policy-map policy1
  class class1
    set dscp 5

  class class2
    set dscp ef
```

After you configure the settings shown for voice packets at the edge, all intermediate routers are configured to provide low-latency treatment to the voice packets, as follows:

```
class-map voice
  match dscp ef
policy-map qos-policy
  class voice
    priority level 1
    police rate percent 10
```

The service policy configured in this section is not yet attached to an interface. For information on attaching a service policy to an interface, see the Modular Quality of Service Overview on Cisco IOS XR Software module.

QoS Group Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the input direction on a HundredGigE 0/7/0/1. The qos-group value is set to 1.

```
class-map match-any class1
  match protocol ipv4
  match access-group ipv4 101

policy-map policy1
  class class1
    set qos-group 1
  !
interface HundredGigE 0/7/0/1
  service-policy input policy1
```

Note The **set qos-group** command is supported only on an ingress policy.

Discard Class Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the input direction on a HundredGigE 0/7/0/1. The discard-class value is set to 1.

```
class-map match-any class1
  match protocol ipv4
  match access-group ipv4 101

policy-map policy1
  class class1
    set discard-class 1
  !
interface HundredGigE 0/7/0/1
  service-policy input policy1
```

CoS Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the output direction on a HundredGigE 0/7/0/1. The 802.1p (CoS) bits in the Layer 2 header are set to 1.

```
class-map match-any class1
  match protocol ipv4
  match access-group ipv4 101

policy-map policy1
  class class1
    set cos 1
  !
interface  HundredGigE 0/7/0/1
interface  HundredGigE 0/7/0/1.100
  service-policy output policy1
```

MPLS Experimental Bit Imposition Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the input direction on a HundredGigE 0/7/0/1. The MPLS EXP bits of all imposed labels are set to 1.

```
class-map match-any class1
  match protocol ipv4
  match access-group ipv4 101

policy-map policy1
  class class1
    set mpls exp imposition 1
 !
interface HundredGigE 0/7/0/1
  service-policy input policy1
```

Note The **set mpls exp imposition** command is supported only on an ingress policy.

MPLS Experimental Topmost Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the output direction on a HundredGigE 0/7/0/1. The MPLS EXP bits on the TOPMOST label are set to 1:

```
class-map match-any class1
  match mpls exp topmost 2

policy-map policy1
  class class1
    set mpls exp topmost 1
  !
interface HundredGigE 0/7/0/1
  service-policy output policy1
```

QoS Policy Propagation using BGP: Examples

These are the IPv4 and IPv6 QPPB examples:

Applying Route Policy: Example

In this example, BGP is being configured for the IPv4 address family:

```
router bgp 100
 bgp router-id 19.19.19.19
 address-family ipv4 unicast
  table-policy qppbv4_dest
 !
 neighbor 10.10.10.10
  remote-as 8000
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
```

In this example, BGP is being configured for the IPv6 address family:

```
router bgp 100
 bgp router-id 19.19.19.19
 address-family ipv6 unicast
  table-policy qppbv6_dest
 !
 neighbor 1906:255::2
  remote-as 8000
  address-family ipv6 unicast
   route-policy pass-all in
   route-policy pass-all out
```

Applying QPPB on a Specific Interface: Example

This example shows applying QPPBv4 (address-family IPv4) for a desired interface:

```
config
interface POS0/0/0/0
ipv4 address 10.1.1.1
ipv4 bgp policy propagation input qos-group destination
end
commit
!
```

This example shows applying QPPBv6 (address-family IPv6) for a desired interface:

```
config
interface POS0/0/0/0
ipv6 address 1906:255::1/64
ipv6 bgp policy propagation input qos-group destination
end
commit
!
```

Route Policy Configuration: Examples

QPPB Source Prefix Configuration: Example

This configuration is an example of configuring QPPB source prefix:

```
route-policy qppb-src10-20
  if source in (201.1.1.0/24 le 32) then
    set qos-group 10
  elseif source in (201.2.2.0/24 le 32) then
    set qos-group 20
  else
    set qos-group 1
  endif
  pass
end-policy
!

