Configuring QoS on Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2)

- Prerequisites for QoS, on page 1
- Restrictions for QoS on Wired Targets, on page 2
- Restrictions for Egress Queuing Policy, on page 4
- Information About QoS, on page 6
- How to Configure QoS, on page 32
- Monitoring QoS, on page 65
- Configuration Examples for QoS, on page 66
- Where to Go Next, on page 80
- Additional References for QoS, on page 80
- Feature History for QoS, on page 81

Prerequisites for QoS

Before configuring standard Quality of Service (QoS), you must have a thorough understanding of these items:

- Standard QoS concepts.
- Classic Cisco IOS QoS.
- Understanding of QoS implementation.
- Understanding of Virtual output queueing (V0Q) architecture.
- Types of applications used and the traffic patterns on your network.
- Traffic characteristics and needs of your network. For example, is the traffic on your network bursty? Do you need to reserve bandwidth for voice and video streams?
- Bandwidth requirements and speed of the network.
- Location of congestion points in the network.

Restrictions for QoS on Wired Targets

A target is an entity where a policy is applied, and a wired target can be a port.

The following are the restrictions for applying QoS features on a device for a wired target:

- For a type queueing policymap, a maximum of eight queuing classes are supported on a device port for a wired target.
- For a nonqueueing policymap, policer is only supported in ingress. Up to 32 policers are supported per ingress policymap per port. Each policer has counter for conform, exceed, and violate.
- A maximum of only 1599 policymaps can be created.
- A total of eight priority levels (0 to 7) are supported, but only one priority level can be configured for each classmap. A class without priority level configured is priority level 0, which is referred to as nonpriority class.
- In a hierarchical policy, overlapping actions between parent and child are not allowed, except when a policy has the port shaper in the parent policy and the queuing features in the child policy.
- For hierarchical QoS (H-QoS) applied in the ingress direction, policing in both the parent and the child is not supported in a QoS hierarchy.
- Marking in both the parent and the child is not supported in a QoS hierarchy.
- On Cisco Catalyst C9600X, QOS shaping is not supported for less than 1.35G. Attempting to configure a shaping rate less than 1.35 Gbps results in the console displaying the error message:

  ```
  Per-port Shape rate must be greater than 1.35G.
  ```

- Empty classes are supported.
- The conform action must be transmit under a policer within a policy map.
- Marking action is not supported in the egress type queueing policy. To support remark on egress, a new policy map must be attached, which can be based on DSCP or PREC or CoS or MPLS EXP or QoS-group, but not based on ACL.
- For Generic Routing Encapsulation (GRE) tunnel interface, a policymap can be attached only to physical members, and not on the logical tunnel interface.
- Classification counters have the following specific restrictions:
  - Classification counters count packets instead of bytes.
  - Filter-based classification counters are not supported.
  - Only QoS configurations with marking or policing can trigger the classification counter.
  - The classification counter is only port based, and aggregation is not performed.
  - As long as there is policing or marking action in the policy, the class will have classification counters.
  - When there are multiple match statements in a class, the traffic counter is cumulative for all the match statements in the class.
  - A class without policer in ingress or egress policymap is assigned to a classification counter. For an ingress policy, 32 unique counters per port are supported, and for egress policy, eight unique

counters per port are supported. If an ingress or egress policy has more than 32/8 classes without policers, the extra classes will share the same match counter. A maximum of 256 classes are supported per policy on the wired port for the wired target.

