Classify Packets to Identify Specific Traffic

- Classify Packets to Identify Specific Traffic, on page 1
- Packet Classification Overview, on page 1
- Packet Classification on Your Router, on page 3
- Traffic Class Elements, on page 12
- Default Traffic Class, on page 13
- Create a Traffic Class, on page 13
- Traffic Policy Elements, on page 15
- Create a Traffic Policy, on page 16
- Attach a Traffic Policy to an Interface, on page 16

Classify Packets to Identify Specific Traffic

Read this section to get an overview of packet classification and the different packet classification types for your router.

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.

When traffic descriptors are used to classify traffic, the source agrees to adhere to the contracted terms and the network promises a quality of service. This is where traffic policers and traffic shapers come into the picture. Traffic policers and traffic shapers use the traffic descriptor of a packet—that is, its classification—to ensure adherence to the contract.

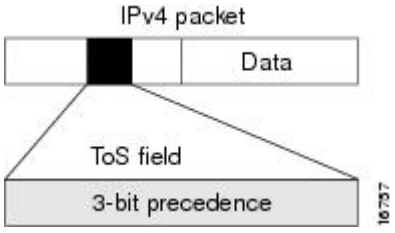
The Modular Quality of Service (QoS) command-line interface (MQC) is used to define the traffic flows that must be classified, where each traffic flow is called a class of service, or class. Later, a traffic policy is created and applied to a class. All traffic not identified by defined classes fall into the category of a default class.

Note From Cisco IOS XR Release 7.2.12 onwards, you can classify packets on Layer 2 transport interfaces using Layer 3 header values. However, this feature applies only to the main interface (physical and bundle interfaces), and not on the sub-interfaces.

Specification of the CoS for a Packet with IP Precedence

Use of IP precedence allows you to specify the CoS for a packet. You can create differentiated service by setting precedence levels on incoming traffic and using them in combination with the QoS queuing features. So that, each subsequent network element can provide service based on the determined policy. IP precedence is usually deployed as close to the edge of the network or administrative domain as possible. This allows the rest of the core or backbone to implement QoS based on precedence.

Figure 1: IPv4 Packet Type of Service Field



You can use the three precedence bits in the type-of-service (ToS) field of the IPv4 header for this purpose. Using the ToS bits, you can define up to eight classes of service. Other features configured throughout the network can then use these bits to determine how to treat the packet in regard to the ToS to grant it. These other QoS features can assign appropriate traffic-handling policies, including congestion management strategy and bandwidth allocation. For example, queuing features such as LLQ can use the IP precedence setting of the packet to prioritize traffic.

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. 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.

Each precedence corresponds to a name. IP precedence bit settings 6 and 7 are reserved for network control information, such as routing updates. These names are defined in RFC 791.

IP Precedence Value Settings

By default, the routers leave 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 and LLQ features can use the IP precedence bits.

IP Precedence Compared to IP DSCP Marking

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

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

Packet Classification on Your Router

On your router, there are two types of packet classification systems:

- In the ingress direction, QoS map and Ternary Content Addressable Memory (TCAM).

Note TCAM is not supported on fixed-configuration routers (where router interfaces are built in). It is supported only on modular routers (that have multiple slots that allow you to change the interfaces on the router).

- In the egress direction, QoS map.

When a policy is matching only on Differentiated Services Code Point (DSCP) or precedence value (also called DSCP or Precedence-based classification), the system selects map-based classification system; else, it selects TCAM.

The TCAM is an extension of the Content Addressable Memory (CAM) table concept. A CAM table takes in an index or key value (usually a MAC address) and looks up the resulting value (usually a switch port or VLAN ID). Table lookup is fast and always based on an exact key match consisting of two input values: 0 and 1 bits.

The QoS map is a table-based classification system for traffic packets.

