Cisco IOS XR Modular Quality of Service Configuration Guide for the Cisco CRS Router, Release 6.1.x

First Published: 2016-11-01

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

This guide describes the Cisco IOS XR QoS configurations. The preface for the *Modular QoS Configuration Guide for Cisco CRS Routers* contains these sections:

Changes to this Document

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

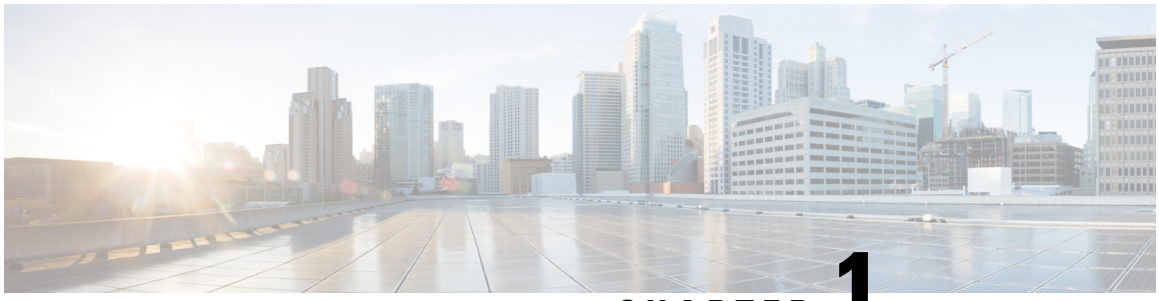
Date	Summary
November 2016	Initial release of this document.

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

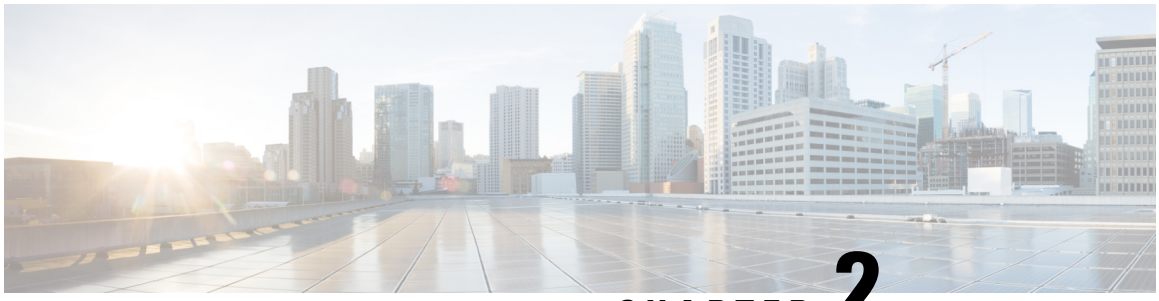
New and Changed QoS Features

New and Changed QoS Features

Table 1: New and Changed Features

Feature	Description	Changed in Release	Where Documented
No new features in this release	NA	Release 6.1.2	NA

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 traffic flow is identified, it can be marked as a traffic class.

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

Marking of a traffic flow is performed by:

- Setting IP Precedence or DSCP bits in the IP Type of Service (ToS) byte.
- Setting Class of Service (CoS) bits in the Layer 2 headers (Ethernet, Spatial Reuse Protocol [SRP], and so on).
- 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.

Note The Cisco CRS Series Modular Services Card 140G (CRS-MSC-140G) does not support SRP interfaces.

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 Software module 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 traffic 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
- Low-priority port for assured-forwarding (AF) traffic, and
- Best-effort port (BE) for low-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.

Interfaces Supported on CRS-MSC-140G

Some interfaces that are used in examples in this guide and are supported on the Cisco CRS Series Modular Services Card 40G (CRS-MSC-40G), are not supported on the Cisco CRS Series Modular Services Card 140G (CRS-MSC-140G). For information about interfaces supported on the CRS-MSC-140G, see the Compatibility Matrix table and the Supported PLIMs table in the CRS system description documents.

Supported Capability Matrix

Feature	Support on CRS-1	Support on CRS-3	Support on CRS-X	Notes
Classification (Match Criteria)				
ANY	No	No	Yes	—
ALL	No	No	Yes	—
NOT	Yes	Yes	Yes	For CRS-X cards, the **match not** classification is not supported for **match access-group**, **match class-map**, or **match vlan** commands.
ACCESS-GROUP	Yes	Yes	Yes	—
COS INNER	Yes	Yes	Yes	—
COS	Yes	Yes	Yes	
DSCP	Yes	Yes	Yes	IP DSCP is supported for IPv4 and IPv6.
PRECEDENCE	Yes	Yes	Yes	—
MPLS EXPERIMENTAL TOPMOST	Yes	Yes	Yes	—
VLAN	Yes	Yes	Yes	Inner label matching is not supported for QOS on QinQ interfaces.
QOS-GROUP	Yes	Yes	Yes	—
PROTOCOL	Yes	Yes	Yes	—
DISCARD-CLASS	Yes	Yes	Yes	—
VPLS	Yes	Yes	Yes	—
DESTINATION ADDRESS MAC	Yes	Yes	No	—
SOURCE ADDRESS MAC	Yes	Yes	No	—
Frame Relay DLCI	Yes	No	No	
Marking				

Feature	Support on CRS-1	Support on CRS-3	Support on CRS-X	Notes
COS	Yes	Yes	Yes	For CRS-X cards, users can create up to four sets with the following combinations: • Ingress: set qos-group+ set discard-class + Two additional sets • Egress: set inner cos + set outer cos + Two additional sets
DISCARD-CLASS	Yes	Yes	Yes	—
QOS-GROUP	Yes	Yes	Yes	—
DSCP	Yes	Yes	Yes	—
DSCP TUNNEL	Yes	Yes	Yes	—
PRECEDENCE	Yes	Yes	Yes	—
GRE Tunnel Precedence	Yes	Yes	No	—
GRE Tunnel DSCP	Yes	Yes	No	—
MPLS EXPERIMENTAL TOPMOST	Yes	Yes	Yes	—
MPLS (EXPERIMENTAL) IMPOSITION	Yes	Yes	Yes	—
Spatial Reuse Protocol (SRP) priority	Yes	Yes	No	—
ATM-CLP (cell loss priority)	Yes	No	No	—
Queuing				
Ingress Queuing	Yes	Yes	No	—
Ingress Priority	Yes	Yes	Yes	—
Egress Queuing	Yes	Yes	Yes	—

Feature	Support on CRS-1	Support on CRS-3	Support on CRS-X	Notes
Egress Priority	Yes (priority level 1)	Yes (priority level 1, 2)	Yes (priority level 1,2)	—
Congestion Management				
Bandwidth	Yes	Yes	Yes	For CRS-X cards, **bandwidth**, **bandwidth remaining** and **shape** are supported only on Egress.
Bandwidth Remaining	Yes	Yes	Yes	
Shape	Yes	Yes	Yes	
Congestion Avoidance				
Random Detect	Yes	Yes	Yes	For CRS-X cards, random detect and queue limit is supported only on egress.
Random Detect Discard-class-based	Yes	Yes	Yes	
Random Detect COS	Yes	Yes	Yes	
Random Detect MPLS EXP	Yes	Yes	Yes	
Random Detect Precedence	Yes	Yes	Yes	
Random Detect DSCP	Yes	Yes	Yes	
Queue Limit	Yes	Yes	Yes	
HQOS	Yes (2-level)	Yes (2-level)	Yes (3-level)	For CRS-3 cards, up to two level of H-QoS is supported. For CRS-X cards, up to three levels of H-QoS is supported.
Rate Limiting				
1R2C	Yes	Yes	Yes	—
1R3C	Yes	Yes	Yes	
2R3C	Yes	Yes	Yes	
QPPB				
QoS Policy Propagation through BGP	No	Yes	Yes	For CRS-X cards, QPPB is supported from Release 5.1.3.

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

Related Topic	Document Title
Initial system bootup and configuration	*Cisco IOS XR Getting Started Guide for the* Cisco CRS Router
Interface types supported on the CRS-MSC-140G	Compatibility Matrix table and Supported PLIMs table in the CRS system description documents
Master command reference	Cisco CRS Router Master Command Listing
QoS commands	Cisco IOS XR Modular Quality of Service Command Reference for the Cisco CRS Router
User groups and task IDs	*"Configuring AAA Services on Cisco IOS XR Software"* module of *Cisco IOS XR System Security Configuration Guide*

Standards

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

MIBs

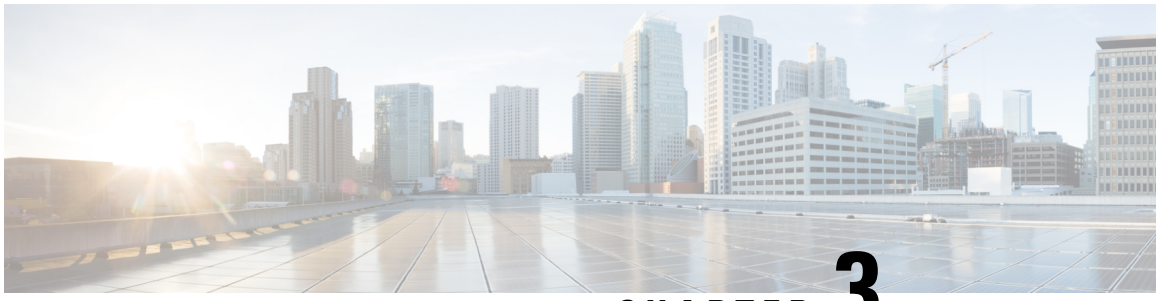
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

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

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



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 2.0	The Congestion Avoidance feature was introduced.
Release 3.6.0	The default queue limit and WRED thresholds was based on the guaranteed service rate of the queue.
Release 3.9.0	Removed the restriction about configuring either the **shape average, bandwidth, or bandwidth remaining percent** commands in the user defined policy map class for the **random-detect** command to take effect (CSCsl69571).
Release 4.0.0	In calculations for the average queue size, indicated that the exponential weight factor is not configurable (CSCeg75763).

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.

Random Detect and Clear Channel ATM SPAs

On Clear Channel ATM SPAs, random detect uses 1024 hardware profiles in the SPA. The profiles maintain configuration details about the thresholds, so that they can be shared across multiple **random-detect** statements in the same class with matching threshold values (and units). The SPA maintains WRED and tail drop statistics for each threshold in a class.

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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.
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 global configuration mode.

	Command or Action	Purpose
Step 9	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface TenGigE 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.
Step 11	**commit**	

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.

Note Only a maximum of 8 random-detect statements can be configured per class within a policy-map. To perform WRED on more than 8 types of traffic that is defined by MPLS EXP, ACL, qos-group, discard-class, IP Prec, or IP DSCP, an ingress policy must be used to allocate traffic types into groups. This is done by setting either qos-group or discard-class markings. The egress policy can then match up to 8 qos-groups or discard-classes.

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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 512 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 the value of the *min-threshold* argument 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**.

	Command or Action	Purpose
		• 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]*
	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	**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.

	Command or Action	Purpose
Step 10	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 11	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface pos 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	**commit**	

Example

In the following example, 10 traffic types are matched and a qos-group value of ‘1’ is set for all packets that match this class on ingress. When the packets arrive on the egress card, the qos-group value is used to match the traffic to the class 'output_oc192' and WRED is executed on all traffic in that class.

```
class-map input_oc192
    match precedence routine priority network internet
    match dscp af41 af 43
    match mpls experimental topmost 1 2 3 4
    set qos-group 1

class-map output_oc192
    match qos-group 1

policy-map egress_oc192
    class output_oc192
        random detect 1000 5079
```

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:

default queue limit (in bytes) = (<200|100|10> ms * queue_bandwidth kbps) / 8

Note You can configure the queue limit in all the priority classes.

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.
- On the Cisco CRS Series Modular Services Card 140G (CRS-MSC-140G), a police action *must* be configured in the same class as the priority action. A class configuration that includes a priority action but no police action is not valid. Such a configuration is rejected.

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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 global configuration 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 POS 0/2/0/0`	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	**commit**	

Additional References

These sections provide references related to implementing QoS congestion avoidance.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco IOS XR Getting Started Guide for the* Cisco CRS Router
Master command reference	Cisco CRS Router Master Command Listing
QoS commands	Cisco IOS XR Modular Quality of Service Command Reference for the Cisco CRS Router
User groups and task IDs	*"Configuring AAA Services on Cisco IOS XR Software"* module of *Cisco IOS XR System Security Configuration Guide*

Standards

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

MIBs

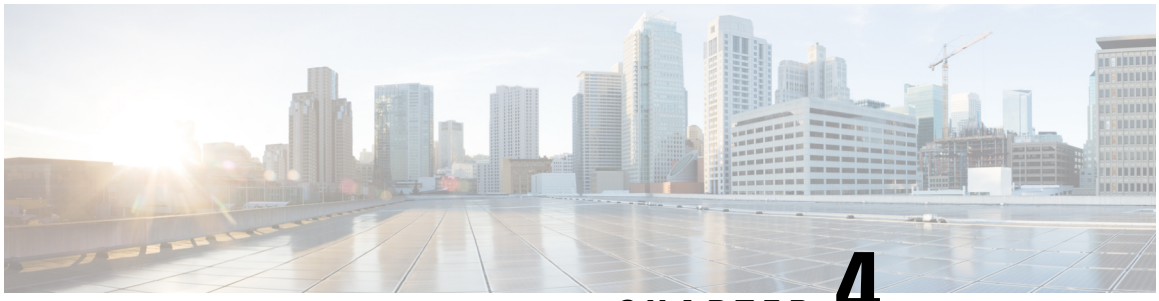
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

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

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



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 2.0	The Congestion Avoidance feature was introduced.
Release 3.2	The **police** command was changed to the **police rate** command and the syntax changed.
Release 3.4.0	The **police rate** command enters policy map police configuration mode to configure the conform, exceed and violate actions. The following new commands were added: **conform-action**, **exceed-action** and **violate-action**. The **cos**, **qos-group**, and **transmit** actions were added to the policer.

Release	Modification
Release 3.6.0	Increased Class scale from 32 to 512 classes. Unallocated remaining bandwidth is equally distributed among all the queueing classes that do not have remaining bandwidth configured explicitly. For shape and police percentage parameters in child policy, reference is relative to the maximum rate of the parent. For bandwidth percentage parameters in child policy, reference is relative to the minimum bandwidth of the parent class. If bandwidth is not configured in parent class, guaranteed service rate of parent class is used as reference.
Release 3.8.0	The multi-action set/policer was supported. The **set qos-group** ingress policer marking was supported.
Release 3.9.2	The Policer Granularity andShaper Granularity features were introduced.
Release 4.0.0	Because they are not supported, removed these sections: • QoS-Group-Based Queueing • Traffic Policing on Layer 2 ATM Interfaces • Attaching a Service Policy to the Attachment Circuits (AC) example • Configuring Dual Queue Limit example High Priority Propagation and Layer-all Accounting features were introduced on the Cisco CRS Series Modular Services Card 140G (CRS-MSC-140G). On the CRS-SMC-140G, a police action *must* be configured in the same class as the priority action (configuration change from the Cisco CRS Series Modular Services Card 40G CRS-MSC-40G).
Release 4.2.1	Configured Accounting 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.

For Clear Channel ATM SPAs, all queue-based actions are offloaded to the SPA and are performed by the SPA. Clear Channel ATM subinterfaces support eight queues per subinterface. On egress subinterfaces, you can configure a service policy with a maximum of seven non-default classes with queueing actions. Other classes must *not* have queueing actions.

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.

The Cisco CRS implements a slight variation of the MDRR scheduling mechanism called packet-by-packet MDRR (P2MDRR). Using P2MDRR, queues are scheduled after every packet is sent compared to MDRR in which queues are scheduled after a queue is emptied. All non-high-priority queues with minimum bandwidth guarantees use P2MDRR.

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.

For Clear Channel ATM subinterfaces, the priority queue cannot be configured on the default class.

High-Priority Propagation

The CRS-MSC-140G supports 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.

High-priority traffic under all ports and groups is serviced before any low-priority traffic. This means that the scope of priority assignment at the queue level is global — it is not limited to the parent group (such as on CRS-MSC-40G) or port. 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 group level for an egress interface policy is not supported.

Policer Requirement

On the CRS-MSC-140G, a policer must be configured to limit the traffic entering priority queues. The policer rate cannot exceed the shape rate configured for the group or port.

Note This requirement does not apply to fabric QoS polices, because police actions in fabric QoS policies are not supported.

On the Cisco CRS Series Modular Services Card 40G (CRS-MSC-40G), a priority action can be configured with or without a police action in the same class. Example 1 shows a valid configuration on the CRS-MSC-40G that includes only a priority action:

Example 1 Class Configured With Priority Action Only (CRS-MSC-40G)

```
policy-map prio_only_policy
 class prec1
  priority level 1
 !
 class class-default
 !
 end-policy-map
!
```

On the Cisco CRS Series Modular Services Card 140G (CRS-MSC-140G), a police action *must* be configured in the same class as the priority action. A class configuration that includes a priority action but no police action is not valid. Such a configuration is rejected.

To use existing CRS-MSC-40G QoS configurations on the CRS-MSC-140G, add a police action to all classes that have a priority action. In Example 2, the class configuration in Example 1 is modified to include a police action:

Example 2 Class Configured With Priority Action and Police Action (CRS-MSC-140G

```
policy-map prio_and_police_policy
```

```
 class prec1
  priority level 1
  police rate percent 20
  !
 !
 class class-default
 !
 end-policy-map
!
```

Note On the CRS-MSC-40G, on egress Layer 3 ATM subinterfaces, if more than one class is configured with priority, a policer must be configured in the same classes.

Ingress and Egress Queuing on the CRS-MSC-140G

- Ingress Queuing Only:

  The smallest step size supported is 32 kbps for groups and 32/3 kbps (10.67 kbps) for queues. Step size increases with the rate value. Rounding error does not exceed 0.4 per cent or 8 kbps, whichever is higher.

- 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 on the CRS-MSC-140G

- Minimum bandwidth of a group must be equal to or greater than the sum of queue minimum bandwidths and the police rates of the high priority classes under the group. 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. In the event of oversubscription, the actual minimum bandwidth that a group or queue receives is proportional to its configured value.

Configured Accounting

Configured Accounting controls the type of overhead and packet length for statistics, policing shaping and queuing. The account option can be specified with a service-policy when applying a policy to an interface. For bundle interfaces, the configured accounting option is applied to all member interfaces.

The configured accounting option is available on ingress and egress policing, queuing and statistics for CRS-MSC-140G. In CRS-MSC-40G, the configured accounting option is not available for queuing.

This table shows the packet length used during QoS for various accounting options on the MSC-140:

Configured Accounting Option	Policing	Queuing	Statistics
Default	layer-all	layer 2	layer 2
Layer2	layer 2	layer 2	layer 2
No Layer2	layer 3	layer 3	layer 3

This table shows the packet length used during QoS for various accounting options on the MSC-40:

Configured Accounting Option	Ingress Policing	Ingress Queuing	Ingress Statistics	Egress Policing	Egress Queuing	Egress Statistics
Default	layer 2	layer 3	layer 2	layer 2	layer 2	layer 2
Layer2	layer 2	layer 3	layer 2	layer 2	layer 2	layer 2
No Layer2	layer 3	layer 3	layer 3	layer 3	layer 2	layer 3

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.

Traffic Shaping for ATM on Layer 2 VPN

The **shape** command under the PVC submode is applicable to the attachment circuits (AC) in the virtual circuit (VC) mode and the virtual path (VP) mode.

Layer 2 and Layer 3 ATM VC interfaces support VC shaping. This is not an MQC QoS configuration where shaping is configured in a service policy. Shaping is configured on the ATM interface, directly under the VC.

For ATM Layer 3 subinterfaces, shaping is not supported in the egress direction.

VC shaping cannot be configured, removed, or modified on an interface that already has an egress service policy configured.

Note The default shape is UBR at line rate.

Layer-All Accounting

The CRS-MSC-140G uses "layer-all" accounting. For Ethernet interfaces, this translates to 20 bytes of Layer 1 overhead in addition to the Layer 2 overhead. Cisco CRS Series Modular Services Card 40G (CRS-MSC-40G) does Layer 3 QoS accounting for ingress queueing. CRS-MSC-40G does Layer 2 QoS accounting for egress queueing, egress policing, and ingress policing.

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.

Note Configured values take into account the Layer 2 encapsulation applied to traffic. This applies to both ingress and egress policing. For POS/SDH transmission, the encapsulation is considered to be 4 bytes. For Ethernet, the encapsulation is 14 bytes; whereas for 802.1Q, the encapsulation is 18 bytes.

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

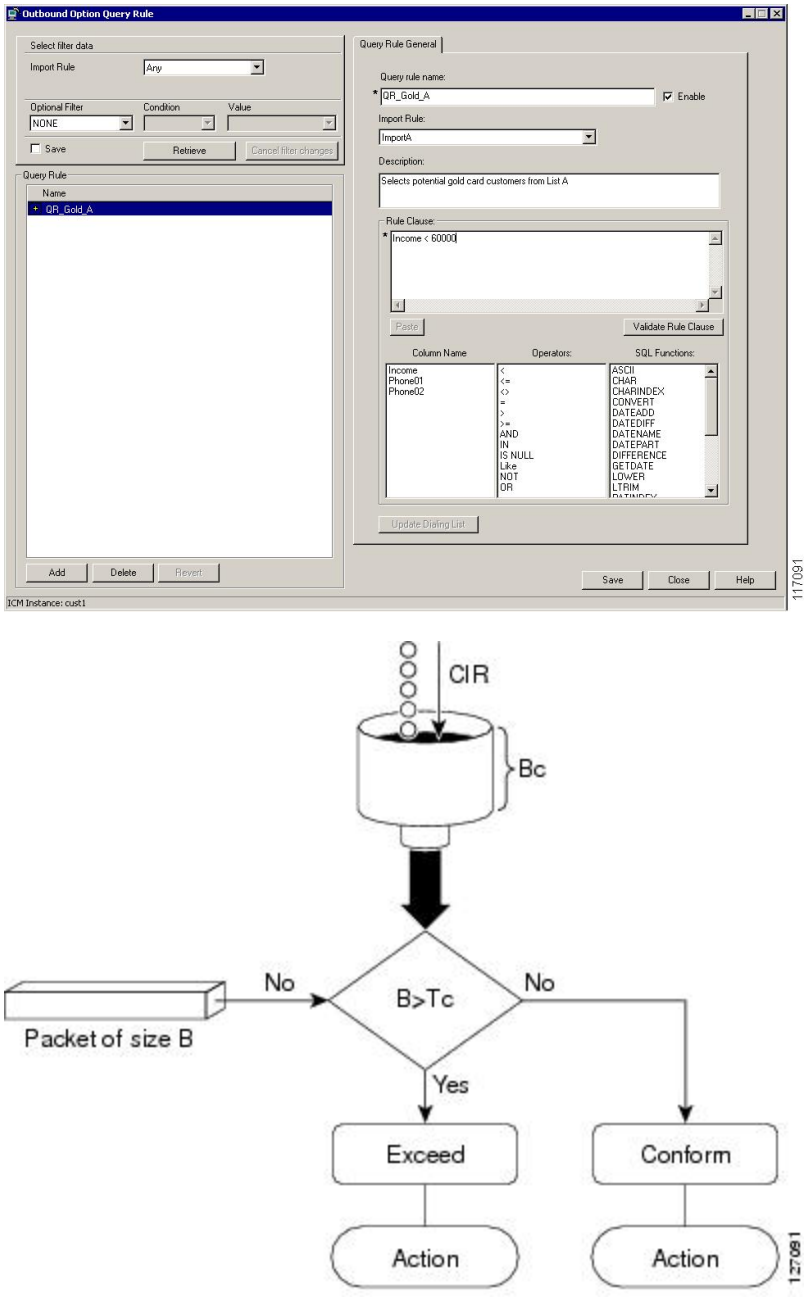
This section describes the single-rate mechanism.