- The device supports 15 unique combinations of policer exceed markdown and policer violate markdown tables. A policymap must use the same combination, which means different classes in the same policymap must use the same table map for exceed markdown. The same is applicable for violate markdown.
- Overlapping and marking actions are not supported in H-QoS policy.
- Policer value can only be configured in either the parent or the child, not both.
- Queuing actions are supported only on DSCP, CoS, QoS-group, IP precedence, and EXP based classification.
- Application Visibility and Control (AVC) and Network Based Application Recognition (NBAR) based QoS are not supported.
- Classification is not supported for the following:
  - Virtual Private LAN Services (VPLS)
  - Layer 2 and MAC
  - Packet length for fixed and range
  - Real-Time Transport Protocol (RTP) for header and type
  - Access control entries (ACEs)
- Conditional markdown using multiple table-maps in the same policy-map is not supported.
- GRE tunnel QoS for policing and marking is not supported.
- Locator ID Separation Protocol (LISP) QoS is not supported.
- QoS ACL ternary content addressable memory (TCAM) for egress is not supported.
- QoS metadata for App-ID entries are not supported.
- Security group tag (SGT) aware QoS is not supported.
- StackWise Virtual Link (SVL) QOS is not supported.
- Policing in egress direction is not supported.
- Policing and queueing are not supported on SVI and tunnel interfaces.
- Object group-based ACLs with QoS ACL based classification are not supported.
- Remark with ACL based classification is not supported in egress direction.
- For Broadcast, Unknown unicast and Multicast (BUM) traffic, QoS Queue statistics visibility is not supported in **show policy-map type queue interface** command.
- Only 2 output queuing statistics can be viewed for multicast traffic.
- Only 8 queuing per port are supported for queuing of traffic.

- Queuing policy-map can be classified only using traffic-class, and the set option in queuing policy-map is not supported.
- Classification using IPv6 based ACE using tcp flag in it is not supported
- Egress remarking or set option is supported based on DSCP, CoS, precedence, or MPLS EXP. A new policy-map must be defined and applied on the interface in egress direction. In case of MPLS, the egress remarking policy on label imposition node works only if there is an EXP based marking policy on the ingress interface.
- The table map should be of the same QoS type tag. For example, table map from DSCP to CoS and CoS to precedence are not supported.
- Aggregate policing for egress is not supported.
- Policing is not supported for outgoing packets.
- Control-plane policy packet counter statistics update is not supported.
- For hierarchical QoS (HQoS) police applied in the ingress direction, child-level policing is not supported. Policing of traffic is based on the police rate value defined in parent-level policing.
- Egress police with QoS group classification along with DSCP/PREC/COS/EXP based classification are not supported.

The following are the restrictions and considerations for applying QoS features on EtherChannel and channel member interfaces:

- Queuing policy is supported only on EtherChannel member ports, and nonqueuing policy is supported only on EtherChannel interface.
- When a nonqueuing service policy is applied to channel members, it gets rejected with a message:

```
Only queueing type policy is supported on member interfaces.
```

- Auto QoS is supported on EtherChannel member ports.

Restrictions for Egress Queuing Policy

- Each class-map matches only one of the traffic classes, the range of which is 0 to 7. **class-default** always matches traffic class 0. Also, no other class can match traffic class 0.
- If there is no **set traffic-class** in the ingress policy map, the QoS tag value is mapped to traffic class 0 or 7 by default.
- To select queues that are different from the system default mapping, configure the traffic class using the ingress policy map. The traffic class that is configured affects VOQ selection for all the egress ports.
- Each unique combination of traffic-classes in a type queuing policy-map requires a separate traffic class profile. The number of traffic class profiles are limited to eight for main interfaces.
- On Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2) statistics for multicast and Layer 2 traffic dropped by queuing police, due to exceeded shape limit value, are not displayed in the total packet drop statistics on the interface. It displays statistics only for unicast traffic.

- Each priority level, when configured, must be configured to the class that matches the corresponding traffic class as shown in the following table:

Note This is applicable only to priority level-based queuing policy map.

Table 1: Priority Level to Traffic Class Mapping

Priority Level	Traffic Class
P1 (highest)	7
P2	6
P3	5
P4	4
P5	3
P6	2
P7	1
None (lowest)	0

- Each priority level—ranging from 1 to 7—can be configured to one class only. That is, there cannot be more than one class at the same priority level. Each class that does not have a priority level configured, is referred to as priority normal. Note that there can be multiple priority normals.
- Priority level must start from 1. Therefore, if there is only one priority class, it must be priority level 1.
- If all the priority levels that are configured in a policy map are sorted, they must be contiguous. In other words, you cannot skip a priority level. For example, P1, P2, P4 (skipping P3), is not allowed. This condition can be met after adding or removing class from a type queueing policy map.
- On Cisco Catalyst 9500X Series Switches, you can not attach a class-map that matches both, DSCP and MPLS experimental bit (EXP), to an ingress policy map.
- Only these actions are supported in the queuing policy:
  - Priority
  - Shape

Note The set option is not available in the queueing policy map.

