# 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.

### Line Card, SIP, and SPA Support

| Feature | ASR 9000 Ethernet Line Cards | SIP 700 for the ASR 9000 |
|---|---|---|
| Classification Based on DEI | yes | no |
| Class-Based Unconditional Packet Marking | yes | yes |
| In-Place Policy Modification | yes | yes |
| IPv6 QoS | yes | yes |
| Packet Classification and Marking | yes | yes |
| Policy Inheritance | yes | yes |
| Port Shape Policies | yes | no |
| Shared Policy Instance | yes | no |

### Feature History for Configuring Modular QoS Packet Classification and Marking on Cisco ASR 9000 Series Routers

| Release | Modification |
|---|---|

| | |
|---|---|
| Release 3.7.2 | The Class-Based Unconditional Packet Marking feature was introduced on ASR 9000 Ethernet Line Cards.<br>The IPv6 QoS feature was introduced on ASR 9000 Ethernet Line Cards. (QoS matching on IPv6 ACLs is not supported.)<br>The Packet Classification and Marking feature was introduced on ASR 9000 Ethernet Line Cards. |
| Release 3.9.0 | The Class-Based Unconditional Packet Marking feature was supported on the SIP 700 for the ASR 9000.<br>The Packet Classification and Marking feature was supported on the SIP 700 for the ASR 9000.<br>The Policy Inheritance feature was introduced on ASR 9000 Ethernet Line Cards and on the SIP 700 for the ASR 9000.<br>The Shared Policy Instance feature was introduced on ASR 9000 Ethernet Line Cards. |
| Release 4.0.0 | The Classification Based on DEI feature was introduced on ASR 9000 Ethernet Line Cards.<br>The In-Place Policy Modification feature was introduced on ASR 9000 Ethernet Line Cards and on the SIP 700 for the ASR 9000.<br>The IPv6 QoS feature was supported on the SIP 700 for the ASR 9000.<br>Support for three stand-alone marking actions and three marking actions as part of a policer action in the same class was added on the SIP 700 for the ASR 9000. (ASR 9000 Ethernet Line Cards support two stand-alone marking actions and two marking actions as part of a policer action in the same class.) |
| Release 4.0.1 | Support for the port shape policies feature was introduced on ASR 9000 Ethernet Line Cards. |
| Release 4.2.1 | QoS on satellite feature was added. |
| Release 5.1.1 | The QoS Offload on satellite feature was added. |
| | The Inter Class Policer Bucket Sharing feature was added. This feature is applicable to all ASR 9000 Enhanced Ethernet line cards except the first generation and SIP 700 line cards. |
| Release 5.1.2 | The QoS Offload on 901 feature was added. |
| | Port Shaper Policy Support on L2 Fabric ICL Interface |
| Release 5.2.0 | The QPPB on GRE Tunnel Interfaces feature was added. |
| Release 6.0 | Classification Support for Ethernet-Services ACL was added. |
| Release 6.1.2 | New Chapter, "Configuring QoS on the Satellite System" was created that contains information about QoS Offload. |
| Release 6.2.1 | Support for Multiple QoS Policy feature was added. |

• Prerequisites for Configuring Modular QoS Packet Classification, on page 3
• Information About Configuring Modular QoS Packet Classification, on page 3
• ICMP, ICMPv6 Fragment Packet and Message Type QoS Classification Enhancement, on page 41
• How to Configure Modular QoS Packet Classification, on page 46
• Configuring Policer Bucket Sharing, on page 67
• Overview of Multiple QoS Policy Support, on page 70
• Configuration Examples for Configuring Modular QoS Packet Classification, on page 85
• Additional References, on page 93

# Prerequisites for Configuring Modular QoS Packet Classification

These prerequisites are required for configuring modular QoS packet classification and marking 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.

**Note** IPv6-based classification is supported only on Layer 3 interfaces.

# 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*.

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" section on page 18.

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 function of these commands is described more thoroughly in the Cisco ASR 9000 Series Aggregation Services Routers Modular Quality of Service Command Reference. The traffic class configuration task is described in the "Creating a Traffic Class" section on page 32.

**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.

# 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 Modular Quality of Service (MQC) (for example, the **policy-map** *policy1* command creates a traffic policy named *policy1*). The traffic class that is used to classify traffic to the specified traffic policy is defined in class map configuration mode. After choosing the traffic class that is used to classify traffic to the traffic policy, the user can enter the QoS features to apply to the classified traffic.

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

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

The function of these commands is described more thoroughly in the *Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Command Reference*.

The traffic policy configuration task is described in "Creating a Traffic Policy" section on page 38.

### Limitation

Review the limitations prior to configuring service packet classification using policy-maps on interfaces:

- 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.
- Applying QoS policies on both a main interface and its subinterface in the same direction is not supported. Configure the policy either on the main interface or on the subinterface.

## 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 the "Configuring Modular QoS Congestion Avoidance on Cisco ASR 9000 Series Routers" module in this guide

## Bundle Traffic Policies

When a policy is bound to a bundle, the same policy is programmed on every bundle member (port). For example, if there is a policer or shaper rate, the same rate is configured on every port. Traffic is scheduled to bundle members based on the load balancing algorithm.

A policy can be bound to:

- Bundles
- Bundle Layer 3 subinterfaces
- Bundle Layer 2 subinterfaces (Layer 2 transport)

Both ingress and egress traffic is supported. Percentage-based policies and absolute rate-based policies are supported. However, for ease of use, it is recommended to use percentage-based policies.

## Shared Policy Instance

After the traffic class and traffic policy have been created, Shared Policy Instance (SPI) can optionally be used to allow allocation of a single set of QoS resources and share them across a group of subinterfaces, multiple Ethernet flow points (EFPs), or bundle interfaces.

Using SPI, a single instance of qos policy can be shared across multiple subinterfaces, allowing for aggregate shaping of the subinterfaces to one rate. All of the subinterfaces that share the instance of a QoS policy must

belong to the same physical interface. The number of subinterfaces sharing the QoS policy instance can range from 2 to the maximum number of subinterfaces on the port.

For bundle interfaces, hardware resources are replicated per bundle member. All subinterfaces that use a common shared policy instance and are configured on a Link Aggregation Control Protocol (LAG) bundle must be load-balanced to the same member link.

When a policy is configured on a bundle EFP, one instance of the policy is configured on each of the bundle member links. When using SPI across multiple bundle EFPs of the same bundle, one shared instance of the policy is configured on each of the bundle member links. By default, the bundle load balancing algorithm uses hashing to distribute the traffic (that needs to be sent out of the bundle EFPs) among its bundle members. The traffic for single or multiple EFPs can get distributed among multiple bundle members. If multiple EFPs have traffic that needs to be shaped or policed together usingSPI, the bundle load balancing has to be configured to select the same bundle member (hash-select) for traffic to all the EFPs that belong the same shared instance of the policy. This ensures that traffic going out on all the EFPs with same shared instance of the policy use the same policer/shaper Instance.

This is normally used when the same subscriber has many EFPs, for example, one EFP for each service type, and the provider requires shaping and queuing to be implemented together for all the subscriber EFPs.

## Policy Inheritance

When a policy map is applied on a physical port, the policy is enforced for all Layer 2 and Layer 3 subinterfaces under that physical port.

# 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.

# Support for 16 Queues

The ASR 9000 traffic manager (TM) for the enhanced Ethernet line cards now supports up to 16 Queues. The extension is from 8 queues to 16 queues at leaf level called L4 in a QoS policy.

The capabilities of each mode are:

- 8 Q-mode—8 L4 flows per L3 class. Up to 32000 L3 classes in TM.
- 16 Q-mode—16 L4 flows per L3 class. Up to 16000 L3 classes in TM.

This table provides the different service profiles supported in different modes:

| Mode | Service Profile |
|---|---|
| 8Q | 1 priority-1 queue, 1 priority-2 queue, 6 normal priority queue |
| 16Q | 1 priority-1 queue, 1 priority-2 queue, 14 normal priority queue |
| 8Q | 1 priority-1 queue, 2 priority-2 queues, 5 normal priority queue (BNG Only) |
| 16Q | 1 priority-1 queue, 2 priority-2 queues, 13 normal priority queue (BNG Only) |
| 8Q | 1 priority-1 queue, 1 priority-2 queue, 1 priority- 3 queue, 5 normal priority queue |
| 16Q | 1 priority-1 queue, 1 priority-2 queue, 1 priority- 3 queue, 13 normal priority queue |

The L3, L4 service profiles in 16 Q-mode are similar to that of the 8 Q-mode, with just an increase in the number of normal priority queues.

**Restrictions**

The support for 16 queues has these restrictions:

- Supports only the enhanced Ethernet line cards.
- When 16Q-mode policy is applied on all interfaces, the number of interface scale will be 4K interface.

# Class-based Unconditional Packet Marking Feature and Benefits

This feature provides users with a means for efficient packet marking by which the users can differentiate packets based on the designated markings.