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 CRS-MSC-140G

- 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

The Multiple Action Set feature allows you to mark packets with multiple action sets (conditional and unconditional) through a class map.

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

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, or Layer 2 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, or Layer 2 data paths that are applicable.

Layer 3 IP Packets	Layer 3 MPLS Packets	Layer 2 Packets
DSCP or precedence	MPLS experimental imposition	MPLS experimental imposition
tunnel DSCP or tunnel precedence	MPLS experimental topmost	discard-class
MPLS experimental imposition	discard-class	qos-group
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, or Layer 2 data paths that are applicable.

Layer 3 IP Packets	Layer 3 MPLS Packets	Layer 2 Packets
DSCP or precedence	MPLS experimental topmost	cos or srp-priority2
cos or srp-priority	cos or srp-priority2	discard-class
discard-class	discard-class	—

Note

- Both cos and srp-priority packets are mutually exclusive; srp-priority is not supported on the Cisco CRS Series Modular Services Card 140G (CRS-MSC-140G).

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.

Table 4 shows the supported conditional policer marking operations.

Note None of the following class-based conditional policer marking operations are supported on ATM interfaces.

Marking Operation	Layer 2 Ingress			Layer 2 Egress			Layer 3 Ingress			Layer 3 Egress		
	PAC	CAC	p-C	PAC	CAC	p-C	Phy	SIf	P-SIf	Phy	SIf	P-SIf
prec	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
prec tunnel	N	N	N	N	N	N	Y	Y	Y	N	N	N
DSCP	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
DSCP tunnel	N	N	N	N	N	N	Y	Y	Y	N	N	N

Marking Operation	Layer 2 Ingress			Layer 2 Egress			Layer 3 Ingress			Layer 3 Egress		
CoS	N	N	N	Y	Y	Y	N	N	N	Y	Y	Y
discard class	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
EXP, imposition	Y	Y	Y	N	N	N	Y	Y	Y	N	N	N
EXP, topmost	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
qos-group	N	N	N	Y	Y	Y	N	N	N	Y	Y	Y
qos-group	Y	Y	Y	N	N	N	Y	Y	Y	N	N	N

[1] (1) p-C=physical interface with underlying CACs.
[2] (2) Phy=physical interface.
[3] (3) SIf=subinterface.
[4] (4) P-SIf=physical interface with underlying subinterfaces.
[5] (5) Not supported on the Cisco CRS-MSC-140G.

Note For a list of supported unconditional marking operations, see the *Configuring Modular Quality of Service Packet Classification on Cisco IOS XR Software* module.

Policer Granularity and Shaper Granularity

Table 5 shows the default policer granularity values.

SPA Interface Processor	Policer Granularity Default Value
Cisco 12000 SIP-401	64 kbps
Cisco 12000 SIP-501	64 kbps
Cisco 12000 SIP-601	64 kbps
Cisco CRS Series Modular Services Card 40G	244 kbps

Table 6 shows the default shaper granularity values.

SPA Interface Processor	Shaper Granularity Default Value
Cisco CRS Series Modular Services Card 40G	256 kbps

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

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.

Bundle QoS Granularity

The Bundle QoS Granularity feature supports shaper and policer rate configuration in units of per-thousand/per-million which provides the ability to provision shape/police rates down to 1 Mbps on link aggregation (LAG) interfaces even with 100 GE bundle members.

Bundle QoS granularity supports both, ingress and egress policy-maps. The supported bundle-member types are, 100GE, 10GE, 40GE, OC768, OC192.

Limitations for Bundle QoS Granularity

These are the limitations for Bundle QoS Granularity:

- The maximum number of bundle members allowed are 64.
- No support for non-bundle interfaces.
- Policer or shaper can be only up to 16G per class on 40G linecard, and 128G for 140G linecard.
- Maximum number of supported classes across all service-policy instances for each linecard is 32000, depending on the use of the Ternary Content Addressable Memory (TCAM) memory.
- On 140G linecard, the number of classes supported for each policy map in bundle interfaces is 512/The number of members in the bundle).

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

On ATM interfaces, if there are other bandwidth commands configured in the same class, the **bandwidth remaining** command cannot be configured.

Restrictions

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

A policy map can have all class bandwidths specified in kilobits per second or percentages but not a mixture of both in the same class.

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

Note In the ingress direction, bandwidth calculations do not include Layer 2 overhead because Layer 2 headers are stripped off when a packet is received. In other instances, the bandwidth calculations include the Layer 2 encapsulation. In the case of PoS/SDH, the encapsulation is 4 bytes; for Ethernet, the encapsulation is 14 bytes; and for Dot1Q, the encapsulation is 18 bytes.

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. **commit**
15. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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:**	Specifies the name of the class whose policy you want to create or change.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-pmap)# class class1`	
Step 4	**bandwidth** {*rate* [*units*]| **percent** *value*} **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth percent 50`	Specifies the bandwidth allocated for a class belonging to a policy map and enters the policy map class configuration mode. In this example, class class1 is guaranteed 50 percent of the interface bandwidth.
Step 5	**bandwidth remaining percent** *value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	Specifies how to allocate leftover bandwidth to various classes. **Note** The remaining bandwidth of 40 percent is shared by class class1 and class2 (see Steps 8 and 9) in a 20:80 ratio: class class1 receives 20 percent of the 40 percent, and class class2 receives 80 percent of the 40 percent.
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 7	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class2`	Specifies the name of a different class whose policy you want to create or change.
Step 8	**bandwidth** {*rate* [*units*] | **percent** *value*} **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth percent 10`	Specifies the bandwidth allocated for a class belonging to a policy map. In this example, class class2 is guaranteed 10 percent of the interface bandwidth.
Step 9	**bandwidth remaining percent** *value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 80`	Specifies how to allocate leftover bandwidth to various classes. **Note** The remaining bandwidth of 40 percent is shared by class class1 and class2 (see Steps 8 and 9) in a 20:80 ratio: class class1 receives 20 percent of the 40 percent, and class class2 receives 80 percent of the 40 percent.
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 global configuration 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 POS 0/2/0/0`	Enters interface configuration mode and configures an interface.
Step 13	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 14	**commit**	
Step 15	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface POS 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Low-Latency Queueing with Strict Priority Queueing

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

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

Restrictions

- Unused priority queues cannot be used for a different priority level.
- The eight priority levels can be configured only on egress of main physical interface or main bundle interface.
- Eight priority levels work on Cisco ASR 9000 High Density 100GE Ethernet line cards only.
- The policy-map with eight priorities must have only one queuing class at the parent level of the priority class.
- If the policy-map has a parent class, the parent class cannot have bandwidth configured.

- Within a policy map, you can give one or more classes priority status. When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is queued to the same single priority queue.
- The **shape average**,**bandwidth**, and **random-detect**commands cannot be configured in the same class with the **priority** command.
- On the CRS-MSC-140G, a policer must be configured to limit the traffic entering priority queues. The policer rate cannot exceed the shape rate configured for the group or port.

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. **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. **commit**
13. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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*]]	**percent** *percentage*] **Example:**	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.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-pmap-c)# police rate 250`	
Step 5	**exceed-action** *action* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c-police)# exceed-action drop`	Configures the action to take on packets that exceed the rate limit.
Step 6	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map class configuration mode.
Step 7	**priority**[**level** *priority_level*] **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# priority level 1`	Specifies priority to a class of traffic belonging to a policy map. If no priority level is configured, the default is priority 1.
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 9	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to Global Configuration mode.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface POS 0/2/0/0`	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	**commit**	
Step 13	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:**	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# show policy-map interface POS 0/2/0/0`	

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

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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.

	Command or Action	Purpose
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 global configuration mode.
Step 7	Specifies the name of the class whose policy you want to create or change.**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface POS 0/2/0/0`	Enters interface configuration mode and configures an interface.
Step 8	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 9	**commit**	
Step 10	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface POS 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Policing (Two-Rate Color-Blind)

Traffic policing allows you to control the maximum rate of traffic sent or received on an interface.

Restrictions

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. **commit**
13. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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: **atm-clp** *value*—Sets the cell loss priority (CLP) bit. **cos** *value*—Sets the class of service value. Range is 0 to 7.

	Command or Action	Purpose
		discard-class *value*—Sets the discard class on IP Version 4 (IPv4) or Multiprotocol Label Switching (MPLS) packets. Range is 0 to 7. **dscp** [**tunnel**] *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** [**tunnel**] *precedence*—Sets the IP precedence and sends the packet. • **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 global configuration mode.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface pos 0/5/0/0`	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.

	Command or Action	Purpose
Step 12	**commit**	
Step 13	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface POS 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Policing (2R3C)

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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`	(Use with SIP 700 line card, ingress only) Creates or modifies a class map that can be attached to one or more interfaces to specify a matching policy and enters the class map configuration mode.
Step 3	**match [not] fr-de***fr-de-bit-value* **Example:**	(Use with SIP 700 line card, ingress only) Specifies the matching condition:

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config)# match not  fr-de 1`	• Match *not* fr-de 1 is typically used to specify a conform-color packet. • Match fr-de 1 is typically used to specify an exceed-color packet.
Step 4	**policy-map** *policy-name* **Example:** `RP/0/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-color** *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c-police)# conform-color match-not-frde`	(Use with SIP 700 line card, ingress only) Configures the class-map name to assign to conform-color packets.
Step 8	**exceed-color** *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c-police)# exceed-color match-frde`	(Use with SIP 700 line card, ingress only) Configures the class-map name to assign to exceed-color packets.
Step 9	**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

	Command or Action	Purpose
		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 10	**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 11	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
Step 12	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 13	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 14	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface`	Enters configuration mode and configures an interface.
Step 15	**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 16	**commit**	
Step 17	**show policy-map interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router# show policy-map interface`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Shaper Granularity

Use the Shaper Granularity feature to configure the shaper granularity value so that the shape rate you specify is a multiple of the shaper granularity.

Restrictions

The Shaper Granularity feature has these limitations:

- Supported on Cisco CRS Series Modular Services Card 40G.
- Shaper granularity values apply to the SIP and to all SPAs that are installed on the SIP.
- The line card must be reloaded, for the configured shape granularity to take effect.
- Effective shape rate is a multiple of the shaper granularity.

SUMMARY STEPS

1. **configure**
2. **hw-module qos output shape granularity** *granularity* **location** *interface-path-id*
3. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**hw-module qos output shape granularity** *granularity* **location** *interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# hw-module qos output shape granularity 128 location 0/4/CPU0`	Configures the specified shaper granularity on the output interface.
Step 3	**commit**	

Configuring Police Rate for LAG Granularity

Granularity of police rate for link aggregation (LAG) bundles can be defined using the following keywords:

- **police rate per-thousand**(equals to 0.1%)
- **police rate per-million** (equals to 0.001%)

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **police rate** { *rate* | **per-thousand** *rate-per-thousand* | **per-million** *rate-per-million* | **percent** *value* }

5. **exit**
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map p1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
Step 4	**police rate** { *rate* | **per-thousand** *rate-per-thousand* | **per-million** *rate-per-million* | **percent** *value* } **Example:** `RP/0/RP0/CPU0:router (config-pmap-class) # police rate per-thousand 1000`	Specifies the police rate. The configured value is the committed information rate per-million/ per-thousand (as the case may be) of the link bandwidth.
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 6	**commit**	

Configuring Shape Average for LAG Granularity

The granularity of shape average for link aggregation (LAG) bundles is defined using these keywords:

- **shape average per-thousand**(equals 0.01%)
- **shape average per-million**(equals 0.001%)

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **shape average** { *rate* | **per-thousand** *rate-per-thousand* | **per-million** *rate-per-million* | **percent** *value* }

5. **exit**
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map p1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy is to be created or changed.
Step 4	**shape average** { *rate* | **per-thousand** *rate-per-thousand* | **per-million** *rate-per-million* | **percent** *value* } **Example:** `RP/0/RP0/CPU0:router (config-pmap-class) # shape average per-thousand 1000`	Specifies the shape average rate. The configured value is the committed information rate per-million or per-thousand (as the case may be) of the link bandwidth.
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 6	**commit**	

Configuring Accounting

The steps that follow describe how to configure accounting.

Note NoLayer2 accounting option is not supported on layer 2 interfaces.

SUMMARY STEPS

1. **configure**
2. **interface** *interface-path-id*
3. **service-policy**{**input | output | type** } *service-policy-name* **account**{**layer2** | **no layer2**}
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router configure`	Enters global configuration mode.
Step 2	**interface** *interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface gig 0/6/5/5`	Configures the interface.
Step 3	**service-policy** {**input** | **output** | **type** } *service-policy-name* **account** {**layer2** | **no layer2**} **Example:** `RP/0/RP0/CPU0:router(config-if)# service-policy input p1 account layer2`	Configures accounting.
Step 4	**commit**	

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 shows the display output for the previous policy map configuration:

```
RP/0/RP0/CPU0:router# show policy-map interface bundle-pos 1 input

Bundle-POS1 input: p2
  Class voip
    Classification statistics                    (packets/bytes)      (rate - kbps)
            Matched                             : 0/0      0
```

```
            Transmitted                         : 0/0      0
            Total Dropped                       : 0/0      0
    Queueing statistics
            Vital           (packets)           : 0
    Queueing statistics
            Queue ID                            : 38
            High watermark  (packets)           : 0
            Inst-queue-len  (bytes)             : 0
            Avg-queue-len   (bytes)             : 0
            TailDrop Threshold(bytes)           : 47923200
            Taildropped(packets/bytes)          : 0/0
Class default
    Classification statistics                     (packets/bytes)      (rate - kbps)
            Matched                             : 0/0      0
            Transmitted                         : 0/0      0
            Total Dropped                       : 0/0      0
    Queueing statistics
            Vital           (packets)           : 0
    Queueing statistics
            Queue ID                            : 36
            High watermark  (packets)           : 0
            Inst-queue-len  (bytes)             : 0
            Avg-queue-len   (bytes)             : 0
            TailDrop Threshold(bytes)           : 239616000
            Taildropped(packets/bytes)          : 0/0
```

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
```

Service Fragment Configurations: Example

This example shows the service-fragment premium being created.

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

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

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

Policer Granularity: Example

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 police rate and the hardware programmed police rate. The configured value will be applied only for all future traffic policies configured on the interface.

This example shows how to set the police rate deviation tolerance to 4%, on an input interface:

```
hw-module qos input police granularity 4 location 0/1/CPU0
```

Use the show hw-module qos {input | output} police granularity location commands to verify the policer granularity.

```
show hw-module qos input police granularity location 0/1/CPU0

============================
    QOS POLICE GRANULARITY
============================

 Location     Rate Deviation
              Tolerance (%)
==========    ==============
 0/1/CPU0           4
----------------------------
```

Shaper Granularity: Example

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 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.

This example shows how to set the shaper granularity to 128 kbps:

```
hw-module qos output shape granularity 128 location 0/1/CPU0
```

Use the show hw-module qos output shape granularity location command to verify theshaper granularity. The Configured Shape Granularity is the user-configured shaper granularity. The LC reload value indicates if a line card reload will be required in order to bring the configured shaper granularity rate into effect. If a configured shaper granularity is not applied, the HW Programmed Granularity is applied.

```
show hw-module qos output shape granularity location 0/1/CPU0

============= ====================================
               QOS SHAPING GRANULARITY
              ====================================
              Configured    HW            LC
 Location     Shape         Programmed    reload
              Granularity   Granularity   (Y / N)
============  ===========   ============ ========
 0/1/CPU0         ---         256Kbps        N
------------------------------------------------
```

Configuring granularity for LAG bundles: Examples

This example shows how to configure police-rate:

```
config
     policy-map p1
     class c1
      police-rate per-thousand 100
     end-policy-map
```

```
     exit
   !
```

The **show run policy-map** command displays police-rate details:

```
show run policy-map p1
policy-map Police8
 class Pre5
  priority level 1
  police rate per-thousand 100
  !
 !
 class Pre1
   !
 class class-default
 !
 end-policy-map

!
```

This example shows how to configure shape average:

```
config
     policy-map p1
     class c1
      shape average per-million 1000
     end-policy-map
     exit
    !
```

The **show run policy-map** command displays shape average details:

```
show run policy-map p1
policy-map p3
 class class-default
  shape average per-million 1000
!
end-policy-map
!
```

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

Related Topic	Document Title
Initial system bootup and configuration	*Cisco IOS XR Getting Started Guide for the* Cisco CRS Router
Master command reference	Cisco CRS Router Master Command Listing
QoS commands	Cisco IOS XR Modular Quality of Service Command Reference for the Cisco CRS Router
User groups and task IDs	*"Configuring AAA Services on Cisco IOS XR Software"* module of *Cisco IOS XR System Security Configuration Guide*

Standards

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

MIBs

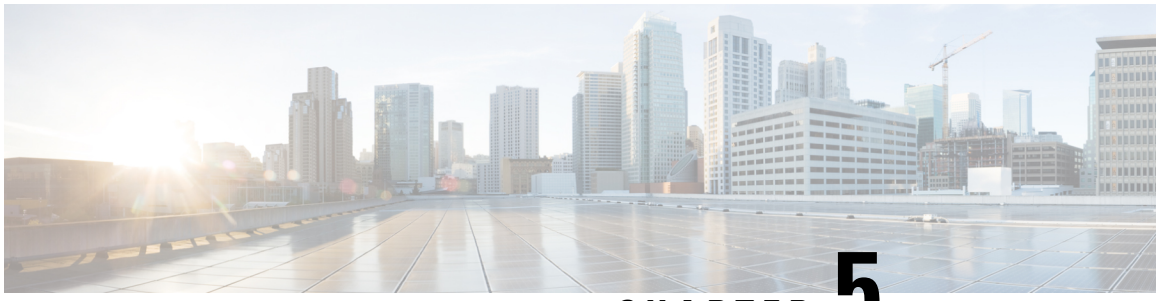
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

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

Technical Assistance

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



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 2.0	The Packet Classification feature was introduced.
Release 3.2	IPv6 addressing was supported for the **match dscp** and **match precedence** commands.
Release 3.3.0	The Hierarchical Ingress Policing feature was introduced. The following commands were added: • **match cos** • **match vlan** The **qos-group** keyword was added to the **set discard-class** command. The **discard-class** keyword was added to the **set qos-group** command. The maximum number of classes permitted per policy map increased from 16 to 32. The **not** keyword was added to all **match** commands, except the **match access-group** command.

Release 3.4.0	The **match vlan** command range was changed from 0 to 8096 to 1 to 4094.
Release 3.6.0	The Fabric Quality of Service Policies and Classes content was moved to a new module. See *Configuring Fabric Quality of Service Policies and Classes on Cisco IOS XR Software*. Policy Propagation using BGP (QPPB) feature was introduced. The maximum number of classes permitted for each policy map was increased from 32 to 512. The **match not vlan** command was not supported. The **match-all** keyword was removed from the **class-map** command.
Release 3.8.0	The In-Place Policy Modification feature was introduced. Support for VPLS QoS was introduced. The Unconditional Multiple Action Set feature was introduced. Support for the ATM CLP bit was introduced.
Release 3.9.0	Added support for **match precendence tunnel** and **match dscp tunnel** for some Layer 2 ingress interfaces (CSCsv50430).
Release 4.0.0	Removed the for ATM on Layer 2 VPN section—it is not supported.
Release 4.1.0	Support for dynamic modification of interface bandwidth was introduced.

PrerequisitesforConfiguringModularQoSPacketClassification

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.

For ATM, ingress QoS is implemented in the modular services card (MSC). Egress marking and policing are implemented in the MSC. Egress queueing features are implemented in the shared port adapter (SPA). For egress features that are implemented in the SPA, classification is still done in the MSC.

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 The **class-map** command supports the **match-any** keyword only. The **match-all** keyword is not supported.

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]** 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. The **match-any** keyword is the only evaluation option supported.

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 2: Supported Traffic Class Match Criteria

Match Criteria	Layer 2 Ingress			Layer 2 Egress			Layer 3 Ingress			Layer 3 Egress		
	PAC	CAC	p-C	PAC	CAC	p-C	Phy	SIf	P-SIf	Phy	SIf	P-SIf
prec	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
dscp	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
vlan	Y	N	Y	Y	N	Y	Y	N	Y	Y	N	Y
CoS	Y	Y	Y	Y	Y	Y	Y	Y9	Y9	N	N	N
qos-group	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
destaddress	N	N	N	Y	Y	Y	N	N	N	Y	Y	Y
EXP	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
protocol	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
access-group	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
mac, src	Y	Y	Y	N	N	N	N	N	Y	N	N	N
mac, dst	N	N	N	Y	Y	Y	N	N	N	N	N	Y
spp-priority	N	N	N	N	N	N	Y	Y	Y	N	N	N
vpls known	Y	Y	Y	N	N	N	N	N	Y	N	N	Y

Match Criteria	Layer 2 Ingress			Layer 2 Egress			Layer 3 Ingress			Layer 3 Egress		
vpls unknown	Y	Y	Y	N	N	N	N	N	Y	N	N	Y
vpls multicast	Y	Y	Y	N	N	N	N	N	Y	N	N	Y
match atm-clp	Y	Y	N	N	N	N	N	Y	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 function of these commands is described more thoroughly in the *Cisco IOS XR Modular Quality of Service Command Reference for the* Cisco CRS Router.

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 512 traffic classes can be associated to a single traffic policy. The 512 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 *Cisco IOS XR Modular Quality of Service Command Referencefor the* Cisco CRS Router.

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.

Port Shape Policies

When a port shaping policy is applied to a main interface, individual regular service policies can also be applied on its subinterfaces. Port shaping policy maps have these restrictions:

- class-default is the only allowed class map.
- The shape class action is the only allowed class action.
- They can only be configured in the egress direction.
- They can only be applied to main interfaces, not to subinterfaces.
- Two- and three- level policies are not supported. Only one level or flat policies are supported.

If any of the above restrictions are violated, the configured policy map is applied as a regular policy, not a port shaping policy.

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.

- Use CoS unconditional packet marking to assign packets to set the priority value of 802.1p/Inter-Switch Link (ISL) packets. The router uses the CoS value to determine how to prioritize packets for transmission and can use this marking to perform Layer 2-to-Layer 3 mapping. To set the Layer 2 CoS value of an outgoing packet, use the **set cos** command in policy map 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 3: Supported Class-based Unconditional Packet Marking Operations

Marking Operation	Layer 2 Ingress			Layer 2 Egress			Layer 3 Ingress			Layer 3 Egress		
	PAC	CAC	p-C	PAC	CAC	p-C	Phy	SIf	P-SIf	Phy	SIf	P-Sif
prec	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
prec tunnel	Y	Y	Y	N	N	N	Y	Y	Y	N	N	N

Marking Operation	Layer 2 Ingress			Layer 2 Egress			Layer 3 Ingress			Layer 3 Egress		
dscp	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
dscp tunnel	Y	Y	Y	N	N	N	Y	Y	Y	N	N	N
CoS	N	N	N	Y	Y	Y	N	N	N	Y	Y	Y
srp-priority	N	N	N	Y	Y	Y	N	N	N	Y	Y	Y
qos-group	Y	Y	Y	N	N	N	Y	Y	Y	N	N	N
discard-class	Y	Y	Y	N	N	N	Y	Y	Y	N	N	N
EXP, imposition	Y	Y	Y	N	N	N	Y	Y	Y	N	N	N
EXP, topmost	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y
atm-clp	N	N	N	N	N	N	N	N	N	N	Y	N

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.

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	Layer 2 Packets
discard-class	dscp or precedence	MPLS experimental imposition	MPLS experimental imposition
qos-group	tunnel dscp or tunnel precedence	MPLS experimental topmost	tunnel dscp or tunnel precedence
—	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

Both cos and srp-priority packets are marked as independent from other markings. In addition to the cos and srp-priority 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 and srp-priority 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 or srp-priority	dscp or precedence MPLS Experimental Imposition	MPLS experimental topmost MPLS Experimental Imposition

Note

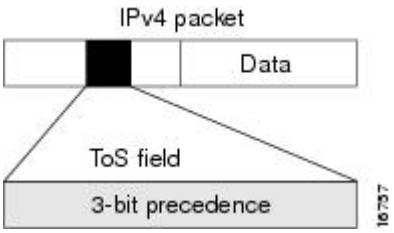
- Both cos and srp-priority packets are mutually exclusive.
- Both DSCP and precedence packets are mutually exclusive.

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 4: 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.
- QoS Policy Propagation Using Border Gateway Protocol (QPPB). See QoS Policy Propagation Using Border Gateway Protocol.

Note The CRS-MSC-140G does not support QPPB.

TCP Establishment DSCP Marking/ Set IP Precedence/DSCP for NTP

The Differentiated Services Code Point (DSCP) field in an IP packet which helps enables different levels of service to be assigned to network traffic. Marking is a process, which helps to modify QOS fields incoming and outgoing packets. You can use marking commands in traffic classes, which are referenced in the policy map. You can configure the following marking features:

- DSCP
- IP Precedence
- CoS

Each IP packet is marked with a DSCP code and assigned to corresponding level of service. DSCP is a combination of IP Precedence and Type of Service fields. The TCP Establishment DSCP Marking/Set IP Precedence feature sets Network Time Protocol (NTP) with the DSCP field. NTP packets can be based on either IPv4 and IPV6 based respectively. The NTP sets DSCP/TOS field under either v4 or v6 IP headers. The DSCP level can be configured through the NTP configuration. The configured level will be set across NTP packets throughout IP layer.

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.

Configuring DSCP for source IPv4 address for NTP Packets

To mark a packet by setting the **IP DSCP** value for **NTP** packets, use the following commands (given below) starting in global configuration mode. These commands permit configuring the DSCP for source addresses, to mark **NTP** packets, so that the marked **NTP** packets are treated as per the DSCP markings. There are different code point values available for different services:

SUMMARY STEPS

1. **configure**
2. **ntp** {*ipv4*} **dscp**
3. **end** or **commit**
4. **show processes ntpd**
5. **show ntp associations**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure**
	Example:
	`RP/0/0/CPU0:Router#configure`
	`Mon Aug 10 14:35:04.826 IST`
	`RP/0/0/CPU0:Router(config)#`
		Enters the global configuration mode.

Step 2	**ntp** {*ipv4*} **dscp**
	Example:
	`RP/0/0/CPU0:Router(config)#ntp ipv4 dscp ?`
	`<0-63> Differentiated services codepoint value`
	`af11 Match packets with AF11 dscp (001010)`
	`af12 Match packets with AF12 dscp (001100)`
	`af13 Match packets with AF13 dscp (001110)`
	`af21 Match packets with AF21 dscp (010010)`
	`af22 Match packets with AF22 dscp (010100)`
	`af23 Match packets with AF23 dscp (010110)`
	`af31 Match packets with AF31 dscp (011010)`
	`af32 Match packets with AF32 dscp (011100)`
	`af33 Match packets with AF33 dscp (011110)`
	`af41 Match packets with AF41 dscp (100010)`
	`af42 Match packets with AF42 dscp (100100)`
	`af43 Match packets with AF43 dscp (100110)`
	`cs1 Match packets with CS1(precedence 1) dscp (001000)`
	`cs2 Match packets with CS2(precedence 2) dscp (010000)`
	`cs3 Match packets with CS3(precedence 3) dscp (011000)`
	`cs4 Match packets with CS4(precedence 4) dscp (100000)`
	`cs5 Match packets with CS5(precedence 5) dscp (101000)`
	`cs6 Match packets with CS6(precedence 6) dscp (110000)`
	`cs7 Match packets with CS7(precedence 7) dscp (111000)`
	`default Match packets with default dscp (000000)`
	`ef Match packets with EF dscp (101110)`
		Specifies Differentiated services code point (dscp) value. The range is from 0 to 63. The default value is 0.

Step 3	**end** or **commit**
	Example:
	`RP/0/RP0/CPU0:Router(config)#commit`
	Example:
	`RP/0/0/CPU0:Router#exit`
	`(`
		Commit- saves the configuration changes and remains within the configuration session.
		End- prompts the 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 4	**show processes ntpd**
	Example:
	`RP/0/RP0/CPU0:Router#show processes ntpd`
	`Mon Jun 22 20:25:18.026 IST`
	`Job Id: 208`
	`PID: 2540`
	`Executable path: /pkg/bin/ntpd`
	`Instance #: 1`
	`Version ID: 00.00.0000`
	`Respawn: ON`
	`Respawn count: 1`
	`Last started: Fri Jun 19 16:04:14 2015`
	`Process state: Run`
	`Package state: Normal`
	`Process group: dlrsc`
	`core: MAINMEM`
	`Max. core: 0`
	`Level: 120`
	`Placement: None`
	`startup_path: /pkg/startup/ip_ntp.startup`
	`Ready: 2.444s`
	`Process cpu time: 0.074s user, 0.031s kernel, 1.005s total`
	`PID TID CPU Stack Pri State Run Time CPU use NAME`
	`2540 2978 1 92K 15 Sleeping 3:04:20:59s 0.000s ntpd`
	`2540 2975 4 92K 15 Sleeping 3:04:20:59s 0.001s ntpd`
	`2540 2947 5 92K 15 Sleeping x3:04:20:59s 0.027s chkpt_evm`
	`2540 2943 1 92K 15 Sleeping 3:04:21:00s 0.000s ITAL Server Thr`
	`2540 2914 5 92K 15 Sleeping 3:04:21:00s 0.000s async`
	`2540 2810 3 92K 15 Sleeping 3:04:21:00s 0.000s EnXR internal:mmap_peer_threa`
	`2540 2760 2 92K 15 Sleeping 3:04:21:00s 0.011s ntpd`
	`2540 2540 1 92K 15 Sleeping 3:04:21:03s 0.064s ntpd`
		Displays the ntpd process

Step 5	**show ntp associations**
	Example:
	`Router# show ntp associations`
	`Sat Feb 14 13:53:18.468 UTC`
		Shows the status of NTP associations.

	Command or Action	Purpose
	address ref clock st when poll reach delay offset disp *~223.255.254.254 171.68.38.65 2 121 128 377 2.44 -0.260 4.537 * sys_peer, # selected, + candidate, - outlayer, x falseticker, ~ configured	

Configure DSCP CS7 (Precedence 7)

Before you begin

The IP DSCP value in the class map command using the following commands, starting with the global configuration mode.

SUMMARY STEPS

1. **configure**
2. **ntp** *ipv4* **dscp** *cs7*
3. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** RP/0/0/CPU0:Router#configure Mon Aug 10 14:35:04.826 IST RP/0/0/CPU0:Router(config)#	Enters the global configuration mode.
Step 2	**ntp** *ipv4* **dscp** *cs7*	Configures options in **DSCP** for a particular source address in IPv4 packets.
Step 3	**end** or **commit** **Example:** RP/0/RP0/CPU0:Router(config)#commit	Commit Command saves the configuration changes and remains within the configuration session. End Command prompts the 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.

Configure IPv4 DSCP Precedence

Before you begin

The following steps help you to configure **DSCP** precedence:

SUMMARY STEPS

1. **configure**
2. **ntp** {*ipv4* } **precedence** *codepoint_value*
3. **ntp** { *ipv4*} **precedence**
4. **end** or **commit**
5. **show ntp status**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure**
	Example:
	`RP/0/0/CPU0:Router#configure`
	`Mon Aug 10 14:35:04.826 IST`
	`RP/0/0/CPU0:Router(config)#`
		Enters the global configuration mode.

Step 2	**ntp** {*ipv4* } **precedence** *codepoint_value*
	Example:
	`RP/0/0/CPU0:Router(config)#ntp ipv4 precedence ?`
	`<0-7> Precedence value`
	`critical Match packets with critical precedence (5)`
	`flash Match packets with flash precedence (3)`
	`flash-override Match packets with flash override precedence (4)`
	`immediate Match packets with immediate precedence (2)`
	`internet Match packets with internetwork control precedence (6)`
	`network Match packets with network control precedence (7)`
	`priority Match packets with priority precedence (1)`
	`routine Match packets with routine precedence (0)`
		Sets the ntp [IPv4] precedence. It ranges from 0 to 63.

Step 3	**ntp** { *ipv4*} **precedence** **Example:**	Sets precedence values.

	Command or Action	Purpose
	RP/0/RP0/CPU0:Router(config)#ntp ipv4 precedence internet	
Step 4	**end** or **commit** **Example:** RP/0/RP0/CPU0:Router(config)#commit **Example:** RP/0/0/CPU0:Router#exit (	Commit- saves the configuration changes and remains within the configuration session. End- prompts the 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 ntp status** **Example:** RP/0/RP0/CPU0:Router#show ntp status Mon Aug 10 14:35:04.826 IST Clock is synchronized, stratum 3, reference is 223.255.254.254 nominal freq is 1000000000.0000 Hz, actual freq is 30440042.8893 Hz, precision is 2**22 reference time is D889D255.BF525356 (13:55:33.747 UTC Sat Feb 14 2015) clock offset is -0.413 msec, root delay is 3.569 msec root dispersion is 20.55 msec, peer dispersion is 4.04 msec loopfilter state is 'CTRL' (Normal Controlled Loop), drift is 0.0000318515 s/s system poll interval is 128, last update was 117 sec ago	Displays NTP status.

Configure IPv6 DSCP precedence

Before you begin

You can configure **DSCP** precedence:

SUMMARY STEPS

1. **configure**
2. **ntp ,** source { *ipv6* } **dscp** *codepoint_value*

3. **ntp** { *ipv6* } **precedence** *codepoint_value*
4. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:Router(config)#`	Enters the global configuration mode.

Step 2	**ntp ,** source { *ipv6* } **dscp** *codepoint_value*
	Example:
	`RP/0/0/CPU0:Router(config)#ntp ipv6 dscp ?`
	`<0-63> Differentiated services codepoint value`
	`af11 Match packets with AF11 dscp (001010)`
	`af12 Match packets with AF12 dscp (001100)`
	`af13 Match packets with AF13 dscp (001110)`
	`af21 Match packets with AF21 dscp (010010)`
	`af22 Match packets with AF22 dscp (010100)`
	`af23 Match packets with AF23 dscp (010110)`
	`af31 Match packets with AF31 dscp (011010)`
	`af32 Match packets with AF32 dscp (011100)`
	`af33 Match packets with AF33 dscp (011110)`
	`af41 Match packets with AF41 dscp (100010)`
	`af42 Match packets with AF42 dscp (100100)`
	`af43 Match packets with AF43 dscp (100110)`
	`cs1 Match packets with CS1(precedence 1) dscp (001000)`
	`cs2 Match packets with CS2(precedence 2) dscp (010000)`
	`cs3 Match packets with CS3(precedence 3) dscp (011000)`
	`cs4 Match packets with CS4(precedence 4) dscp (100000)`
	`cs5 Match packets with CS5(precedence 5) dscp (101000)`
	`cs6 Match packets with CS6(precedence 6) dscp (110000)`
	`cs7 Match packets with CS7(precedence 7) dscp (111000)`
	`default Match packets with default dscp (000000)`
	`ef Match packets with EF dscp (101110)`
		Configures options in **DSCP** for a particular source address in IPV6 packets. The value ranges between 0 - 63.

Step 3	**ntp** { *ipv6* } **precedence** *codepoint_value* **Example:**	Sets **ntp** *ipv6* **precedence**

	Command or Action	Purpose
	RP/0/0/CPU0:Router(config)#ntp ipv6 precedence ? <0-7> Precedence value critical Match packets with critical precedence (5) flash Match packets with flash precedence (3) flash-override Match packets with flash override precedence (4) immediate Match packets with immediate precedence (2) internet Match packets with internetwork control precedence (6) network Match packets with network control precedence (7) priority Match packets with priority precedence (1) routine Match packets with routine precedence (0)	
Step 4	**end** or **commit** **Example:** RP/0/RP0/CPU0:Router(config)#commit **Example:** RP/0/0/CPU0:ios#exit (	Commit- saves the configuration changes and remains within the configuration session. End- prompts the 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.

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.

Note The Cisco CRS router does not support ip-precedence-based QPPB (support only for qos-based QPPB).

QoS on PWHE

QoS on Pseudo-wire Head End (PWHE) enables enhanced L3VPN and L2VPN service on a service-provider-edge router. The available PWHE types are PW-Ether main interfaces, PW-Ether subinterfaces, and PW interworking (IW) interfaces.

Note The PWHE-Ether subinterfaces and PW-IW interfaces are supported from Release 5.1.1 onwards.

For more information on PWHE-QoS, refer the *Modular QoS Configuration Guide for Cisco CRS Routers* .

Supported Features

Features of QoS on PWHE:

- IPv4 and IPv6 address-families are supported.
- Policy maps on both ingress and egress PWHE. Both ingress and egress support policing, marking, and queuing within hardware limitations.
- Policies at the port for the transit traffic can be applied simultaneously with policies for PWHE interfaces.
- Policy is replicated on all PWHE members. This means the rate specified in the PWHE policy-map is limited to the lowest rate of all the pin down members. For example, if the PWHE interface has both 1G and 10G pin down members, the rate is limited to 1G. if the 10G member has a shaper of 900 mbps, the rate of the PWHE interface policy is limited to 900 mbps.
- Port shaping policy on the member interface will impact the PWHE traffic passing through that port.
- Policy maps can be applied on PW-Ether subinterface.
- PW-Ether subinterfaces inherit policy on its main PW-Ether interface.
- PW-Ether subinterface can have policy configured as shared policy instance (SPI).

- PW-Ether main and subinterface policies may co-exist.
- L2 multicast and flood over PW-Ether interface are supported.
- L3 multicast over PW-Ether interface are supported.
- Independent of line card co-existence mode, percentage based rate at the lowest policy level in PW-Ether main and subinterface policies is supported.

Note In the same policy, the grand parent level is lower than parent level, and parent level is lower than child level.

Limitations

- QoS accounting doesnot include pseudowire header.
- 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.
- QoS configuration for a PWHE interface is allowed even if the generic-interface-list is not attached or the generic interface list has no configured interfaces. If the PWHE interface configuration holds a feature that is not supported, it may not be alerted to the operator until after the generic interface list with associated interfaces is attached.

  In such cases, one scenario could be that the invalid QoS configuration is detected after a generic-interface-list configuration is applied, resulting in that specific PWHE remaining down. An IOS error message is displayed, asking for user intervention and an automatic retry is initiated.

  Sample error message for an invalid QOS configuration:

  RP/0/RSP0/CPU0:2019 Jan 19 13:52:06.255 CET: pwhe_ma[404]: %L2-L2VPN_MA_PWHE-3-REPLICATION_FAILED : Interface replication failure; interface name: PW-Ether2098; error code: 'qos-ea' detected the 'warning' condition 'Cannot bind policy-map to both main interface and sub-interface in the same direction'. Replication will be retried.

  Refer to the IOS error message and remove the invalid QoS configuration under the PWHE interface. This issue is resolved after the invalid configuration is removed and the PWHE interface comes up automatically.

Bandwidth Distribution

PWHE and non-PWHE traffic on the same pin down member share scheduling resources. It is recommended to configure bandwidth remaining in the parent class-default of PWHE policies to control the distribution of excess bandwidth between PWHE and non-PWHE traffic.

Bandwidth remaining command can be used in the parent default class of PWHE policies allowing user to control the distribution of excess bandwidth between various PWHE interfaces and physical interface.

QoS Accounting

- The packet length when performing QoS functions (policing, shaping, statistics, etc.) will be based on the customer IP packet, customer L2 header and the configured additional overhead.
- QoS statistics will include the customer IP packet, customer L2 header and configured additional overhead.

Note For PW-IW interfaces, the packet length used for QoS accounting does not contain customer L2 header.

- Outer MPLS headers (VC label, transport labels, etc.) and outer L2 header (Layer 2 encap of the underlying physical interface) will not be included in the packet length when performing QoS on the PWHE virtual interface.

Classification and Marking Support

Marking for PW-Ether in ingress and egress direction

- Marking of customer IP header, qos-group and discard-class will be supported.
- Marking of EXP bits for all imposed MPLS labels will be supported for PWHE main interface and PW-Ether subinterfaces.
- EXP for imposed labels can be set in an ingress or an egress policy attached to a PWHE interface.

Note For non-PWHE interfaces, EXP for imposed labels can only be set in an ingress policy. This is an exception made for PWHE interfaces because more labels are imposed on the customer IP packet after processing the egress QoS policy.

- For unconditional markings in ingress direction, the following fields can be marked - DSCP/precedence, EXP for imposed labels, qos-group and discard-class.
- For unconditional markings in egress direction, the following fields can be marked - DSCP/precedence, discard-class and EXP for imposed labels.
- For conditional policer markings in ingress direction, at most two of the following fields can be marked - DSCP/precedence, EXP for imposed labels, qos-group and discard-class.
- For conditional policer markings in egress direction, the following fields can be marked - DSCP/precedence, discard-class and EXP for imposed labels.

L2 header based classification and marking support

The Table-1, Table-2 and Table-3 summarizes the L2 header based classification and marking support on different PWHE interfaces.

Table 5: Supported L2 header based classification and marking for PW-Ether VC type 4 interface

PW-Ether VC type 4		
Classification	**Ingress**	**Egress**
SRC MAC	Yes	No
DEST MAC	Yes	No

DEI	Yes	No
DEI Inner	No	No
COS	Yes	No
COS Inner	No	No
VLAN	Yes	No
VLAN Inner	No	No
Marking		
DEI	No	No
COS	No	No
COS Inner	No	No

Table 6: Supported L2 header based classification and marking for PW-Ether VC type 5 interface

PW-Ether VC type 5		
Classification	**Ingress**	**Egress**
SRC MAC	Yes	No
DEST MAC	Yes	No
DEI	Yes	Yes
DEI Inner	No	No
COS	Yes	Yes
COS Inner	Yes	Yes
VLAN	No	No
VLAN Inner	No	No
Marking		
DEI	No	Yes
COS	No	Yes
COS Inner	No	Yes

Note The classification and marking applied on PW-Ether main interface are inherited by its subinterfaces without policy.

Table 7: Supported L2 header based classification and marking for PW-Ether L3 subinterface VC type 5

PW-Ether L3 subinterface VC type 5		
Classification	**Ingress**	**Egress**
SRC MAC	Yes	No
DEST MAC	Yes	No
DEI	Yes	Yes
DEI Inner	No	No
COS	Yes	Yes
COS Inner	Yes	Yes
VLAN	Yes	Yes
VLAN Inner	Yes	Yes
Marking		
DEI	No	Yes
COS	No	Yes
COS Inner	No	Yes

For PW-Ether L2 subinterface VC type 5, all classification and marking are supported.

L2 classification and marking are not supported for PW-IW interface VC type 11.

Policing and Queuing support

All the policing features supported on normal L3 interfaces will be supported on PWHE main interface and subinterface too.

Queuing

	Ingress and Egress Queues	**Ingress and Egress Policers**
PWHE interface with no policy map	Each PWHE member has per port default queues. Both the ingress and egress traffic will use the members port default queue.	Not applicable
PWHE interface with a policy map	Any ingress and egress queues in the policymaps would be replicated on each PWHE member.	Any ingress and egress policer in the policymaps would be replicated per each PWHE member.

Note If PWHE member is a bundle, policy maps will be replicated on bundle members.

Statistics

Show commands of a PWHE virtual interface and PWHE subinterface QoS policy will provide ingress / egress statistics;

- per pin down member.
- per bundle member if the bundle is a pin down member.
- aggregated stats on the whole PWHE interface.
- shared policy instance per pin down member.
- aggregated stats on the whole bundle if the bundle is a pin down member.
- PWHE aggregate shaper stats aggregates all queuing stats of all subinterfaces.

Co-existence of PWHE Main and Subinterface Policies

A line card (LC) can be configured to allow PWHE aggregate shaper policy to co-exist with subinterface policies. This mode is known as co-existence mode. The PWHE aggregate shaper policy will only have a class-default with shape and bandwidth remaining actions. If no PWHE subinterface policy exists, PWHE main interface can have up to 3 level-queuing hierarchical policy.

The co-existence mode with subinterface queuing policies is known as co-existence queuing mode. The co-existence mode with subinterface non-queuing policies is known as co-existence non-queuing mode.

As shown in below examples, PWHE aggregate shaper policy can have:

- only shape action
- only bandwidth remaining action
- shape and bandwidth remaining actions.

Here is the example for PWHE aggregate shaper policy with only shape action:

```
policy-map pwhe-aggregate-shaper
class class-default
shape average 1 gbps
!
end-policy-map
!
end
```

Here is the example for PWHE aggregate shaper policy with only bandwidth remaining action:

```
policy-map pwhe-aggregate-shaper
class class-default
bandwidth remaining ratio 20
!
end-policy-map
!
end
```

Here is the example for PWHE aggregate shaper policy with shape and bandwidth remaining actions:

```
policy-map pwhe-aggregate-shaper
class class-default
shape average 1 gbps
bandwidth remaining ratio 20
```

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

Note It is recommended to configure shape and bandwidth remaining actions for PWHE aggregate shaper policy.

Restrictions

These restrictions apply while configuring co-existence mode:

- If co-existence mode is configured for all LCs in ingress direction then co-existence mode configuration for specified LC in ingress will be rejected. But co-existence configuration for specified LC in egress will be accepted provided there is no co-existence mode configured for all LCs in egress direction.
- If any PWHE main or subinterface has policy configured on a LC, configuring or not configuring co-existence mode will take effect after the LC reloads.
- If no PWHE main or subinterface has policy configured on a LC, configuring or not configuring co-existence mode will take effect immediately on the LC. It is recommended to commit the co-existence mode change before adding QoS policies on the PWHE main or subinterfaces.
- In the co-existence queuing mode, policy applied on PWHE subinterface will have up to 2-levels of queuing. Configuring a 3-level queuing policy on PWHE subinterface will be rejected.
- In the co-existence non-queuing mode, only non-queuing policies on subinterfaces are allowed to co-exist with the PWHE aggregate shaper. If PWHE main interface does not have policy, then subinterface policy can have up to 2-level of queuing.
- When a LC is not in co-existence mode, the PWHE main interface and subinterfaces cannot have policies at the same time. But each can have policy if the other does not.
- The traffic for PWHE main interface and subinterfaces without queuing policy will use the pin down interface default queue. The behavior is consistent whether the LC is in co-existence mode or not.
- In co-existence queuing, non-queuing mode or co-existence disabled (default) mode, applying a non-aggregate shaper policy on PWHE main interface is allowed if subinterface policy does not exists. The non-aggregate shaper policy can have up to 3-levels of queuing. If non-aggregate shaper policy applied on PWHE main interface is a queuing policy, it impacts traffic on the PWHE main interface and subinterfaces because the traffic is moving from the port default queues to the new queues created for the PWHE.
- After PWHE subinterface policies are applied, in-place modification of the PWHE aggregate shaper is also allowed but after the modification the policy should still be a PWHE aggregate shaper.
- PWHE subinterface policy co-existing with PWHE aggregate shaper is allowed to be configured as SPI.

PW-Ether Subinterface Policy

QoS policies can be applied on PW-Ether subinterfaces when there is no policy applied on the main PW-Ether interface.

Restrictions

- When the LC is not in co-existence mode, policies supported on regular subinterface are supported on PW-Ether subinterface too.
- Percentage based rate on the lowest level is supported on policy applied on PW-Ether subinterface.

Note In the same policy, the grand parent level is lower than parent level, and parent level is lower than child level.

- When LC is not in co-existence mode, service-policy on the PW-Ether main interface is rejected if there is a service-policy already applied on any of its PW-Ether subinterfaces .

PW-Ether Subinterface Shared Policy Instance

PW-Ether subinterface supports shared policy instance (SPI). SPI on PW-Ether subinterface functions similarly to SPI on bundle subinterfaces.

Restrictions

- SPI is only supported on PW-Ether subinterfaces. Configuring SPI on PWHE main interface will be rejected.
- When a policy is applied on PW-Ether subinterface with the SPI, a single instance of the same policy is created on each pin down member.
- SPI name is unique across all PW-Ether main interfaces and bundle interfaces.

Scale Information

QoS on PWHE supports:

- 8000 PWHE interface per system.
- 1792 PWHE interface per line card (LC).
- 8 physical or bundle interfaces per generic interface list.
- 4096 sub-interfaces per PW-Ether interface.
- 20,000 total subinterfaces per LC.

Note The scale numbers are supported if configuration is applied properly so that queuing resource is not exhausted.

Policy Instantiation

The various scenarios of QoS on PWHE are discussed here:

- If any member interface has policies applied to them, only non PWHE traffic will be subjected to those policies. An exception to this is a configured port shaper.
- QoS policy applied on the PWHE main interface or PWHE subinterface is instantiated on pin-down member. If the pin-down member is a bundle, then the policy is instantiated on each bundle member .
- The supported policy combinations on PWHE main and subinterfaces for line card (LC) in any mode are:

- Non-queuing policy on PWHE main interface and no policy on subinterfaces.
- No policy on PWHE main interface and no policy or non-queuing policy on subinterfaces.
- No policy on PWHE main interface. 2-level queuing policies on subinterface with or without SPI.
- 1, 2, or 3-level queuing policy on PWHE main interface. No policies on subinterface.

- The supported policy combinations on PWHE main and subinterfaces for LC not in the co-existence mode are:
  - No policy on PWHE main interface. 3-level queuing policies on subinterfaces with or without SPI.

Note In ingress direction, policies with priority but no queuing actions in the policy-map will use the member port default queues. In egress direction, priority is treated as queuing action so dedicated queue is created for it.

- The supported policy combinations on PWHE main and subinterfaces for LC in the co-existence queuing mode are:
  - PWHE aggregate shaper policy on the PWHE main interface. Non-queuing policies on subinterfaces with or without SPI.
  - PWHE aggregate shaper policy on the PWHE main interface. Up to 2 level queuing policies on subinterfaces with or without SPI.

Note In the ingress direction, the PWHE subinterface polices with priority but no queuing action in the policy-map will use the queues created for the PWHE main interface. In the egress direction, priority is treated as queuing action so dedicated queues will be created for the subinterface. If the PWHE main interface does not have queuing policy, its subinterface with non-queuing policies will use the pindown interface default queues.

- The supported policy combination on PWHE main and subinterfaces for LC in the co-existence non-queuing mode is:
  - PWHE aggregate shaper on the PWHE main interface. Non-queuing policies on subinterfaces.

Note In ingress and egress direction, the PWHE subinterface policies with priority but no queuing action in the policy-map will use the queues created for the PWHE main interface. If the PWHE main interface does not have queuing policy, its subinterface policies with priority but no queuing action will use the pin-down interface default queues.

Note When PWHE interface is created, and no PWHE QoS policy is applied on it, PWHE traffic will pass through the member interface default queues.

PWHE without QoS policy

The following two cases represent the default behavior of the PWHE interfaces:

- PWHE ingress to core facing egress (access to core) - DSCP/ precedence value from customer IP packet is copied to EXP of all imposed labels (VPN and transport)in the core-facing direction.
- PWHE egress (core to access) - DSCP/precedence value from customer IP packet is copied to EXP of all imposed labels (VC and transport) in the access-facing direction.

Configuring QoS on PWHE: Example.

The example shows how to configure QoS on PWHE main interface or subinterfaces. The example configuration can not be applied on PWHE main and subinterfaces at the same time.

```
policy-map pw_child_in
class voip
 priority level 1
  police rate percent 1
  !
!
class video
  police rate percent 10
  !
  priority level 2
!
class data
  police rate percent 70 peak-rate percent 100
   exceed-action transmit
   violate-action drop
  !
!
class class-default
  police rate percent 19 peak-rate percent 100
   exceed-action transmit
   violate-action drop
  !
!
end-policy-map
!
policy-map pw_parent_in
class class-default
  service-policy pw_child_in
  police rate 100 mbps
   child-conform-aware
  !
!
end-policy-map
!

policy-map pw_child_out
class voip
  priority level 1
  police rate 1 mbps
  !
!
class data
  bandwidth remaining percent 70
  random-detect discard-class 3 40 ms 50 ms
!
class video
  priority level 2
```

```
  police rate 10 mbps
  !
!
class class-default
  random-detect discard-class 1 20 ms 30 ms
!
end-policy-map
!
policy-map pw_parent_out
class class-default
  service-policy pw_child_out
  shape average 100 mbps
!
end-policy-map
!

interface pw-ether 1
service-policy input pw_parent_in
service-policy output pw_parent_out
!
```

The example shows how to apply the PWHE aggregate shaper on PWHE main interface and another policy on its subinterface when the LC is in co-existence mode:

Note

- Use the **hw-module qos-mode pwhe-aggregate-shaper sub-interface** { **queuing** | **non-queuing** } { **ingress** | **egress** } command to enable co-existence mode on the LC.
- For the following example to work, the LC must be in co-existence queuing mode.
- When LC is in co-existence mode, apply only PWHE aggregate shaper policy on PWHE main interface.

```
policy-map pwhe-aggregate-shaper
class class-default
shape average 1 gbps
bandwidth remaining ratio 20
!
end-policy-map
!
policy-map pw_parent_out
class class-default
service-policy pw_child_out
shape average 100 mbps
!
end-policy-map
!

interface pw-ether 1
service-policy output pwhe-aggregate-shaper
!

interface pw-ether 1.1
service-policy output pw_parent_out
```

For other PWHE related information, please refer the

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 two 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.

Ingress Queuing Support

Ingress queuing is disabled for some line cards.

The tables below list out the ingress queuing support for fixed port and modular line cards.

Note Ingress queuing is not supported on ASR9K-SIP-700 line cards.

Fixed port Line Card

LC type	Ingress Queuing Support
A9K-24X10GE-TR/- SE	Yes
A9K-36X10GE-TR/ -SE	No
A9K-2X100GE-TR/ -SE	No
A9K-1X100GE-TR/ -SE	No
A9K-8X100G-LB-SE / -TR	No
A9K-8X100GE-SE / -TR	No
A99-8X100GE-SE / -TR	No
A9K-4X100GE-SE / -TR	Yes
A99-12X100GE	No
A9K-4X100GE	No
A9K-48X10GE-1G-SE/-TR	No

LC type	Ingress Queuing Support
A9K-24X10GE-1G-SE/-TR	No

Modular Line Card

Note The A9K-MOD400-SE/TR line cards are supported from Cisco IOS XR Release 5.3.2, and the A9K-MOD200-SE/TR line cards are supported from Cisco IOS XR Release 6.0.1.

For minimum software release versions of the new MPAs that are supported on the Cisco ASR 9000 Series 400G (A9K-MOD400-SE/TR) and 200G Modular Line Cards (A9K-MOD200-SE/TR), see Table 5 and Table 6 respectively.

LC type	EP type	Ingress Queuing Support
A9K-MOD80-TR/ -SE	A9K-MPA-20X1GE	Yes
A9K-MOD80-TR/ -SE	A9K-MPA-4X10GE	No **Note** To enable ingress queueing on these line cards, run the command **hw-module all qos-mode ingress-queue-enable**.
A9K-MOD80-TR/ -SE	A9K-MPA-2X10GE	Yes
A9K-MOD80-TR/ -SE	A9K-MPA-1X40GE	No
A9K-MOD400-SE	A9K-MPA-8X10GE	Yes
A9K-MOD400-SE	A9K-MPA-20X10GE	No **Note** To enable ingress queueing on these line cards, run the command **hw-module all qos-mode ingress-queue-enable**.
A9K-MOD400-SE	A9K-MPA-2X100GE	No
A9K-MOD400-SE	A9K-MPA-1X100GE	No
A9K-MOD400-SE	A9K-MPA-2X100GE	Yes
A9K-MOD400-SE	A9K-MPA-32X1GE	Yes
A9K-MOD200-SE	A9K-MPA-8X10GE	No
A9K-MOD200-SE	A9K-MPA-4X10GE	Yes

LC type	EP type	Ingress Queuing Support
A9K-MOD200-SE	A9K-MPA-2X10GE	Yes
A9K-MOD200-SE	A9K-MPA-2X40GE	No
A9K-MOD200-SE	A9K-MPA-1X40GE	Yes
A9K-MOD200-SE	A9K-MPA-20X1GE	Yes
A9K-MOD200-SE	A9K-MPA-32X1GE	Yes
A9K-MOD200-SE	A9K-MPA-1X100GE	No
A9K-MOD200-SE	A9K-MPA-10X10GE	No
A9K-24X10GE-1G	4X1GE , 4X10GE	No
A9K-48X10GE-1G	4X1GE , 4X10GE	No
A99-12X100GE/A9K-4X100GE	QSFP-4X10G	No
A99-12X100GE/A9K-4X100GE	QSFP-1X40G	No
A99-12X100GE/A9K-4X100GE	QSFP-1x100G	No
ASR9901	All types	No
A9K-MOD160-TR/ -SE	A9K-MPA-20X1GE	Yes
A9K-MOD160-TR/ -SE	A9K-MPA-4X10GE	Yes
A9K-MOD160-TR/ -SE	A9K-MPA-2X10GE	Yes
A9K-MOD160-TR/ -SE	A9K-MPA-1X40GE	No
A9K-MOD160-TR/ -SE	A9K-MPA-2X40GE	No
A9K-MOD160-TR/ -SE	A9K-MPA-8X10GE	No
A9K-MOD200-TR/ -SE	A9K-MPA-20X1GE	Yes
A9K-MOD200-TR/ -SE	A9K-MPA-4X10GE	Yes
A9K-MOD200-TR/ -SE	A9K-MPA-2X10GE	Yes
A9K-MOD200-TR/ -SE	A9K-MPA-1X40GE	Yes
A9K-MOD200-TR/ -SE	A9K-MPA-2X40GE	No
A9K-MOD200-TR/ -SE	A9K-MPA-8X10GE	No
A9K-MOD400-TR/ -SE	A9K-MPA-20X1GE	Yes
A9K-MOD400-TR/ -SE	A9K-MPA-4X10GE	Yes
A9K-MOD400-TR/ -SE	A9K-MPA-2X10GE	Yes

LC type	EP type	Ingress Queuing Support
A9K-MOD400-TR/ -SE	A9K-MPA-1X40GE	Yes
A9K-MOD400-TR/ -SE	A9K-MPA-2X40GE	No
A9K-MOD400-TR	A9K-MPA-8X10GE	Yes
A9K-MOD400-TR	A9K-MPA-20X10GE	No
A9K-MOD400-TR	A9K-MPA-2X100GE	No
A9K-MOD400-TR	A9K-MPA-1X100GE	No
ASR9001-LC	Chassis fixed 4X10GE	No
ASR9001-LC	A9K-MPA-4X10GE	No
ASR9001-LC	A9K-MPA-2X10GE	No
ASR9001-LC	A9K-MPA-1X40GE	No
ASR9001-LC	A9K-MPA-20X1GE	No

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.

Dynamic Modification of Interface Bandwidth

This section describes the dynamic modification of interface bandwidth feature.

Gigabit Ethernet SPA with Copper SFP

For a Gigabit Ethernet interface containing copper small form-factor pluggable (SFP) modules, the interface bandwidth could change dynamically due to autonegotiation. If the new interface bandwidth is compatible with the current interface and feature configuration, no user action is required. If the new bandwidth is *not* compatible, the system displays an error notification message, and continues to handle traffic on a best-efforts basis. User action is required in this situation to overcome the error condition.

The content of the error notification is based on the policy states listed in the Policy States.

This feature is supported on the copper Gigabit Ethernet interfaces in these shared port adapters (SPAs):

- SPA-5X1GE-V2
- SPA-8X1GE-V2
- SPA-10X1GE-V2

POS Interfaces On SPA-8xOC12-POS

The dynamic modification of interface bandwidth feature is supported on the SPA-8xOC12-POS. The bandwidth of the POS interface changes when an OC-12 SFP is removed and replaced with an OC-3 SFP, or when an OC-3 SFP is replaced with an OC-12 SFP. If the new interface bandwidth is compatible with the current interface and feature configuration, no user action is required. If the new bandwidth is *not* compatible, the system displays an error notification message, and continues to handle traffic on a best-efforts basis. User action is required in this situation to overcome the error condition.

The content of the error notification is based on the policy states listed in the Policy States.

Policy States

During the dynamic bandwidth modification process, if the modification is successful, the system does not display policy state information. However, if an error occurs, the system places the interface in one of these states and provides a policy-state error notification:

- Verification—This state indicates an incompatibility of the configured QoS policy with respect to the new interface bandwidth value. The system handles traffic on a best-efforts basis and some traffic drops can occur.
- Hardware programming—This state indicates a hardware programming failure caused by one of these conditions:
  - The modification of the interface default QoS resources encountered hardware update failures.
  - The modification of QoS resources (associated with the applied QoS policy) encountered hardware update failures.

With either of these failures, hardware programming could be in an inconsistent state, which can impact features such as policing, queueing and marking. Therefore, the system disables QoS policy in hardware for these error conditions.

- Reset—In response to user reconfiguration of QoS policy, the system attempts to apply the new policy but fails; the fallback to the previous QoS policy also fails.

If you receive notification of any of these policy states, you need to reconfigure the QoS policy to clear this condition.

Use the **show qos interface** and **show policy-map interface** commands to query the QoS state of the interface. The system displays the QoS policy status if the interface is in one of the error states (verification, hardware programming, or reset), but does not display it if the state is active.

Example

```
RP/0/RP1/CPU0:router# show qos interface gigabitEthernet 0/7/5/0 input
Wed Dec  8 09:00:29.092 UTC
NOTE:- Configured values are displayed within parentheses
Interface GigabitEthernet0/7/5/0  --   input policy
Total number of classes:       2
Interface Bandwidth:           1000000 kbps
QoS Policy Status:             Failed to verify QoS policy parameters < verification state
------------------------------------------------
Level1 class                              =   c1
Ingressq Queue ID                         =   53 (LP queue)
Queue belongs to Port                     =   10
Queue Max. BW.                            =   150016 kbps (150 mbits/sec)
Weight                                    =   10 (BWR not configured)
Guaranteed service rate                   =   150000 kbps
TailDrop Threshold                        =   1875000 bytes / 100 ms (default)
Policer not configured for this class
WRED not configured for this class

Level1 class                              =   class-default
Ingressq Queue ID                         =   32 (Port default LP queue)
Queue belongs to Port                     =   10
Queue Max. BW.                            =   1000064 kbps (default)
Weight                                    =   10 (BWR not configured)
Guaranteed service rate                   =   500000 kbps
TailDrop Threshold                        =   6250000 bytes / 100 ms (default)
Policer not configured for this class
WRED not configured for this class
```

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.

Statistics on the Clear Channel ATM SPAs

On egress policies, match statistics are obtained from the MSC, while drop statistics are obtained from the SPA. However, the rate calculation for SPA statistics is not supported. Rate calculation is displayed only on demand.

Inter-Class Policer Bucket Sharing

Based on different classification criteria, inter-class policer bucket sharing feature allows policer bucket sharing among different classes in a hierarchical QoS model, within the modular quality of service command line (MQC) construct, to achieve multirate policing of the same packet. In this feature, the classification of the incoming packet happens only once. However, the policer bucket is shared among classes; that is the same token bucket is used even though a match happens against different classes.

This feature includes following components:

Policer Bucket Shared

The policer bucket shared feature defines and shares a policer node entity. The defined policer bucket is shared among multiple classes.

Here is a sample configuration that defines and shares policer bucket instance *sp1*:

```
policy-map parent
        class long-distance
          police bucket shared sp1 rate 1 mbps
```

In this configuration, a policy-map for class long-distance traffic type is created to police at 1Mbps and the policer bucket is shared.

Policer Bucket Referred

The policer bucket referred feature refers a defined policer bucket instance. The reference to the policer bucket could be across policy level, a parent can refer a child policer, or vice versa, and one policer node can be referred by multiple classes across a policy map.

Here is a sample configuration that refers shared policer bucket instance *sp1*:

```
policy-map voip-child
        class long-distance-voip
         police bucket referred sp1
```

In this configuration, a policy-map for class long-distance-voip traffic type is created and the shared policer bucket *sp1* is referred.

Interface Support

Inter-class policer bucket sharing feature is supported only in the ingress direction. This section describes supported and non-supported interfaces for inter-class policer bucket sharing feature.

Table 8: Supported and non-supported interfaces

Supported Interfaces	1G/10G/100GE Physical interfaces
	L2 and L3 sub-interfaces
	Bundle ports
	Bundle sub-interfaces
Non-supported Interfaces	Bridge Virtual Interface (BVI)
	Satellite interfaces
	Pseudowire Headend (PWHE) interfaces

Note Inter-class policer bucket sharing feature is supported only on the ASR 9000 Enhanced Ethernet Line Card.

Classification Support for Ethernet-Services ACL

You can configure class of service (QoS) classification based on a match for partial MAC address (such as Organizationally Unique Identifier (OUI)) using the **match access-group ethernet-service** command. This command creates a match criteria for a class map based on the specified ethernet-service access control list (ACL) containing MAC addresses.

For example, you can create an ethernet-service ACL such as the following:

```
ethernet-services access-list acl1
 20 permit 2222.3300.0000 0000.00ff.ffff any
 30 permit 1111.2200.0000 0000.00ff.ffff any
 40 permit 1212.2300.0000 0000.00ff.ffff any
!
```

The ethernet-service ACL can be used in the class map to match the MAC addresses as follows:

```
class-map NID-123
  match access-group ethernet-service acl1
end-class-map
```

Note

- You can provide multiple values for the **ethernet-service** match type in a configuration; only the first value is considered for the match criteria. Subsequent values indicated in the match statement are ignored for classification.
- The capture statements in an ethernet-service ACL are ignored.
- An ethernet-service ACL should have only permit statements. If there are any deny statements, the policy is rejected.
- If you specify a value for the **Ether-Type** keyword using the **match access-group ethernet-service** command, the value is ignored.

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.
- 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** [**not**] **cos inner** [*inner-cos-value*] [*inner-cos-value0...inner-cos-value7*]
6. **match destination-address mac** *destination-mac-address*
7. **match source-address mac** *source-mac-address*
8. **match** [**not**] **discard-class** *discard-class-value* [*discard-class-value1 ... discard-class-value6*]
9. **match** [**not**] **dscp** [**ipv4** | **ipv6**] *dscp-value* [*dscp-value ... dscp-value*]
10. **match** [**not**] **mpls experimental topmost** *exp-value* [*exp-value1 ... exp-value7*]
11. **match** [**not**] **precedence** [**ipv4** | **ipv6**] *precedence-value* [*precedence-value1 ... precedence-value6*]

12. **match** [**not**] **protocol** *protocol-value* [*protocol-value1 ... protocol-value7*]
13. **match** [**not**] **qos-group** [*qos-group-value1 ... qos-group-value8*]
14. **match vlan [inner]** *vlanid* [vlanid1 ... vlanid7]
15. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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** [**not**] **cos inner** [*inner-cos-value*] [*inner-cos-value0...inner-cos-value7*] **Example:** `RP/0/RP0/CPU0:router match cos inner 7`	(Optional) Specifies an *inner-cos-value* in a class map to match packets. The inner-cos-value arguments are specified as an integer from 0 to 7.
Step 6	**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 7	**match source-address mac** *source-mac-address* **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match source-address mac 00.00.00`	(Optional) Configures the match criteria for a class map based on the specified source MAC address.

	Command or Action	Purpose
Step 8	**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 9	**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 10	**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 11	**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 12	**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 13	**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. • **match qos-group** command is supported only for an egress policy.

	Command or Action	Purpose
Step 14	**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 15	**commit**	

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 *Cisco IOS XR Modular Quality of Service Command Reference for the* Cisco CRS Router.

For conceptual information, see Traffic Policy Elements.

Restrictions

A maximum of 512 classes (including Level 1 and Level 2 hierarchical classes as well as implicit default classes) can be applied to one policy map.

For a hierarchical policy (with Level 1 and Level 2 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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.
Step 5	**commit**	

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 Cisco IOS XR Modular Quality of Service Command Referencefor the Cisco CRS Router.

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. **commit**
5. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface gigabitethernet 0/1/0/9`	Configures an interface and enters the interface configuration mode.
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	**commit**	
Step 5	**show policy-map interface** *type interface-path-id* [**input**	**output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface gigabitethernet 0/1/0/9`	(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
- SRP priority (egress only)
- Discard class (ingress only)
- ATM CLP

For a list of supported unconditional marking criteria action, see Table 2.

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

Restrictions

The CRS-MSC-140G does not support SRP priority.

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 cos** [**inner**] *cos-value*
9. **set mpls experimental** {**imposition** | **topmost**} *exp-value*
10. **set srp-priority** *priority-value*
11. **set discard-class** *discard-class-value*
12. **set atm-clp**
13. **exit**
14. **exit**
15. **interface** *type* interface-path-id
16. **service-policy** {**input** | **output**]} *policy-map*
17. **commit**
18. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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:**	Sets the precedence value in the IP header.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-pmap-c)# set precedence 1`	
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* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set qos-group 31`	Sets the QoS group identifiers on IPv4 or MPLS packets. 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 cos** [**inner**] *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. • For Layer 2 interfaces, the set cos command: Is rejected on ingress or egress policies on a main interface. Is accepted but ignored on ingress policies on a subinterface. Is supported on egress policies on a subinterface. • For Layer 3 interfaces, the set cos command: Is ignored on ingress policies on a main interface. Is rejected on ingress policies on a subinterface. Is supported on egress policies on main interfaces and subinterfaces.

	Command or Action	Purpose
Step 9	**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 10	**set srp-priority** *priority-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set srp-priority 3`	Sets the spatial reuse protocol (SRP) priority value of an outgoing packet. **Note** This command can be used only in service policies that are attached in the output direction of an interface.
Step 11	**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 12	**set atm-clp** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set atm-clp`	Sets the cell loss priority (CLP) bit.
Step 13	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 14	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 15	**interface** *type* interface-path-id **Example:** `RP/0/RP0/CPU0:router(config)# interface pos 0/2/0/0`	Configures an interface and enters the interface configuration mode.
Step 16	**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 17	**commit**	

	Command or Action	Purpose
Step 18	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface pos 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring 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 IP precedence and/or 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 Cisco IOS XR Routing Configuration Guide . To configure CEF, see Cisco IOS XR IP Address and Services Configuration Guide .
- Configure a BGP community list or access list.
- Define the route policy. Set the IP precedence and/or 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 or configure QPPB on the GRE Tunnel interfaces.
- Configure and enable a QoS Policy to use the above classification (IP precedence or 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 IP precedence or 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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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	**commit**	

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode.

	Command or Action	Purpose
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	**commit**	

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.

Restrictions

- The Cisco CRS Router only supports the setting of the QoS group ID using QPPB.

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface pos 0/2/0/0`	Enters interface configuration mode and associates one or more interfaces to the VRF.
Step 3	**ipv4 | ipv6 bgp policy propagation input**{**ip-precedence**|**qos-group**} {**destination**[*ip-precedence* {*destination*|*source*}] {**source**[*ip-precedence* {*destination*|*source*}] **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 bgp policy propagation input qos-group destination`	Enables QPPB on an interface

	Command or Action	Purpose
Step 4	**commit**	

Configuring QPPB on the GRE Tunnel Interfaces

This task applies QPPB to a GRE tunnel 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 tunnel-ip***number*
3. **ipv4 address** *ipv4-address subnet-mask*
4. **ipv6 address** *ipv6-prefix/prefix-length*
5. **ipv4 | ipv6 bgp policy propagation input**{**ip-precedence**|**qos-group**} {**destination**[*ip-precedence* {*destination*|*source*}] {**source**[*ip-precedence* {*destination*|*source*}]
6. **tunnel source** *type path-id*
7. **tunnel destination** *ip-address*
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface tunnel-ip***number* **Example:** `RP/0/RP0/CPU0:router(config)# interface tunnel-ip 4000`	Enters interface configuration mode and associates one or more interfaces to the VRF.
Step 3	**ipv4 address** *ipv4-address subnet-mask* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.1.1 255.255.255.0`	Assigns an IP address and subnet mask to the tunnel interface.
Step 4	**ipv6 address** *ipv6-prefix/prefix-length* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv6 address 100:1:1:1::1/64`	Specifies an IPv6 network assigned to the interface.
Step 5	**ipv4 | ipv6 bgp policy propagation input**{**ip-precedence**|**qos-group**} {**destination**[*ip-precedence* {*destination*|*source*}] {**source**[*ip-precedence* {*destination*|*source*}]	Enables QPPB on the GRE tunnel interface

	Command or Action	Purpose
	Example: `RP/0/RP0/CPU0:router(config-if)# ipv4 bgp policy propagation input qos-group destination`	
Step 6	**tunnel source** *type path-id* **Example:** `RP/0/RP0/CPU0:router(config-if)# tunnel source TenGigE0/2/0/1`	Specifies the source of the tunnel interface.
Step 7	**tunnel destination** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-if)# tunnel destination 100.100.100.20`	Defines the tunnel destination.
Step 8	**commit**	

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

The hierarchical ingress policing is supported at two levels:

- Parent level
- Child level

Restrictions

The Modular QoS command-line interface (MQC) provides hierarchical configuration, with the following limitations:

- The parent level consists of class defaults or class maps that match VLAN (in the case of nCmD policy).
- The child level consists of a flat policy that can be configured with the supported action or the **class-map** command.
- The parent policer value is used as a reference bandwidth for the child policy whenever required.

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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.
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	**commit**	

Matching the ATM CLP Bit

You can use the **match atm clp** command to enable packet matching on the basis of the ATM cell loss priority (CLP).

SUMMARY STEPS

1. **configure**
2. **class-map** *class-name*
3. **match atm clp**
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map** *class-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map c1`	Enters class map configuration mode. Creates a class map to be used for matching packets to the class whose name you specify.
Step 3	**match atm clp** **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match atm clp`	Sets the ATM cell loss priority (CLP) bit setting for all packets that match the specified traffic class. The ATM CLP bit value can be 0 or 1.
Step 4	**exit** **Example:** `RP/0/RP0/CPU0:router(config-cmap)# exit`	Returns the router to the global configuration mode.

Configuring Policer Bucket Sharing

Perform these tasks to enable policer bucket sharing in the ingress direction.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match precedence**[*number* | *name*]
4. **end-class-map**
5. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
6. **match precedence**[*number* | *name*]

7. **end-class-map**
8. **policy-map** [**type qos**] *policy-name*
9. **class** *class-name*
10. **police bucket shared** *policer instance name***rate***value*
11. **exit**
12. **class** *class-name*
13. **police bucket referred** *policer instance name*
14. **exit**
15. **end-policy-map**
16. **interface** *type interface-path-id*
17. **service-policy input** *policy-map*
18. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map class1`	Creates a class map to be used for matching packets to the class specified and enters the class map configuration mode. If you specify **match-any**, any one 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 match criteria.
Step 3	**match precedence**[*number* | *name*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match precedence 5`	Identifies IP precedence values as match criteria. The range is from 0 to 7. Reserved keywords can be specified, instead of numeric values.
Step 4	**end-class-map** **Example:** `RP/0/RP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 5	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map class2`	Creates a class map to be used for matching packets to the class specified and enters the class map configuration mode. If you specify **match-any**, any one 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 match criteria.

	Command or Action	Purpose
Step 6	**match precedence**[*number* | *name*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match precedence 1`	Identifies IP precedence values as match criteria. The range is from 0 to 7. Reserved keywords can be specified, instead of numeric values.
Step 7	**end-class-map** **Example:** `RP/0/RP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 8	**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 9	**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 10	**police bucket shared** *policer instance name***rate***value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# policer bucket shared policy1 rate 2Mbps`	Defines and shares a policer bucket. In this example, shared policer bucket *policy1* is created to rate limit traffic at 2Mbps.
Step 11	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 12	**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.
Step 13	**police bucket referred** *policer instance name* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# policer bucket referred policy1`	Refers a shared policer bucket. In this example, policer bucket *policy1* is referred.
Step 14	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.

	Command or Action	Purpose
Step 15	**end-policy-map** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# end-policy-map`	Ends the policy map configuration.
Step 16	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface gigabitethernet 100/0/0/0`	Configures an interface and enters the interface configuration mode.
Step 17	**service-policy input** *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-if)# service-policy input policy1`	Attaches a policy map to an input interface to be used as the service policy for that interface.
Step 18	**commit**	

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.

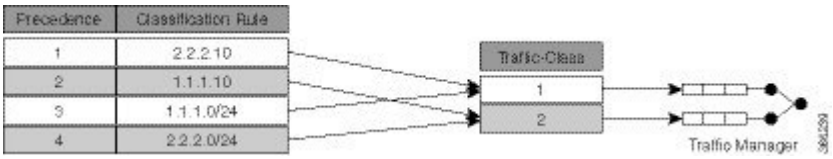
In order to overcome this limitation, the “Multiple QoS Policy Support” feature is introduced. This feature enables the users to apply aggregated actions to various classes of traffic and apply multiple QoS policies on an interface.

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—



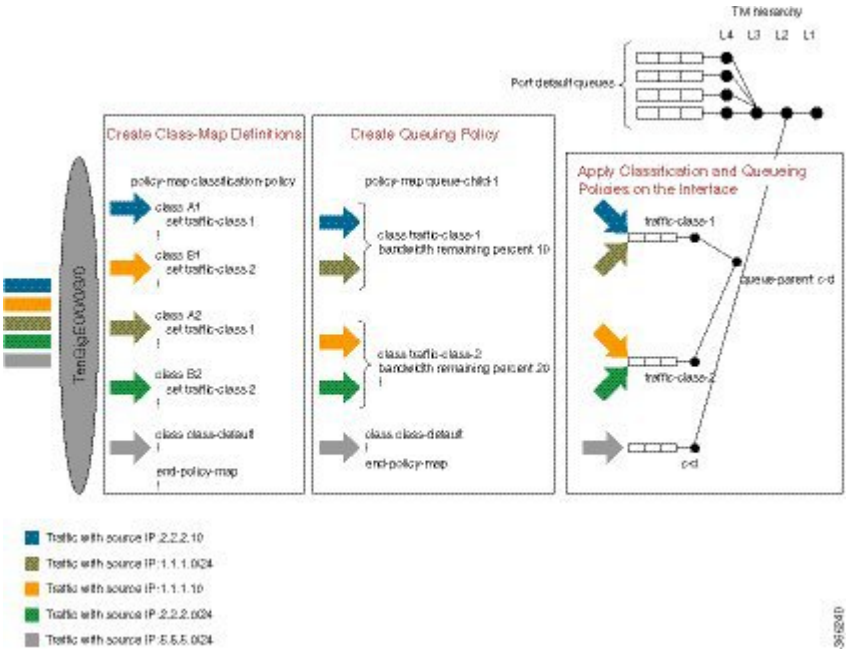
In this example, if the traffic packet matches 2.2.2.10 or 1.1.1.0/24, then the traffic packet is forwarded to the queue that is associated with traffic-class 1. And if the traffic packet matches 1.1.1.10 or 2.2.2.0/24, then the traffic packet is forwarded to the queue that is associated with traffic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

Policies Already Applied on the Interface			Policies that are yet to be Applied on the Interface			Accepted
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**	
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

Main/Bundle Interface			Sub/Bundle Sub Interface	Comments
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 aborted.

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 aborted 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 created 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 pos 0/1/0/0
  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.

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 POS interface 0/1/0/0. The IP precedence bit in the ToS byte is set to 1:

```
policy-map policy1
  class class1
    set precedence 1
!
interface pos 0/1/0/0
  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 GigabitEthernet interface 0/1/0/9. 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 GigabitEthernet 0/1/0/9
  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 POS interface 0/1/0/0. 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 POS interface 0/1/0/0
  service-policy input policy1
```

Note The unconditional **set discard-class** command is supported only on a Cisco CRS ingress policy.

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 10-Gigabit Ethernet interface, TenGigE0/1/0/0. 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 TenGigE0/1/0/0
interface TenGigE0/1/0/0.100
  service-policy output policy1
```

Note The **set cos** command is supported only on egress.

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 10-Gigabit Ethernet interface, TenGigE0/1/0/0. 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 TenGigE0/1/0/0
  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 10-Gigabit Ethernet interface, TenGigE0/1/0/0. 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 TenGigE0/1/0/0
  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
!
```

Applying QPPB on a GRE Tunnel Interface: Example

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

```
config
interface tunnel-ip 4000
ipv4 address 10.1.1.1
ipv4 bgp policy propagation input qos-group destination
tunnel source TenGigE0/2/0/1
tunnel destination 145.12.5.2
end
commit
!
```

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

```
config
interface tunnel-ip 3000
ipv6 address 1906:255::1/64
ipv6 bgp policy propagation input qos-group destination
tunnel source TenGigE0/2/0/1
tunnel destination 145.12.5.2
end
commit
!
```

Route Policy Configuration: Examples

BGP Community Configuration: Example

This configuration is an example of configuring route policy in a BGP community:

```
route-policy qppb-comm6200
  if  community matches-any (61100:10, 61200:20, 61300:30) then
    set qos-group 11
  elseif  community matches-any (62100:10, 62200:20, 62300:30) then
    set qos-group 12
  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-comm6200
  network 201.32.21.0/24
  network 201.37.5.0/24
 !
 neighbor 201.1.0.2
  remote-as 61100
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.2.0.2
  remote-as 61200
```

```
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.32.21.2
  remote-as 62100
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.32.22.2
  remote-as 62200
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !

interface GigabitEthernet 0/1/0/9
 service-policy output comm-p1-out
 ipv4 address 201.1.0.1 255.255.255.0
 ipv4 bgp policy propagation input qos-group destination
 negotiation auto
 !

interface GigabitEthernet 0/1/5/6
 service-policy input comm-p1-in
 ipv4 address 201.32.21.1 255.255.255.0
 ipv4 bgp policy propagation input qos-group source
 negotiation auto

!
```

Autonomous System Path Configuration: Example

This configuration is an example of configuring a route policy in an autonomous system path:

```
policy-map p11-in
 class class-qos10
  set discard-class 3
  shape average percent 20
 !
 class class-qos20
  set precedence flash-override
  shape average percent 30
 !
 class class-qos30
  set precedence critical
  shape average percent 40
 !
 class class-qos11
  set precedence priority
  shape average percent 10
 !
 class class-qos12
  set precedence immediate
  shape average percent 20
 !
 class class-qos13
  set precedence flash
  shape average percent 30
```

```
 !

policy-map p11-out
 class class-qos10
  set precedence priority
  police rate percent 20
  !
 !
 class class-qos20
  set precedence immediate
  police rate percent 30
  !
 !
 class class-qos30
  set precedence critical
  police rate percent 40
  !
 !
 class class-qos11
  set precedence immediate
  police rate percent 20
  !
 !
 class class-qos12
  set precedence flash
  police rate percent 30
  !
 !
 class class-qos13
  set precedence flash-override
  police rate percent 40
  !
 !
 class class-default
 !
 end-policy-map
!


route-policy qppb-as6000
  if as-path in (ios-regex '61100, 61200, 61300') then
    set qos-group 10
  elseif as-path in (ios-regex '62100, 62200, 62300') then
    set qos-group 20
  elseif as-path in (ios-regex '63100, 63200, 63300') then
    set qos-group 30
  elseif as-path neighbor-is '61101' or as-path neighbor-is '61201' then
    set qos-group 11
  elseif as-path neighbor-is '61102' or as-path neighbor-is '61202' then
    set qos-group 12
  elseif as-path neighbor-is '61103' or as-path neighbor-is '61203' then
    set qos-group 13
  else
    set qos-group 2
  endif
end-policy
!

router bgp 100
 bgp router-id 10.10.10.10
 address-family ipv4 unicast
  table-policy qppb-as6000
  network 201.32.21.0/24
  network 201.37.5.0/24
```

```
 !
 neighbor 201.1.0.2
  remote-as 61100
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.2.0.2
  remote-as 61200
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.32.21.2
  remote-as 62100
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.32.22.2
  remote-as 62200
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
interface GigabitEthernet 0/0/5/4
 service-policy output p11-out
 ipv4 address 201.1.0.1 255.255.255.0
 ipv4 bgp policy propagation input qos-group destination
 negotiation auto
 !

interface GigabitEthernet 0/1/5/6
 service-policy input p11-in
 ipv4 address 201.32.21.1 255.255.255.0
 ipv4 bgp policy propagation input qos-group source
 negotiation auto
 !
```

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 GigabitEthernet0/0/5/4
 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 GigabitEthernet0/1/5/6
 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 GigabitEthernet0/2/2/1
 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 GigabitEthernet0/2/2/1.2
 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 GigabitEthernet0/2/2/1.3
 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 GigabitEthernet0/2/2/1.4
 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 GigabitEthernet0/2/2/1.5
 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

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 GigabitEthernet0/1/5/7
 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 POS 0/6/0/1
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.

Configuring Inter Class Policer Bucket Sharing: Example

In this example, policer bucket *policy1* is defined and shared by class *class1*. The shared policer bucket *policy1* is referred by class *class2*.

```
configure
class-map class1
 match precedence 5
 !
class-map class2
 match precedence 1
 !
policy-map parent
  class class1
    police bucket shared policy1 rate 2 mbps
  class class2
    police bucket referred policy1
end-policy-map
!
```

Additional References

These sections provide references related to implementing packet classification.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco IOS XR Getting Started Guide for the* Cisco CRS Router
Master command reference	Cisco CRS Router Master Command Listing
QoS commands	Cisco IOS XR Modular Quality of Service Command Reference for the Cisco CRS Router
User groups and task IDs	*"Configuring AAA Services on Cisco IOS XR Software"* module of *Cisco IOS XR System Security Configuration Guide*
Initial system bootup and configuration	*Cisco IOS XR Getting Started Guide for the* Cisco CRS Router

Standards

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

MIBs

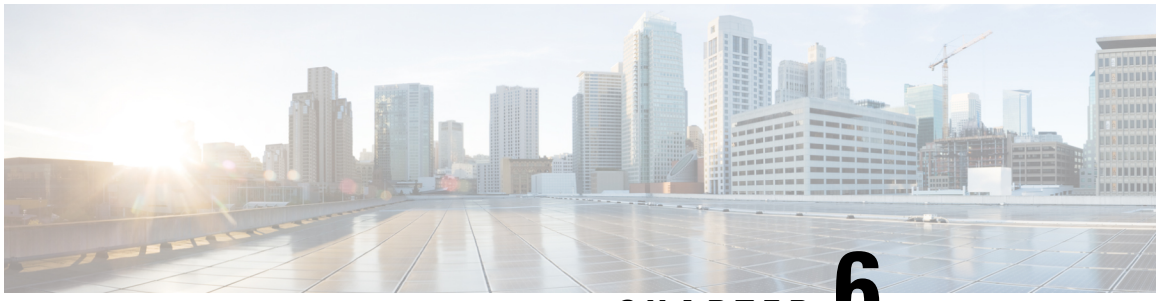
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

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

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 6

Modular QoS Deployment Scenarios

This module provides deployment scenarios use cases for specific QoS features or for QoS implementations of features that are described in other technology guides such as L2VPN or MPLS.

Feature History for QoS Deployment Scenarios on Cisco IOS XR Software

Release	Modification
Release 3.2.2	The QoS with SRP feature was introduced.
Release 3.8.0	The Qos on Multicast VPN feature was introduced. The VPLS QoS feature was introduced.
Release 4.0.0	The QinQ QoS feature was introduced(for Layer 2 only). The Hierarchical VPLS QoS feature was introduced.
Release 4.1.0	QoS on Pseudowire Headend (PWHE) Interfaces feature was introduced. Information regarding match criteria for inner VLAN was added to this document.
Release 5.3.2	VPLS(QoS) CRS fetaure was introduced on CRS-X.

Hierarchical VPLS QoS

Hierarchical VPLS (H-VPLS) is an extension of basic VPLS to provide scaling and operational benefits. H-VPLS partitions a network into several edge domains that are interconnected using an MPLS core. H-VPLS

provides a solution to deliver Ethernet multipoint services over MPLS. The use of Ethernet switches at the edge offers significant technical and economic advantages. H-VPLS also allows Ethernet point-to-point and multipoint Layer 2 VPN services, as well as Ethernet access to high-speed Internet and IP VPN services.

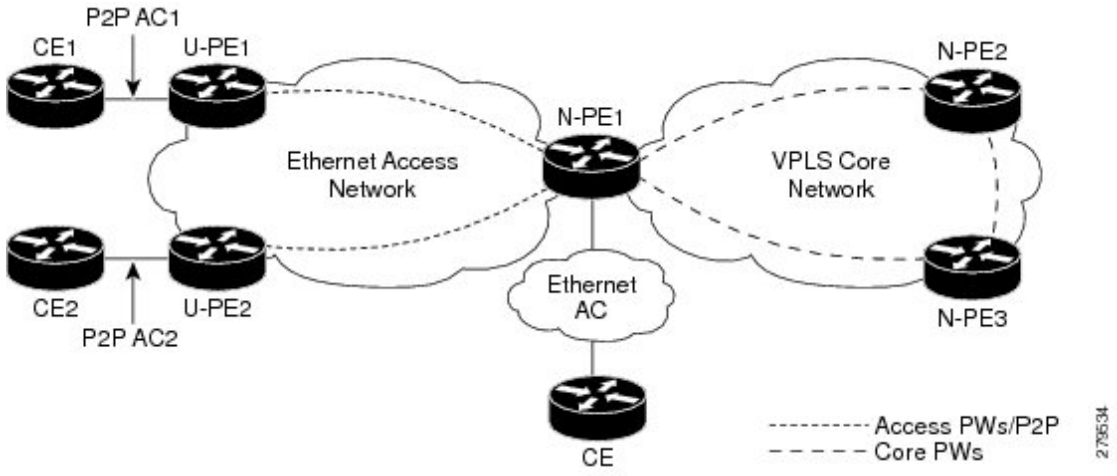
Note The Cisco CRS Series Modular Services Card 140G (CRS-MSC-140G) does not support Hierarchical VPLS QoS.

Hierarchical VPLS with Pseudowire Access

This figure shows the pseudowire (PW) access for H-VPLS. The edge domain can be an MPLS access network. In an H-VPLS configuration, the N-PE router forwards L2VPN packets from an Access PW to a Core PW and also forwards packets from a Core PW to an Access PW.

When forwarding packets, the N-PE router removes the virtual circuit (VC) label and also adds a VC label. The **set mpls experimental topmost** command, the **set mpls experimental imposition** command, or both of these commands can be applied to a QoS policy in a PW <-> PW configuration. Standard VPLS QoS actions, such as **match vlan inner**, **match cos inner**, and **set cos inner**, apply to H-VPLS QoS as well.

Figure 3: Pseudowire Access for H-VPLS



This table summarizes the actions taken with respect to the MPLS experimental value in a PW<-> PW configuration.

	Ingress/Egress	
Default	Ingress Egress	MPLS experimental value from the VC is copied to all imposed labels
set mpls experimental topmost	Ingress	MPLS experimental value from the QoS policy is copied to all imposed labels
set mpls experimental topmost	Egress	MPLS experimental value from the QoS policy is copied to the outermost labels

	Ingress/Egress	
set mpls experimental imposition	Ingress	MPLS experimental value from the QoS policy is copied to all imposed labels
set mpls experimental topmost and **set mpls experimental imposition**	Ingress	MPLS experimental value from the **set mpls experimental imposition** command in the QoS policy is copied to all imposed labels

Hierarchical VPLS QoS: Example

In this example, the N-PE router forwards L2VPN packets from an Access PW to a Core PW and also forwards packets from a Core PW to an Access PW. The example defines classes and specifies experimental values in the topmost MPLS label as match criteria for a class map to match the MPLS label. A QOS policy is applied to the ingress Core interfaces in the MPLS Access Network or the MPLS Core Network.

In this H-VPLS QoS configuration, the U-PE (customer) router and the N-PE (service provider) router have different QoS policies. The policy on the U-PE router specifies a high experimental value so traffic can receive more bandwidth and is sent as fast as possible to minimize jitter and delay. (topmost values of 4, 5, and 6.)

The N-PE router needs to balance traffic from various sources. The N-PE router remarks the packets by adding a label with a lower experimental value (imposition values of 1 and topmost values of 2).