  - Bandwidth remaining ratio
  - Queue-limit
  - Random Early Detection (RED)

- Match

Information About QoS

The following sections provide information about QoS.

QoS Components

QoS consists of the following key components:

- Classification: This is the process of distinguishing one type of traffic from another based upon access control lists (ACLs), Differentiated Services Code Point (DSCP), Class of Service (CoS), and other factors.
- Marking and mutation: Marking is used on traffic to convey specific information to a downstream device in the network, or to carry information from one interface in a device to another. When traffic is marked, QoS operations on that traffic can be applied. This can be accomplished directly using the **set** command or through a table map, which takes input values and translates them directly to values in the output.
- Shaping: This is the process of imposing a maximum rate of traffic while regulating the traffic rate in such a way that downstream devices are not subjected to congestion. Shaping in the most common form is used to limit the traffic sent from a physical or logical interface.
- Policing: This is used to impose a maximum rate on a traffic class. If the rate is exceeded, then a specific action is taken as soon as the event occurs.

Note Policing of traffic is supported only in the ingress direction.

- Queuing: This is used to prevent traffic congestion. Traffic is sent to specific queues for servicing and scheduling, based upon bandwidth allocation. Traffic is then scheduled or sent out through the port. Traffic class is used as the classification value for queueing.

QoS Terminology

The following terms are used interchangeably in this QoS configuration guide:

- Upstream (direction towards the device) is the same as ingress.
- Downstream (direction from the device) is the same as egress.

Information About QoS

A default policy is always present for the device, and the traffic-class and discard-class are set based on the default policy. Without QoS, the device offers best-effort service for each packet, regardless of the packet contents or size.

The following are specific features provided by QoS:

- Low latency
- Bandwidth guarantee
- Buffering capabilities and dropping disciplines
- Traffic policing
- Enables the changing of the attribute of the frame or packet header
- Relative services

Modular QoS CLI

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

Virtual Output Queuing

In traditional methods of traffic management, traffic is sent to the egress output queues without taking into consideration the egress interface availability to transmit. Therein lies the problem as well. In case of traffic congestion, traffic may get dropped at the egress port. That means the network resources spent getting the packets from the ingress input queue across the switch fabric to the output queues at egress are wasted. That is not all—the same input queue buffers traffic meant for different egress ports, so congestion on one egress port could affect traffic on another port, an event referred to as head-of-line-blocking. Virtual output queueing (VOQ) resolves this problem.

In a VOQ architecture, a separate virtual queue for each egress port is maintained for the physical buffer of each input port. That is, unlike in the traditional method where one input queue will direct traffic to multiple output queues, every output queue will have a dedicated virtual queue. Therefore, in case of congestion, virtual queue only for that particular egress port will be blocked and transmitted only when the egress port has enough resources.

Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2) uses VOQ architecture in accordance with the Cisco IOS Modular QoS CLI (MQC).

Supported QoS Features for Wired Access

The following table describes the supported QoS features for wired access.

Table 2: Supported QoS Features for Wired Access

Feature	Description
Supported targets	• 10-Gigabit Ethernet • 40-Gigabit Ethernet • 25-Gigabit Ethernet • 100-Gigabit Ethernet • EtherChannel member ports **Note** Queuing is supported only on EtherChannel member ports and policing is supported on port-channel interface. • SVI **Note** It is supported only in Layer 2 member ports on Cisco Catalyst 9600 Series Supervisor 2 Module (C9600-SUP- 2).
Configuration sequence	QoS policy installed using the **service-policy** command.
Supported number of queues at port level	Up to eight queues supported on a port.

Supported classification mechanism	• DSCP
	• IP precedence
	• MPLS experimental bits (EXP)
	• CoS
	• QoS group
	• ACL membership including:
	  • IPv4 ACLs
	  • IPv6 ACLS

Hierarchical QoS

H-QoS allows you to perform:

- Hierarchical classification: Traffic classification is based upon traffic class.

- Hierarchical shaping: Shaping can also be configured at multiple levels in the hierarchy.

Note For the parent you only have a configuration class default, and the only action for the class default, is shaping.

QoS Implementation

Typically, networks operate on a best-effort delivery basis, which means that all traffic has equal priority and an equal chance of being delivered in a timely manner. When congestion occurs, all traffic has an equal chance of being dropped.