router bgp 100
 bgp router-id 10.10.10.10
 address-family ipv4 unicast
  table-policy qppb-src10-
 !
 neighbor 201.1.1.2
  remote-as 62100
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.2.2.2
  remote-as 62200
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 202.4.1.1
  remote-as 100
  address-family ipv4 unicast
  !
 !
!

policy-map p1-in
 class class-qos10
  set discard-class 4
  shape average percent 20
 !
 class class-qos20
```

```
  set precedence critical
  shape average percent 30
 !
 class class-default
 !
 end-policy-map
!
policy-map p2-out
 class class-qos10
  set precedence priority
  police rate percent 30
 !
 class class-qos20
  set precedence immediate
  police rate percent 40
 !
 class class-default
 !
 end-policy-map
!

interface HundredGigE 0/7/0/1
 service-policy output p1-in
 ipv4 address 201.1.0.1 255.255.255.0
 ipv4 bgp policy propagation input qos-group source
 negotiation auto
 !

interface HundredGigE 0/7/0/1
 service-policy input p2-out
 ipv4 address 201.32.21.1 255.255.255.0
 ipv4 bgp policy propagation input qos-group destination
 negotiation auto
 !
```

QPPB Destination Prefix Configuration: Example

This configuration is an example of QPPB destination prefix:

```
route-policy qppb-des10to20
  if destination in (10.10.0.0/16 le 28) then
    set qos-group 10
  elseif destination in (10.11.0.0/16 le 28) then
    set qos-group 11
  elseif destination in (10.12.0.0/16 le 28) then
    set qos-group 12
  elseif destination in (10.13.0.0/16 le 28) then
    set qos-group 13
  elseif destination in (10.14.0.0/16 le 28) then
    set qos-group 14
  elseif destination in (10.15.0.0/16 le 28) then
    set qos-group 15
  elseif destination in (20.20.0.0/16 le 28) then
    set qos-group 20
  else
    set qos-group 1
  endif
  pass
end-policy
!

router bgp 100
 bgp router-id 10.10.10.10
```

```
 address-family ipv4 unicast
  table-policy qppb-des10to20
 !
 neighbor 201.1.1.2
  remote-as 62100
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.1.2.2
  remote-as 62102
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.1.3.2
  remote-as 62103
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.1.4.2
  remote-as 62104
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.1.5.2
  remote-as 62105
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !


policy-map p11-in
 class class-qos10
  set precedence priority
  shape average percent 30
 !
 class class-qos11
  set precedence immediate
  shape average percent 10
 !
 class class-qos12
  set precedence flash
  shape average percent 15
 !
 class class-qos13
  set precedence flash-override
  shape average percent 20
 !
 class class-qos14
  set precedence flash
  shape average percent 25
 !
 class class-qos15
  set precedence flash-override
  shape average percent 30
```

```
 !
 class class-qos20
  set precedence critical
  shape average percent 40
 !
 class class-default
 !
 end-policy-map
!
policy-map p1-out
 class class-qos10
  set precedence priority
  police rate percent 30
 !
 class class-qos20
  set precedence critical
  police rate percent 40
 !
 class class-default
 !
 end-policy-map
!

interface HundredGigE 0/7/0/1
 service-policy output p1-out
 ipv4 address 201.1.0.1 255.255.255.0
 ipv4 bgp policy propagation input qos-group source
 negotiation auto
!
interface HundredGigE 0/7/0/1
 service-policy input p11-in
 ipv4 address 201.1.2.1 255.255.255.0
 ipv4 bgp policy propagation input qos-group destination
 dot1q vlan 2
!
interface HundredGigE 0/7/0/1
 service-policy input p11-in
 ipv4 address 201.1.3.1 255.255.255.0
 ipv4 bgp policy propagation input qos-group destination
 dot1q vlan 3
!
interface HundredGigE 0/7/0/1
 service-policy input p11-in
 ipv4 address 201.1.4.1 255.255.255.0
 ipv4 bgp policy propagation input qos-group destination
 dot1q vlan 4
!
interface HundredGigE 0/7/0/1
 service-policy input p11-in
 ipv4 address 201.1.5.1 255.255.255.0
 ipv4 bgp policy propagation input qos-group destination
 dot1q vlan 5
!
```

Hierarchical Ingress Policing: Example

This configuration is an example of typical hierarchical ingress policing:

```
policy-map parent
 class class-default
  service-policy child
  police rate percent 50
```

```
  conform-action transmit
  exceed-action drop
```

Hierarchical policing allows service providers to provision the bandwidth that is available on one link among many customers. Traffic is policed first at the child policer level and then at the parent policer level.

In this example, the child policy specifies a police rate of 40 percent. This is 40 percent of the *transmitted* rate in the parent policy. The parent policy specifies a police rate of 50 percent. This is 50 percent of the interface rate. The child policy remarks traffic that exceeds the conform rate; the parent policy drops traffic that exceeds the conform rate.