Improve ACL Scaling Using Peering QoS

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
Improve ACL Scaling Using Peering QoS	Release 7.3.2	This feature merges the functions of QoS and security access control lists (ACLs). This combination enables using the ACL filter with the Object Group ACL, which provides a vastly improved ACL scale due to much lower TCAM usage. Before this functionality was introduced, ACLs applied for QoS group actions consumed a sizeable number of TCAM entries, reducing the available scale of the feature.

Peering QoS is an ingress QoS classification feature that allows you to merge the functions of QoS ACLs and security ACLs. It does so by enabling you to set QoS group actions for every access control entry (ACE) in a security ACL, thus avoiding multiple entries (for QoS and security) per ACE. You can then use this merged ACL with the Object Group ACL feature to apply the ACL filter (permit or deny) for ACEs. Object Group ACLs are also known as 'compressed ACLs' because the object group compresses multiple individual IP addresses into object groups. Also, in an object group-based ACL, you can create a single ACE that uses an object group name instead of creating many ACEs. This ability to 'merge' and 'compress' ACLs saves significant TCAM space and provides a vastly improved ACL scale for QoS policies.

Essential Points about Merging ACLs

- Ensure that you merge the ACLs (set QoS group actions for every ACE in the security ACL) before attaching them to the interface.
- The ACL merge is order-dependent. This means that ACEs are programmed in the order they appear in the ACL.

Guidelines and Restrictions for Peering QoS

- Only Layer 3 interfaces support peering QoS. Configurations on Layer 2 are rejected.
- Peering QoS is supported only in the ingress direction.
- Peering QoS policies and regular QoS policies can coexist on the same line card, but only if you attach them to different interfaces.
- You can attach the same peering QoS policy to multiple interfaces on the same line card.
- To account for IPv4 and IPv6 traffic separately, configure unique QoS group values for IPv4 and IPv6 security ACLs.

- Traffic marked with MPLS EXP bits on a peering QoS-configured interface is matched with **class-default** in a class map configured as **match-any** (default) for MPLS → MPLS flows.
- Subinterfaces inherit peering QoS policies applied to main interfaces, but they don't inherit ACLs. Ensure that you configure security ACLs (matching those on main interfaces) on all subinterfaces; otherwise, all subinterfaces traffic is subjected to **class-default** action, thus affecting their priority weightage.
- You can only configure **match qos-group** for peering QoS policies. Any other **qos-group** command is rejected.
- You can insert, delete, and modify ACEs in security ACLs.

Configuring Peering QoS for ACL Scaling

To configure peering QoS on an interface:

1. Configure security ACL and set **qos-group** per ACE. Else, **qos-group** is set to its default value of 0, affecting the priority weightage for traffic on the interface.
2. Configure peering QoS policy matching on **qos-group** that you set in the security ACL. Set QoS group actions such as remark, policer, traffic-class, DSCP, precedence, and discard-class.
3. Attach the security ACL and peering QoS ACL to the interface.

```
/*Configure security ACL, in this example: ipv4-sec-acl*/
Router(config)#ipv4 access-list ipv4-sec-acl

/*set qos-group per ACE; you can do this because of peering QoS that enables the use of a
single entry per ACE instead of multiple entries */
Router(config-ipv4-acl)#10 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence priority set
qos-group 1
 Router(config-ipv4-acl)#20 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence immediate set
qos-group 2
 Router(config-ipv4-acl)30 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence flash set qos-group
 3
 Router(config-ipv4-acl)40 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence flash-override
set qos-group 4
 Router(config-ipv4-acl)50 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence critical set
qos-group 5
 Router(config-ipv4-acl)#60 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence internet set
qos-group 6
 Router(config-ipv4-acl)#70 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence network set
qos-group 7
Router(config-ipv4-acl)#exit

/*Configure peering QoS policy matching for each qos-group that you set in security ACL*/
Router(config)#class-map match-any grp-7
Router(config-cmap)#match qos-group 7
Router(config-cmap)#end-class-map
Router(config)#class-map match-any grp-6
Router(config-cmap)#match qos-group 6
Router(config-cmap)#end-class-map
Router(config)#class-map match-any grp-5
Router(config-cmap)#match qos-group 5
Router(config-cmap)#end-class-map
Router(config)#class-map match-any grp-4
Router(config-cmap)#match qos-group 4
Router(config-cmap)#end-class-map
Router(config)#class-map match-any grp-3
Router(config-cmap)#match qos-group 3
```

```
Router(config-cmap)#end-class-map
Router(config)#class-map match-any grp-2
Router(config-cmap)#match qos-group 2
Router(config-cmap)#end-class-map
Router(config)#class-map match-any grp-1
Router(config-cmap)#match qos-group 1
Router(config-cmap)#end-class-map
Router(config)#class-map match-any class-default
Router(config-cmap)#end-class-map


/*Set Qos actions in the policy map configured, in this example: set prec, set tc, and set
 dscp*/
Router(config)#policy-map ingress_qosgrp_to_Prec-TC
Router(config-pmap)#class grp-7
Router(config-pmap-c)#set precedence 1
Router(config-pmap-c)#set traffic-class 7
Router(config-pmap-c)#exit
Router(config-pmap)#class grp-6
Router(config-pmap-c)#set precedence 1
Router(config-pmap-c)#set traffic-class 6
Router(config-pmap-c)#exit
Router(config-pmap)#class grp-5
Router(config-pmap-c)#set precedence 2
Router(config-pmap-c)#set traffic-class 5
Router(config-pmap-c)#exit
Router(config-pmap)#class grp-4
Router(config-pmap-c)#set precedence 2
Router(config-pmap-c)#set traffic-class 4
Router(config-pmap-c)#exit
Router(config-pmap)#class grp-3
Router(config-pmap-c)#set traffic-class 3
Router(config-pmap-c)#set dscp ef
Router(config-pmap-c)#exit
Router(config-pmap)#class grp-2
Router(config-pmap-c)#set precedence 3
Router(config-pmap-c)#set traffic-class 2
Router(config-pmap-c)#exit
Router(config-pmap)#class grp-1
Router(config-pmap-c)#set precedence 4
Router(config-pmap-c)#set traffic-class 1
Router(config-pmap-c)#exit
Router(config-pmap)#class class-default
Router(config-pmap-c)#set precedence 5
Router(config-pmap-c)#exit
Router(config-pmap)#end-policy-map

/*Attach the security acl with match qos-groups, to the interface*/
Router(config)#int bundle-Ether 350
Router(config-if)#ipv4 access-group ipv4-sec-acl ingress

/*Attach the policy map with qos actions that you set in the security acl, to the interface*/
Router(config-if)#service-policy input ingress_qosgrp_to_DSCP_TC_qgrp
Router(config-if)#commit
Router(config-if)#exit
```

You have successfully used peering QoS to merge and compress security and QoS ACLs and achieved vastly improved ACL scales for QoS policies.

Running Configuration

```
ipv4 access-list ipv4-sec-acl
  10 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence priority set qos-group 1
  20 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence immediate set qos-group 2
  30 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence flash set qos-group 3
  40 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence flash-override set qos-group 4
  50 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence critical set qos-group 5
  60 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence internet set qos-group 6
  70 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence network set qos-group 7
!
class-map match-any grp-7
 match qos-group 7
 end-class-map
!
class-map match-any grp-6
 match qos-group 6
 end-class-map
!
class-map match-any grp-5
 match qos-group 5
 end-class-map
!
class-map match-any grp-4
 match qos-group 4
 end-class-map
!
class-map match-any grp-3
 match qos-group 3
 end-class-map
!
class-map match-any grp-2
 match qos-group 2
 end-class-map
!
class-map match-any grp-1
 match qos-group 1
 end-class-map
!
class-map match-any class-default
 end-class-map
!
policy-map ingress_qosgrp_to_Prec-TC
 class grp-7
  set precedence 1
  set traffic-class 7
 !
 class grp-6
  set precedence 1
  set traffic-class 6
 !
 class grp-5
  set precedence 2
  set traffic-class 5
 !
 class grp-4
  set precedence 2
  set traffic-class 4
 !
 class grp-3
  set traffic-class 3
  set dscp ef
 !
 class grp-2
```

```
  set precedence 3
  set traffic-class 2
 !
 class grp-1
  set precedence 4
  set traffic-class 1
 !
 class class-default
  set precedence 5
 !
 end-policy-map
!
int bundle-Ether 350
 ipv4 access-group ipv4-sec-acl ingress
!
int bundle-Ether 350
 service-policy input ingress_qosgrp_to_DSCP_TC_qgrp
```