This 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 Layer 2 class-of-service (CoS) value.
- Mark packets by setting inner and outer CoS tags for an IEEE 802.1Q tunneling (QinQ) configuration.
- Mark packets by setting the value of the *qos-group* argument.
- Mark packets by setting the value of the *discard-class* argument.

**Note** *qos-group* and *discard-class* are variables internal to the router, and are not transmitted.

**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
- Cisco ASR 9000 fourth Generation family of QSFP-based dense 100GE 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, can be used 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 packets to a QoS group. To set the QoS group identifier on MPLS packets, use the **set qos-group** command in policy map class configuration mode.

**Note** Setting the QoS group identifier does not automatically prioritize the packets for transmission. You must first configure an egress policy that uses the QoS group.

- Use CoS unconditional packet marking to assign packets to set the priority value of IEEE 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.

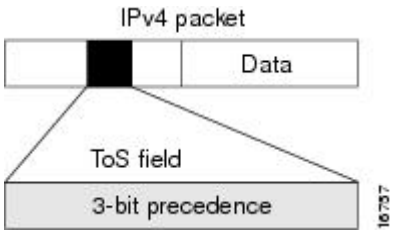
The configuration task is described in the Configuring Class-based Unconditional Packet Marking, page 46.

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

# 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 1: 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, 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. This allows the rest of the core or backbone to implement QoS based on precedence.

The configuration task is described in the "Configuring Class-based Unconditional Packet Marking" section on page 46.

## 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 1: IP Precedence Values*

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

**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.

# Classification Based on DEI

You can classify traffic based on the Drop Eligible Indicator (DEI ) bit that is present in 802.1ad frames and in 802.1ah frames. Default DEI marking is supported. The set DEI action in policy maps is supported on 802.1ad packets for:

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

**Note** The set DEI action is ignored for traffic on interfaces that are not configured for 802.1ad encapsulation.

## Default DEI Marking

| Incoming Packet | | Default DEI on Imposed 802.1ad Headers |
|---|---|---|
| 802.1q packet | None | 0 |
| 802.1ad packet | None | DEI of top-most tag of the incoming packet |
| 802.1q packet translated to 802.1ad packet<br>or<br>802.1ad packet | set dei {0 \| 1} | 0 or 1<br>Based on DEI value in the set action |

# 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

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

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/0/CPU0:Router#configure`<br>`Mon Aug 10 14:35:04.826 IST`<br>`RP/0/0/CPU0:Router(config)#` | Enters the global configuration mode. |
| **Step 2** | **ntp** {*ipv4*} **dscp**<br>**Example:**<br>`RP/0/0/CPU0:Router(config)#ntp ipv4 dscp ?`<br>`<0-63> Differentiated services codepoint value`<br>`af11 Match packets with AF11 dscp (001010)`<br>`af12 Match packets with AF12 dscp (001100)`<br>`af13 Match packets with AF13 dscp (001110)`<br>`af21 Match packets with AF21 dscp (010010)`<br>`af22 Match packets with AF22 dscp (010100)`<br>`af23 Match packets with AF23 dscp (010110)`<br>`af31 Match packets with AF31 dscp (011010)`<br>`af32 Match packets with AF32 dscp (011100)`<br>`af33 Match packets with AF33 dscp (011110)`<br>`af41 Match packets with AF41 dscp (100010)`<br>`af42 Match packets with AF42 dscp (100100)`<br>`af43 Match packets with AF43 dscp (100110)`<br>`cs1 Match packets with CS1(precedence 1) dscp (001000)`<br>`cs2 Match packets with CS2(precedence 2) dscp (010000)`<br>`cs3 Match packets with CS3(precedence 3) dscp (011000)`<br>`cs4 Match packets with CS4(precedence 4) dscp (100000)`<br>`cs5 Match packets with CS5(precedence 5) dscp (101000)`<br>`cs6 Match packets with CS6(precedence 6) dscp (110000)`<br>`cs7 Match packets with CS7(precedence 7) dscp (111000)`<br>`default Match packets with default dscp (000000)`<br>`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**<br>**Example:**<br>`RP/0/RP0/CPU0:Router(config)#commit`<br>**Example:** | Commit- saves the configuration changes and remains within the configuration session.<br>End- prompts the user to take one of these actions:<br>• Yes- Saves configuration changes and exits the configuration session. |

| | Command or Action | Purpose |
|---|---|---|
| | `RP/0/0/CPU0:Router#exit`<br>`(` | • No-Exits the configuration session without committing the configuration changes.<br>• Cancel-Remains in the configuration session, without committing the configuration changes. |
| **Step 4** | **show processes ntpd**<br>**Example:**<br>`RP/0/RP0/CPU0:Router#show processes ntpd`<br>`Mon Jun 22 20:25:18.026 IST`<br>`Job Id: 208`<br>`PID: 2540`<br>`Executable path: /pkg/bin/ntpd`<br>`Instance #: 1`<br>`Version ID: 00.00.0000`<br>`Respawn: ON`<br>`Respawn count: 1`<br>`Last started: Fri Jun 19 16:04:14 2015`<br>`Process state: Run`<br>`Package state: Normal`<br>`Process group: dlrsc`<br>`core: MAINMEM`<br>`Max. core: 0`<br>`Level: 120`<br>`Placement: None`<br>`startup_path: /pkg/startup/ip_ntp.startup`<br>`Ready: 2.444s`<br>`Process cpu time: 0.074s user, 0.031s kernel, 1.005s total`<br>`PID TID CPU Stack Pri State Run Time CPU use NAME`<br>`2540 2978 1 92K 15 Sleeping 3:04:20:59s 0.000s ntpd`<br>`2540 2975 4 92K 15 Sleeping 3:04:20:59s 0.001s ntpd`<br>`2540 2947 5 92K 15 Sleeping x3:04:20:59s 0.027s chkpt_evm`<br>`2540 2943 1 92K 15 Sleeping 3:04:21:00s 0.000s ITAL Server Thr`<br>`2540 2914 5 92K 15 Sleeping 3:04:21:00s 0.000s async`<br>`2540 2810 3 92K 15 Sleeping 3:04:21:00s 0.000s EnXR internal:mmap_peer_threa`<br>`2540 2760 2 92K 15 Sleeping 3:04:21:00s 0.011s ntpd`<br>`2540 2540 1 92K 15 Sleeping 3:04:21:03s 0.064s ntpd` | Displays the ntpd process |
| **Step 5** | **show ntp associations**<br>**Example:** | Shows the status of NTP associations. |

| | Command or Action | Purpose |
|---|---|---|
| | Router# show ntp associations<br>Sat Feb 14 13:53:18.468 UTC<br><br>address ref clock st when poll reach delay offset disp<br>*~223.255.254.254 171.68.38.65 2 121 128 377 2.44 -0.260 4.537<br>* 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

**Procedure**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>RP/0/0/CPU0:Router#configure<br>Mon Aug 10 14:35:04.826 IST<br>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**<br>**Example:**<br>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:<br>• Yes- Saves configuration changes and exits the configuration session.<br>• 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. |

# 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

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/0/CPU0:Router#configure`<br>`Mon Aug 10 14:35:04.826 IST`<br>`RP/0/0/CPU0:Router(config)#` | Enters the global configuration mode. |
| **Step 2** | **ntp** {*ipv4* } **precedence** *codepoint_value*<br>**Example:**<br>`RP/0/0/CPU0:Router(config)#ntp ipv4 precedence ?`<br>`<0-7> Precedence value`<br>`critical Match packets with critical precedence (5)`<br>`flash Match packets with flash precedence (3)`<br>`flash-override Match packets with flash override precedence (4)`<br>`immediate Match packets with immediate precedence (2)`<br>`internet Match packets with internetwork control precedence (6)`<br>`network Match packets with network` | Sets the ntp [IPv4] precedence. It ranges from 0 to 63. |

| | Command or Action | Purpose |
|---|---|---|
| | `control precedence (7)`<br>`  priority  Match packets with priority precedence (1)`<br>`  routine  Match packets with routine precedence (0)` | |
| **Step 3** | **ntp** { *ipv4*} **precedence**<br>**Example:**<br>`RP/0/RP0/CPU0:Router(config)#ntp ipv4 precedence internet` | Sets precedence values. |
| **Step 4** | **end** or **commit**<br>**Example:**<br>`RP/0/RP0/CPU0:Router(config)#commit`<br>**Example:**<br>`RP/0/0/CPU0:Router#exit`<br>`(` | Commit- saves the configuration changes and remains within the configuration session.<br>End- prompts the user to take one of these actions:<br>• Yes- Saves configuration changes and exits the configuration session.<br>• No-Exits the configuration session without committing the configuration changes.<br>• Cancel-Remains in the configuration session, without committing the configuration changes. |
| **Step 5** | **show ntp status**<br>**Example:**<br>`RP/0/RP0/CPU0:Router#show ntp status`<br>`Mon Aug 10 14:35:04.826 IST`<br>`Clock is synchronized, stratum 3, reference is 223.255.254.254`<br>`nominal freq is 1000000000.0000 Hz, actual freq is 30440042.8893 Hz, precision is 2**22`<br>`reference time is D889D255.BF525356 (13:55:33.747 UTC Sat Feb 14 2015)`<br>`clock offset is -0.413 msec, root delay is 3.569 msec`<br>`root dispersion is 20.55 msec, peer dispersion is 4.04 msec`<br>`loopfilter state is 'CTRL' (Normal Controlled Loop), drift is 0.0000318515 s/s`<br>`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

