Configuring Modular QoS Service Packet Classification

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

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

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

Release	Modification
Release 5.0.0	This feature was introduced.

Prerequisites for Configuring Modular QoS Packet Classification

These prerequisites are required for configuring modular QoS packet classification on your network:

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

- You must be familiar with Cisco IOS XR QoS configuration tasks and concepts.

Information About Configuring Modular QoS Packet Classification

Packet Classification Overview

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

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

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

Traffic Class Elements

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

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

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

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

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

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

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

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

Table 1: Supported Traffic Class Match Criteria

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

Note

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

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

Traffic Policy Elements

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

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

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

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

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

Limitation

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

Default Traffic Class

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

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

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

Class-based Unconditional Packet Marking Feature and Benefits

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

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

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

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

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

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

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

  For example, weighted random early detection (WRED), a congestion avoidance technique, uses IP precedence values to determine the probability that a packet is dropped. In addition, low-latency queuing (LLQ) can then be configured to put all packets of that mark into the priority queue.

- Use QoS unconditional packet marking to assign MPLS packets to a QoS group. The router uses the QoS group to determine how to prioritize packets for transmission. To set the QoS group identifier on MPLS packets, use the **set qos-group** command in policy map class configuration mode.

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

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

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

Table 2: Supported Class-based Unconditional Packet Marking Operations

Marking Operation	Layer 3 Ingress			Layer 3 Egress		
	Phy	SIf	P-SIf	Phy	SIf	P-Sif
prec	Y	Y	Y	Y	Y	Y
dscp	Y	Y	Y	Y	Y	Y
CoS	N	N	N	Y	Y	Y
qos-group	Y	Y	Y	N	N	N
discard-class	Y	Y	Y	N	N	N
EXP, imposition	Y	Y	Y	N	N	N
EXP, topmost	Y	Y	Y	Y	Y	Y

Note

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

Unconditional Multiple Action Set

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

Unconditional Ingress Markings

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

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

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

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

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

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

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

Unconditional Egress Markings

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

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

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

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

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

Note

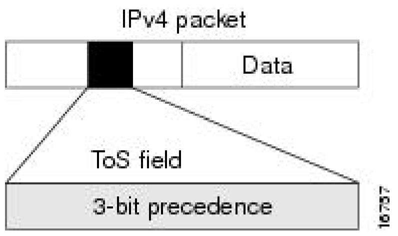
- Both DSCP and precedence packets are mutually exclusive.

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

Specification of the CoS for a Packet with IP Precedence

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

Figure 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, although IP precedence is not a queuing feature, queuing features, such as LLQ, can use the IP precedence setting of the packet to prioritize traffic.

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

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

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

IP Precedence Bits Used to Classify Packets

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

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

Table 3: IP Precedence Values

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

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

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

IP Precedence Value Settings

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

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

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

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

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

IP Precedence Compared to IP DSCP Marking

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

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

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

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

QoS Policy Propagation Using Border Gateway Protocol

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

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

QPPB supports both the IPv4 and IPv6 address-families.

QPPB allows you to classify packets based on:

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

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

Hierarchical Ingress Policing

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

- Parent level
- Child level

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

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

In-Place Policy Modification

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

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

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

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

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

Recommendations for Using In-Place Policy Modification

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

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

Bidirectional Forwarding Detection Echo Packet Prioritization

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

If ingress QOS policy is present on the interface on which BFD echo packets are received and switched back, then the BFD echo packets are marked as vital packets (when received) and all QOS actions of the matching class except for taildrop and WRED are performed on the packets. The packets are then sent to the high priority queue in the ingressq ASIC reserved for transit control traffic, overriding the queue selected by the ingress QOS policy.

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

How to Configure Modular QoS Packet Classification

Creating a Traffic Class

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

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

For conceptual information, see the Traffic Class Elements.