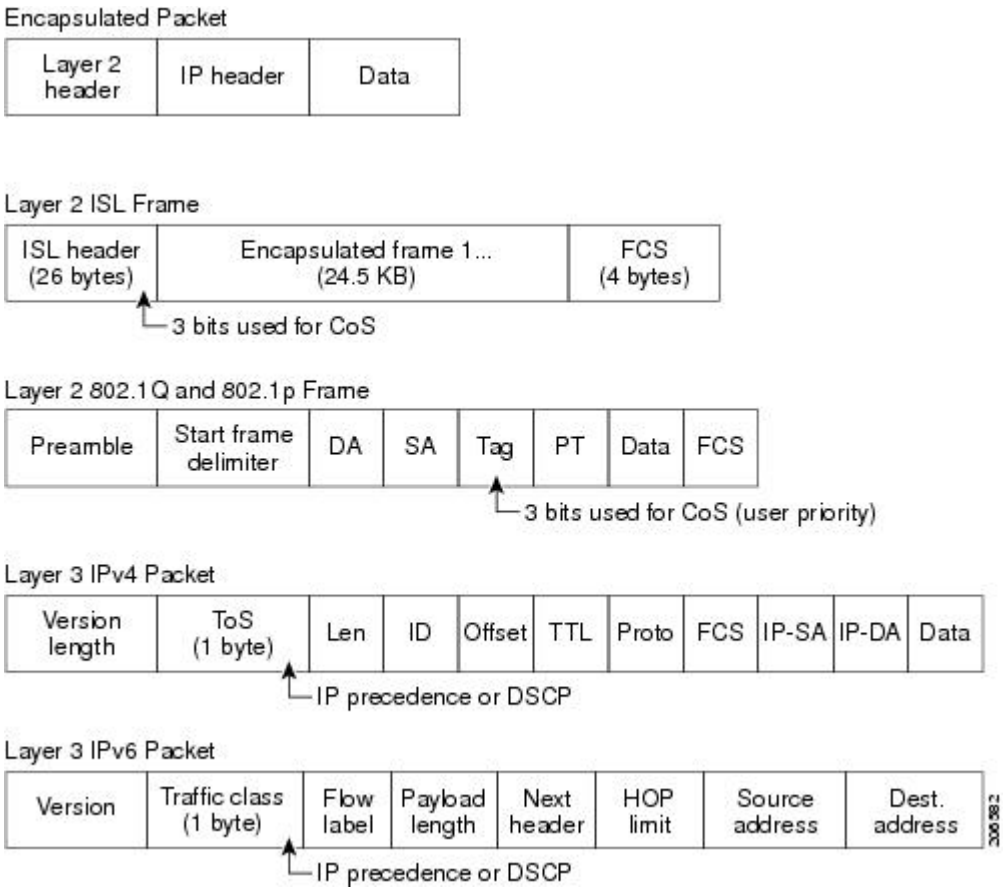
When you configure the QoS feature, you can select specific network traffic, prioritize it according to its relative importance, and use congestion-management and congestion-avoidance techniques to provide preferential treatment. Implementing QoS in your network makes network performance more predictable and bandwidth utilization more effective.

QoS implementation is based on the Differentiated Services (Diff-Serv) architecture, a standard from the Internet Engineering Task Force (IETF). This architecture specifies that each packet is classified upon entry into the network.

The classification is carried in the IP packet header, using six bits from the deprecated IP type of service (ToS) field to carry the classification (*class*) information. Classification can also be carried in the Layer 2 frame.

The special bits in the Layer 2 frame or a Layer 3 packet are shown in the following figure:

Figure 1: Classification Layers of QoS in Frames and Packets



Layer 2 Frame Prioritization Bits

Layer 2 Inter-Switch Link (ISL) frame headers have a 1-byte User field that carries an IEEE 802.1p class of service (CoS) value in the three least-significant bits. On ports configured as Layer 2 ISL trunks, all the traffic is in ISL frames.

Layer 2 802.1Q frame headers have a 2-byte Tag Control Information field that carries the CoS value in the three most-significant bits, which are called User Priority bits. On ports configured as Layer 2 802.1Q trunks, all the traffic is in 802.1Q frames, except for the traffic in the native VLAN.

Other frame types cannot carry Layer 2 CoS values.