**Procedure**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RP0/CPU0:Router(config)#` | Enters the global configuration mode. |
| **Step 2** | **ntp ,** source { *ipv6* } **dscp** *codepoint_value*<br>**Example:**<br>`RP/0/0/CPU0:Router(config)#ntp ipv6 dscp ?`<br>`<0-63> Differentiated services codepoint value`<br>`af11 Match packets with AF11 dscp (001010)`<br>`af12 Match packets with AF12 dscp (001100)`<br>`af13 Match packets with AF13 dscp (001110)`<br>`af21 Match packets with AF21 dscp (010010)`<br>`af22 Match packets with AF22 dscp (010100)`<br>`af23 Match packets with AF23 dscp (010110)`<br>`af31 Match packets with AF31 dscp (011010)`<br>`af32 Match packets with AF32 dscp (011100)`<br>`af33 Match packets with AF33 dscp (011110)`<br>`af41 Match packets with AF41 dscp (100010)`<br>`af42 Match packets with AF42 dscp (100100)`<br>`af43 Match packets with AF43 dscp (100110)`<br>`cs1 Match packets with CS1(precedence 1) dscp (001000)`<br>`cs2 Match packets with CS2(precedence 2) dscp (010000)`<br>`cs3 Match packets with CS3(precedence 3) dscp (011000)`<br>`cs4 Match packets with CS4(precedence 4) dscp (100000)`<br>`cs5 Match packets with CS5(precedence 5)` | Configures options in **DSCP** for a particular source address in IPV6 packets. The value ranges between 0 - 63. |

| | Command or Action | Purpose |
|---|---|---|
| | `dscp (101000)`<br>`  cs6      Match packets with CS6(precedence 6)`<br>`dscp (110000)`<br>`  cs7      Match packets with CS7(precedence 7)`<br>`dscp (111000)`<br>`  default  Match packets with default dscp (000000)`<br><br>`  ef       Match packets with EF dscp (101110)` | |
| **Step 3** | **ntp** { *ipv6* } **precedence** *codepoint_value*<br>**Example:**<br>`  RP/0/0/CPU0:Router(config)#ntp ipv6 precedence ?`<br>`  <0-7>           Precedence value`<br>`  critical        Match packets with critical precedence (5)`<br>`  flash           Match packets with flash precedence (3)`<br>`  flash-override  Match packets with flash override precedence (4)`<br>`  immediate       Match packets with immediate precedence (2)`<br>`  internet        Match packets with internetwork control precedence (6)`<br>`  network         Match packets with network control precedence (7)`<br>`  priority        Match packets with priority precedence (1)`<br>`  routine         Match packets with routine precedence (0)` | Sets **ntp** *ipv6* **precedence** |
| **Step 4** | **end** `or` **commit**<br>**Example:**<br>`RP/0/RP0/CPU0:Router(config)#commit`<br>**Example:**<br>`RP/0/0/CPU0:ios#exit`<br>`(` | Commit- saves the configuration changes and remains within the configuration session.<br>End- prompts the user to take one of these actions:<br>• Yes- Saves configuration changes and exits the configuration session.<br>• No-Exits the configuration session without committing the configuration changes.<br>• 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.

# 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.

## 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 for QoS on PWHE

### Supported Interfaces

You can configure PWHE in Layer 3 mode on PWHE-Ether main and subinterfaces.

### QoS Accounting Scope

QoS accounting, which measures and records the packet length when performing QoS functions such as policing, shaping, and gathering statistics, doesn't include PW headers.

### QoS Configuration Rules

- **match** commands are optional when configuring QoS on PWHE. However, you must configure at least one match criterion for a class.
- When using the **match access-group** command to configure the match criteria for a class map on the basis of the specified access control list (ACL), QoS classification based on the packet length or time to live (TTL) field in the IPv4 and IPv6 headers is not supported.

### L2 Header Based Classification and Marking Scope

L2 header classification and marking are not supported on L3 PWHE interfaces.

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

### Egress QoS group classification

Egress classification based on QoS-group is not supported for BUM traffic in AC-to-PW flows.

## 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 2: 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 3: 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 4: 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 *L2VPN and Ethernet Services Configuration Guide for Cisco ASR 9000 Series Routers*

# Port Shaper Policy Support on L2 Fabric ICL Interface

In L2 fabric mode, a port shaper policy can be applied on the inter-chassis link (ICL) sub-interface. The port shaper policy applied on the ICL sub-interface helps to control the traffic going out on all satellite access ports, and efficiently handles the oversubscribed backhaul ethernet virtual circuits (EVC). This port shaper policy applies to all the satellite interfaces hosted under the ICL sub-interface.

### Restrictions

- Support to add or remove the port shaper policy is implemented only when the nV satellite configuration is present on the ICL sub-interface.
- Only the port shaper policy can be configured under the L2 Fabric ICL in egress direction.
- Ability to shape all satellite ports under a single satellite in a simple ring mode is not supported.
- When operating in L2 fabric and simple ring mode, only upto 2 levels of QoS policies are supported on the satellite access ports. The user-defined class configuration is not supported at parent level.
- Pseudowire Headend (PW-HE) and Broadband Network Gateway (BNG) configurations are not supported under the satellite interfaces in L2 fabric or simple ring mode.

## Configuring Port Shaper Policy on the ICL Interface in L2 Fabric Mode

Perform this task to apply the port shaper policy on the L2 fabric inter-chassis link (ICL) ethernet virtual circuits (EVC). This procedure applies the port shaper in the egress direction of the ICL EVC.

### Before you begin

The nV satellite configuration must be applied on the inter-chassis link (ICL) interface.

### SUMMARY STEPS

1. **configure**
2. **policy-map** [ **type qos** ] *policy-name*
3. **class** *class-name*
4. **shape** {*shape* [*units*] | **average** *value*}
5. **exit**
6. **end-policy-map**

7. **interface** *type interface-path-id*
8. **service-policy output** *policy-map*
9. Use the **commit** or **end** command.

## DETAILED STEPS

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **policy-map** [ **type qos** ] *policy-name*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# policy-map icl_ps` | Creates or modifies a policy map that can be attached to one or more ICL interfaces to specify a service policy and enters the policy map configuration mode. |
| **Step 3** | **class** *class-name*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap)# class class-default` | Specifies the name of the class whose policy you want to create or change. |
| **Step 4** | **shape** {*shape* [*units*] \| **average** *value*}<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap-c)# shape average 400 mbps` | Specifies the port shape allocated for a class belonging to a policy map. |
| **Step 5** | **exit**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap)# exit` | Returns the router to policy map configuration mode. |
| **Step 6** | **end-policy-map**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap)# end-policy-map` | Ends the policy map configuration. |
| **Step 7** | **interface** *type interface-path-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# interface TenGigabitEthernet 0/1/0/0.1` | Configures an interface and enters the sub-interface configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 8** | **service-policy output** *policy-map*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-satellite-fabric-link)# service-policy output icl_ps` | Attaches a policy map to an output interface to be used as the service policy for the ICL interface. |
| **Step 9** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

# Functionality Differences Between ASR 9000 Series High Density Ethernet Line Card Generations

The table lists some fundamental differences in functionalities between the third, fourth, and fifth generations of the ASR 9000 Series High Density Ethernet line cards. See the data sheets for more information on these line cards.