Verification

Run the **show interface** command for the interface to which you attached the security and QoS ACLs.

```
Router#show run int bundle-Ether 350
interface Bundle-Ether350
service-policy input ingress_qosgrp_to_DSCP_TC_qgrp
ipv4 address 11.25.0.1 255.255.255.0
ipv6 address 2001:11:25:1::1/64
ipv4 access-group ipv4-sec-acl ingress
!
```

Peering QoS for ABF

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
ACL-Based Forwarding (ABF) support with peering QoS	Release 7.3.3	This feature gives you the flexibility to configure next-hop addresses for ACEs in the merged (QoS and security) ACL instead of the route that is selected by a routing protocol. You can configure VRF-select or VRF-aware next-hop addresses. This feature enables you to have QoS and ABF functionalities within the same ACEs.

Starting with Release 7.3.3, the Cisco 8000 Series Routers support ACL-based forwarding with peering QoS. ACL-Based Forwarding (ABF) is a policy-based routing feature wherein the router forwards traffic matching specific ACL rules to a user-specified next-hop instead of the route selected by a routing protocol. The Peering QoS feature merges the QoS ACLs and security ACLs to avoid multiple entries (QoS and security) per ACE. In ABF support with peering QoS, you could configure next-hop addresses for ACEs in the merged (QoS and security) ACL. The next-hop address is made use for forwarding the incoming packets matching the permit ACEs to their destination. Here, ABF supports both VRF-select and VRF-aware redirect. In VRF-select, the next-hop consists of VRF only, and the VRF-aware next-hop consists of both VRF and IP addresses.

Configuration

1. Configure security ACL, in this example: abf-acl

```
Router(config)#ipv4 access-list abf-acl
```

2. Set qos-group per ACE; you can do this because of peering QoS that enables the use of a single entry per ACE instead of multiple entries.

```
Router(config-ipv4-acl)#10 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence priority set
qos-group 1 nexthop1 vrf VRF1 nexthop2 vrf VRF2 nexthop3 vrf VRF3
Router(config-ipv4-acl)#20 permit tcp 135.0.0.0/8 217.0.0.0/8 precedence priority set
qos-group 2 nexthop1 vrf vrf3 nexthop2 vrf vrf2
Router(config-ipv4-acl)#30 permit tcp 135.0.0.0/8 217.0.0.0/8 match-all +ack +psh set
qos-group 3 nexthop1 vrf vrf2 nexthop2 vrf vrf3 nexthop3 vrf vrf1
Router(config-ipv4-acl)#40 permit tcp 135.0.0.0/8 217.0.0.0/8 precedence priority set
qos-group 4 nexthop1 vrf vrf1 nexthop2 vrf vrf2 nexthop3 vrf vrf3
Router(config-ipv4-acl)#exit
```

3. Configure peering QoS policy matching for each qos-group that you set in security ABF ACL.

```
Router(config)#class-map match-any grp-4
Router(config-cmap)#match qos-group 4
Router(config-cmap)#end-class-map
Router(config)#class-map match-any grp-3
Router(config-cmap)#match qos-group 3
Router(config-cmap)#end-class-map
Router(config)#class-map match-any grp-2
Router(config-cmap)#match qos-group 2
Router(config-cmap)#end-class-map
Router(config)#class-map match-any grp-1
Router(config-cmap)#match qos-group 1
Router(config-cmap)#end-class-map
Router(config)#class-map match-any class-default
Router(config-cmap)#end-class-map
```