```
interface HundredGigE 0/7/0/1
 service-policy input parent
 mac-accounting ingress
 mac-accounting egress
!

class-map match-any customera
 match vlan 10-30
 end-class-map
!
class-map match-any customerb
 match vlan 40-70
 end-class-map
!
class-map match-any precedence-5
 match precedence 5
 end-class-map
!
!
policy-map child
 class precedence-5
  priority level 1
  police rate percent 40
  !
 !
 class class-default
  police rate percent 30
   conform-action set precedence 2
   exceed-action set precedence 0
  !
 !
 end-policy-map
!

policy-map parent
 class customera
  service-policy child
  police rate percent 50
   conform-action transmit
   exceed-action drop
  !
 !
 class customerb
  service-policy child
  police rate percent 20
   conform-action transmit
   exceed-action drop
  !
 !
 class class-default
 !
```

```
 end-policy-map
!
```

In-Place Policy Modification: Example

This configuration is an example of in-place policy modification:

Defining a policy map:

```
configure
policy-map policy1
class class1
set precedence 3
commit
```

Attaching the policy map to an interface:

```
configure
interface HundredGigE 0/7/0/1
service-policy output policy1
commit
```

Modifying the precedence value of the policy map:

```
configure
policy-map policy1
class class1
set precedence 5
commit
```

Note The modified policy *policy1* takes effect on all the interfaces to which the policy is attached. Also, you can modify any class-map used in the policy-map. The changes made to the class-map takes effect on all the interfaces to which the policy is attached.

Additional References

These sections provide references related to implementing packet classification.

Related Documents

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

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

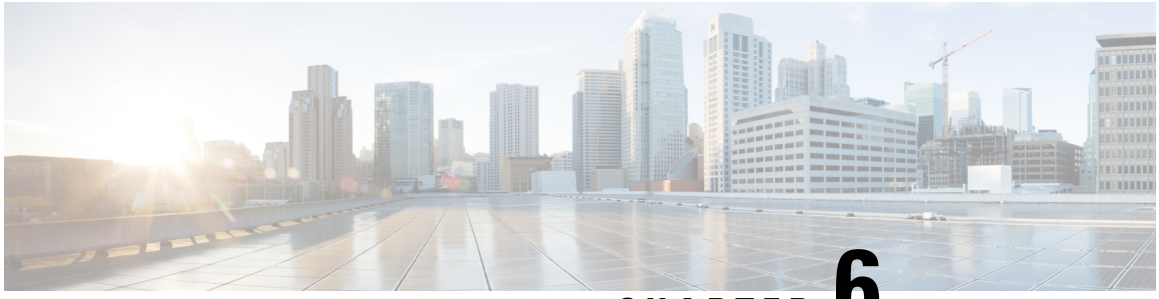
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR softw Cisco MIB Locator found at the following URL and choo under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.sht

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



CHAPTER 6

Configuring Fabric QoS Policies and Classes

This module provides the conceptual and configuration information for fabric QoS.

Feature History for Configuring Fabric Quality of Service Policies and Classes on Cisco IOS XR Software

Release	Modification
Release 5.0.0	This feature was introduced.
Release 6.6.1	Support for assured forwarding (AF) priority was introduced.

Prerequisites for Configuring Fabric Quality of Service Policies and Classes

This prerequisite is required for configuring modular fabric QoS 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.

Information About Configuring Fabric Quality of Service Policies and Classes

Overview

The fabric queue selection mechanism is known as fabric QoS. There are four queues per destination port: (high priority) HP1, HP2, assured-forwarding (AF) and best effort (BE).

Note By default, internal control traffic is placed in the high-priority queue.

You can configure a fabric QoS policy that defines classification criteria for selecting high-priority or low-priority queue. This is applied to the secure domain router (SDR) (this may be the whole router if no individual service domain routers are configured) and affects all fabricq ASICs in the logical router.

Of the four levels of priority—HP1, HP2, AF, and BE—AF is unused. From Release 6.6.1 onwards, support for AF is enabled. The enhancement allows for AF and BE traffic to be supported, even while prioritizing traffic for BE. This provides better quality of service whenever there is a fabric congestion.

A maximum of four classescan be specified within the policy. A class known as *class-default* is automatically created and equates to the BE queues. The name of this class cannot be altered. Any name may be applied to the classes that equate to the priority and AF ports or queues.

Note The **class-map** for fabric QoS checks that all the IPv4, IPv6 and MPLS matches the conditions configured for all the incoming packets. The match also supports Class of Service (CoS).

Fabric QoS policy class maps are restricted to matching a subset of these classification options:

- precedence
- dscp
- qos-group
- discard-class
- mpls experimental topmost

Note To match on **qos-group** or **discard-class**, an ingress QoS policy must be applied, setting the required values for **qos-group** or **discard-class**. Both of these variables have local significance only and are not recognized outside of the router.

The fabricq queue selection mechanism is known as Fabric QoS. To provide class of service to the traffic under fabric congestion scenarios, configure Fabric QoS. The platform-independent user interface allows you to configure an MQC policy on the switch fabric queues. This policy is global for all line cards on the router.

Ingress Policy and Fabric QoS Policy Interaction

If the ingress QoS policy remarks certain traffic with values that the fabric QoS policy class-maps are to match on, then the remarked traffic is matched and placed in the appropriate port or queues. This provides the ability for the ingress QoS policy and the fabric QoS policy to complement each other, rather than potentially conflicting.

It is important to remember that if an ingress QoS policy is applied to an interface and the fabric QoS policy has been applied to the router, then the ingress MSC RX PSE is required to perform two classification cycles.

How to Configure Fabric Quality of Service Policies and Classes

Creating a Traffic Class

See the "Creating a Traffic Class" section in the "Configuring Modular Quality of Service Packet Classification on Cisco IOS XR Software" module.

Creating a Fabric QoS Service Policy

SUMMARY STEPS

1. **configure**
2. **class-map** *class-map-name*
3. **match precedence***class-mapprecedence value*
4. **policy-map** *policy-name*
5. **class** *class-name*
6. **priority** [**level** *priority-level*]
7. **exit**
8. **exit**
9. **switch-fabric service-policy** *policy_name*
10. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**class-map** *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map class201`	Creates a class map to be used for matching packets to the class whose name you specify and enters the class map configuration mode. If you specify match-any, one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. This is the default.

	Command or Action	Purpose
Step 3	**match precedence***class-mapprecedence value* **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match 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-cmap)# 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//CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.
Step 6	**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.
Step 7	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to XR Config mode.
Step 9	**switch-fabric service-policy** *policy_name* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# switch-fabric service-policy policy1`	Configures a service policy for the switch fabric.
Step 10	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.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuration Examples for Configuring Fabric Quality of Service Policies and Classes

Configuring Fabric Quality of Service Policies and Classes: Example

This configuration is an example of how the packets are matched:

```
class-map match-all High
match not mpls experimental topmost 0
match not precedence routine
end-class-map
```

The match is based on the outermost protocol with these results:

- **IP packets:**
  - IP packets with ToS0 (Type of Service) is False, because `match not mpls experimental topmost 0` is True and `match not precedence routine` is False. So both the options are False.
  - IP packets with ToS1 to ToS7 is True, because `match not mpls experimental topmost 0` is True and `match not precedence routine` is True. So both the options are True.
- **MPLS packets:**
  - MPLS packets with EXP0 (Experimental bits) is False, because `match not mpls experimental topmost 0` is False and `match not precedence routine` is True. So both the options are False.
  - MPLS packets with EXP1 to EXP7 is True, because `match not mpls experimental topmost 0` is True and `match not precedence routine` is True. So both the options are True.

This configuration is an example of a possible fabric QoS policy:

```
class-map match-any llq
 match mpls experimental topmost 5
 match precedence critical
!
class-map match-any business
 match mpls experimental topmost 3
 match precedence flash
!
policy-map fabric_qos
 class llq
  priority
 !
```

To apply the policy, use the **switch-fabric service-policy** command with the *policy-name* argument.

This example shows an ingress QoS policy working in conjunction with a fabric QoS policy. The fabric QoS policy is shown first, followed by the ingress QoS policy:

```
class-map match-any llq
 match qos-group 5
!
class-map match-any business&games
 match qos-group 3
!
policy-map fabric_qos
 class llq
  priority
 !
!
class-map match-any voip
 match mpls experimental topmost 5
 match precedence critical
 match dscp cs5
!
class-map match-any business
 match mpls experimental topmost 4
 match dscp cs4
 match precedence flash-override
!
class-map match-any broadband-games
 match mpls experimental topmost 3
 match dscp cs3
 march precedence flash
!

policy-map input-qos
 class voip
  priority level 1
  police rate percent 20
  conform-action set qos-group 5
class business
  set qos-group 3
 class broadband-games
  set qos-group 3
```

Additional References

These sections provide references related to implementing fabric QoS policies and classes.

Related Documents

QoS commands	Cisco IOS XR Modular Quality of Service Command Refere
User groups and task IDs	*"Configuring AAA Services on Cisco IOS XR Software"* mod *System Security Configuration Guide*

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

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

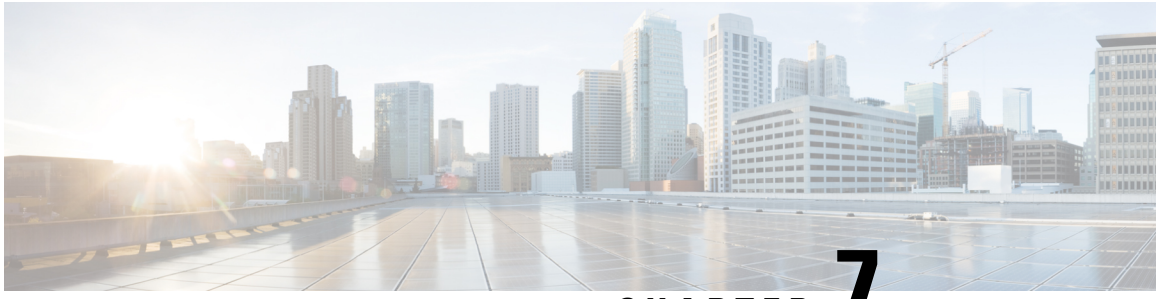
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR softw Cisco MIB Locator found at the following URL and choo under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.sht

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



CHAPTER 7

Configuring Modular QoS on Link Bundles

A link bundle is a group of one or more ports that are aggregated together and treated as a single link. This module describes QoS on link bundles.

Feature History for Configuring Modular QoS on Link Bundles

Release	Modification
Release 5.0.0	This feature was introduced.

Link Bundling Overview

The Link Bundling feature allows you to group multiple point-to-point links together into one logical link and provide higher bidirectional bandwidth, redundancy, and load balancing between two routers. A virtual interface is assigned to the bundled link. The component links can be dynamically added and deleted from the virtual interface.

The virtual interface is treated as a single interface on which one can configure an IP address and other software features used by the link bundle. Packets sent to the link bundle are forwarded to one of the links in the bundle.

A link bundle is simply a group of ports that are bundled together and act as a single link. The advantages of link bundles are as follows:

- Multiple links can span several line cards and SPAs to form a single interface. Thus, the failure of a single link does not cause a loss of connectivity.
- Bundled interfaces increase bandwidth availability, because traffic is forwarded over all available members of the bundle. Therefore, traffic can move onto another link if one of the links within a bundle fails. You can add or remove bandwidth without interrupting packet flow..

All links within a bundle must be of the same type. For example, a bundle can contain all Ethernet interfaces, or it can contain all POS interfaces, but it cannot contain Ethernet and POS interfaces at the same time.