| Functionality | Third Generation of ASR 9000 Series High Density Ethernet Line Cards | Fourth and Fifth Generations of ASR 9000 Series High Density Ethernet Line Cards |
|---|---|---|
| Ingress queuing | Supported.<br>See **Ingress Queuing support**for details. | Not supported |
| Pre-configured profile values | Supported | Not supported |
| Default queue limit | 100 ms | 20 ms |
| Queue length measurement unit | packets or particles | KB |
| Queue length for Weighted Random Early Detection (WRED) | Instantaneous queue length. Run the **show policy-map interface** command to view the counter readings. | Average queue length. Run the **show policy-map interface** command to view the counter readings. |

| Functionality | Third Generation of ASR 9000 Series High Density Ethernet Line Cards | Fourth and Fifth Generations of ASR 9000 Series High Density Ethernet Line Cards |
|---|---|---|
| Shaper granularity | 8 kbps for all scheduler hierarchies | • 100 kbps for scheduler hierachies L2—L4<br>• 400 kbps for scheduler hierachy L1<br>In certain use cases, traffic shaper rates may be rounded down. For example, in a two or three-level Hierarchical QOS policy with only a shaper action in the parent or grandparent policy applied to the main interface, if the configured shaper rate is<br>• Non-multiples of 4—shaper rate is rounded down to the nearest multiple of 4 due to L1 shaper granularity of 400 kbps.<br>• Multiples of 4—configured shaper rate is honored.<br>For values in non-multiples of 4, to ensure the configured shaper rate is honored, apply the policy to L2 or L3 sub-interfaces, or add a marking action to the parent or grandparent policy when applying it to the main interface. |
| Policer granularity | 8 kbps | 1 kbps |
| Default burst size for policer | 100 ms | 100 ms |
| Default burst size for shaper | 100 ms | 1 ms |
| Particle granularity | 256 bytes | 32 bytes |

| Functionality | Third Generation of ASR 9000 Series High Density Ethernet Line Cards | Fourth and Fifth Generations of ASR 9000 Series High Density Ethernet Line Cards |
|---|---|---|
| Combination for priority and normal queues | Support for two modes:<br>• 3 priority queues and 5 normal queues<br>• 2 priority queues and 6 normal queues | 2 priority queues and 6 normal queues.<br>No support for priority level 3. |
| QoS classification format ID 5 | Supported | Not supported |
| Traffic drop | Packet drops on account of a full egress queue, or drops by WRED are performed in the Traffic Manager after the packet has exited the NP pipeline. The following commands account for these drops:<br>• **show controllers np tm counters**<br>• **show policy-map interface** | Because of better integration of the NP pipeline with the Traffic Manager, the egress queue state is verified while the NP pipeline is still processing the packet. The following commands account for drops because of a full egress queue or drops by WRED:<br>• **show controllers np tm counters**<br>• **show policy-map interface**<br>• **show controllers np counters** |
| Multi-rate interfaces | Supported | Not supported |

# Prioritize BFD traffic over logical bundle with QoS on third and fifth generation of ASR 9000 Series high density ethernet line cards

Bidirectional forwarding detection over logical bundle (BLB) is a BFD implementation that:

- is implemented over VLAN interfaces
- uses a single BFD session for the entire bundle interface, monitoring the path as a whole, and
- monitors the health of a bundle interface as a single entity, rather than monitoring each physical link individually.

### Automatic line card assignment for BFD sessions

Users cannot decide which line card should handle a BFD session on an ASR 9000 router with a mix of third and fifth generation high density ethernet line cards within a logical bundle. The router automatically assigns the session to one of these physical links in the bundle based on the availability of the line cards.

This table provides information about how to handle BFD traffic on an ASR 9000 router with a mix of third and fifth generation high density ethernet line cards.

*Table 5: Handling of BFD Traffic*

| If... | Then... |
|---|---|
| the router has ASR 9000 third generation high density ethernet line cards in a logical bundle with QoS and there is congestion at the egress interface, | the BFD packets are dropped. |
| the router has ASR 9000 fifth generation high density ethernet line cards in a logical bundle with QoS, | the line cards bypass the QoS processing for BFD traffic. |
| the BFD session moves from a third generation to a fifth generation high density ethernet line card within the bundle due to a line card failure or system restart, | any previously set CoS or precedence values are bypassed, and QoS does not apply anymore. |
| the BFD session moves from a fifth generation to a third generation high density ethernet line card, | the CoS or precedence values are applied and the packets are dropped if there is congestion. |

### Apply QoS on the egress interface to prevent BFD traffic drops

Configure QoS at the egress interface to ensure that the BFD packets are prioritized, regardless of whether they originate from a third or a fifth generation high density ethernet line card in the bundle.

# 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 |

| LC type | Ingress Queuing Support |
|---|---|
| A9K-4X100GE | No |
| A9K-48X10GE-1G-SE/-TR | No |
| A9K-24X10GE-1G-SE/-TR | No |
| A99-48X10GE-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<br>**Note**<br>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<br>**Note**<br>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 | Yes |
| A9K-MOD400-SE | A9K-MPA-2X100GE | Yes |
| A9K-MOD400-SE | A9K-MPA-32X1GE | Yes |

| LC type | EP type | Ingress Queuing Support |
|---|---|---|
| A9K-MOD200-SE/ -TR | A9K-MPA-8X10GE | No<br>**Note** To enable ingress queueing on these line cards, run the command **hw-module all qos-mode ingress-queue-enable**. |
| A9K-MOD200-SE | A9K-MPA-4X10GE | Yes |
| 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 |

| LC type | EP type | Ingress Queuing Support |
|---|---|---|
| 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 |
| 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 | Yes |
| 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.

If the policy-modification is successful, the modified policy takes effect on all the interfaces to which the policy is attached. The configuration session is blocked until the policy modification is complete.

However, if the policy modification fails on any one of the interfaces, an automatic rollback is initiated to ensure that the pre-modification policy is in effect on all the interfaces. The configuration session is blocked until the rollback is complete on all affected interfaces.

If unrecoverable errors occur during in-place policy modification, the policy is put into an inconsistent state on target interfaces. Use the **show qos inconsistency** command to view inconsistency in each location. (This command is supported only on ASR 9000 Ethernet Line Cards). The configuration session is blocked until the modified policy is effective on all interfaces that are using the policy. No new configuration is possible until the configuration session is unblocked.

When a QoS policy attached to an interface is modified, there might not be any policy in effect on the interfaces in which the modified policy is used for a short period of time.

**Note** The QoS statistics for the policy that is attached to an interface are 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, there might not be any policy in effect on the interfaces in which the modified policy is used. For this reason, 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.

### Policy States

- 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.

## 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 in both the egress and ingress directions. This section describes supported and non-supported interfaces for inter-class policer bucket sharing feature.

***Table 6: 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.

# ICMP, ICMPv6 Fragment Packet and Message Type QoS Classification Enhancement

***Table 7: Feature History Table***

| Feature Name | Release Information | Feature Description |
|---|---|---|
| ICMP, ICMPv6 Fragment Packet and Message Type QoS Classification Enhancement | Release 24.3.1 | In this release, we have enhanced network congestion management and security against distributed denial-of-service (DDoS) fragment attacks by enabling more granular classification and policing of ICMP and ICMPv6 traffic.<br><br>This is made possible by enabling support for QoS policies that can classify and police ICMP and ICMPv6 fragments and message types in both flat QoS and hierarchical QoS (HQoS).<br><br>Prior to release 24.3.1, this capability was limited to classifying ICMP and ICMPv6 fragments in HQoS and message types in flat QoS only. |

This feature enables accurate identification of ICMP and ICMPv6 fragment packets in flat QoS scenarios, utilizing Access Control Lists (ACLs) for effective Modular QoS Command-Line Interface (MQC) policing. This ensures that fragmented packets are now classified into a separate class, allowing for distinct policy rules

and better congestion control. As a result, it improves network security by mitigating ICMP and ICMP6 fragment attacks.

This feature also supports classification of ICMP and ICMPv6 message types, specifically echo and echo-reply, in the HQoS scenarios. By leveraging ACLs for MQC policing within child-queues, this feature improves the granularity of traffic management, enabling the deployment of differentiated policy rules for these message types and enhancing overall network performance.

### ICMP and ICMPv6 Fragments and Message Types: QoS Classification Enhancements

Internet Control Message Protocol (ICMP) and its IPv6 variant, ICMPv6, are routing protocols integral to network diagnostics and error reporting. They facilitate communication between network devices to relay error messages and operational information.

There are two types of classification for ICMP and ICMPv6 packets:

- **Fragmentation Classification: ICMP/ICMPv6 Fragments**—Fragmentation occurs when packets are too large to be transmitted in a single frame and must be divided into smaller fragments. Proper classification of these fragments is crucial for effective congestion management and security, as it allows for the application of specific policy rules to mitigate potential fragment-based distributed denial-of-service (DDoS) attacks.
- **Message Type Classification: Echo and Echo-Reply**—These message types are used primarily for diagnostic purposes, such as the ping command, which tests the reachability of a host on an IP network. Differentiating these message types in hierarchical QoS scenarios allows for more granular traffic management and the application of tailored policy rules, enhancing network performance and reliability.

# Configure ICMP, ICMPv6 Fragment Packets for Flat QoS

### Procedure

**Step 1** Create IPv4 and IPv6 ACLs with fragment match.

- For ICMP (IPv4) ACLs, use the **ipv4 access-list** command.

  **Example:**

```
router (config)#ipv4 access-list ICMP-fragment-rate-limit
router(config-ipv4-acl)#permit icmp any any fragments
router(config-ipv4-acl)#exit
```

- For ICMPv6 (IPv6) ACLs, use the **ipv6 access-list** command.