Restrictions

- All **match** commands specified in this configuration task are considered optional, but you must configure at least one match criterion for a class.
- For the **match access-group** command, QoS classification based on the packet length or TTL (time to live) field in the IPv4 and IPv6 headers is not supported.
- For the **match access-group** command, when an ACL list is used within a class-map, the deny action of the ACL is ignored and the traffic is classified based on the specified ACL match parameters.
- The **match discard-class** command is not supported on the Asynchronous Transfer Mode (ATM) interfaces.
- When QoS policy-maps use ACLs to classify traffic, ACEs of ACLs consume some amount of TCAM memory of the line card. Each QoS policy-map for ASR9000 supports up to a maximum of 3072 TCAM IPv4 entries. If you cross the limit, IOS XR fails to apply this policy-map with the insufficient memory available error. If you encounter this error, decrease the number of ACEs in ACLs for the policy-map. This error typically appears when using nested policy-maps, where ACEs in ACLs on different levels are multiplied.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match [not] access-group [ipv4| ipv6]** *access-group-name*
4. **match** [**not**] **cos** [*cos-value*] [*cos-value0 ... cos-value7*]
5. **match destination-address mac** *destination-mac-address*
6. **match source-address mac** *source-mac-address*
7. **match** [**not**] **discard-class** *discard-class-value* [*discard-class-value1 ... discard-class-value6*]

8. **match** [**not**] **dscp** [**ipv4** | **ipv6**] *dscp-value* [*dscp-value* ... *dscp-value*]
9. **match** [**not**] **mpls experimental topmost** *exp-value* [*exp-value1* ... *exp-value7*]
10. **match** [**not**] **precedence** [**ipv4** | **ipv6**] *precedence-value* [*precedence-value1* ... *precedence-value6*]
11. **match** [**not**] **protocol** *protocol-value* [*protocol-value1* ... *protocol-value7*]
12. **match** [**not**] **qos-group** [*qos-group-value1* ... *qos-group-value8*]
13. **match vlan [inner]** *vlanid* [vlanid1 ... vlanid7]
14. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
	Example:
	`RP/0/RP0/CPU0:router(config)# class-map class201`
		Creates a class map to be used for matching packets to the class whose name you specify and enters the class map configuration mode.
		If you specify **match-any**, one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. This is the default. If you specify **match-all**, the traffic must match all the match criteria.

Step 3	**match [not] access-group** [**ipv4**| **ipv6**] *access-group-name* **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match access-group ipv4 map1`	(Optional) Configures the match criteria for a class map based on the specified access control list (ACL) name.
Step 4	**match** [**not**] **cos** [*cos-value*] [*cos-value0* ... *cos-value7*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match cos 5`	(Optional) Specifies a *cos-value* in a class map to match packets. The *cos-value* arguments are specified as an integer from 0 to 7.
Step 5	**match destination-address mac** *destination-mac-address* **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match destination-address mac 00.00.00`	(Optional) Configures the match criteria for a class map based on the specified destination MAC address.
Step 6	**match source-address mac** *source-mac-address* **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match source-address mac 00.00.00`	(Optional) Configures the match criteria for a class map based on the specified source MAC address.
Step 7	**match** [**not**] **discard-class** *discard-class-value* [*discard-class-value1* ... *discard-class-value6*] **Example:**	(Optional) Specifies a *discard-class-value* in a class map to match packets. The *discard-class-value* argument is specified as an integer from 0 to 7. The **match discard-class** command is supported only for an egress policy. The **match discard-class** command is

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cmap)# match discard-class 5`	not supported on the Asynchronous Transfer Mode (ATM) interfaces.
Step 8	**match** [**not**] **dscp** [**ipv4** | **ipv6**] *dscp-value* [*dscp-value ... dscp-value*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match dscp ipv4 15`	(Optional) Identifies a specific DSCP value as a match criterion. • Value range is from 0 to 63. • Reserved keywords can be specified instead of numeric values. • Up to eight values or ranges con be used per match statement.
Step 9	**match** [**not**] **mpls experimental topmost** *exp-value* [*exp-value1 ... exp-value7*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match mpls experimental topmost 3`	(Optional) Configures a class map so that the three-bit experimental field in the topmost Multiprotocol Label Switching (MPLS) labels are examined for experimental (EXP) field values. The value range is from 0 to 7.
Step 10	**match** [**not**] **precedence** [**ipv4** | **ipv6**] *precedence-value* [*precedence-value1 ... precedence-value6*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match precedence ipv4 5`	(Optional) Identifies IP precedence values as match criteria. • Value range is from 0 to 7. • Reserved keywords can be specified instead of numeric values.
Step 11	**match** [**not**] **protocol** *protocol-value* [*protocol-value1 ... protocol-value7*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match protocol igmp`	(Optional) Configures the match criteria for a class map on the basis of the specified protocol.
Step 12	**match** [**not**] **qos-group** [*qos-group-value1 ... qos-group-value8*] **Example:** `RP/0/RP0/CPU0:router(config-cmap)# match qos-group 1 2 3 4 5 6 7 8`	(Optional) Specifies service (QoS) group values in a class map to match packets. • *qos-group-value* identifier argument is specified as the exact value or range of values from 0 to 63. • Up to eight values (separated by spaces) can be entered in one match statement. • **match qos-group** command is supported only for an egress policy.
Step 13	**match vlan [inner]** *vlanid* [vlanid1 ... vlanid7] **Example:**	(Optional) Specifies a VLAN ID or range of VLAN IDs in a class map to match packets. • *vlanid* is specified as an exact value or range of values from 1 to 4094.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cmap)# match vlan vlanid vlanid1`	• Total number of supported VLAN values or ranges is 8.
Step 14	**commit**	