Note POS interfaces are not supported in Release 5.0.0.

Cisco IOS XR software supports these methods of forming bundles of Ethernet interfaces:

- IEEE 802.3ad—Standard technology that employs a Link Aggregation Control Protocol (LACP) to ensure that all the member links in a bundle are compatible. Links that are incompatible or have failed are automatically removed from a bundle.
- EtherChannel —Cisco proprietary technology that allows the user to configure links to join a bundle, but has no mechanisms to check whether the links in a bundle are compatible.POS Channel applies to POS interfaces.

Load Balancing

Load balancing is supported on all links in the bundle. Load balancing function is a forwarding mechanism to distribute traffic over multiple links based on Layer 3 routing information in the router. There are two types of load balancing schemes:

- Per-Destination Load Balancing
- Per-Packet Load Balancing

When a traffic stream arrives at the router, per-packet load balancing allows the traffic to be evenly distributed among multiple equal cost links. Per-packet schemes make routing decision based on round-robin techniques, regardless of the individual source-destination hosts.

Only Per-Destination Load Balancing is supported.

Per-destination load balancing allows the router to distribute packets over one of the links in the bundle to achieve load sharing. The scheme is realized through a hash calculating based on the source-destination address and user sessions.

When the per-destination load balancing is enabled, all packets for a certain source-destination pair will go through the same link, though there are multiple links available. In other words, per-destination load balancing can ensure that packets for a certain source-destination pair could arrive in order.

QoS and Link Bundling

All Quality of Service (QoS) features, currently supported on Layer 3 physical interfaces, are also supported on all Link Bundle interfaces. All QoS features currently supported on Layer 3 physical sub interfaces are also supported on bundle VLANs.

QoS is configured on Link Bundles in primarily the same way that it is configured on individual interfaces. However, there are some important restrictions that should be noted:

- Parameters for QoS actions can only be specified as percentages.
- WRED and queue limit parameters can only be configured in time units.

These guidelines describe the characteristics of QoS behavior on bundle interfaces:

- QoS features are configured only on the Link Bundle interfaces, never on individual members.
- For egress QoS policies, a unique set of queues defined by the policy are assigned to each physical interface on each linecard hosting a member link. As a result, for queuing actions (e.g. shaping), egress QoS policy acts on the bandwidth provided by the individual member interfaces. This means that the actual shape rate experienced by a traffic class may be less than the configured value (as a percentage of the bundle bandwidth), if its traffic is not load-balanced evenly across all member links. For policer actions, egress QoS policy acts on the aggregate bandwidth provided by the interfaces on that linecard.

When a member link is added to a bundle with output QoS configured, the policy-map of the bundle is applied to the member link.

Example 2 shows the output QoS policy supported on link bundles.

Example 2 : Output QoS policy supported on link bundles