Layer 2 CoS values range from 0 for low priority to 7 for high priority.

Layer 3 Packet Prioritization Bits

Layer 3 IP packets can carry either an IP precedence value or a Differentiated Services Code Point (DSCP) value. QoS supports the use of either value because DSCP values are backward-compatible with IP precedence values.

IP precedence values range from 0 to 7. DSCP values range from 0 to 63.

End-to-End QoS Solution Using Classification

All the switches and the routers that access the internet rely on class information to provide the same forwarding treatment to packets with the same class information, and different treatment to packets with different class information. The class information in the packet can be assigned by end hosts or by switches or routers along the way, based on a configured policy, detailed examination of the packet, or both. Detailed examination of the packet is expected to occur closer to the edge of the network, so that the core switches and routers are not overloaded with this task.

Switches and routers along the path can use the class information to limit the amount of resources allocated per traffic class. The behavior of an individual device when handling traffic in the Diff-Serv architecture is called per-hop behavior. If all the devices along a path provide a consistent per-hop behavior, you can construct an end-to-end QoS solution.

Implementing QoS in your network can be a simple task or a complex task, depending on the QoS features offered by your internetworking devices, the traffic types and patterns in your network, and the granularity of control that you need over incoming and outgoing traffic.

Packet Classification

Packet classification is the process of identifying a packet as belonging to one of several classes in a defined policy, based on certain criteria. The Modular QoS CLI (MQC) is a policy-class based language. The policy class language is used to define the following:

- Class map template with one or several match criteria
- Policy map template with one or several classes associated to the policy map

The policy map template is then associated to one or several interfaces on the device.

Packet classification is the process of identifying a packet as belonging to one of the classes defined in the policy map. The process of classification will exit when the packet being processed matches a specific filter in a class. This is referred to as first-match exit. If a packet matches multiple classes in a policy, irrespective of the order of classes in the policy map, it will still exit the classification process after matching the first class.

If a packet does not match any of the classes in the policy, it is classified into the default class in the policy. Every policy map has a default class, which is a system-defined class to match the packets that do not match any of the user-defined classes.

Packet classification can be categorized into the following types:

- Classification based on information that is propagated with the packet
- Classification based on information that is device specific
- Hierarchical classification

Classification Based on Information Propagated with a Packet

Classification based on information that is part of a packet, and propagated either end-to-end or between hops, typically includes the following:

- Classification based on Layer 3 or 4 headers
- Classification based on Layer 2 information

Classification Based on Layer 3 or Layer 4 Header

This is the most common deployment scenario. Numerous fields in the Layer 3 and Layer 4 headers can be used for packet classification.

At the most granular level, this classification methodology can be used to match an entire flow. For this deployment type, an access control list (ACL) can be used. ACLs can also be used based on various subsets of the flow, for example, source IP address only, destination IP address only, or a combination of both.

Classification can also be done based on the precedence or DSCP values in the IP header. The IP precedence field is used to indicate the relative priority with which a particular packet needs to be handled. It is made up of three bits in the IP header's type of service (ToS) byte.

The following table shows the different IP precedence bit values and their names.

Table 3: IP Precedence Values and Names

IP Precedence Value	IP Precedence Bits	IP Precedence Names
0	000	Routine
1	001	Priority
2	010	Immediate
3	011	Flash
4	100	Flash Override
5	101	Critical
6	110	Internetwork control
7	111	Network control

Note All the routing control traffic in a network use the IP precedence value 6 by default. IP precedence value 7 is also reserved for network control traffic. Therefore, we do not recommend the use of IP precedence values 6 and 7 for user traffic.

The DSCP field is made up of 6 bits in the IP header, and is being standardized by the Internet Engineering Task Force (IETF) Differentiated Services Working Group. The original ToS byte, which contained the DSCP bits, has been renamed the DSCP byte. The DSCP field is part of the IP header, similar to IP precedence. The DSCP field is a super set of the IP precedence field. Therefore, the DSCP field is used and is set in ways that are similar to what was described with respect to IP precedence.

Note

- The DSCP field definition is backward-compatible with the IP precedence values.
- Some fields in the Layer 2 header can also be set using a policy.