  **Example:**

```
router (config)#ipv6 access-list ICMP-fragment-rate-limit-IPv6
router(config-ipv6-acl)#permit icmpv6 any any fragments
router(config-ipv6-acl)#exit
```

**Step 2** Create two class maps, one for IPv4 and IPv6, and attach the respective ACLs to the class maps.

- For ICMP (IPv4) class maps:

  **Example:**

```
router (config)#class-map match-any ICMP-fragment-rate-limit
router(config-cmap)#match access-group ipv4 ICMP-fragment-rate-limit
router(config-cmap)#end-class-map
router(config-cmap)#exit
```

- For ICMPv6 (IPv6) class maps:

  **Example:**

```
router (config)#class-map match-any ICMP-fragment-rate-limit
router(config-cmap)#match access-group ipv6 ICMP-fragment-rate-limit-IPv6
router(config-cmap)#end-class-map
router(config-cmap)#exit
```

**Step 3** Create a flat QoS policy map to attach to an interface.

**Example:**

```
router(config)#policy-map ICMP-fragment
router(config-pmap)#class ICMP-fragment-rate-limit
router(config-pmap-c)#police rate 10 mbps burst 1875000 bytes peak-burst 3750000 bytes
router(config-pmap-c-police)#exit
router(config-pmap-c)#exit
router(config-pmap)#exit
```

Within the policy map, specify the class type and apply traffic policing and shaping to it.

**Step 4** Attach the policy map to an input interface.

**Example:**

```
router (config)#interface TenGigE0/4/0/10
router(config-if)#service-policy output ICMP-fragment
router(config-if)#commit
```

**Step 5** Verify ICMP, ICMPv6 fragment packets in a flat QoS scenario.

**Example:**

```
router#show policy-map interface tenGigE 0/4/0/10 output

TenGigE0/4/0/10 output: ICMP-fragment
Class ICMP-fragment-rate-limit
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :             1374137/714551240           41596
      Transmitted         :              332712/173010240           9998 => 10 Mbps policing
      Total Dropped       :             1041425/541541000           31598
    Policing statistics                (packets/bytes)     (rate - kbps)
      Policed(conform)    :              332712/173010240           9998
      Policed(exceed)     :             1041425/541541000           31598
      Policed(violate)    :                   0/0                   0
      Policed and dropped :             1041425/541541000
      Policed and dropped(parent policer)  : N/A
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :             1374137/714551240           41596
    Transmitted         :              332712/173010240           9998
    Total Dropped       :             1041425/541541000           31598
```

In the configuration examples, the configured policy rate is 10 Mbps.

# Configure ICMP, ICMPv6 Message Types for Hierarchical QoS

### Procedure

**Step 1** Create IPv4 and IPv6 ACLs with message type match.

- For ICMP (IPv4) ACLs, use the **ipv4 access-list** command.

  **Example:**

```
router (config)#ipv4 access-list ICMP-fragment-rate-limit
router(config-ipv4-acl)#permit icmp any any echo
router(config-ipv4-acl)#permit icmp any any echo-reply
router(config-ipv4-acl)#exit
```

- For ICMPv6 (IPv6) ACLs, use the **ipv6 access-list** command.

  **Example:**

```
router (config)#ipv6 access-list ICMP-fragment-rate-limit-IPv6
router(config-ipv6-acl)#permit icmpv6 any any echo
router(config-ipv4-acl)#permit icmpv6 any any echo-reply
router(config-ipv6-acl)#exit
```

**Step 2** Create two class maps, one for IPv4 and IPv6, and attach the respective ACLs to the class maps.

- For ICMP (IPv4) class maps:

  **Example:**

```
router (config)#class-map match-any ICMP-fragment-rate-limit
router(config-cmap)#match access-group ipv4 ICMP-fragment-rate-limit
router(config-cmap)#end-class-map
router(config-cmap)#exit
```

- For ICMPv6 (IPv6) class maps:

  **Example:**

```
router (config)#class-map match-any ICMP-fragment-rate-limit
router(config-cmap)#match access-group ipv6 ICMP-fragment-rate-limit-IPv6
router(config-cmap)#end-class-map
router(config-cmap)#exit
```

**Step 3** Create a hierarchical QoS (HQoS) policy map to attach to an interface.

- For ICMP (IPv4) policy maps:

  **Example:**

```
router(config)#policy-map ICMP-rate-limit
router(config-pmap)#class ICMP-rate-limit
router(config-pmap-c)#police rate 3 gbps burst 16777215 bytes peak-burst 16777215 bytes
router(config-pmap-c-police)#exit
router(config-pmap-c)#exit
router(config-pmap)#exit
router(config)#policy-map HQOS-ICMP-rate-limit
router(config-pmap)#class class-default
router(config-pmap-c)#service-policy ICMP-rate-limit
router(config-pmap-c-police)#exit
```

- For ICMPv6 (IPv6) policy maps:

  **Example:**

```
router(config)#policy-map ICMP-rate-limit
router(config-pmap)#class ICMP-rate-limit-IPv6
router(config-pmap-c)#police rate 3 gbps burst 16777215 bytes peak-burst 16777215 bytes
router(config-pmap-c-police)#exit
router(config-pmap-c)#exit
router(config-pmap)#exit
router(config)#policy-map HQOS-ICMP-rate-limit
router(config-pmap)#class class-default
router(config-pmap-c)#service-policy ICMP-rate-limit-IPv6
router(config-pmap-c-police)#exit
```

In both examples, the child policy map `ICMP-rate-limit` is referenced within the `class-default` of the parent policy map `HQOS-ICMP-rate-limit`.

**Step 4** Attach the policy map to an input interface.

**Example:**

```
router (config)#interface TenGigE0/2/0/39
router(config-if)#service-policy output HQOS-ICMP-rate-limit
router(config-if)#commit
```

**Step 5** Verify ICMP, ICMPv6 message types transmission rate in the HQoS policing.

**Example:**

```
router#show policy-map interface ten0/2/0/39 output

Fri Nov  3 14:29:59.768 KST

TenGigE0/2/0/39 output: ICMP-rate-limit

Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                  0/0                   0
    Transmitted         :                  0/0                   0
    Total Dropped       :                  0/0                   0

  Policy ICMP-type Class ICMP-rate-limit
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            47704736/12212412416          3710115
    Transmitted         :            38639920/9891819520           3000260
    Total Dropped       :             9064816/2320592896           709855
  Policing statistics                (packets/bytes)     (rate - kbps)
    Policed(conform)    :            38639920/9891819520           3000260
    Policed(exceed)     :             9064816/2320592896           709855
    Policed(violate)    :                  0/0                   0
    Policed and dropped :             9064816/2320592896
  Queueing statistics
    Queue ID                            : 36970
    High watermark                      : N/A
    Inst-queue-len  (kbytes)            : 0
    Avg-queue-len   (kbytes)            : 0
    Taildropped(packets/bytes)          : 0/0
    Queue(conform)      :            38639920/9891819520           3000260
    RED random drops(packets/bytes)     : 0/0


  Policy ICMP-type 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                            : 36970
  High watermark                      : N/A
  Inst-queue-len   (kbytes)           : 0
  Avg-queue-len    (kbytes)           : 0
  Taildropped(packets/bytes)          : 0/0
  Queue(conform)      :                  0/0                   0
  RED random drops(packets/bytes)     : 0/0
```

In the configuration examples, the configured policy rate is 3 Gbps.

# 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, page 16.

**Restrictions**

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

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

This error typically appears when using nested policy-maps, where ACEs in ACLs on different levels are multiplied.

## SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match [not] access-group** [**ipv4| ipv6| ethernet-service** ] *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. Use the **commit** or **end** command.