```
class-map match-any exp1-top
 match mpls experimental topmost 4
 end-class-map
!
class-map match-any exp2-top
 match mpls experimental topmost 5
 end-class-map
!
class-map match-any exp3-top
 match mpls experimental topmost 6
 end-class-map

policy-map RX-Core
 class exp1-top
  set mpls experimental imposition 1
  shape average percent 20
 !
 class exp2-top
  set mpls experimental imposition 1
  set mpls experimental topmost 2
    shape average percent 20

 class exp3-top
  set mpls experimental topmost 2
  shape average percent 20

 class class-default
  set mpls experimental imposition 1
  shape average percent 20
 !
 end-policy-map
```

QinQ QoS

IEEE 802.1Q-in-Q VLAN Tagging expands the VLAN space by tagging the already tagged packets, thus producing a “double-tagged” frame. Double-tagging is useful for service providers, because it allows them to use VLANs internally while mixing traffic from clients that are already VLAN-tagged.

Q-in-Q allows service providers to use a single VLAN to transport most or all of a single customer's VLANs (inner VLAN tag) across their MAN/WAN backbone. The service provider adds an extra 802.1Q tag (outer VLAN tag) to customer traffic in the switches or routers at the edge of the service provider network. The outer VLAN tag assigns a unique VLAN ID to each customer in the service provider network. The outer VLAN tag keeps each customer's VLAN traffic segregated and private.

In the below figure, Customer ABC and Customer XYZ use the same VLAN ID of 27. However, these customers do not receive each other’s traffic, because the service provider tags each customer’s incoming frame with a unique outer VLAN tag. Customer ABC is assigned outer VLAN Tag 9 and Customer XYZ is assigned outer VLAN tag 10.

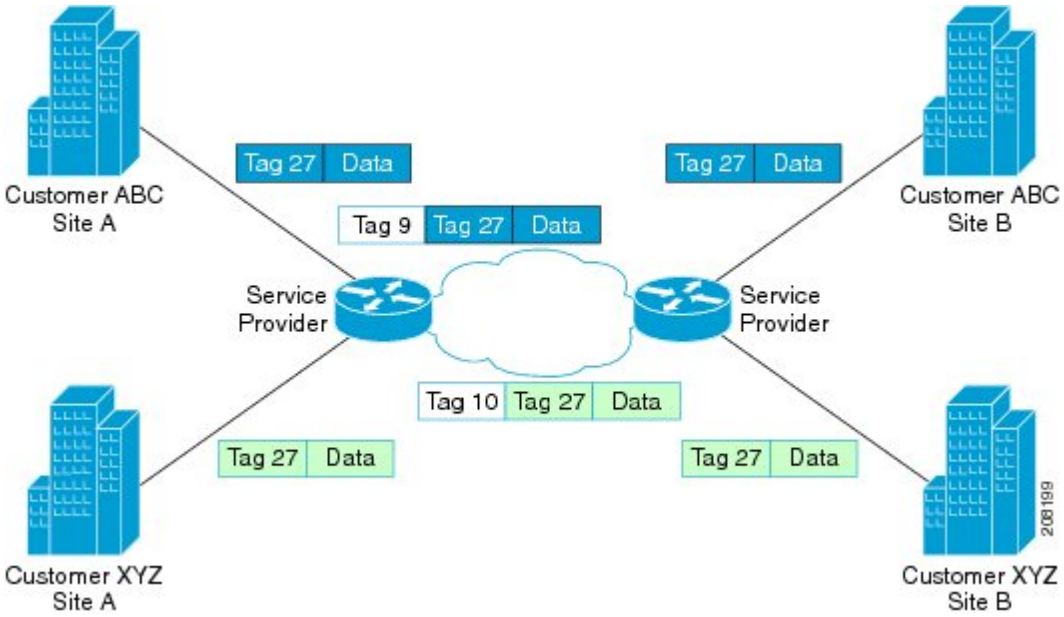
Classify and mark traffic for a specific customer, such as Customer XYZ, based on the inner (customer) VLAN tag using the **inner** keyword with the **match cos**, **match vlan**, and **set cos** commands.

The **match cos**, **match vlan**, and **set cos** commands classify and mark any traffic that matches the outer (service provider) VLAN tag; this traffic includes Customer XYZ traffic and customer ABC traffic.

The service provider removes the outer VLAN tag on traffic that exits the service provider network before it is sent to Customer ABC and Customer XYZ.

Note Some customers use three layers of VLAN tags. In this case, using the **inner** keyword with the **match cos** and **match vlan** commands classifies traffic traffic on the VLAN tag, which is right next to the outermost VLAN tag. Using the **inner** keyword with the **set cos** command marks CoS bits of the VLAN tag which is right next to the outermoset VLAN tag (what this document refers to as the inner (customer) VLAN tag).

Figure 4: QinQ Tagging



Note The CRS-MSC-140G does not support QinQ QoS.

Q-in-Any QoS

As in a QinQ configuration, a Q-in-Any interface receives “double-tagged” packets but only the outer VLAN ID is specified and inner VLAN ID can be any number in the range of 1-4094. Traffic on a Q-in-Any interface can be classified and marked based on the inner (customer) VLAN tag using the **inner** keyword with the **match cos**, **match vlan** and **set cos** commands.

Marking and Classifying Packets

To classify and mark packets in QinQ QoS or Q-in-Any QoS configurations based on the inner (customer) VLAN tag, use the **inner** keyword in the following commands:

	Supported On	On Layer 2	On Layer 3
match cos inner	ingress egress	Main interfaces Subinterfaces	Main interfaces

	Supported On	On Layer 2	On Layer 3
match vlan inner	ingress egress	Main interfaces Subinterfaces	Main interfaces
set cos inner	**egress: conditional and unconditional marking**	Main interfaces Subinterfaces	Main interfaces

Match on Inner CoS: Example

In this example, traffic with an outer VLAN ID of 2 and an inner VLAN ID of 3 enters the QinQ attachment circuit (AC). If traffic has a CoS value of 1, 3, or 5 based on the inner VLAN tag, it matches class ic.

```
policy-map p2
 class ic
  police rate percent 30
  !
  bandwidth remaining percent 40
 !
 class class-default
 !
 end-policy-map
!
class-map match-any ic
 match cos inner 1 3 5
 end-class-map
!
interface GigabitEthernet0/5/0/0.2 l2transport
 dot1q vlan 2 3
 service-policy input p2
!
```

Match Criteria for Inner VLANs: Example

This section provides guidance for configuring match criteria for inner VLANs on Q-in-Any and QinQ attachment circuits (ACs).

Match Criteria for Inner VLAN on Q-in-Any AC

In this example, all traffic with an outer VLAN ID of 1 enters the Q-in-Any AC, but the inner VLAN ID can be any value. However, only the traffic with an inner VLAN ID of 1, 2, or 3 matches the class provisioned in the policy-map (that is, class iv).

```
policy-map p1
 class iv
  shape average percent 30
  set qos-group 1
 !
 class class-default
 !
 end-policy-map
!
class-map match-any iv
 match vlan inner 1 2 3
 end-class-map
```

```
!
interface GigabitEthernet0/5/0/0.1 l2transport
 dot1q vlan 1 any
 service-policy input p1
!
```

Match Criteria for Inner VLAN on QinQ AC

The treatment of match criteria for inner VLAN on a QinQ attachment circuit (AC) does not function in the same manner as it does on a Q-in-Any AC (as described in the Match Criteria for Inner VLANs: Example). You should *not* attempt to configure match criteria for inner VLAN on QinQ ACs.

Explanation

During the configuration process, it is possible to configure an invalid match on the inner VLAN and commit the configuration. However, if you configure a match on inner VLAN with an ID different from the inner VLAN ID configured on the QinQ AC, the traffic on the AC might not be directed to the correct class.

Caution We recommend that you *not* configure match criteria for inner VLAN on QinQ ACs.

Example

In this example, the user has configured class iv1 with match on inner VLAN 1, but has configured an inner VLAN ID of 2 on the QinQ AC. Now, if traffic with an outer VLAN of 4 and inner VLAN of 2 enters the QinQ AC, the system matches the first class with inner VLAN ID (class iv1).

```
interface GigabitEthernet0/5/0/0.4 l2transport
 dot1q vlan 4 2
 service-policy input p1
!

policy-map p1
 class iv1
  shape average percent 30
  set qos-group 1
 !
 class iv2
  bandwidth remaining percent 30
 !
 class class-default
 !
 end-policy-map
!

class-map match-any iv1
 match vlan inner 1
 end-class-map
!

class-map match-any iv2
 match vlan inner 2
 end-class-map
!
```

Set Inner CoS: Example

In this example, traffic with outer VLAN ID of 3 and inner VLAN ID of 2 exits through the QinQ AC. If the traffic matches class qg1 or class qg2, it is marked with the inner CoS value specified in class qg1 or class qg2, respectively.

```
policy-map p3
 class qg1
  police rate percent 30 peak-rate percent 50
   conform-action set cos inner 1
   exceed-action set cos inner 2
   violate-action set cos inner 3
  !
 !
 class qg2
  set cos inner 4
 !
 class class-default
 !
 end-policy-map
!
class-map match-any qg1
 match qos-group 1
 end-class-map
!
class-map match-any qg2
 match qos-group 2
 end-class-map
!
interface GigabitEthernet0/5/0/0.3 l2transport
 dot1q vlan 3 2
 service-policy output p3
!
```

QoS on Multicast VPN

QoS on Multicast VPN: Example

Supporting QoS in an mVPN-enabled network requires conditional and unconditional marking of the DSCP or precedence bits onto the tunnel header. Unconditional marking marks the DSCP or precedence tunnel as a policy action. Conditional marking marks the DSCP or precedence values on the tunnel header as a policer action (conform, exceed, or violate).

Unconditional Marking

```
class-map c1
  match vlan 1-10

policy-map p1
 class c1
  set precedence tunnel 3
```

Conditional Marking

```
policy-map p2
 class c1

  police rate percent 50
  conform action set dscp tunnel af11
  exceed action set dscp tunnel af12
```

QoS with SRP

Spatial bandwidth reuse (SRP) is possible due to the packet destination-stripping property of SRP. Older technologies incorporate source stripping, where packets traverse the entire ring until they are removed by the source. Even if the source and destination nodes are next to each other on the ring, packets continue to traverse the entire ring until they return to the source to be removed. SRP provides more efficient use of available bandwidth by having the destination node remove the packet after it is read. This provides more bandwidth for other nodes on the SRP ring.

SRP rings consists of two counter rotating fibers, known as outer and inner rings, both concurrently used to carry data and control packets. SRP uses both explicit control packets and control information piggybacked inside data packets (control packets handle tasks such as keepalives, protection switching, and bandwidth control propagation). Control packets propagate in the opposite direction from the corresponding data packets, ensuring that the data takes the shortest path to its destination. The use of dual fiber-optic rings provides a high-level of packet survivability. In the event of a failed node or a fiber cut, data is transmitted over the alternate ring.

SRP rings are media independent and can operate over a variety of underlying technologies, including SONET/SDH, wavelength division multiplexing (WDM), and dark fiber. This ability to run SRP rings over any embedded fiber transport infrastructure provides a path to packet-optimized transport for high- bandwidth IP networks.

To distinguish between the two rings, one is referred to as the “inner” ring and the other as the “outer” ring. SRP operates by sending data packets in one direction (downstream) and sending the corresponding control packets in the opposite direction (upstream) on the other fiber. This allows SRP to use both fibers concurrently to maximize bandwidth for packet transport and to accelerate control signal propagation for adaptive bandwidth utilization, and for self-healing purposes.

Note The CRS-MSC-140G does not support QoS on SRP interfaces.

QoS with SRP: Example

This example shows how to configure two quality-of-service (QoS) classes. One is for voice traffic and is identified by an MPLS experimental bit value of 4; the second is control traffic that is identified by an IP precedence value of 6. Both classes of traffic are sent to the SRP high priority queue and are marked with high SRP priority (4 and 6).

```
Last configuration change at 04:56:06 UTC Tue Sep 06 2005 by lab
!
hostname router
```

```
class-map match-any ctrl
 match precedence internet
!
class-map match-any voice
 match mpls experimental topmost 4
!
policy-map srp-policy
 class voice
  police cir 2000000
  set cos 4
  priority
 !
 class ctrl
  priority
  set cos 6
 !
!
interface SRP0/7/0/0
 description "Connected to 3-nodes ring"
 service-policy output srp-policy
 ipv4 address 30.30.30.2 255.255.255.0
```

VPLS and VPWS QoS

To support QoS on a virtual private LAN service (VPLS)-enabled network, packets can be classified based on these VPLS-specific match criteria:

- Match on vpls broadcast
- Match on vpls known
- Match on vpls unknown
- Match on vpls multicast
-

Note VPLS-specific classification is performed only in the ingress direction. VPLS-specific and VPWS-specific classification are performed only in the ingress direction.

This example illustrates a typical VPLS topology with this configuration (in PE class c1):

```
class c1
  match vpls known
!
class c2
  match vpls unknown
!
class c3
  match vpls multicast
!
class c4
  match vpls broadcast
!
policy-map p1
  class c1
   set qos-group2
```

```
!
class c2
  set qos-group3
!
class c3
  set discard-class4
!
class c4
  set discard-class 5
!
```

The following figure illustrates a typical VPLS topology. The VPLS network is a mesh of pseudowires (PWs) interconnected to bridge domains in the routers. Each of the provider edge (PE) routers has a bridge domain. Each PW is a bridge port into the bridge domain. The customer edge (CE) connection into each PE router is an attachment circuit (AC) bridge port into the same bridge domain. QoS configuration commands are applied to the AC that connects to the CE router on the one end and the bridge domain of the PE router on the other.

VPLS and VPWS QoS: Example

In the VPLS-enabled network:

- If a unicast packet arrives on the ingress interface of the PE router with a known MAC address (which means the destination MAC address of the packet is found in the MAC forwarding table), it matches class c1.
- If a unicast packet arrives on the ingress interface of the PE router with an unknown MAC address (which means the destination MAC address of the packet is not found in the MAC forwarding table), it matches class c2.
- If a VPLS multicast packet arrives on the ingress interface of the PE router, it matches class c3.
- If a VLPS broadcast packet arrives on the ingress interface of the PE router, it matches class c4.

The packets that meet the VPLS-specific match criteria receive QoS treatment according to the policy actions defined in the policy.

Note Packets with unknown destination MAC address, multicast packets, and broadcast packets are flooded.

Note The CRS-MSC-140G does not support VPLS QoS.

Note The CRS-X supports VPLS QoS.

VPLS QoS: Example

In this example, the packets that meet the VPLS-specific match criteria receive QoS treatment according to the policy actions defined in the policy. Apply the policy to the VPLS AC interface.

```
class-map match-any c1
 match vpls known
 end-class-map
!

class-map match-any c2
 match vpls unknown
 end-class-map
!

class-map match-any c3
 match vpls broadcast
 end-class-map
!

class-map match-any c4
 match vpls multicast
 end-class-map
!

policy-map p2
 class c1
  set qos-group 3
  set mpls experimental imposition 4
  shape average percent 40
 !

 class c2
  bandwidth remaining percent 10
  set mpls experimental imposition 5
 !

 class c3
  police rate percent 30
  set mpls experimental imposition 6
  !

 class c4
  bandwidth remaining percent 10
  set mpls experimental imposition 7
  !

class class-default
 !
 end-policy-map
!

class-map match-any c1
 match vpls known
 end-class-map
!

class-map match-any c2
 match vpls unknown
 end-class-map
!

class-map match-any c3
 match vpls multicast
 end-class-map
!

policy-map p2
```

```
 class c1
  set qos-group 3
  set mpls experimental imposition 4
  shape average percent 40
 !

 class c2
  bandwidth remaining percent 10
  set mpls experimental imposition 5
 !

 class c3
  bandwidth remaining percent 10
  set mpls experimental imposition 7
  !

class class-default
 !
 end-policy-map
!
```

QoS on Pseudowire Headend

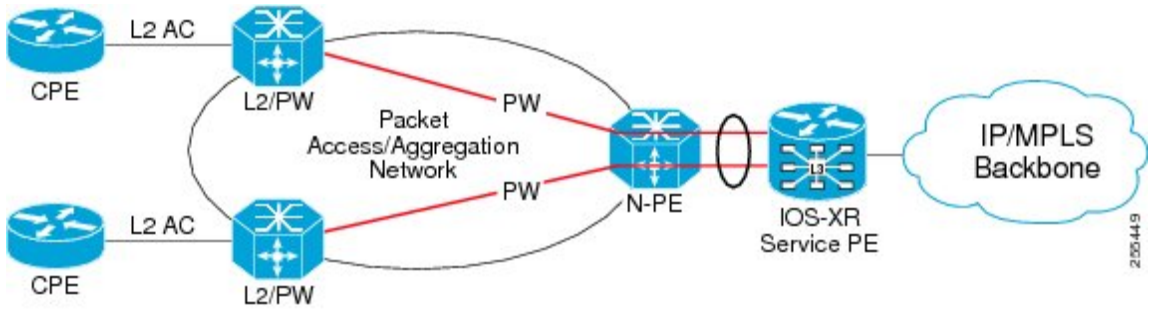
A pseudowire (PW) headend (PWHE) virtual interface originates as a PW on an access node (the Layer 2 PW feeder node) and terminates on a Layer 3 service instance, such as a VRF instance, on the service provider router (Cisco CRS Router). All ingress and egress QoS functions can be configured on the interface, including policing, shaping, queuing, and hierarchical policies.

Note The system supports only hierarchical QoS policy on the PWHE interface, a single default class in the parent policy, and policing or shaping on the parent. Nonhierarchical policies are not supported.

Note The Cisco CRS Series Modular Services Card 140G (CRS-MSC-140G) does not support QoS on PWHE interfaces.

This figure illustrates PWs originating on access node (L2/PW feeder node) and terminating on the service provider edge (Service PE). The composite Layer 2 attachment circuit (AC) plus PW segment forms a point-to-point virtual link that functions and behaves the same as traditional Layer 3 links. Although the PWHE interface is virtual, the traffic corresponding to that interface is sent and received over a single physical interface.

Figure 5: Example of Pseudowire Headend Virtual Interface



Note For information on creating the PWHE interface, see *Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router*.

The QoS policy on PWHE virtual interface can coexist with the QoS policy on its underlying physical interfaces. These two figures (one each for the ingress and egress directions) show the application of QoS for physical and PWHE traffic.

Figure 6: Ingress QoS Policy Application for Physical and PWHE Traffic

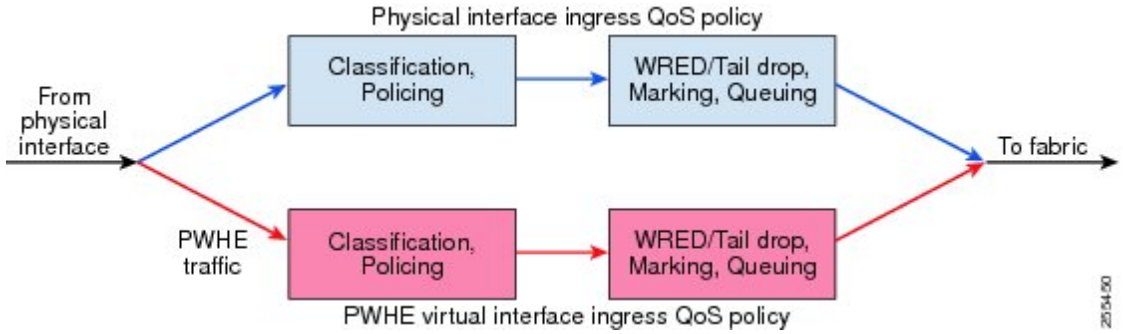
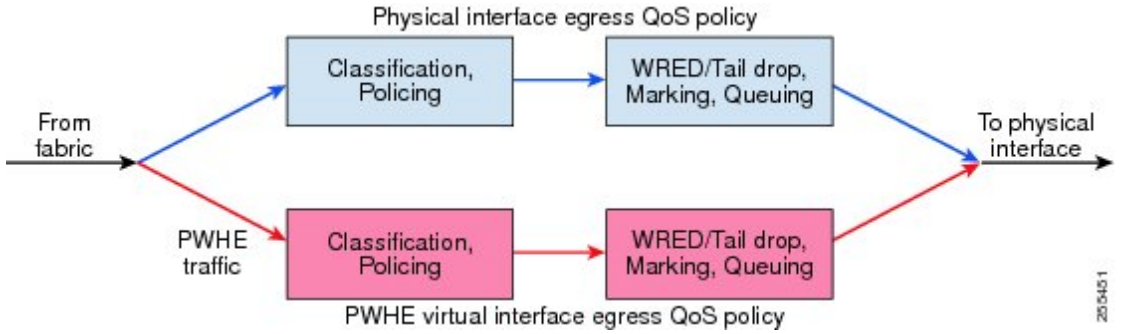


Figure 7: Egress QoS Policy Application for Physical and PWHE Traffic



The PWHE virtual interface is provisioned as a regular Layer 3 interface, and all QoS features that are supported on regular Layer 3 interfaces are supported on PWHE virtual interfaces. All ingress and egress QoS functions can be configured on the interface, including policing, shaping, queuing, and hierarchical policies. These details apply to header fields for traffic on PWHE interfaces:

- QoS classification and marking functions are based on the inner packet IP header and Layer 2 header fields. Marking of EXP bits for all imposed labels is supported in both ingress and egress QoS policies.
- If you configure Layer 2 overhead on the PWHE interface, the system adds this Layer 2 overhead to the packet length and then performs QoS functions.

The common MQC restrictions for QoS policies on Layer 3 interfaces apply to QoS policies for PWHE interfaces. In addition, these restrictions apply to QoS policies on PWHE interfaces:

- Configure the policy as a hierarchical policy with a single default class in the parent policy. Nonhierarchical policies are not supported.
- Configure either a police or shape command in the parent default class with an absolute rate. Percentage units are not allowed for police and shape commands in the parent default class.
- If you configure a police command without a shape command, you must configure a drop action within either the exceed action or the violate action.

QoS on Pseudowire Headend: Example

This example shows how to configure QoS policy on a PWHE interface. The **show** command at the end of the example displays the PWHE QoS statistics.