Creating a Traffic Policy

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

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

These class-actions are supported:

- bandwidth—Configures the bandwidth for the class. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.
- police—Police traffic. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.
- priority—Assigns priority to the class. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.
- queue-limit—Configures queue-limit (tail drop threshold) for the class. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.
- random-detect—Enables Random Early Detection. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.
- service-policy—Configures a child service policy.
- set—Configures marking for this class. See the Class-based Unconditional Packet Marking Feature and Benefits.
- shape—Configures shaping for the class. See the Configuring Modular Quality of Service Congestion Management on Cisco IOS XR Software module.

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

For conceptual information, see Traffic Policy Elements.

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

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

SUMMARY STEPS

1. **configure**
2. **policy-map** [**type qos**] *policy-name*

3. **class** class-name
4. **set precedence** [**tunnel**] *precedence-value*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

Step 3	**class** class-name
	Example:
	`RP/0/RP0/CPU0:router(config-pmap)# class class1`
		Specifies the name of the class whose policy you want to create or change.

Step 4	**set precedence** [**tunnel**] *precedence-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set precedence 3`	Sets the precedence value in the IP header. **Note** • A policy configured with the set precedence tunnel command or the set dscp tunnel command can be applied on any Layer 3 interface in the ingress direction.
Step 5	**commit**	

Attaching a Traffic Policy to an Interface

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

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

Prerequisites

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

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/1`	Configures an interface and enters the interface configuration mode.
Step 3	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 4	**commit**	
Step 5	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RP0/CPU0:router# show policy-map interface HundredGigE 0/7/0/1`	(Optional) Displays statistics for the policy on the specified interface.

Configuring Class-based Unconditional Packet Marking

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

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

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

SUMMARY STEPS

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

3. **class** *class-name*
4. **set precedence**
5. **set dscp**
6. **set qos-group** *qos-group-value*
7. **set cos** *cos-value*
8. **set mpls experimental** {**imposition** | **topmost**} *exp-value*
9. **set discard-class** *discard-class-value*
10. **exit**
11. **exit**
12. **interface** *type* interface-path-id
13. **service-policy** {**input** | **output**]} *policy-map*
14. **commit**
15. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

Step 3	**class** *class-name*
	Example:
	`RP/0/RP0/CPU0:router(config-pmap)# class class1`
		Specifies the name of the class whose policy you want to create or change and enters the policy class map configuration mode.

Step 4	**set precedence** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set precedence 1`	Sets the precedence value in the IP header.
Step 5	**set dscp** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set dscp 5`	Marks a packet by setting the DSCP in the ToS byte.
Step 6	**set qos-group** *qos-group-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set qos-group 31`	Sets the QoS group identifiers on IPv4 or MPLS packets. The **set qos-group** command is supported only on an ingress policy.
Step 7	**set cos** *cos-value* **Example:**	Sets the specific IEEE 802.1Q Layer 2 CoS value of an outgoing packet. Values are from 0 to7.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-pmap-c)# set cos 7`	Sets the Layer 2 CoS value of an outgoing packet. • This command should be used by a router if a user wants to mark a packet that is being sent to a switch. Switches can leverage Layer 2 header information, including a CoS value marking. • Packets entering an interface cannot be set with a CoS value.
Step 8	**set mpls experimental** {**imposition** | **topmost**} *exp-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set mpls experimental imposition 3`	Sets the experimental value of the MPLS packet top-most or imposition labels. **Note** The **imposition** keyword can be used only in service policies that are attached in the ingress policy.
Step 9	**set discard-class** *discard-class-value* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# set discard-class 3`	Sets the discard class on IP Version 4 (IPv4) or Multiprotocol Label Switching (MPLS) packets. **Note** This command can be used only in service policies that are attached in the ingress policy.
Step 10	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 11	**exit** **Example:** `RP/0/RP0/CPU0:router(config-pmap)# exit`	Returns the router to XR Config mode.
Step 12	**interface** *type* interface-path-id **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/1`	Configures an interface and enters the interface configuration mode.
Step 13	**service-policy** {**input** | **output**]} *policy-map* **Example:** `RP/0/RP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 14	**commit**	
Step 15	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:**	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# show policy-map interface HundredGigE 0/7/0/1`	