Classification Based on Layer 2 Header

The CoS method can be used to perform classification based on the Layer 2 header information. Classification is based on the three bits in the Layer 2 header based on the IEEE 802.1p standard. This usually maps to the ToS byte in the IP header.

Classification Based on Information that is Device Specific

A device also provides classification mechanisms in scenarios that are available where classification is not based on information in the packet header or payload.

At times you might be required to aggregate traffic coming from multiple input interfaces into a specific class in the output interface. For example, multiple customer edge routers might be going into the same access device on different interfaces. The service provider might want to police all the aggregate voice traffic going into the core to a specific rate. However, the voice traffic coming in from the different customers could have different ToS settings. QoS group-based classification is a feature that is useful in these scenarios.

Policies configured on the input interfaces set the QoS group to a specific value, which can then be used to classify the packets in the policies enabled on the output interface.

The QoS group is a field in the packet data structure that is internal to a device. It is important to note that a QoS group is an internal label to the device and is not a part of the packet header.

QoS Wired Model

To implement QoS, the device must perform the following tasks:

- Traffic classification: Distinguish packets or flows from one another.
- Traffic marking and policing: Assign a label to indicate the given quality of service as the packets move through the device, and then make the packets comply with the configured resource usage limits.
- Queuing and scheduling: Provide different treatment in all the situations where resource contention exists.
- Shaping: Ensure that the traffic sent from the device meets a specific traffic profile.

Ingress Port Activity

The following activities occur at the ingress port of a device:

- Classification: Classifying a distinct path for a packet by associating it with a QoS label. For example, the device maps the CoS or DSCP in the packet to a QoS label to distinguish one type of traffic from another. The QoS label that is generated identifies all future QoS actions to be performed on this packet.
- Marking: Marking evaluates the policer and configuration information for the action to be taken when a packet is out of profile and determines what to do with the packet (pass through a packet without modification, mark down the QoS label in the packet, or drop the packet). Traffic can be marked with traffic-class to hit a particular VoQ. The traffic-class ranges from 0 to 7. Marking can also be done without policing.

Egress Port Activity

The following activities occur at the egress port of the device:

- Policing: Policing is not supported for outgoing packets.

- Marking: Marking rewrites the QoS tags such as DSCP and CoS in the packet so that the next hop can apply QoS based on the new value. You must configure a separate policy-map (different from the type queueing policy-map described in the next bullet point) and attach it to the egress direction of an interface by using the **service-policy output policy-name** command. This policy-map can only contain marking action using the **set qos-tag value** command. For example, `set dscp cs6` or `set cos 5`.
- Queuing: Queuing selects which VoQ to use, and applies congestion management such as queue-limit and random-detect. Packets admitted into VoQ are scheduled to the corresponding egress output queueing. Shape, priority, and bandwidth remaining ratio affects how packets are scheduled. VoQ selection happens on the ingress side by setting the traffic-class in the ingress policy if configured, or it follows the default QoS tag to traffic-class mapping which only uses traffic-class 0 and 7. On the egress side, a separate **type queueing** policy-map can be attached to the interface unless the default queuing configuration is acceptable. This policy-map can only use class-maps that match traffic-class (class-default matches traffic-class 0 by default). The **service-policy type queueing output policy-name** command is used to attach the queuing policy-map to an interface. The policy-name used must be a policy-name defined using the **policy-map type queueing** command.

Note Queuing is only supported using traffic class.

Classification

Classification is the process of distinguishing one kind of traffic from another by examining the fields in the packet. Classification is enabled only if QoS is enabled on a device. By default, QoS is enabled in a device.

During classification, a device performs a lookup and assigns a QoS label to a packet. The QoS label identifies all the QoS actions to be performed on a packet and from which queue the packet is sent.

Access Control Lists

You can use IP standard and extended IP to define a group of packets with the same characteristics (class). You can also classify IP traffic based on IPv6 ACLs.

In the QoS context, the permit and deny actions in the access control entries (ACEs) have different meanings from security ACLs:

- If a match with a permit action is encountered (first-match principle), the specified QoS-related action is taken.
- If multiple ACLs are configured on a port, the lookup stops after the packet matches the first ACL with a permit action, and QoS processing begins.

Note Deny-based ACL classification is not supported. Only permit-based ACL is supported.