4. Set QoS actions in the policy map configured, in this example: set prec, set tc, and set dscp

```
Router(config)#policy-map edge_qos_policy
Router(config-pmap)#class grp-4
Router(config-pmap-c)#set precedence 2
Router(config-pmap-c)#set traffic-class 4
Router(config-pmap-c)#exit
Router(config-pmap)#class grp-3
Router(config-pmap-c)#set traffic-class 3
Router(config-pmap-c)#set dscp ef
Router(config-pmap-c)#exit
Router(config-pmap)#class grp-2
Router(config-pmap-c)#set precedence 3
Router(config-pmap-c)#set traffic-class 2
Router(config-pmap-c)#exit
Router(config-pmap)#class grp-1
Router(config-pmap-c)#set precedence 4
Router(config-pmap-c)#set traffic-class 1
Router(config-pmap-c)#exit
Router(config-pmap)#class class-default
Router(config-pmap-c)#set precedence 5
Router(config-pmap-c)#exit
Router(config-pmap)#end-policy-map
```

5. Attach the security acl with set qos-groups, to the interface.

```
Router(config)#int bundle-Ether 350
Router(config-if)#ipv4 access-group abf-acl ingress
```

6. Attach the policy map with QoS actions that you set in step 4, to the interface.

```
Router(config-if)#service-policy input edge_qos_policy
Router(config-if)#commit
Router(config-if)#exit
```

You have successfully configured peering QoS with ABF.

Running Configuration

```
ipv4 access-list abf-acl
10 permit ipv4 135.0.0.0/8 217.0.0.0/8 precedence priority set qos-group 1 nexthop1 vrf
VRF1 nexthop2 vrf VRF2 nexthop3 vrf VRF3
20 permit tcp 135.0.0.0/8 217.0.0.0/8 precedence priority set qos-group 2 nexthop1 vrf vrf3
 nexthop2 vrf vrf2
30 permit tcp 135.0.0.0/8 217.0.0.0/8 match-all +ack +psh set qos-group 3 nexthop1 vrf vrf2
 nexthop2 vrf vrf3 nexthop3 vrf vrf1
40 permit tcp 135.0.0.0/8 217.0.0.0/8 precedence priority set qos-group 4 nexthop1 vrf vrf1
 nexthop2 vrf vrf2 nexthop3 vrf vrf3
!
class-map match-any grp-4
match qos-group 4
end-class-map
!
class-map match-any grp-3
match qos-group 3
end-class-map
!
class-map match-any grp-2
match qos-group 2
end-class-map
!
class-map match-any grp-1
match qos-group 1
end-class-map
!
class-map match-any class-default
end-class-map
!
policy-map edge_qos_policy
class grp-4
set precedence 2
set traffic-class 4
!
class grp-3
set traffic-class 3
set dscp ef
!
class grp-2
set precedence 3
set traffic-class 2
!
class grp-1
set precedence 4
set traffic-class 1
!
class class-default
set precedence 5
!
end-policy-map
```

```
!
int bundle-Ether 350
ipv4 access-group abf-acl ingress
!
int bundle-Ether 350
service-policy input edge_qos_policy
```

Verification

Run the **show interface** command for the interface to which you attached the security and QoS ABF ACLs.

```
Router#show run int bundle-Ether 350
interface Bundle-Ether350
service-policy input edge_qos_policy
ipv4 address 11.25.0.1 255.255.255.0
ipv6 address 2001:11:25:1::1/64
ipv4 access-group abf-acl ingress
!
```

Classify and Remark Layer 3 Header on Layer 2 Interfaces

When you need to mark packets for Layer 2 interface traffic that flows across bridge domains and bridge virtual interfaces (BVIs), you can create a mixed QoS policy. This policy has both map-based and TCAM-based classification class-maps. The mixed policy ensures that both bridged (Layer 2) and Bridge Virtual Interface (BVI, or Layer 3) traffic flows are classified and remarked.