## DETAILED STEPS

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*<br>**Example:**<br>`RP/0/RSP0/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.<br>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| ethernet-service** ] *access-group-name*<br>**Example:**<br>`RP/0/RSP0/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.<br>**Note**<br>You can provide multiple values in a configuration; only the first value is considered for the match criteria. The subsequent values indicated in the match statement are ignored for classification. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 4** | **match** [**not**] **cos** [*cos-value*] [*cos-value0 ... cos-value7*]<br>**Example:**<br>`RP/0/RSP0/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*]<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/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. |
| **Step 8** | **match** [**not**] **discard-class** *discard-class-value* [*discard-class-value1 ... discard-class-value6*]<br>**Example:**<br>`RP/0/RSP0/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.<br>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*]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-cmap)# match dscp ipv4 15` | (Optional) Identifies a specific DSCP value as a match criterion.<br>• Value range is from 0 to 63.<br>• Reserved keywords can be specified instead of numeric values.<br>• 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*]<br>**Example:**<br>`RP/0/RSP0/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. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 11** | **match** [**not**] **precedence** [**ipv4** \| **ipv6**] *precedence-value* [*precedence-value1 ... precedence-value6*]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-cmap)# match precedence ipv4 5` | (Optional) Identifies IP precedence values as match criteria.<br>• Value range is from 0 to 7.<br>• Reserved keywords can be specified instead of numeric values. |
| **Step 12** | **match** [**not**] **protocol** *protocol-value* [*protocol-value1 ... protocol-value7*]<br>**Example:**<br>`RP/0/RSP0/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*]<br>**Example:**<br>`RP/0/RSP0/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.<br>• *qos-group-value* identifier argument is specified as the exact value or range of values from 0 to 63.<br>• Up to eight values (separated by spaces) can be entered in one match statement.<br>• **match qos-group** command is supported only for an egress policy. |
| **Step 14** | **match vlan [inner]** *vlanid* [vlanid1 ... vlanid7]<br>**Example:**<br>`RP/0/RSP0/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.<br>• *vlanid* is specified as an exact value or range of values from 1 to 4094.<br>• Total number of supported VLAN values or ranges is 8. |
| **Step 15** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

# Creating a Traffic Policy

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

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

These class-actions are supported:

- bandwidth—Configures the bandwidth for the class. See the "Configuring Modular Quality of Service Congestion Management on Cisco ASR 9000 Series Routers" module in this guide.
- police—Police traffic. See the "Configuring Modular Quality of Service Congestion Management on Cisco ASR 9000 Series Routers" module in this guide.
- priority—Assigns priority to the class. See the "Configuring Modular Quality of Service Congestion Management on Cisco ASR 9000 Series Routers" module in this guide.
- queue-limit—Configures queue-limit (tail drop threshold) for the class. See the "Configuring Modular QoS Congestion Avoidance on Cisco ASR 9000 Series Routers" module in this guide.
- random-detect—Enables Random Early Detection. See the "Configuring Modular QoS Congestion Avoidance on Cisco ASR 9000 Series Routers" module in this guide.
- service-policy—Configures a child service policy.
- set—Configures marking for this class. See the "Class-based Unconditional Packet Marking Feature and Benefits" section on page 20.
- shape—Configures shaping for the class. See the "Configuring Modular Quality of Service Congestion Management on Cisco ASR 9000 Series Routers" module in this guide.

For additional commands that can be entered as match criteria, see the *Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Command Reference*.

For conceptual information, see "Traffic Policy Elements" section on page 17.

## SUMMARY STEPS

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

**DETAILED STEPS**

**Procedure**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **policy-map** [ **type qos** ] *policy-name*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# policy-map policy1` | Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode. |
| **Step 3** | **class** class-name<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap)# class class1` | Specifies the name of the class whose policy you want to create or change. |
| **Step 4** | **set precedence** [ **tunnel** ] *precedence-value*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap-c)# set precedence 3` | Sets the precedence value in the IP header. |
| **Step 5** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

# Attaching a Traffic Policy to an Interface

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

For additional commands that can be entered in policy map class configuration mode, see the Cisco ASR 9000 Series Aggregation Services RoutersModular Quality of Service Command Reference..

**Prerequisites**

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

## SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **service-policy** {**input** | **output**} *policy-map*
4. Use the **commit** or **end** command.
5. **show policy-map interface** *type interface-path-id* [**input** | **output**]

## DETAILED STEPS

**Procedure**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **interface** *type interface-path-id*<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1` | Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface. |
| **Step 4** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |
| **Step 5** | **show policy-map interface** *type interface-path-id* [**input** | **output**]<br>**Example:** | (Optional) Displays statistics for the policy on the specified interface. |

| | Command or Action | Purpose |
|---|---|---|
| | `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet 0/1/0/9` | |

## Attaching a Shared Policy Instance to Multiple Subinterfaces

After the traffic class and traffic policy are created, you can optionally use the **service-policy (interface)** configuration command to attach a shared policy instance to multiple subinterfaces, and to specify the direction in which the policy should be applied (either on packets coming into or leaving the subinterface).

**Note** A shared policy can include a combination of Layer 2 and Layer 3 subinterfaces.

For additional commands that can be entered in policy map class configuration mode, see the Cisco ASR 9000 Series Aggregation Services Routers Modular Quality of Service Command Reference.

**Prerequisites**

A traffic class and traffic policy must be created before attaching a shared policy instance to a subinterface.

**Restrictions**

Shared policy instance across multiple physical interfaces is not supported.

### SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **service-policy** {**input** | **output**} *policy-map* [**shared-policy-instance** instance-name]
4. Use the **commit** or **end** command.
5. **show policy-map shared-policy-instance** *instance-name* [**input** | **output**] **location** *rack/slot/module*

### DETAILED STEPS

**Procedure**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **interface** *type interface-path-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/1/0/0.1` | Enters interface configuration mode and configures a subinterface. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 3** | **service-policy** {**input** \| **output**} *policy-map* [**shared-policy-instance** instance-name]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1 shared-policy-instance Customer1` | Attaches a policy map to an input or output subinterface to be used as the service policy for that subinterface.<br>• In this example, the traffic policy evaluates all traffic leaving that interface. |
| **Step 4** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |
| **Step 5** | **show policy-map shared-policy-instance** *instance-name* [**input** \| **output**] **location** *rack/slot/module*<br>**Example:**<br>`RP/0/RSP0/CPU0:router# show policy-map shared-policy-instance Customer1 location 0/1/0/7.1` | (Optional) Displays statistics for the policy on the specified shared policy instance subinterface. |

## Attaching a Shared Policy Instance to Bundle Interfaces or EFP Bundles

After the traffic class and traffic policy are created, you can optionally use the **service-policy (interface)** configuration command to attach a shared policy instance to bundle interfaces and to bundle EFPs, and to specify the direction in which the policy should be applied (either on packets coming into or leaving the subinterface).

For additional commands that can be entered in policy map class configuration mode, see the *Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Command Reference*.

**Prerequisites**

A traffic class and traffic policy must be created before attaching a shared policy instance to bundle interfaces or EFP bundles.

**Restrictions**

Shared policy instance across multiple physical interfaces is not supported.

### SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **service-policy** {**input** \| **output**} *policy-map* [**shared-policy-instance** instance-name]
4. Use the **commit** or **end** command.

5. **show policy-map shared-policy-instance** *instance-name* [**input** | **output**] **location** *location-id*

## DETAILED STEPS

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **interface Bundle-Ether** *bundle-id*<br>**Example:**<br>`RP/0/RP1/CPU0:router(config)# interface Bundle-Ether 100.1 l2transport` | Enters interface configuration mode and configures a bundle interface. |
| **Step 3** | **service-policy** {**input** \| **output**} *policy-map* [**shared-policy-instance** instance-name]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1 shared-policy-instance Customer1` | Attaches a policy map to an input or output bundle interface to be used as the service policy for that subinterface.<br>• In this example, the traffic policy evaluates all traffic leaving that interface. |
| **Step 4** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |
| **Step 5** | **show policy-map shared-policy-instance** *instance-name* [**input** \| **output**] **location** *location-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router# show policy-map shared-policy-instance Customer1 location 0/rsp0/cpu0` | (Optional) Displays statistics for the policy at the specified shared policy instance location. |

# Configuring Class-based Unconditional Packet Marking

*Table 8: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Set IP Marking for SRv6 Encapsulation | Release 24.2.1 | With this feature support for IP marking for SRv6 packets that are encapsulated, there are some important updates to the QoS behavior.<br>This is an explicit packet marking feature that applies only to ingress QoS policies.<br>**CLI:** This feature introduces the **set ip encapsulation** command. |

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

- IP precedence value
- IP DSCP value
- IP Encapsulation value
- QoS group value (ingress only)
- CoS value ( egress only on Layer 3 subinterfaces)
- MPLS experimental value
- Discard class

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

**Note** Choose only two **set** commands per class.

### Procedure

**Step 1** **configure**

**Example:**

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

**Step 2** **policy-map** *policy-name*

**Example:**

```
RP/0/RSP0/CPU0:router(config)# policy-map policy1
```

Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

**Step 3** **class** *class-name*

**Example:**

```
RP/0/RSP0/CPU0:router(config-pmap)# class class1
```

Specifies the name of the class whose policy you want to create or change and enters the policy class map configuration mode.

**Step 4** **set precedence**

**Example:**

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set precedence 1
```

Sets the precedence value in the IP header.

**Step 5** **set dscp**

**Example:**

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set dscp 5
```

Marks a packet by setting the DSCP in the ToS byte.

**Step 6** **set ip encapsulation class-of-service***cos-value*

**Example:**

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set ip encapsulation class-of-service 55
```

Sets the traffic class imposition for SRv6 encapsulation.

**Step 7** **set qos-group** *qos-group-value*

**Example:**

```
RP/0/RSP0/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 8** **set cos** *cos-value*

**Example:**

```
RP/0/RSP0/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 9** **set cos** [**inner**] *cos-value*

**Example:**

```
RP/0/RSP0/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.