Configuring QoS Policy Propagation Using Border Gateway Protocol

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

Policy Propagation Using BGP Configuration Task List

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

Overview of Tasks

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

- Configure BGP and Cisco Express Forwarding (CEF). To configure BGP, see Routing Command Reference for Cisco NCS 6000 Series Routers . To configure CEF, see IP Addresses and Services Command Reference for Cisco NCS 6000 Series Routers.
- Configure a BGP community list or access list.
- Define the route policy. Set the QoS group ID, based on the BGP community list, BGP autonomous system path, access list, source prefix address or destination prefix address.
- Apply the route policy to BGP.
- Configure QPPB on the desired interfaces .
- Configure and enable a QoS Policy to use the above classification (QoS group ID). To configure committed access rate (CAR), WRED and tail drop, see the Configuring Modular QoS Congestion Avoidance on Cisco IOS XR Software module.

Defining the Route Policy

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

Prerequisites

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

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **set qos-group** *qos-group-value*
4. **commit**

DETAILED STEPS

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

Applying the Route Policy to BGP

This task applies the route policy to BGP.

Prerequisites

Configure BGP and CEF.

SUMMARY STEPS

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

DETAILED STEPS

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

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-bgp-af) # table-policy qppb a1`	
Step 5	**commit**	

Configuring QPPB on the Desired Interfaces

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

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface HundredGigE 0/7/0/1`	Enters interface configuration mode and associates one or more interfaces to the VRF.
Step 3	**ipv4 | ipv6 bgp policy propagation input**{|**qos-group**} {**destination**[{*destination*|*source*}] {**source**[{*destination*|*source*}] **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 bgp policy propagation input qos-group destination`	Enables QPPB on an interface
Step 4	**commit**	

QPPB Scenario

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

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

Configuring Hierarchical Ingress Policing

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config)# policy-map parent`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

Step 3	**class** *class-name*
	Example:
	`RP/0/RP0/CPU0:router(config-pmap)# class class-default`
		Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.

Step 4	**service-policy** *policy-name* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# service-policy child`	Specifies the service-policy as a QoS policy within a policy map.
Step 5	**police rate percent** *percentage* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c)# police rate percent 50`	Configures traffic policing and enters policy map police configuration mode.
Step 6	**conform-action** *action* **Example:** `RP/0/RP0/CPU0:router(config-pmap-c-police)# conform-action transmit`	Configures the action to take on packets that conform to the rate limit. The allowed action is **transmit** that transmits the packets.
Step 7	**exceed-action** *action* **Example:**	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/RP0/CPU0:router(config-pmap-c-police)# exceed-action drop`	
Step 8	**commit**	

Overview of Multiple QoS Policy Support

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

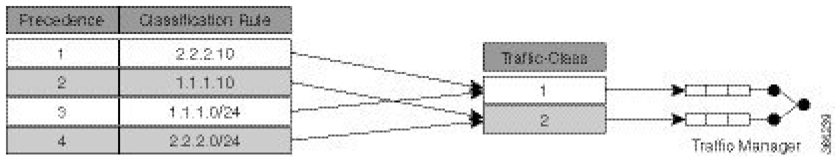
In order to overcome this limitation, the “Multiple QoS Policy Support” feature is introduced. This feature enables the users to apply aggregated actions to various classes of traffic and apply multiple QoS policies on an interface.