```
policy-map out-sample
  class voice
    priority level 1
    police rate percent 10
  class premium
    bandwidth percent 30
    queue-limit 100 ms
  class class-default
    queue-limit 100 ms
```

Aggregate Bundle QoS Mode

Aggregated Bundle QoS allows the shape, bandwidth, police rates, and burst values to be distributed between the active members of a bundle where a QoS policy-map is applied. For instance, consider that the traffic is load-balanced among the members of the bundle. In aggregate mode, the bundle ethernet traffic is shaped to 10 Mbps to match the configuration of QoS policy.

When the policy is applied on a member of the bundle, a ratio can be calculated based on the total bandwidth of the bundle to that of the bandwidth of a member in the bundle. For example, if the bandwidth of the bundle is 20 Gbps, and the bandwidth of a member in the bundle is 10 Gbps, then the ratio will be 2:1.

A change in the bundle (with a member down, added, removed or activated) or mode results in the automatic recalculation of QoS rate.

The user QoS policy is invalid when applied to the bundle interface under the following scenarios:

- A 10 Gbps interface and 40 Gbps interface are part of a bundle, and the 40 Gbps interface is inactive. Currently, QoS policy is also programmed on non-active members. When programming the 40Gbps bundle member, the bundle bandwidth is 10 Gpbs, but the member bandwidth is 40 Gbps. The ratio of bundle bandwidth to member bandwidth does not work for this member.
- Consider a shape of 15 Gbps. This action is valid on bundles with multiple 10G active members, but invalid when only one member is active and that member is in QoS inconsistent state. To view inconsistency details for the QoS policy, run the **show qos inconsistency** command in EXEC mode. This scenario is also applicable during the reload of a router where only few interfaces in line cards (LC) becomes available before the rest of the interfaces in all LCs.
- A failure in the hardware when programming a rate change during the bundle bandwidth change or when a new member is added to the bundle.

- If an interface has the QoS policy configured to an absolute value, you cannot change the aggregated bundle mode from enabled to disabled. You must modify the policy or remove it before attempting to disable the aggregate bundle mode.
- An invalid policy combination, with the absolute values of port shaper less than the policy shape rate is accepted without an error in console or log file.
- You apply QoS policy on bundle interfaces for BNG subscribers. In this case, there is no aggregation across bundle members, and the policy is applied individually. For the policy to work correctly, modify the rate according to the number of members in the bundle.
- You apply QoS policy on PWHE and BVI interfaces. Both these interfaces don't support bundle aggregate mode.

Load Balancing in Aggregate Bundle QoS

Load balancing requires a large number of flows in order to distribute the traffic among the members of the bundle. Ensure that load is balanced evenly among the members of the bundle before using the aggregate bundle QoS mode. If the under-lying traffic is only a few tunnels (GRE, TE-TUNNELS), it may be possible that the load balancing is not distributing the traffic evenly and may cause problems.

For example, consider bandwidth of a bundle is 20 Gbps, and the bandwidth of a member in the bundle is 10 Gbps. If the traffic is not load balanced, the aggregate traffic output may not reach 10 Mbps even when more than 10 Mbps is sent to the bundle-ether interface.

QoS Policy in Aggregate bundle mode

The following table shows the behavior of aggregate QoS policy mode to various actions:

Action	Behavior
Policing and Shaping / Bandwidth	**Percentage**: No change. The percentage is calculated based o max-rate / interface bandwidth **PPS / Absolute rate**: Divide the rate based on bandwidth rat **Burst-size**: If `time-units`, no change. Convert `time-units` t based on `service-rate` If configured in absolute value, divide the absolute value bas bandwidth ratio
Wred / Queue-Limit Threshold	**Time-Units**: No Change. Use the `service-rate` to convert to **Absolute value**: Divide the absolute value based on bandwid

Enabling Aggregate Bundle QoS

To enable the aggregate bundle QoS, perform these steps:

SUMMARY STEPS

1. **configure**

2. **hw-module all qos-modeaggregate-bundle-mode**
3. **class** *class-name*
4. **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	**hw-module all qos-modeaggregate-bundle-mode** **Example:** `RP/0/RSP0/CPU0:router(config)# hw-module all qos-mode aggregate-bundle-mode`	Enters policy map configuration mode. When aggregated bundle mode changes, QoS polices on bundle interfaces and sub-interfaces are modified automatically. A reload of the line card is not required.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Enters policy map class configuration mode. Specifies the name of the class whose policy you want to create or change.

Step 4	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# end`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# commit`	

Policing using Aggregate Bundle QoS

```
policy-map grand-parent
 class class-default
  service-policy parent
  police rate 200 mbps burst 1 kbytes
 end-policy-map

policy-map parent
 class class-default
  service-policy child
  police rate 300 mbps
 end-policy-map

policy-map child
 class 3play-voip
  police rate 10 mbps burst 10 kbytes peak-rate 20 mbps peak-burst 20 kbytes
   conform-color red-cos
```

```
    conform-action set precedence 1
    exceed-action set precedence 2
    violate-action drop

class 3play-video
  police rate 15 mbps burst 10 kbytes peak-rate 30 mbps peak-burst 20 kbytes
    conform-color yellow-cos
    conform-action set precedence 1
    exceed-action set precedence 2
    violate-action drop
   !
  !
  class 3play-premium
   police rate 25 mbps burst 10 kbytes peak-rate 35 mbps peak-burst 20 kbytes
    conform-color green-cos
    conform-action set precedence 1
    exceed-action set precedence 2
    violate-action drop
   !
  !
  class class-default
   police rate 6 mbps
   !
  !
  end-policy-map
!
```

Additional References

These sections provide references related to implementing QoS on Link Bundles.

Related Documents

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

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 softw Cisco MIB Locator found at the following URL and choo under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.sht

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