After a class map has been defined with the ACL, you can use the class map name in a policy map. A policy might contain multiple classes, with actions specified for each one of them. A policy might include commands to classify the class as a particular aggregate, for example, assign a DSCP, or rate-limit the class. This policy is then attached to a particular port on which it becomes effective.

Note Layer 2-based ACLs are not supported.

Class Maps

A class map is a mechanism that you use to name a specific traffic flow (or class) and isolate it from all other traffic. The class map defines the criteria used to match against a specific traffic flow to further classify it. The criteria can include matching the access group defined by the ACL or matching a specific list of DSCP or IP precedence values or CoS values or traffic-class values. If you have more than one type of traffic that you want to classify, you can create another class map and use a different name.

Note You can create multiple ACLs in one class, but only the first entry will be considered. By default the class criteria is **match-all**. You cannot use ACLs with one matching TCP and UDP for **match-all** as one packet cannot be TCP and UDP at the same time. You can use it with **match-any** which can match all TCP and UDP packets.

You create a class map by using the **class-map** global configuration command or the **class** policy-map configuration command. You should use the **class-map** command when the map is shared among mulitple policies. When you enter the **class-map** command, the device enters the class-map configuration mode.

You can create a default class by using the **class class-default** policy-map configuration command. The default class is system-defined and cannot be configured. Unclassified traffic (traffic that does not meet the match criteria specified in the traffic classes) is treated as default traffic.

Note When configuring a class-map for traffic-class, only one match traffic-class is supported.

Time-to-Live Classification

Note IPv6 time-to-live (TTL) is not supported.

You can classify packets based on the ACL map. You can set TTL as a criterion in the ACL list and perform a TTL check on the incoming packet. The access control entry is used to check the IPv4 TTL to match the value on the incoming packet. The classified packet is either marked or policed based on the policy-map action. Queueing cannot be configured on this classification.

The following is an example of TTL classification:

```
policy-map TTL_MATCH
 class IPV4_TTL
  police rate 6000000000
  set dscp af23

ip access-list extended IPV4_TTL
 permit ip any any ttl eq 100
 permit tcp any any ttl ne 150

!
```

```
Device#show run class-map IPV4_TTL
class-map match-all IPV4_TTL
 match access-group name IPV4_TTL
!

Device#show policy-map interface hun1/0/47

HundredGigE1/0/47

  Service-policy input: TTL_MATCH

    Class-map: IPV4_TTL (match-all)
    553567424 packets
    Match: access-group name IPV4_TTL
    police:
    rate 6000000000 bps, burst 187500000 bytes
    conformed 22983406600 bytes; actions:
    transmit
    exceeded 32375773000 bytes; actions:
    drop
    conformed 588922000 bps, exceeded 830894000 bps
    QoS Set
    dscp af23

    Class-map: class-default (match-any)
    2184433710 packets
    Match: any
```

Policy Maps

A policy map specifies which traffic class to act on. Actions can include the following:

- Setting a specific DSCP or IP precedence value in the class-map
- Setting a CoS value in the class-map
- Setting a traffic-class in the class-map
- Specifying the traffic bandwidth limitations and the action to take when the traffic is out of profile

Before a policy map can be effective, you must attach it to a port.

You create and name a policy map using the **policy-map** global configuration command. When you enter this command, the device enters the policy-map configuration mode. In this mode, you specify the actions to take on a specific traffic class by using the **class** or **set** policy-map configuration and policy-map class configuration commands.

The policy map can also be configured using the **police** and **bandwidth** policy-map class configuration commands, which define the policer, the bandwidth limitations of the traffic, and the action to take if the limits are exceeded. In addition, the policy-map can further be configured using the **priority** policy-map class configuration command, to schedule priority for the class or the queuing policy-map class configuration command, **queue-limit**.

To enable the policy map, you attach it to a port by using the **service-policy** interface configuration command.

Priority based queuing is supported and only in egress direction. Policy-map can be applied to an interface in ingress direction only.