Guidelines

- A class-map with TCAM classification may not match bridged traffic. TCAM entries match only routed traffic while map entries match both bridged and BVI traffic.
- A class-map with map-based classification matches both bridged and BVI traffic.

Example

```
ipv4 access-list acl_v4
10 permit ipv4 host 100.1.1.2 any
20 permit ipv4 host 100.1.100.2 any
ipv6 access-list acl_v6
10 permit tcp host 50:1:1::2 any
20 permit tcp any host 50:1:200::2
class-map match-any c_match_acl
match access-group ipv4 acl_v4 ! This entry does not match bridged traffic
match access-group ipv6 acl_v6 ! This entry does not match bridged traffic
match dscp af11 This entry matches bridged and BVI traffic
class-map match-all c_match_all
match protocol udp ! This entry does not match bridged traffic
match prec 7
class-map match-any c_match_protocol
match protocol tcp ! This entry, and hence this class does not match bridged traffic
class-map match-any c_match_ef
match dscp ef ! This entry/class matches bridged and BVI traffic
class-map match-any c_qosgroup_1 This class matches bridged and BVI traffic
!
match qos-group 1
policy-map p_ingress
class c_match_acl
set traffic-class 1
set qos-group 1
```

```
!
class c_match_all
set traffic-class 2
set qos-group 2
!
class c_match_ef
set traffic-class 3
set qos-group 3
!
class c_match_protocol
set traffic-class 4
set qos-group 4
policy-map p_egress
class c_qosgroup_1
set dscp af23
interface FourHundredGigE0/0/0/0
l2transport
service-policy input p_ingress
service-policy output p_egress
!
!
interface FourHundredGigE0/0/0/1
ipv4 address 200.1.2.1 255.255.255.0
ipv6 address 2001:2:2::1/64
service-policy input p_ingress
service-policy output p_egress
```

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 - to specify various criteria for classifying packets.
- An instruction on how to evaluate these **match** commands (if more than one **match** command exists in the traffic class)

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.

This table shows the details of match types supported on the router.

Match Type Supported	Min, Max	Max Entries	Support for Match NOT	Support for Ranges	Direction Supported on Interfaces
IPv4 DSCP IPv6 DSCP	(0,63)	64	Yes	Yes	Ingress
DSCP					Egress

Match Type Supported	Min, Max	Max Entries	Support for Match NOT	Support for Ranges	Direction Supported on Interfaces
IPv4 Precedence IPv6 Precedence	(0,7)	8	Yes	No	Ingress
Precedence					Egress
MPLS Experimental Topmost	(0,7)	8	Yes	No	Ingress Egress
Access-group	Not applicable	8	No	Not applicable	Ingress
QoS-group	(1,7)	7	No	No	Egress
Protocol	(0, 255)	1	Yes	Not applicable	Ingress

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.

For egress classification, match on **qos-group** (1-7) is supported. Match **qos-group 0** cannot be configured. The class-default in the egress policy maps to **qos-group 0**.

This example shows how to configure a traffic policy for the default class:

```
configure
 policy-map ingress_policy1
 class class-default
  police rate percent 30
 !
```

Create 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 **match** commands in class-map configuration mode, as needed.

Guidelines

- 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.