**Step 10** **set mpls experimental** {**imposition** | **topmost**} *exp-value*

**Example:**

```
RP/0/RSP0/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 11** **set srp-priority** *priority-value*

**Example:**

```
RP/0/RSP0/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 12** **set discard-class** *discard-class-value*

**Example:**

```
RP/0/RSP0/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 13** **set atm-clp**

**Example:**

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set atm-clp
```

Sets the cell loss priority (CLP) bit.

**Step 14** **exit**

**Example:**

```
RP/0/RSP0/CPU0:router(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

**Step 15** **exit**

**Example:**

```
RP/0/RSP0/CPU0:router(config-pmap)# exit
```

Returns the router to global configuration mode.

**Step 16** **interface** *type* interface-path-id

**Example:**

```
RP/0/RSP0/CPU0:router(config)# interface pos 0/2/0/0
```

Configures an interface and enters the interface configuration mode.

**Step 17** **service-policy** {**input** | **output**]} *policy-map*

**Example:**

```
RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1
```

Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.

**Step 18** Use the **commit** or **end** command.

**commit** —Saves the configuration changes, and remains within the configuration session.

**end** —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration mode, without committing the configuration changes.

**Step 19** **show policy-map interface** *type interface-path-id* [**input** | **output**]

**Example:**

```
RP/0/RSP0/CPU0:router# show policy-map interface pos 0/2/0/0
```

(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

# Configuring 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.

**Restrictions**

- IPv4 and IPv6 QPPB with egress QoS policy is supported on all Ethernet and SIP-700 line cards.
- IPv4 and IPv6 QPPB with ingress QoS policy is supported on the first generation ASR9000 Ethernet line cards.

### SUMMARY STEPS

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

### DETAILED STEPS

**Procedure**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **route-policy** *name*<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap-c)# set qos-group 30` | Sets the QoS group identifiers. The set qos-group command is supported only on an ingress policy. |
| **Step 4** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

# Applying the Route Policy to BGP

This task applies the route policy to BGP.

**Prerequisites**

Configure BGP and CEF.

## SUMMARY STEPS

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

## DETAILED STEPS

**Procedure**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **router bgp** *as-number*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router bgp 120` | Enters BGP configuration mode. |
| **Step 3** | **address-family** { **ipv4** | **ipv6** } *address-family-modifier*<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp-af) # table-policy qppb a1` | Configures the routing policy for installation of routes to RIB. |
| **Step 5** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

# Configuring QPPB on the Desired Interfaces

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

## SUMMARY STEPS

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

## DETAILED STEPS

**Procedure**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **interface** *type interface-path-id*<br>**Example:**<br>`RP/0/RSP0/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*}]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# ipv4 bgp policy propagation input qos-group destination` | Enables QPPB on an interface |
| **Step 4** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes, and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration mode, without committing the configuration changes. |

# 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. Use the **commit** or **end** command.

## DETAILED STEPS

**Procedure**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **interface tunnel-ip***number*<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/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**} | Enables QPPB on the GRE tunnel interface |

| | Command or Action | Purpose |
|---|---|---|
| | {**destination**[*ip-precedence* {*destination*\|*source*}] {**source**[*ip-precedence* {*destination*\|*source*}]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# ipv4 bgp policy propagation input qos-group destination` | |
| **Step 6** | **tunnel source** *type path-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# tunnel source TenGigE0/2/0/1` | Specifies the source of the tunnel interface. |
| **Step 7** | **tunnel destination** *ip-address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# tunnel destination 100.100.100.20` | Defines the tunnel destination. |
| **Step 8** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes, and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration mode, without committing the configuration changes. |

# QPPB Scenario

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

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

# Configuring Hierarchical Ingress Policing

### SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*

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

## DETAILED STEPS

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **policy-map** *policy-name*<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap-c)# police rate percent 50` | Configures traffic policing and enters policy map police configuration mode. |
| **Step 6** | **conform-action** *action*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-action transmit` | Configures the action to take on packets that conform to the rate limit. The allowed action is **transmit** that transmits the packets. |
| **Step 7** | **exceed-action** *action*<br>**Example:** | Configures the action to take on packets that exceed the rate limit. The allowed action is **drop** that drops the packet. |

| | Command or Action | Purpose |
|---|---|---|
| | `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action drop` | |
| **Step 8** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

# Configuring Policer Bucket Sharing

Perform these tasks to enable policer bucket sharing in both the egress and ingress directions.

### 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. **service-policy output** *policy-map*
19. Use the **commit** or **end** command.

## DETAILED STEPS

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*<br>**Example:**<br>`RP/0/RSP0/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.<br>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* ]<br>**Example:**<br>`RP/0/RSP0/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**<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/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.<br>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 6** | **match precedence**[ *number* \| *name* ]<br>**Example:**<br>`RP/0/RSP0/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**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-cmap)# end-class-map` | Ends the class map configuration. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 8** | **policy-map** [ **type qos** ] *policy-name*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# policy-map policy1` | Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode. |
| **Step 9** | **class** *class-name*<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap-c)# policer bucket shared policy1 rate 2Mbps` | Defines and shares a policer bucket.<br>In this example, shared policer bucket *policy1* is created to rate limit traffic at 2Mbps. |
| **Step 11** | **exit**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap-c)# exit` | Returns the router to policy map configuration mode. |
| **Step 12** | **class** *class-name*<br>**Example:**<br>`RP/0/RSP0/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*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap-c)# policer bucket referred policy1` | Refers a shared policer bucket.<br>In this example, policer bucket *policy1* is referred. |
| **Step 14** | **exit**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap-c)# exit` | Returns the router to policy map configuration mode. |
| **Step 15** | **end-policy-map**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pmap)# end-policy-map` | Ends the policy map configuration. |
| **Step 16** | **interface** *type interface-path-id*<br>**Example:** | Configures an interface and enters the interface configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| | `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 100/0/0/0` | |
| **Step 17** | **service-policy input** *policy-map*<br>**Example:**<br>`RP/0/RSP0/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** | **service-policy output** *policy-map*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1` | Attaches a policy map to an output interface to be used as the service policy for that interface. |
| **Step 19** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

# Overview of Multiple QoS Policy Support

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

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

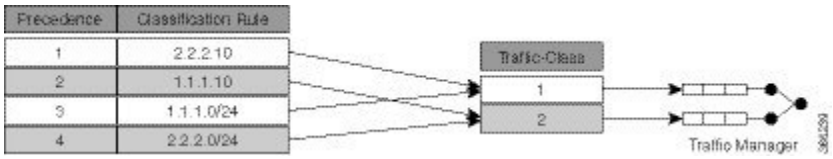
# Use Case — Multiple QoS Policy Support

Consider a scenario where:

- The classification rules must be applied at different precedence levels.
- Each classification rule must be associated with non-queueing actions (that is, policing/marking).
- Multiple classification rules at different precedence levels must be mapped to a traffic-class.

- Each traffic-class or a group of traffic-classes must be associated with a single queue.

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



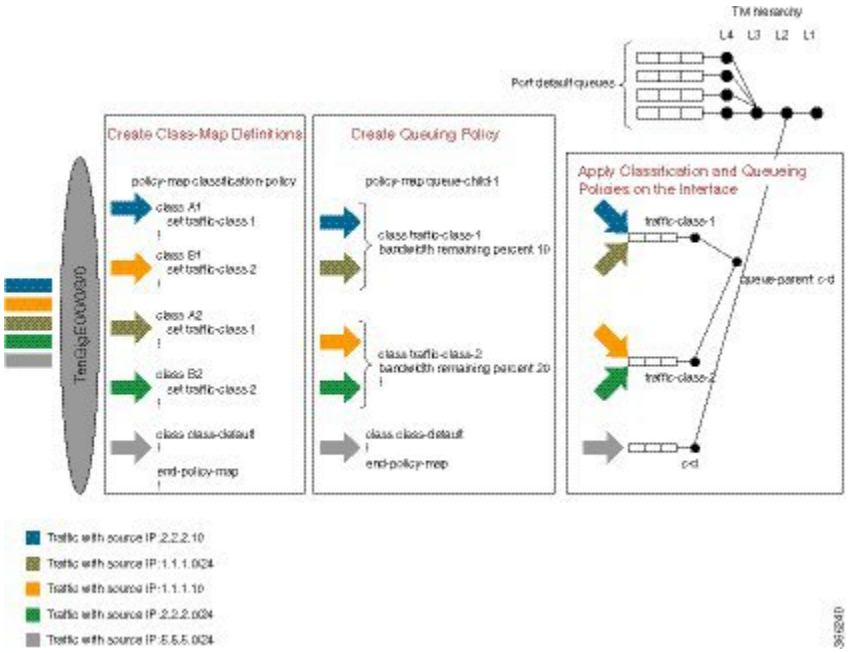

In this example, if the traffic packet matches 2.2.2.10 or 1.1.1.0/24, then the traffic packet is forwarded to the queue that is associated with traffic-class 1. And if the traffic packet matches 1.1.1.10 or 2.2.2.0/24, then the traffic packet is forwarded to the queue that is associated with traffic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

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

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

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

The example below depicts the implementation—




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

# Configuring Multiple QoS Policy Support

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

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

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

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

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

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

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

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

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

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

### Verification

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

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

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