Use Case — Multiple QoS Policy Support

Consider a scenario where:

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

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



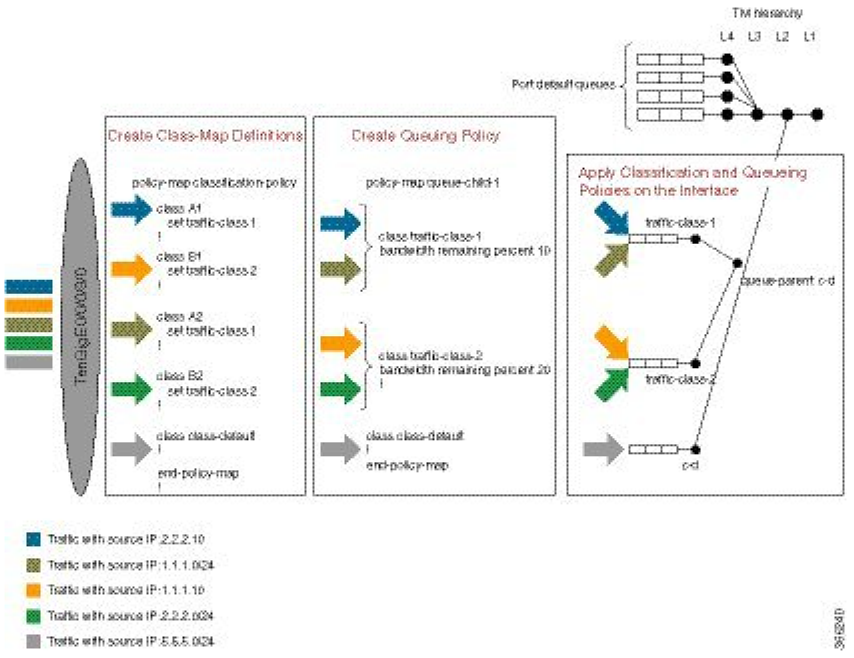
In this example, if the traffic packet matches 2.2.2.10 or 1.1.1.0/24, then the traffic packet is forwarded to the queue that is associated with traffic-class 1. And if the traffic packet matches 1.1.1.10 or 2.2.2.0/24, then the traffic packet is forwarded to the queue that is associated with traffic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

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

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

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

The example below depicts the implementation—



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

Configuring Multiple QoS Policy Support

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

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

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

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

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

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

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

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

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

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

Verification

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

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

```
Router#show qos interface TenGigE 0/0/0/3/0 output
Interface: TenGigE0/0/0/3/0 output
Bandwidth configured: 50000 kbps Bandwidth programed: 50000 kbps
```

```
ANCP user configured: 0 kbps ANCP programed in HW: 0 kbps
Port Shaper programed in HW: 50000 kbps
Policy: queue-parent Total number of classes: 4
----------------------------------------------------------------------
Level: 0 Policy: queue-parent Class: class-default
Matches: traffic-classes : { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
 63,} and no traffic-class
QueueID: N/A
Shape CIR : NONE
Shape PIR Profile : 8 (Grid) Scale: 134 PIR: 49920 kbps  PBS: 624000 bytes
WFQ Profile: 3/9 Committed Weight: 10 Excess Weight: 10
Bandwidth: 0 kbps, BW sum for Level 0: 0 kbps, Excess Ratio: 1
----------------------------------------------------------------------
Level: 1 Policy: queue-child Class: traffic-class-1
Matches: traffic-classes : { 1}
Parent Policy: queue-parent Class: class-default
QueueID: 1040402 (Priority Normal)
Queue Limit: 66 kbytes Abs-Index: 19 Template: 0 Curve: 0
Shape CIR Profile: INVALID
WFQ Profile: 3/19 Committed Weight: 20 Excess Weight: 20
Bandwidth: 0 kbps, BW sum for Level 1: 0 kbps, Excess Ratio: 10
----------------------------------------------------------------------
Level: 1 Policy: queue-child Class: traffic-class-2
Matches: traffic-classes : {2}
Parent Policy: queue-parent Class: class-default
QueueID: 1040403 (Priority Normal)
Queue Limit: 126 kbytes Abs-Index: 29 Template: 0 Curve: 0
Shape CIR Profile: INVALID
WFQ Profile: 3/39 Committed Weight: 40 Excess Weight: 40
Bandwidth: 0 kbps, BW sum for Level 1: 0 kbps, Excess Ratio: 20
----------------------------------------------------------------------
Level: 1 Policy: queue-child Class: class-default
Matches: traffic-classes : { 0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,}
and no traffic-class
Parent Policy: queue-parent Class: class-default
QueueID: 1040404 (Priority Normal)
Queue Limit: 446 kbytes Abs-Index: 52 Template: 0 Curve: 0
Shape CIR Profile: INVALID
WFQ Profile: 3/98 Committed Weight: 139 Excess Weight: 139
Bandwidth: 0 kbps, BW sum for Level 1: 0 kbps, Excess Ratio: 70
----------------------------------------------------------------------
Interface: TenGigE0/0/0/3/0 output
Bandwidth configured: 10000000 kbps Bandwidth programed: 10000000 kbps
ANCP user configured: 0 kbps ANCP programed in HW: 0 kbps
Port Shaper programed in HW: 0 kbps
Policy: classification-policy Total number of classes: 5
----------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: A1
Set traffic-class : 1
QueueID: 0 (Port Default)
Policer Profile: 59 (Single)
Conform: 100000 kbps (100 mbps) Burst: 1250000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
----------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: B1
Set traffic-class : 2
QueueID: 0 (Port Default)
Policer Profile: 60 (Single)
```

```
Conform: 200000 kbps (200 mbps) Burst: 2500000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
----------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: A2
Set traffic-class : 1
QueueID: 0 (Port Default)
Policer Profile: 61 (Single)
Conform: 300000 kbps (300 mbps) Burst: 3750000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
----------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: B2
Set traffic-class : 2
QueueID: 0 (Port Default)
Policer Profile: 62 (Single)
Conform: 400000 kbps (400 mbps) Burst: 5000000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
----------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: class-default
QueueID: 0 (Port Default)
----------------------------------------------------------------------
```