- Use the **not** keyword with the **match** command to perform a match based on the values of a field that are not specified.
- All **match** commands specified in this configuration task are considered optional, but you must configure at least one match criterion for a class.
- 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.
- 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 qos-group**, **traffic-class**, **DSCP/Prec**, and **MPLS EXP** are supported only in egress direction, and these are the only match criteria supported in egress direction
- The egress default class implicitly matches **qos-group** 0.
- Multicast takes a system path that is different than unicast on router, and they meet later on the egress in a multicast-to-unicast ratio of 20:80 on a per interface basis. This ratio is maintained on the same priority level as that of the traffic.
- Egress QoS for multicast traffic treats traffic classes 0-5 as low-priority and traffic classes 6-7 as high priority. Currently, this is not user-configurable.
- Egress shaping does not take effect for multicast traffic in the high priority (HP) traffic classes. It only applies to unicast traffic.
- If you set a traffic class at the ingress policy and do not have a matching class at egress for the corresponding traffic class value, then the traffic at ingress with this class will not be accounted for in the default class at the egress policy map.
- Only traffic class 0 falls in the default class. A non-zero traffic class assigned on ingress but with no assigned egress queue, falls neither in the default class nor any other class.

Configuration Example

You have to accomplish the following to complete the traffic class configuration:

1. Creating a class map
2. Specifying the match criteria for classifying the packet as a member of that particular class

   (For a list of supported match types, see Traffic Class Elements, on page 12.)

```
Router# configure
Router(config)# class-map match-any qos-1
Router(config-cmap)# match qos-group 1
Router(config-cmap)# end-class-map
Router(config-cmap)# commit
```

Use this command to verify the class-map configuration:

```
Router#show class-map qos-1
1) ClassMap: qos-1    Type: qos
    Referenced by 2 Policymaps
```

Also see, Attach a Traffic Policy to an Interface, on page 16.

Related Topics

- Traffic Class Elements, on page 12
- Traffic Policy Elements, on page 15

Traffic Policy Elements

A traffic policy contains three elements:

- Name
- Traffic class
- QoS policies

After choosing the traffic class that is used to classify traffic to the traffic policy, the user can enter the QoS features to be applied to the classified traffic.

The MQC does not necessarily require that the users associate only one traffic class to one traffic policy.

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 router supports 8 classes per policy-map in the ingress direction and 8 classes per policy-map in the egress direction.

This table shows the supported class-actions on the router.

Supported Action Types	Direction supported on Interfaces
bandwidth-remaining	egress
mark	See Packet Marking
police	ingress
priority	egress (level 1 to level 7)
queue-limit	egress
shape	egress
red	egress

RED supports the **discard-class** option; the only values to be passed to the discard-class being 0 and 1.

Create a Traffic Policy

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.

To configure a traffic class, see Create a Traffic Class, on page 13.

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. With dual policy support, you can have two traffic policies, one marking and one queuing attached at the output. See, Attach a Traffic Policy to an Interface, on page 16.

Configuration Example

You have to accomplish the following to complete the traffic policy configuration:

1. Creating a policy map that can be attached to one or more interfaces to specify a service policy
2. Associating the traffic class with the traffic policy
3. Specifying the class-action(s) (see Traffic Policy Elements, on page 15)

```
Router# configure
Router(config)# policy-map  test-shape-1
Router(config-pmap)# class qos-1

/* Configure class-action ('shape' in this example).
Repeat as required, to specify other class-actions */
Router(config-pmap-c)# shape average percent 40
Router(config-pmap-c)# exit

/* Repeat class configuration as required, to specify other classes */

Router(config-pmap)# end-policy-map
Router(config)# commit
```

Related Topics

- Traffic Policy Elements, on page 15
- Traffic Class Elements, on page 12

Attach a Traffic Policy to an Interface

After the traffic class and the traffic policy are created, you must attach the traffic policy to interface, and specify the direction in which the policy should be applied.

Note Hierarchical policies are not supported.

When a policy-map is applied to an interface, the transmission rate counter of each class is not accurate. This is because the transmission rate counter is calculated based on the exponential decay filter.