```
Parent Policy: queue-parent Class: class-default
QueueID: 1040404 (Priority Normal)
Queue Limit: 446 kbytes Abs-Index: 52 Template: 0 Curve: 0
Shape CIR Profile: INVALID
WFQ Profile: 3/98 Committed Weight: 139 Excess Weight: 139
Bandwidth: 0 kbps, BW sum for Level 1: 0 kbps, Excess Ratio: 70
------------------------------------------------------------------------
Interface: TenGigE0/0/0/3/0 output
Bandwidth configured: 10000000 kbps Bandwidth programed: 10000000 kbps
ANCP user configured: 0 kbps ANCP programed in HW: 0 kbps
Port Shaper programed in HW: 0 kbps
Policy: classification-policy Total number of classes: 5
------------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: A1
Set traffic-class : 1
QueueID: 0 (Port Default)
Policer Profile: 59 (Single)
Conform: 100000 kbps (100 mbps) Burst: 1250000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: B1
Set traffic-class : 2
QueueID: 0 (Port Default)
Policer Profile: 60 (Single)
Conform: 200000 kbps (200 mbps) Burst: 2500000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: A2
Set traffic-class : 1
QueueID: 0 (Port Default)
Policer Profile: 61 (Single)
Conform: 300000 kbps (300 mbps) Burst: 3750000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: B2
Set traffic-class : 2
QueueID: 0 (Port Default)
Policer Profile: 62 (Single)
Conform: 400000 kbps (400 mbps) Burst: 5000000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: class-default
QueueID: 0 (Port Default)
------------------------------------------------------------------------
```

# Restrictions for Multiple QoS Policy Support

### Policy Classification Restrictions

- Classification policy must always be executed before the queuing policy. Also, queuing actions are not supported within a classification policy.

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

### Queuing Policy Restrictions

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

### Applying Multiple Services on an Interface Restrictions

- Applying multiple polices is supported only when one policy is a classification policy and the other policy is a queuing policy.
- Applying multiple polices (not more than 2 policies) is supported only in the egress direction. Applying more than 1 policy in the ingress direction is not supported.
- Applying multiple policies is supported only on the following interfaces:
  - Main-interface
  - Sub-interface
  - Bundle interface

- Bundle sub-interface

- Applying Multi policies is not supported on the following interfaces:
  - PWHE
  - GRE
  - BVI
  - Satellite interfaces

- Multi policies are only supported on Cisco ASR 9000 High Density 100GE Ethernet line cards, Cisco ASR 9000 Enhanced Ethernet line cards, and Cisco ASR 9000 Ethernet line cards.
- The same classification policy cannot be applied with different queuing policies on a different interface of the same line card.
- Classification policy and queuing policy cannot be applied with any of the following feature options
  - account
  - service-fragment-parent
  - 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.<br>Sub interface policy is enabled and uses the port shaper rate as the reference bandwidth.<br>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.<br>Both the sub interface policies are enabled and both the policies use the port shaper rate as the reference bandwidth.<br>If port shaper is applied after sub interface policies, then both the applied sub interface policies will be updated with the new reference bandwidth. If the port shaper rate is lower than any sub interface policy rate, then the port shaper policy is rejected. |

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

# Statistics

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

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

### Classification Policy

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

### Queuing policy

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

### Example: Egress Policy Classification Statistics

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

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

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

### Example: Egress Queuing Policy Statistics

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

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

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

### Restrictions for Statistics

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

# Policy Modification

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

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

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

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

### Policy Modification Restrictions

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

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

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

# Supported Features by Multi Policies

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

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

# Configuration Examples for Configuring Modular QoS Packet Classification

## Traffic Classes Defined: Example

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

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

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

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

## Traffic Policy Created: Example

In this example, a traffic policy called policy1 is defined to contain policy specifications for the two classes—class1 and class2. The match criteria for these classes were defined in the traffic classes created in the "Traffic Classes Defined: Example" section on page 68.

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" section on page 68). 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 gigabitethernet 0/1/0/9
  service-policy output policy1
  exit
!
```

# Traffic Policy Attached to Multiple Subinterfaces: Example

The following example shows how to attach an existing traffic policy to multiple subinterfaces. After you define a traffic policy with the **policy-map** command, you can attach it to one or more subinterfaces using the service policy command in subinterface configuration mode.

```
interface gigabitethernet 0/1/0/0.1
  service-policy input policy1 shared-policy-instance ethernet101
  exit
!
interface gigabitethernet 0/1/0/0.2
  service-policy input policy1 shared-policy-instance ethernet101
  exit
```

# Traffic Policy Attached to a Bundle Interface: Example

The following example shows how to attach an existing traffic policy to a bundle interface. After you define a traffic policy with the **policy-map** command, you can attach it to one or more bundle subinterfaces using the service policy command in subinterface configuration mode.

```
interface Bundle-Ether 100.1
  service-policy tripleplaypolicy shared-policy-instance subscriber1
  exit
!
interface Bundle-Ether 100.2
  service-policy output tripleplaypolicy shared-policy instance subscriber1
  exit
```

# EFP Load Balancing with Shared Policy Instance: Example

The following examples show how to configure load balancing of an EFP when SPI is implemented. For additional information on EFP load balancing on link bundles, see the Cisco IOS XR Interface and Hardware Component Configuration Guide.

## Configuring a Bundle Interface: Example

```
interface Bundle-Ether 50
interface gigabitethernet 0/1/0/5
```

```
  bundle id 50 mode active
interface gigabitethernet 0/1/0/8
  bundle id 50 mode active
```

## Configuring Two Bundle EFPs with the Load Balance Options: Example

This example configures the traffic for two bundle EFPs go over the same physical member link.

```
interface Bundle-Ether 50.25 l2transport
  encapsulation dot1q 25
  bundle load-balance hash-select 2
!
interface Bundle-Ether 50.36 l2transport
  encapsulation dot1q 36
  bundle load-balance hash-select 2
```

# 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. Each matching criterion is evaluated to see if the packet matches that criterion. 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 egress policies.

# 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
```

## IP Encapsulation Marking Configuration Example

In this example, a service policy called *policy1* is created. This service policy is associated with a class map called *class1* by using the **class** command. The service policy is attached in the input direction on the interface HundredGigE 0/0/0/24.

```
class-map match-any class1
  match protocol ipv4
  match access-group ipv4 10
  class class-default
end-class-map

policy-map policy1
  class class1
    set ip encapsulation class-of-service 55
  class class-default
end-policy-map
!
interface interface HundredGigE 0/0/0/24
  service-policy input policy1
```

**Guidelines and Limitations:**

- The **set ip encapsulation class-of-service** command is not allowed in egress QoS policies.
- The **set qos-group** action cannot be used in conjunction with the **set ip encapsulation class-of-service** command.
- The IP encapsulation marking (**set ip encapsulation class-of-service**) and the MPLS experimental imposition marking (**set mpls experimental**) features are mutually exclusive. Both configurations are not allowed at the same time.

## 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.

## 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 IEEE 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
```

## 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
!
```

# In-Place Policy Modification: Example

In this example, the precedence is changed from 3 to 5 after the policy is defined and attached to an interface:

Define a class:

```
class-map match-any class1
 match cos 7
 end-class-map
```

Define a policy map that uses the class:

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

Attach the policy map to an interface:

```
interface gigabitethernet 0/6/0/1
 service-policy output policy1
 commit
```

Modify the precedence value of the policy map:

```
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 take effect on all the interfaces to which the policy is attached.

Output from the **show policy-map targets** command indicates that the Gigabit Ethernet interface 0/1/0/0 has one policy map attached as a main policy (as opposed to being attached to a child policy in a hierarchical QoS configuration). Outgoing traffic on this interface is affected if the policy is modified:

```
show policy-map targets

Fri Jul 16 16:38:24.789 DST
1) Policymap: policy1    Type: qos
     Targets (applied as main policy):
       GigabitEthernet0/1/0/0 output
     Total targets: 1

     Targets (applied as child policy):
     Total targets: 0
```

# 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 ASR 9000 Series Aggregation Services Router Gett Guide* |
| QoS commands | *Cisco ASR 9000 Series Aggregation Services Router Mod of Service Command Reference* |
| User groups and task IDs | "Configuring AAA Services on Cisco ASR 9000 Series Ro of Cisco Cisco ASR 9000 Series Aggregation Services Re 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 softw Cisco MIB Locator found at the following URL and choo under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.sht |

# RFCs

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

# Technical Assistance

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