```
policy-map pw_parent_in
  class class-default
    police rate 100 mbps
    service-policy pw_child_in

policy-map pw_child_in
  class voip
    priority
    police rate 1 mbps
  class video
    police rate 10 mbps
    bandwidth remaining percent 20
  class data
    bandwidth remaining percent 70
  class class-default
    bandwidth remaining percent 10

policy-map pw_parent_out
  class class-default
    shape average 100 mbps
    service-policy pw_child_out

policy-map pw_child_out
  class voip
    priority
    police rate 1 mbps
  class video
    bandwidth 10 mbps
  class data
    bandwidth remaining percent 70
    random-detect discard-class 3 40 ms 50 ms
  class class-default
    random-detect discard-class 1 20 ms 30 ms

interface pw-ether 1
 service-policy input pw_parent_in
```

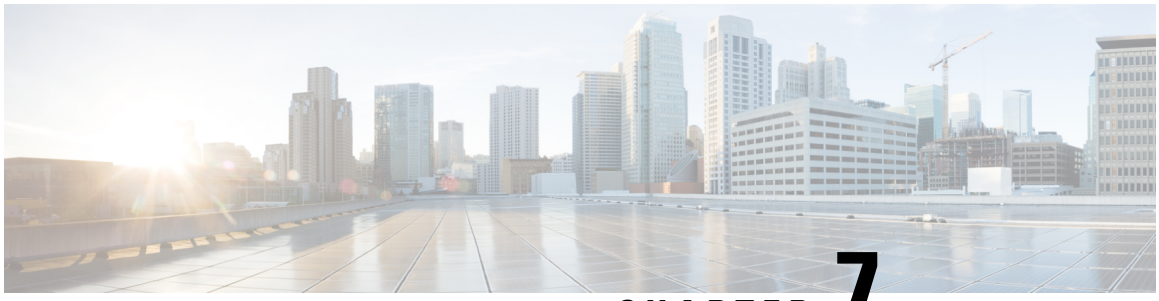
```
 service-policy output pw_parent_out

show policy-map interface pw-ether 1 member GigE 0/0/0/0 location 0/0/CPU0
```

Related Information

The information in this module focuses on the QoS implementation of features that are described in other technology guides. This table indicates the guides where you can find more information about these features.

Feature	Guide
Hierarchical VPLS QoS	*Cisco IOS XR MPLS Configuration Guide for the* Cisco CRS Router Cisco IOS XR *MPL*S Command Reference for the Cisco CRS Router
QinQ QoS	Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router Cisco IOS XR Virtual Private Network Command Reference for the Cisco CRS Router
QoS on Multicast VPN	Cisco IOS XR Multicast Configuration Guide for the Cisco CRS Router Cisco IOS XR Multicast Command Reference for the Cisco CRS Router
QoS with SRP	*Cisco IOS XR Interface and Hardware Component Configuration Guide* for the Cisco CRS Router *Cisco IOS XR Interface and Hardware Component Command Referenc*e for the Cisco CRS Router
VPLS QoS	*Cisco IOS XR MPLS Configuration Guide for the* Cisco CRS Router Cisco IOS XR *MPL*S Command Reference for the Cisco CRS Router



CHAPTER 7

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 3.3.0	The Fabric QoS Policies and Classes feature 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 fabricq queue selection mechanism is known as fabric QoS. Three queues are defined: a high-priority port for internal control traffic and classified high-priority traffic, a low-priority port for assured-forwarding

(AF) traffic, and a best effort port (BE) for low-priority traffic. Each port is assigned 1023 queues. 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.

By default, internal control traffic is placed in the high-priority queue. All other traffic is placed in the best-effort queue.

If an ingress QoS policy is configured classifying certain traffic as being priority, then the priority traffic is placed into the high-priority queue together with the internal control traffic. All other traffic is placed in the best-effort queue.

The user can configure a fabric QoS policy that defines the relative MDRR bandwidths associated with the AF and BE ports. 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.

A maximum of three classes can be specified within the policy, . A class known as *class-default* is automatically created and equates to the BE port or 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.

Within the fabric QoS policy, the only applicable actions are to assign **priority** to the priority class and **bandwidth remaining percent** to the AF and BE classes. The bandwidth remaining percent for the BE class does not need to be specified. It receives all remaining capacity after the appropriate weight has been allocated to the AF class. The user may wish to specifically enter configuration values for the default class just for clarity.

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

Be careful when applying ingress QoS policies when they must interact with a fabric QoS policy. The fabric QoS policy overrides any traffic classification conducted by the ingress policy when determining which traffic should be placed in the priority, AF, or BE queues within the fabricq ASIC. In addition, the fabric QoS policy is used to determine which traffic is placed in the priority queue within the ingressq ASIC fabric queues and the switch fabric ASICs (S2 and S3 stages) rather than any **priority** marking set by the ingress QoS policy.

The **priority** marking performed by the ingress QoS policy is still used when determining packet scheduling in the shape queues within the ingressq ASIC.

For example, if an ingress QoS policy were to classify and mark particular traffic types as being **priority** and a fabric QoS policy were to be applied either marking alternative traffic as being **priority** or not setting **priority** at all, then the ingress policy **priority** statement is effectively ignored in the fabricq, the S2 ASICs, and the S3 ASICs. Use caution to ensure that there is no conflict between the ingress QoS policy and the fabric QoS policy.

A very simplistic illustration would be if an ingress QoS policy uses a class-map to exclude MPLS experimental 3 from the **priority** class, but the fabric QoS policy places MPLS experimental 3 traffic in the **priority** class. In this case, the MPLS experimental 3 traffic is placed in the high-priority S2 and S3 queues and the fabricq high-priority port or queues.

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.

As noted above, fabric QoS is constrained to a subset of the possible **match** criteria that can be used in its class maps. If the ingress QoS policy were to set a **qos-group** marking for all traffic that should be placed in the priority queue and another **qos-group** marking for all traffic to be placed in the AF class, then if the fabric QoS policy class maps matches on the **qos-group** values, the policy is honored end to end. This approach enables multiple ingress QoS policies to interact in the expected manner with a fabric QoS policy.

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. This has an impact on the forwarding capacity of the line card or PLIM, reducing the performance to about 62.5 Mpps.

You can choose not to apply a specific fabric QoS policy, giving the ingress QoS policy the decision on which traffic is placed in the high-priority queues but removing the ability to differentiate between AF and BE classes in the fabricq ASIC.

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

This configuration task explains how to configure a fabric QoS policy.

Restrictions:

- A maximum of three classes can be specified within the service policy. A class known as default is automatically created and equates to the BE port or queues. The name of this class cannot be modified. You can rename the classes that equate to the priority and AF ports or queues.
- Within the fabric QoS policy, the only applicable actions are to assign:

- Priority to the priority class
- Percent of bandwidth remaining to the AF and BE classes.

- The percent of bandwidth remaining for the BE class does not need to be specified.
- For fabric QoS, the supported class-map match types are:
  - precedence
  - dscp
  - qos-group
  - discard-class
  - mpls experimental topmost
  - cos

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. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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.
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. Fabric QoS is supported for IPv4 only.

	Command or Action	Purpose
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/RP0/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 global configuration 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	**commit**	

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:

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
```

Note In the policy-map input-qos command, the **priority** keyword under class voip is for reference rather than for configuration.

Additional References

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

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco IOS XR Getting Started Guide for the* Cisco CRS Router
Master command reference	Cisco CRS Router Master Command Listing
QoS commands	Cisco IOS XR Modular Quality of Service Command Reference for the Cisco CRS Router
User groups and task IDs	*"Configuring AAA Services on Cisco IOS XR Software"* module of *Cisco IOS XR System Security Configuration Guide*

Standards

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

MIBs

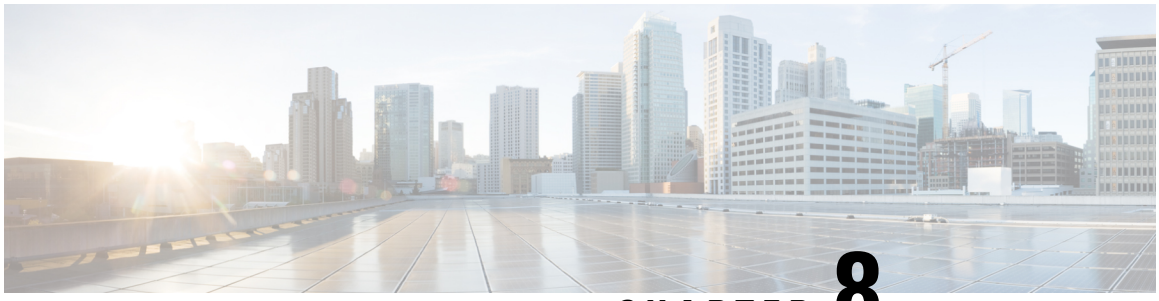
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

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

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 8

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 on Cisco IOS XR Software

Release	Modification
Release 3.8.0	The QoS on Link Bundles 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. For example, you can upgrade from an OC-48c PLIM modular services card to an OC-192 PLIM modular services card without interrupting traffic.

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.

Cisco IOS XR software supports these methods of forming bundles of Ethernet and POS 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 or POS Channel—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. (EtherChannel applies to Ethernet interfaces and 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.
- A policer rate limit greater than 16 GB per class cannot be configured.
- Layer 2 QoS is not supported for bundles.

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.
- For ingress QoS policies, a single set of queues defined by the policy are assigned on each linecard hosting a member link. As a result, ingress QoS policy acts on the aggregate bandwidth provided by the interfaces on that linecard.

QoS for POS link bundling

For POS link bundles, percentage-based bandwidth is supported for policers and output queues. Time-based queue limit is supported for output queues.

Input QoS Policy setup

For input QoS, queuing is not supported and thus bandwidth is used for policer only. As a member link is added or removed from a bundle with input QoS configured, the aggregate bundle bandwidth for that affected line card will change. One input QoS policy instance is assigned for each SIP 700 line card that is part of the POS link bundle.

Output QoS Policy setup

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 POS link bundles.

Example 2 : Output QoS policy supported on POS 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.

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 on parent max-rate / interface bandwidth **PPS / Absolute rate**: Divide the rate based on bandwidth ratio **Burst-size**: If `time-units`, no change. Convert `time-units` to bytes based on `service-rate` If configured in absolute value, divide the absolute value based on bandwidth ratio
Wred / Queue-Limit Threshold	**Time-Units**: No Change. Use the `service-rate` to convert to bytes **Absolute value**: Divide the absolute value based on bandwidth ratio

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.

	Command or Action	Purpose

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

Related Topic	Document Title
Initial system bootup and configuration	*Cisco IOS XR Getting Started Guide for the* Cisco CRS Router
Link Bundling	"Configuring Link Bundling on Cisco IOS XR Software" module of *Cisco IOS XR Interface and Hardware Component Configuration Guide for the the* Cisco CRS Router
Master command reference	Cisco CRS Router Master Command Listing
QoS commands	Cisco IOS XR Modular Quality of Service Command Reference for the Cisco CRS Router
User groups and task IDs	*"Configuring AAA Services on Cisco IOS XR Software"* module of *Cisco IOS XR System Security Configuration Guide*

Standards

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

MIBs

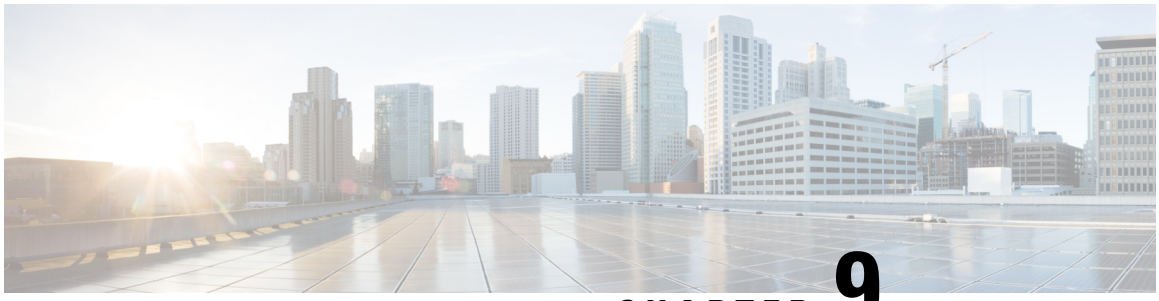
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

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

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 9

Configuring QoS on the Satellite System

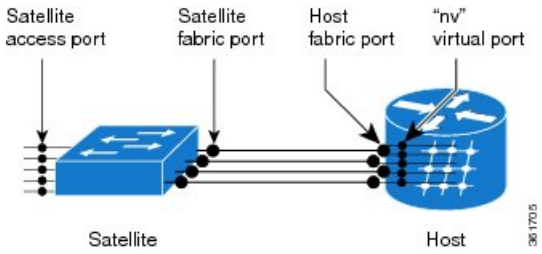
QoS Offload on Satellite

The Satellite System enables you to configure a topology in which one or more satellite switches complement one or more CRS Router, to collectively deploy a single virtual switching system. In this system, the satellite switches act under the management control of the routers. The connections between the CRS Router and the satellite switches are called the Inter-chassis link (ICL), which is established using standard Ethernet interfaces.

The ICL link between the and the satellite gets oversubscribed by the access interfaces on the satellite box. This is because the QoS policies applied on the satellite interfaces are programmed on the CRS Router Line card locally. Therefore, the flow of traffic on the ICL from the satellite switch is not controlled. This leads a loss of high-priority traffic due to congestion on the ICL.

This figure shows the ports where the QoS policies may be applied.

Figure 8: Satellite and Host connection



Benefits of QoS Offload

The QoS offload feature protects the control packets when Satellite fabric links (SFL) is congested. The offloading of QoS policies helps to drop excess traffic at the ingress direction (or access ports) and prioritize the protocol control traffic at the egress direction (or SFL).

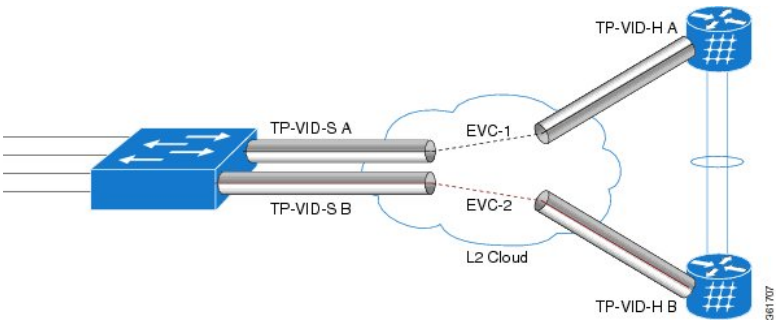
QoS Offload on Different Topologies

The QoS Offload feature is supported on these satellite topologies:

L2 Fabric Architecture

In the L2 Fabric architecture, a satellite is connected to one or more hosts through one or more ethernet virtual circuit (EVC) in the Layer 2 Fabric network. An EVC is identified by two transport VLAN IDs, TPVID- S and TP-VID-H. TP-VID-S is the satellite side transport VLAN ID and TP-VID-H is the host side transport VLAN ID.

Figure 9: L2 Fabric Architecture



Here is the sample configuration that displays the QoS policy being offloaded in the L2 Fabric architecture.

```
interface TenGigabitEthernet 0/1/0/0
nv satellite-fabric-link satellite 100
 remote-ports GigabitEthernet 0/0/0-5
service-policy output <policy-map name>
encapsulation dot1q 20
!
!
```

In this configuration, the ICL link is created on the VLAN and EVC is provisioned as an interface in the host-side. The nV satellite fabric interface is created under this interface. The service-policy is configured in the nv mode and the QoS policy is offloaded on the ICL link where the VLAN is connected to the Host.

Restrictions

- Policy on sub-interfaces is not supported.
- Classification based on access group is not supported.

QoS Offload Scenarios

This section describes the various QoS offload scenarios on different interfaces on different satellite topologies.

Service-policy on Access Port over Physical Interface

In this scenario, the QoS policy is configured on the access port or in the ingress direction over the physical interface of the satellite.

In this example, the policy_A service-policy is directly applied on the Ethernet interface on the satellite. Thus, policy_A stays on the CRS Router and there is no offloading in this example.

```
interface gigabitEthernet 100/0/0/0
 service-policy input/output policy_A
 !
 !
```

In this example, the policy_B service-policy is configured under the nv mode, and the QoS policy is completely offloaded to the Satellite.

```
interface gigabitEthernet100/0/0/0
 nv
  service-policy input policy_B
```

Figure 10: Service-policy on Access Port over Physical Interface (Single Host)

Service-policy on Access Port over Bundle Interface

In this scenario, the QoS policy is configured on the bundle access port or in the ingress direction over the bundle interface of the satellite.

In this example, similar to one for the physical interface, the QoS service-policy is configured under the nv mode and the policy-map is offloaded on the satellite bundle-ether interface.

```
interface GigabitEthernet 100/0/0/1
 bundle-id 1
!
interface GigabitEthernet 100/0/0/1
 bundle-id 1
!
interface bundle-ether 1
 nv
 service-policy input<Policy-map name>
!
!
```

Figure 11: Service-policy on Access Port over Bundle Interface

Service-policy on SFLs over Physical Interface (L2 Fabric)

In this scenario, the QoS policy is configured on the VLAN interface or in the egress direction over the physical interface of the satellite.

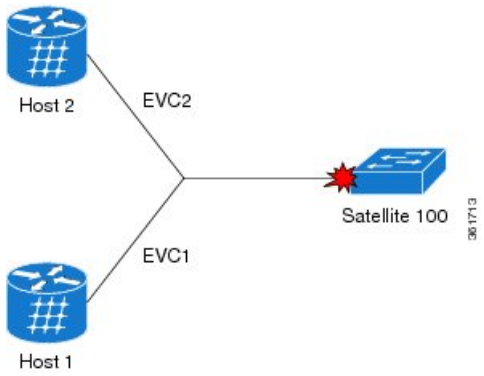
In this example, the QoS policy-map is applied to the host-facing bundle SFL under the nv mode and the QoS policy is offloaded to the satellite 100 (9000v) through one of the EVC of the L2 Fabric network.

```
interface TenGigabitEthernet 0/1/0/0
 nv satellite-fabric-link satellite 100
 remote-ports GigabitEthernet 0/0/0-5
  service-policy output <Policy-map-name>
!
```

In this example, the QoS policy-map is applied to the host-facing bundle SFL under the nv mode and the QoS policy is offloaded to the satellite 100 (901) through one of the EVC of the L2 Fabric network.

```
interface TenGigabitEthernet 0/1/0/0
 nv satellite-fabric-link satellite 100
 remote-ports GigabitEthernet 0/0/0-5
  service-policy output <Policy-map-name>
!
```

Figure 12: Service-policy on SFLs over Physical Interface (L2 Fabric)



Supported Platform-Specific Information for QoS Offload

This section describes the supported capability matrix, various supported classification combinations, and the supported scalability matrix for 9000v and ASR 901 satellites.

Supported Capability Matrix

The example shows how to configure the access policy at the ingress of the access interface and ICL policy at the egress of the ICL interface.

```
interface GigabitEthernet200/0/0/10
nv
  service-policy input access
!
!

interface Bundle-Ether200
ipv4 point-to-point
ipv4 unnumbered Loopback3000
nv
  satellite-fabric-link satellite 200
   service-policy output icl
   redundancy
    iccp-group 10
   !
```

```
   remote-ports GigabitEthernet 0/0/0-43
  !
!
!
policy-map icl
class icl1
  bandwidth remaining percent 5
!
class icl2
  bandwidth remaining percent 4
!
class icl3
  priority level 1
!
class icl4
  bandwidth remaining percent 1
!
class class-default
  bandwidth remaining percent 1
!
end-policy-map
!

RP/0/RSP1/CPU0:vkg3(config)#show running-config policy-map access
policy-map access
class access1
  set qos-group 1
!
class access2
  set qos-group 2
!
class access3
  set qos-group 3
!
class access4
  set qos-group 4
!
class class-default
!
end-policy-map
!

class-map match-any access1
match cos 1
match precedence 1
end-class-map
!
class-map match-any access2
match cos 2
match precedence 2
end-class-map
!
class-map match-any access3
match cos 3
match precedence 3
end-class-map
!
class-map match-any access4
match cos 4
match precedence 4
end-class-map
!
class-map match-any icl1
match qos-group 1
```

```
end-class-map
!
class-map match-any icl2
match qos-group 2
end-class-map
!
class-map match-any icl3
match qos-group 3
end-class-map
!
class-map match-any icl4
match qos-group 4
end-class-map
!
```

The example shows the service policy status of the ICL interface.

```
RP/0/RSP1/CPU0:vkg3(config)#do show qos status interface bundle-ether 200 nv
Bundle-Ether200 direction input: Service Policy not installed

Bundle-Ether200  Satellite: 200 output: icl

     Last Operation Attempted :  IN-PLACE MODIFY
     Status                   :  ACTIVE
```

The example shows the service policy status of the access interface.

```
RP/0/RSP1/CPU0:vkg3(config)#do show qos status interface gigabitEthernet 200/0/0/10 nv

GigabitEthernet200/0/0/10  Satellite: 200 input: access

     Last Operation Attempted :  IN-PLACE MODIFY
     Status                   :  ACTIVE
GigabitEthernet200/0/0/10 direction output: Service Policy not installed
```

Feature	Support on 9000v Platform	Range	Restrictions
Classification			
Ingress			
COS	Yes	0-7	The cos classification is done on the outer vlan tag. **Note** The cos classification based on match-rule is not applicable for untagged packets on the ingress direction.