Configuration Example

You have to accomplish the following to attach a traffic policy to an interface:

1. Creating a traffic class and the associated rules that match packets to the class (see Create a Traffic Class, on page 13)
2. Creating a traffic policy that can be attached to one or more interfaces to specify a service policy (see Create a Traffic Policy, on page 16)
3. Associating the traffic class with the traffic policy
4. Attaching the traffic policy to an interface, in the ingress or egress direction

```
Router# configure
Router(config)#  interface fourHundredGigE 0/0/0/2
Router(config-int)# service-policy output strict-priority
Router(config-int)# commit
```

Running Configuration

```
/* Class-map configuration */

class-map match-any traffic-class-7
 match traffic-class 7
 end-class-map

!class-map match-any traffic-class-6
 match traffic-class 6
 end-class-map

class-map match-any traffic-class-5
 match traffic-class 5
 end-class-map

class-map match-any traffic-class-4
 match traffic-class 4
 end-class-map

class-map match-any traffic-class-3
 match traffic-class 3

class-map match-any traffic-class-2
 match traffic-class 2
 end-class-map

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

```
/* Traffic policy configuration */

policy-map test-shape-1
 class traffic-class-1
  shape average percent 40
 !

policy-map strict-priority
 class tc7
  priority level 1
  queue-limit 75 mbytes
 !
 class tc6
  priority level 2
  queue-limit 75 mbytes
 !
 class tc5
  priority level 3
  queue-limit 75 mbytes
 !
 class tc4
  priority level 4
  queue-limit 75 mbytes
 !
 class tc3
  priority level 5
  queue-limit 75 mbytes
 !
 class tc2
  priority level 6
  queue-limit 75 mbytes
 !
 class tc1
  priority level 7
  queue-limit 75 mbytes
 !
 class class-default
  queue-limit 75 mbytes
 !
 end-policy-map

- - -
- - -

/* Attaching traffic policy to an interface in egress direction */
interface fourHundredGigE 0/0/0/2
 service-policy output strict-priority
 !
```

Verification

```
Router# #show qos int fourHundredGigE 0/0/0/2 output

NOTE:- Configured values are displayed within parentheses Interface FourHundredGigE0/0/0/2
 ifh 0xf0001c0  -- output policy

NPU Id:                        0
Total number of classes:       8
Interface Bandwidth:           400000000 kbps
Policy Name:                   strict-priority
```

```
VOQ Base:                       2400
Accounting Type:                Layer1 (Include Layer 1 encapsulation and above)
------------------------------------------------------------------------------
Level1 Class (HP1)                          =   tc7
Egressq Queue ID                            =   2407 (HP1 queue)
Queue Max. BW.                              =   no max (default)
TailDrop Threshold                          =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP2)                          =   tc6
Egressq Queue ID                            =   2406 (HP2 queue)
Queue Max. BW.                              =   no max (default)
TailDrop Threshold                          =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP3)                          =   tc5
Egressq Queue ID                            =   2405 (HP3 queue)
Queue Max. BW.                              =   no max (default)
TailDrop Threshold                          =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP4)                          =   tc4
Egressq Queue ID                            =   2404 (HP4 queue)
Queue Max. BW.                              =   no max (default)
TailDrop Threshold                          =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP5)                          =   tc3
Egressq Queue ID                            =   2403 (HP5 queue)
Queue Max. BW.                              =   no max (default)
TailDrop Threshold                          =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP6)                          =   tc2
Egressq Queue ID                            =   2402 (HP6 queue)
Queue Max. BW.                              =   no max (default)
TailDrop Threshold                          =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class (HP7)                          =   tc1
Egressq Queue ID                            =   2401 (HP7 queue)
Queue Max. BW.                              =   no max (default)
TailDrop Threshold                          =   74999808 bytes / 2 ms (75 megabytes)
WRED not configured for this class

Level1 Class                                =   class-default
Egressq Queue ID                            =   2400 (Default LP queue)
Queue Max. BW.                              =   no max (default)
Inverse Weight / Weight                     =   1 / (BWR not configured)
TailDrop Threshold                          =   74999808 bytes / 150 ms (75 megabytes)
WRED not configured for this class

!
```

Related Topics

- Traffic Policy Elements, on page 15
- Traffic Class Elements, on page 12