Restrictions for Multiple QoS Policy Support

Policy Classification Restrictions

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

Queuing Policy Restrictions

- Queuing policy can only classify on **traffic-class** field.
    - Valid values for **match traffic-class** are 0-63.

    - Class-maps can match up to 8 discreet traffic-class values or traffic-class ranges.

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

Applying Multiple Services on an Interface Restrictions

- Applying multiple polices is supported only when one policy is a classification policy and the other policy is a queuing policy.
- Applying multiple polices (not more than 2 policies) is supported only in the egress direction. Applying more than 1 policy in the ingress direction is not supported.
- Applying multiple policies is supported only on the following interfaces:
    - Main-interface
    - Sub-interface
    - Bundle interface
    - Bundle sub-interface
- Applying Multi policies is not supported on the following interfaces:
    - PWHE
    - GRE
    - BVI
    - Satellite interfaces
- Multi policies are only supported on Cisco ASR 9000 High Density 100GE Ethernet line cards, Cisco ASR 9000 Enhanced Ethernet line cards, and Cisco ASR 9000 Ethernet line cards.
- The same classification policy cannot be applied with different queuing policies on a different interface of the same line card.
- Classification policy and queuing policy cannot be applied with any of the following feature options
    - account
    - service-fragment-parent
    - 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
No	No	Yes	No	Yes	No	Yes
			Yes	Yes	No	No
			No	No	Yes	No
			No	Yes	Yes	No
			No	Yes	Yes	No
No	Yes	Yes	Any combination			No