Feature	Support on 9000v Platform	Range	Restrictions
IP DSCP	Yes	0-63	IP DSCP is supported for untagged, single-tagged, double-tagged, and mac-in-mac packets on the ingress direction, from the access-side. IP DSCP is supported for IPv4 and IPv6.
IP PREC	Yes	0-7	IP PREC is supported for untagged, single-tagged, double-tagged, and mac-in-mac packets on the ingress direction, from the access-side. IP PR is supported only for IPv4.
MPLS EXPERIMENTAL TOPMOST	Yes	0-7	The mpls experimental topmost feature is supported only for the untagged packets on the ingress direction, from the access-side.
VLAN	Yes	1-4096	The vlan classification is done on the outer vlan tag based on the policies and the cos value applied on the outer vlan tag. **Note** The vlan classification based on outer vlan tag is not applicable for untagged packets on the ingress direction.
Egress			

Feature	Support on 9000v Platform	Range	Restrictions
QOS-GROUP	Yes	1-5	A class-map with multiple "match qos-group" statements is not supported. **Note** • qos-group 0 corresponds to class-default, hence, it cannot be configured. • qos-group 6 and qos-group 7 are reserved, and hence, it cannot be configured.
COS	Yes	0-7	The cos classification is done on the outer vlan tag. **Note** The cos classification based on match-rule is not applicable for untagged packets on the ingress direction.
IP DSCP	Yes	0-63	IP DSCP is supported for untagged, single-tagged, double-tagged, and mac-in-mac packets on the ingress direction, from the access-side. IP DSCP is supported only for IPv4.
IP PREC	Yes	0-7	IP PREC is supported for untagged, single-tagged, double-tagged, and mac-in-mac packets on the ingress direction, from the access-side. IP PR is supported only for IPv4.

Feature	Support on 9000v Platform	Range	Restrictions
MPLS EXPERIMENTAL TOPMOST	Yes	0-7	The mpls experimental topmost feature is supported only for the untagged packets on the ingress direction, from the access-side.
VLAN	Yes	1-4096	The vlan classification is done on the outer vlan tag based on the policies and the cos value applied on the outer vlan tag. **Note** The vlan classification based on outer vlan tag is not applicable for untagged packets on the ingress direction.
Marking			
Ingress			
COS	Yes	0-7	The cos marking is done on the vlan tag that is added by the satellite on the direction towards host.
IP DSCP	Yes	0-63	IP DSCP is supported for untagged, single-tagged, double-tagged, and mac-in-mac packets on the ingress direction, from the access-side.
MPLS EXPERIMENTAL IMPOSITION	No	0-7	—
IP PREC	Yes	0-7	IP PREC is supported for untagged, single-tagged, double-tagged, and mac-in-mac packets on the ingress direction, from the access-side.

Feature	Support on 9000v Platform	Range	Restrictions
QOS-GROUP	Yes	0-5	The qos-group marking feature is only used to redirect packets to a particular queue. The set qos-group 0 on ingress policy is necessary to send the packets to queue 0 on ICL. **Note** If the QoS classification rule at the ICL interface in the egress and ingress direction matches, then the packets are directed to the configured group, else the packets are directed to the class-default group.
Queuing			
Egress			
Bandwidth Percent	Yes	—	For a 9000v satellite, bandwidth value cannot be configured under qos-group 3. A combination of bandwidth types cannot be configured. For example, the bandwidth command can be configured either with kbps, or remaining percent, or remaining ratio, but not with a combination of all.
Bandwidth Remaining Percent	Yes	—	
Bandwidth Remaining Ratio	Yes	—	

Feature	Support on 9000v Platform	Range	Restrictions
Priority	Yes	—	When a priority level is configured at the host, it by default gets configured to priority percent 95 85 on the satellite. The priority action cannot be combined with other queuing actions. Only one class-map with a priority action can be configured. On 9000v satellites, the priority action is only supported under qos-group 3.
Priority Percent	Yes	—	
Random Detect Discard-class-based	No	Discard-class: 0-2 Thresholds: 1-8192000	—
Shape Average	Yes	8000- 10000000000	On 9000v satellites, the shape average command cannot be configured under qos-group 3.
HQOS	No	—	—
Rate Limiting (Only Ingress)			

Feature	Support on 9000v Platform	Range	Restrictions
1R2C	Yes	CIR/PIR: 8000-10000000000 Burst bytes: 1000-256000000 Burst ms:1-2000	The bytes can be configured in milliseconds (ms) only if CIR is in percent. **Note** • CIR stands for Committed Information Rate and PIR stands for Peak Information Rate. • Transmit and marking actions are not supported together.
2R3C	Yes		If the exceed-action command is configured, then violate-action is copied from exceed-action, by default. If the exceed-action is not configured, then violate-action and exceed-action are dropped. **Note** • 2R3C statistics are supported only for conform & violate actions. • Transmit and marking actions are not supported together.

Supported Classification Combination

These are the allowed classification combination in CRS Router:

- COS + IP DSCP
- IP DSCP +VLAN
- COS + VLAN

- IP DSCP + IP PREC

Note The IP DSCP + IP PREC combination is not supported for 9000v.

The table lists the allowed classification combinations in 9000v:

Match-all class map	DSCP + PREC + COS
	PREC + DSCP + VLAN
Match-any class map	VLAN + COS + PREC + DSCP
	DSCP + VLAN + COS
	DSCP + PREC + COS
	VLAN + COS + PREC

Note For NCS 5000 Series Satellite, COS+DSCP match is the only supported classification combination on ingress. For Egress, policies can only match on qos-group (1 per class-map). For Egress offload policies on NCS 5000 Series Satellite, it is mandatory to configure eight class-maps including class-default for eight queues, even if all the class maps are not in use.

Supported Scalability Matrix for 9000v

Class-map with options	Number of Field Programmable (FP) entries needed per policy-map(max 8 classes)	Max policy-maps supported
cos (0-7)	7 + 1 (class default)	2304/8 = 288
ip dscp (0-63)	7 + 1	2304/8 = 288
ip precedence (0-7)	7 + 1	2304/8 = 288
vlan (1-4094)	7 + 1	2304/8 = 288
match-any or match-all with single argument		
cos + dscp cos+ prec cos + vlan dscp + vlan prec + vlan	2 *7 + 1 (class-default) = 15	2304/15 = 153.6

Class-map with options	Number of Field Programmable (FP) entries needed per policy-map(max 8 classes)	Max policy-maps supported
match-any with maximum arguments to the match parameters		
cos (max 4)+ ip precedence (max 4)	8 * 7 + 1 (class-default) = 57	2304/57 = 40.4
cos (4) + ip dscp (8)	12 * 7 + 1 (class-default)= 85	2304/85 = 27.1
cos (4) + vlan (30)	34 * 7 + 1 = 239	2304/239 = 9.6
vlan (30) + ip prec (4)	34 *7 + 1 = 239	2304/239 = 9.6
vlan (30)+ip dscp (8)	38*7 + 1 =267	2304/267 = 8.6
match-all with maximum arguments		
cos (4) + ip dscp (8)	32 *7 + 1=225	2304/225 = 10.2
cos (4) + vlan (30)	120 *7+ 1=841	2304/841 = 2.7
vlan (30) + ip prec (4)	120*7+1=841	2304/841 = 2.7
cos (4) + ip prec (4)	16 *7 +1= 113	2304/113 = 20.3
vlan (30) + ip dscp (8)	240 *7 + 1 = 1681	2304/1681 =1.3

QoS Offload Configuration Overview

Three steps to configure QoS Offload are:

1. Create a class-map of the type ‘qos’.
2. Create a policy-map of the type ‘qos’ using the above configured class map.
3. Bind QoS policy to Satellite interfaces such as physical access, bundle access, physical ICL, and bundle ICL.

To modify a QoS Offload configuration:

1. Modify the class-map or policy-map without unbinding the policy-map from the applied interface.

Note QoS Offload configuration with **police rate** in **pps** unit is not supported.

Sample QoS Offload Configuration

```
class-map match-any my_class
 match dscp 10
 end-class-map
!
policy-map my_policy
 class my_class
 police rate percent 30
!
end-policy-map
!
interface GigabitEthernet100/0/0/9
 ipv4 address 10.1.1.1 255.255.255.0
  nv
   service-policy input my_policy
!
!
```

Prerequisites for QoS Offload Configuration

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. Before configuring the QoS offload feature, you must have these hardware and software installed in your chassis.

- Hardware—Cisco ASR 9000 Series Aggregation Services Routers with Cisco ASR 9000 Enhanced Ethernet line cards as the location of Inter Chassis Links and Cisco ASR9000v
- Software—Cisco IOS XR Software Release 5.2.2 or higher for ASR9000v and ASR 901 satellites.

Offloading Service-policy on Physical Access Port

Perform these tasks to offload the service-policy on the physical access port. This procedure offloads the service-policy in the ingress direction of the Satellite Ethernet interface.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match precedence***precedence-value* [*precedence-value1 ... precedence-value6*]
4. **end-class-map**
5. **policy-map** [**type qos**] *policy-name*
6. **class** *class-name*
7. **set qos-group** *qos-group-value*
8. **exit**
9. **end-policy-map**
10. **interface** *type interface-path-id*
11. **nv**
12. **service-policy input** *policy-map*
13. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map match-any class1`	Creates a class map to be used for matching packets to the class specified 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 precedence***precedence-value* [*precedence-value1 ... precedence-value6*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match precedence 5`	Identifies IP precedence values as match criteria. • Value range is from 0 to 7. • Reserved keywords can be specified instead of numeric values.
Step 4	**end-class-map** **Example:** `RP/0/RP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 5	**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 6	**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 7	**set qos-group** *qos-group-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set qos-group 5`	Sets the QoS group identifiers on IPv4 or MPLS packets.
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map configuration mode.
Step 9	**end-policy-map** **Example:**	Ends the policy map configuration.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-pmap)# end-policy-map`	
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface gigabitethernet 100/0/0/0`	Configures an interface and enters the interface configuration mode.
Step 11	**nv** **Example:** `RP/0/RP0/CPU0:router(config-if)# nv`	Enters the satellite network virtualization (nV) configuration submode.
Step 12	**service-policy input** *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-if-nV)# service-policy input policy1`	Attaches a policy map to an input interface to be used as the service policy for that interface.
Step 13	**commit**	

Offloading Service-policy on Bundle Access Port

Perform these tasks to offload the service-policy on the bundle access port. This procedure offloads the service-policy in the ingress direction of the Satellite Ethernet interface.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match precedence***precedence-value*
4. **end-class-map**
5. **policy-map** [**type qos**] *policy-name*
6. **class** *class-name*
7. **set qos-group** *qos-group-value*
8. **exit**
9. **end-policy-map**
10. **interface** *type interface-path-id*
11. **bundle id** *bundle-id*
12. **nv**
13. **service-policy input** *policy-map*
14. **commit**
15. **exit**
16. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map match-any class2`	Creates a class map to be used for matching packets to the class specified 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 precedence***precedence-value* **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match precedence 6`	Identifies IP precedence values as match criteria. • Value range is from 0 to 7. • Reserved keywords can be specified instead of numeric values.
Step 4	**end-class-map** **Example:** `RP/0/RP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 5	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map policy2`	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 6	**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.
Step 7	**set qos-group** *qos-group-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set qos-group 5`	Sets the QoS group identifiers on IPv4 or MPLS packets.
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map configuration mode.
Step 9	**end-policy-map** **Example:**	Ends the policy map configuration.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-pmap)# end-policy-map`	
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface bundle-ether 1`	Configures an interface and enters the interface configuration mode.
Step 11	**bundle id** *bundle-id* **Example:** `RP/0/RP0/CPU0:router(config-if)# bundle id 1`	Creates a multilink interface bundle with the specified bundle ID.
Step 12	**nv** **Example:** `RP/0/RP0/CPU0:router(config-if)# nv`	Enters the satellite network virtualization (nV) configuration submode.
Step 13	**service-policy input** *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-if-nV)# service-policy input policy2`	Attaches a policy map to an input interface to be used as the service policy for that interface.
Step 14	**commit**	
Step 15	**exit** **Example:** `RP/0/RP0/CPU0:router(config-if)# exit`	Returns the router to global configuration mode.
Step 16	**commit**	

Offloading Service-policy on Physical Satellite Fabric Link

Perform these tasks to offload the service-policy on the physical Satellite Fabric Link (SFL). This procedure offloads the service-policy in the egress direction of SFL.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match qos-group** [*qos-group-value*]
4. **end-class-map**
5. **policy-map** [**type qos**] *policy-name*
6. **class** *class-name*
7. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}

8. **exit**
9. **end-policy-map**
10. **interface** *type interface-path-id*
11. **nv**
12. **satellite-fabric-link satellite** *satellite_id*
13. **remote-ports** *interface_type remote_subslot*
14. **service-policy output** *policy-map*
15. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map match-any class3`	Creates a class map to be used for matching packets to the class specified 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 qos-group** [*qos-group-value*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match qos-group 5`	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. • **match qos-group**command is supported only for an egress policy.
Step 4	**end-class-map** **Example:** `RP/0/RP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 5	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map policy3`	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 6	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class3`	Specifies the name of the class whose policy you want to create or change.

	Command or Action	Purpose
Step 7	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth percent 13`	Specifies the bandwidth allocated for a class belonging to a policy map.
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map configuration mode.
Step 9	**end-policy-map** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# end-policy-map`	Ends the policy map configuration.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface TenGigE 0/1/0/0`	Configures an interface and enters the interface configuration mode.
Step 11	**nv** **Example:** `RP/0/RP0/CPU0:router(config-if)# nv`	Enters the satellite network virtualization (nV) configuration submode.
Step 12	**satellite-fabric-link satellite** *satellite_id* **Example:** `RP/0/RP0/CPU0:router(config-if-nV)# satellite-fabric-link satellite 100`	Specifies an interface as an Interface Control Plane Extender(ICPE) inter-chassis link (ICL). **Note** The Interface Control Plane Extender(ICPE) infrastructure has a mechanism to provide the Control Plane of an interface physically located on the Satellite device in the local Cisco IOS XR software.
Step 13	**remote-ports** *interface_type remote_subslot* **Example:** `RP/0/RP0/CPU0:router(config-satellite-fabric-link)# remote-ports Satellite-Ether 0/0/0-9`	Configures the remote satellite ports 0 to 9.
Step 14	**service-policy output** *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-satellite-fabric-link)# service-policy output policy3`	Attaches a policy map to an output interface to be used as the service policy for that interface.
Step 15	**commit**	

Offloading Service-policy on Bundle SFL

Perform these tasks to offload the service-policy on the bundle Satellite Fabric Link (SFL). This procedure offloads the service-policy in the egress direction of SFL.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match qos-group** [*qos-group-value*]
4. **end-class-map**
5. **policy-map** [**type qos**] *policy-name*
6. **class** *class-name*
7. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}
8. **exit**
9. **end-policy-map**
10. **interface** *type interface-path-id*
11. **bundle id** *bundle-id*
12. **nv**
13. **satellite-fabric-link satellite** *satellite_id*
14. **remote-ports***interface_type remote_subslot*
15. **service-policy output** *policy-map*
16. **commit**
17. **exit**
18. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map match-any class4`	Creates a class map to be used for matching packets to the class specified 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 qos-group** [*qos-group-value*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match qos-group 5`	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 4	**end-class-map** **Example:** `RP/0/RP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 5	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map policy4`	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 6	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class4`	Specifies the name of the class whose policy you want to create or change.
Step 7	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth percent 13`	Specifies the bandwidth allocated for a class belonging to a policy map.
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map configuration mode.
Step 9	**end-policy-map** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# end-policy-map`	Ends the policy map configuration.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface Bundle-Ether 2`	Configures an interface and enters the interface configuration mode.
Step 11	**bundle id** *bundle-id* **Example:** `RP/0/RP0/CPU0:router(config-if)# bundle id 2`	Creates a multilink interface bundle with the specified bundle ID.
Step 12	**nv** **Example:**	Enters the satellite network virtualization (nV) configuration submode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-if)# nv`	
Step 13	**satellite-fabric-link satellite** *satellite_id* **Example:** `RP/0/RP0/CPU0:router(config-if)# satellite-fabric-link satellite 100`	Specifies an interface as an Interface Control Plane Extender(ICPE) inter-chassis link (ICL). **Note** The Interface Control Plane Extender(ICPE) infrastructure has a mechanism to provide the Control Plane of an interface physically located on the Satellite device in the local Cisco IOS XR software.
Step 14	**remote-ports***interface_type remote_subslot* **Example:** `RP/0/RP0/CPU0:router(config-satellite-fabric-link)# remote-ports GigabitEthernet 0/0/0-5`	Configures the remote satellite ports 0 to 5.
Step 15	**service-policy output** *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-satellite-fabric-link)# service-policy output policy4`	Attaches a policy map to an output interface to be used as the service policy for that interface.
Step 16	**commit**	
Step 17	**exit** **Example:** `RP/0/RP0/CPU0:router(config-if)# exit`	Returns the router to global configuration mode.
Step 18	**commit**	

Offloading Service-policy on L2 Fabric Physical SFL

Perform these tasks to offload the service-policy on L2 Fabric physical Satellite Fabric Link (SFL). This procedure offloads the service-policy in the egress direction of SFL.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match qos-group** [*qos-group-value1*]
4. **end-class-map**
5. **policy-map** [**type qos**] *policy-name*
6. **class** *class-name*
7. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}
8. **exit**
9. **end-policy-map**

10. **interface** *type interface-path-id*
11. **encapsulation dot1q***vlan-identifier*
12. **nv**
13. **satellite-fabric-link satellite** *satellite_id*
14. **remote-ports***interface_type remote_subslot*
15. **service-policy output** *policy-map*
16. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RP0/CPU0:router(config)# class-map match-any class5`	Creates a class map to be used for matching packets to the class specified 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 qos-group** [*qos-group-value1*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match qos-group 5`	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. • **match qos-group**command is supported only for an egress policy.
Step 4	**end-class-map** **Example:** `RP/0/RP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 5	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map policy5`	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 6	**class** *class-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap)# class class5`	Specifies the name of the class whose policy you want to create or change.
Step 7	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:**	Specifies the bandwidth allocated for a class belonging to a policy map.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-pmap-c)# bandwidth percent 13`	
Step 8	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map configuration mode.
Step 9	**end-policy-map** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# end-policy-map`	Ends the policy map configuration.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface TenGigabitEthernet 0/1/0/0.1`	Configures an interface and enters the interface configuration mode.
Step 11	**encapsulation dot1q***vlan-identifier* **Example:** `RP/0/RP0/CPU0:router(config-if)# encapsulation dot1q 20`	Defines the encapsulation format as IEEE 802.1Q (**dot1q**), and specifies the VLAN identifier.
Step 12	**nv** **Example:** `RP/0/RP0/CPU0:router(config-subif)# nv`	Enters the satellite network virtualization (nV) configuration submode.
Step 13	**satellite-fabric-link satellite** *satellite_id* **Example:** `RP/0/RP0/CPU0:router(config-if-nV)# satellite-fabric-link satellite 100`	Specifies an interface as an Interface Control Plane Extender(ICPE) inter-chassis link (ICL). **Note** The Interface Control Plane Extender(ICPE) infrastructure has a mechanism to provide the Control Plane of an interface physically located on the Satellite device in the local Cisco IOS XR software.
Step 14	**remote-ports***interface_type remote_subslot* **Example:** `RP/0/RP0/CPU0:router(config-satellite-fabric-link)# remote-ports GigabitEthernet 0/0/0-5`	Configures the remote satellite ports 0 to 5.
Step 15	**service-policy output** *policy-map* **Example:**	Attaches a policy map to an output interface to be used as the service policy for that interface.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-satellite-fabric-link)# service-policy output policy5`	
Step 16	**commit**	

Configuration Examples for QoS Offload

Note While the examples use 1G access ports and 10G fabric ports, the same can be applied to Cisco NCS 5000 series 10G access and 10G/100G fabric ports for supported scenarios.

Offloading Service-policy on Physical Access Port: 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 GigabitEthernet interface 100/0/0/0. This service-policy is configured under the nv mode and thus the QoS policy is offloaded to the satellite.

```
config
class-map match-any class1
 match precedence 6
 end-class-map
!
policy-map policy1
 class class1
  set qos-group 5
 !
interface gigabitEthernet 100/0/0/0
nv
service-policy input policy1
end or commit
```

Offloading Service-policy on Bundle Access Port: Example

In this example, a service-policy called policy2 is created. This service policy is associated to a class map called class2 through the use of the class command. The service policy is then attached in the input direction on a bundle-ether interface with bundle id as 1 that has two bundle member links—GigabitEthernet interface 100/0/0/1 and GigabitEthernet interface 100/0/0/2. This service-policy is configured under the nv mode and thus the QoS policy is offloaded to the satellite bundle-ether interface.

```
config
class-map match-any class2
 match precedence 6
 end-class-map
!
policy-map policy2
 class class2
  set qos-group 5
```

```
  end-policy-map
 !
interface bundle-ether 1
bundle-id 1
nv
service-policy input policy2
end or commit
!
end or commit
```

Offloading Service-policy on Physical SFL: Example

In this example, a service-policy called policy3 is created, which is associated to a class map called class3 through the use of the class command. The service policy is applied to the host-facing satellite fabric link (SFL) on the satellite 100 and attached in the output direction on a TenGigE interface 0/1/0/0. This is configured under the nv mode and thus the QoS policy is offloaded to the satellite.

```
config
class-map match-any class3
 match qos-group 5
 end-class-map
!
policy-map policy3
 class class3
  bandwidth percent 13
 !
interface TenGigE 0/1/0/0
nv satellite-fabric-link satellite 100
remote-ports GigabitEthernet 0/0/0-9
service-policy output policy3
end or commit
```

Offloading Service-policy on Bundle SFL: Example

In this example, a service-policy called policy4 is created, which is associated to a class map called class4 through the use of the class command. The service policy is applied to the host-facing bundle satellite fabric link (SFL) on the satellite 100 and attached in the output direction on the bundle-ether interface with bundle id 2 that has two bundle member links—TengGig interface 0/1/0/0 and TengGig interface 0/1/0/1. This is configured under the nv mode and thus the QoS policy is offloaded to the satellite.

```
config
class-map match-any class4
 match qos-group 5
 end-class-map
!
policy-map policy4
 class class4
  bandwidth percent 13
 !
interface Bundle-ether 2
nv satellite-fabric-link satellite 100
remote-ports GigabitEthernet 0/0/0-5
service-policy output policy4
exit/commit
interface TengGig 0/1/0/0
bundle-id 2
!
interface TengGig 0/1/0/1
```

```
bundle-id 2
!
end or commit
```

Offloading Service-policy on L2 Fabric physical SFL: Example

In this example, a service-policy called policy5 is created, which is associated to a class map called class5 through the use of the class command. The service policy is applied to the host-facing bundle SFL under the nv mode and attached in the output direction on the TenGigabitEthernet 0/1/0/0.1 sub-interface. The QoS policy is offloaded to the satellite 100 in the L2 Fabric network.

```
config
class-map match-any class5
 match qos-group 5
 end-class-map
!
policy-map policy5
 class class5
  bandwidth percent 13
 !
interface TenGigabitEthernet 0/1/0/0.1
encapsulation dot1q 20
nv satellite-fabric-link satellite 100
remote-ports GigabitEthernet 0/0/0-5
service-policy output policy5
end or commit
```