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

Multi Policy and Interface Hierarchy

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

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

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

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

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

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

Statistics

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

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

Classification Policy

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

Queuing policy

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

Example: Egress Policy Classification Statistics

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

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

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

Example: Egress Queuing Policy Statistics

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

TenGigE0/0/0/3/9.1 output: queueing
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :           534226989/534226989000         400067
    Transmitted         :           355884870/355884870000         280381
    Total Dropped       :           106961210/106961210000         119726
  Policy queueing-child Class traffic-class-1
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :           178155114/178155114000         200014
      Transmitted         :           178155114/178155114000         200014
      Total Dropped       :                   0/0                    0
    Queueing statistics
      Queue ID                             : 647264
      High watermark                       : N/A
      Inst-queue-len  (packets)            : 0
      Avg-queue-len                        : N/A
      Taildropped(packets/bytes)           : 0/0
      Queue(conform)      :           178155114/178155114000         200014
      Queue(exceed)       :                   0/0                    0
      RED random drops(packets/bytes)      : 0/0
  Policy queueing-child Class traffic-class-2
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :           178098546/178098546000         200111
      Transmitted         :            71137336/71137336000          80385
      Total Dropped       :           106961210/106961210000         119726
    Queueing statistics
      Queue ID                             : 647265
      High watermark                       : N/A
      Inst-queue-len  (packets)            : 1620
      Avg-queue-len                        : N/A
      Taildropped(packets/bytes)           : 106961210/106961210000
      Queue(conform)      :            71137336/71137336000          80385
      Queue(exceed)       :                   0/0                    0
      RED random drops(packets/bytes)      : 0/0
Policy egress-queueing-child Class class-default
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched            :                  0/0                   0
      Transmitted        :                  0/0                   0
      Total Dropped      :                  0/0                   0
    Queueing statistics
      Queue ID                             : 647266
      High watermark                       : N/A
      Inst-queue-len  (packets)            : 0
      Avg-queue-len                        : N/A
      Taildropped(packets/bytes)           : 0/0
      Queue(conform)     :                  0/0                   0
      Queue(exceed)      :                  0/0                   0
      RED random drops(packets/bytes)      : 0/0
```

Restrictions for Statistics

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

Policy Modification

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

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

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

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

Policy Modification Restrictions

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

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

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

Supported Features by Multi Policies

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

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

Configuration Examples for Configuring Modular QoS Packet Classification

Traffic Classes Defined: Example

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

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

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

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

Traffic Policy Created: Example

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

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

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

Traffic Policy Attached to an Interface: Example

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

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

Default Traffic Class Configuration: Example

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

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

class-map match-any Command Configuration: Example

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

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

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

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

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

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

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

Two-Level Hierarchical Traffic Policy Configuration: Example

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

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

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

Class-based Unconditional Packet Marking: Examples

These are typical class-based unconditional packet marking examples:

IP Precedence Marking Configuration: Example

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

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

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

IP DSCP Marking Configuration: Example

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

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

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

  class class2
    set dscp ef
```

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

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

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

QoS Group Marking Configuration: Example

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

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

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

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

Discard Class Marking Configuration: Example

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

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

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

CoS Marking Configuration: Example

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

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

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

MPLS Experimental Bit Imposition Marking Configuration: Example

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

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

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

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

MPLS Experimental Topmost Marking Configuration: Example

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

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

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

QoS Policy Propagation using BGP: Examples

These are the IPv4 and IPv6 QPPB examples:

Applying Route Policy: Example

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

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

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

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

Applying QPPB on a Specific Interface: Example

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

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

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

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

Route Policy Configuration: Examples

QPPB Source Prefix Configuration: Example

This configuration is an example of configuring QPPB source prefix:

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

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

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

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

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

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

QPPB Destination Prefix Configuration: Example

This configuration is an example of QPPB destination prefix:

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

router bgp 100
 bgp router-id 10.10.10.10
 address-family ipv4 unicast
  table-policy qppb-des10to20
 !
 neighbor 201.1.1.2
  remote-as 62100
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.1.2.2
  remote-as 62102
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.1.3.2
  remote-as 62103
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.1.4.2
  remote-as 62104
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 neighbor 201.1.5.2
  remote-as 62105
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !


policy-map p11-in
 class class-qos10
  set precedence priority
  shape average percent 30
 !
 class class-qos11
  set precedence immediate
  shape average percent 10
 !
 class class-qos12
  set precedence flash
  shape average percent 15
 !
 class class-qos13
  set precedence flash-override
  shape average percent 20
 !
 class class-qos14
  set precedence flash
  shape average percent 25
 !
 class class-qos15
  set precedence flash-override
  shape average percent 30
 !
 class class-qos20
  set precedence critical
  shape average percent 40
 !
 class class-default
 !
 end-policy-map
!
policy-map p1-out
 class class-qos10
  set precedence priority
  police rate percent 30
 !
 class class-qos20
  set precedence critical
  police rate percent 40
 !
 class class-default
 !
 end-policy-map
!

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

Hierarchical Ingress Policing: Example

This configuration is an example of typical hierarchical ingress policing:

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

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

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

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

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

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

In-Place Policy Modification: Example

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

Defining a policy map:

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

Attaching the policy map to an interface:

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

Modifying the precedence value of the policy map:

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

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

Additional References

These sections provide references related to implementing packet classification.

Related Documents

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

Standards

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

MIBs

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

RFCs

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

Technical Assistance

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