The following is an example of how to configure policy-map which can be applied to an interface in ingress direction only.

```
Device# configure terminal
Device(config)# class-map match-all cs2
Device(config-cmap)# match dscp cs2
Device(config-cmap)# exit
Device(config)# class-map match-all cs3
Device(config-cmap)# match dscp cs3
Device(config-cmap)# exit
Device(config)# policy-map pmap
Device(config-pmap)# class cs2
Device(config-pmap-c)# police 1000000000
Device(config-pmap-c)# set dscp cs7
Device(config-pmap-c)# exit
Device(config-pmap)# class cs3
Device(config-pmap-c)# set traffic-class 7
Device(config-pmap-c)# set dscp cs5
Device(config-pmap-c)# police cir 40000000000
Device(config-pmap-c)# conform-action transmit
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# interface HundredGigE1/0/2
Device(config-if)# service-policy input pmap
```

Policy Map on Physical Port

A policy map also has these characteristics:

- A policy map can contain multiple class statements, each with different match criteria and policers.
- A policy map can contain a predefined default traffic class explicitly placed at the end of the map.

  When you configure a default traffic class by using the **class class-default** policy-map configuration command, unclassified traffic (traffic that does not meet the match criteria specified in the traffic classes) is treated as the default traffic class (**class-default**).
- A separate policy-map class can exist for each type of traffic received through a port.

Policing

After a packet is classified and has a DSCP-based, CoS-based, or QoS-group label assigned to it, the policing and marking process can begin.

Policing involves creating a policer that specifies the bandwidth limits for the traffic. Packets that exceed the limits are *out of profile* or *nonconforming*. Each policer decides on a packet-by-packet basis whether the packet is in or out of profile and specifies the actions on the packet. These actions, carried out by the marker, include passing through the packet without modification, dropping the packet, or modifying (marking down) the assigned DSCP or CoS value of the packet.

To avoid out-of-order packets, both conform and nonconforming traffic typically exit the same queue.

Note All traffic, regardless of whether it is bridged or routed, is subjected to a policer, if one is configured. As a result, packets might be dropped or might have their DSCP or CoS fields modified when they are policed and marked.

After you configure the policy map and policing actions, attach the policy-map to a port in ingress direction by using the **service-policy** interface configuration command.

Token-Bucket Algorithm

Policing uses a token-bucket algorithm. As each frame is received by the device, a token is added to the bucket. The bucket has a hole in it and leaks at a rate that you specify as the average traffic rate in bits per second. Each time a token is added to the bucket, the device verifies that there is enough room in the bucket. If there is not enough room, the packet is marked as nonconforming, and the specified policer action is taken (dropped or marked down).

How quickly the bucket fills is a function of the bucket depth (burst-byte), the rate at which the tokens are removed (rate-bps), and the duration of the burst above the average rate. The size of the bucket imposes an upper limit on the burst length and limits the number of frames that can be transmitted back-to-back. If the burst is short, the bucket does not overflow, and no action is taken against the traffic flow. However, if a burst is long and at a higher rate, the bucket overflows, and the policing actions are taken against the frames in that burst.

Marking

Marking is used to convey specific information to a downstream device in the network, or to carry information from one interface in a device to another.

Marking can be used to set certain field/bits in the packet headers, or marking can also be used to set certain fields in the packet structure that is internal to the device. It can also be used with traffic classes for packets to hit specific VoQs. Additionally, the marking feature can be used to define mapping between fields. The following marking methods are available for QoS:

- Packet header
- Device specific information
- Table maps

Packet Header Marking

Marking on fields in the packet header can be classified into two general categories:

- IPv4/v6 header bit marking
- Layer 2 header bit marking

Note Only CoS-based is supported. VLAN ID based is not supported.

The marking feature at the IP level is used to set the precedence or the DSCP in the IP header to a specific value to get a specific per-hop behavior at the downstream device (switch or router), or it can also be used to aggregate traffic from different input interfaces into a single class in the output interface. The functionality is currently supported on both the IPv4 and IPv6 headers.

Marking in the Layer 2 headers is typically used to influence dropping behavior in the downstream devices (switch or router). It works in tandem with the match on the Layer 2 headers. The bits in the Layer 2 header that can be set using a policy map are class of service.

Note Based on whether the packet is Layer 2 or Layer 3, DSCP or CoS bit is marked. On a routed traffic, only DSCP marking is allowed, and on a bridged traffic, only CoS marking is allowed.

Switch-Specific Information Marking

This form of marking includes marking of fields in the packet data structure that are not part of the packets header, so that the marking can be used later in the data path. This is not propagated between the switches. Marking of QoS group falls into this category. This form of marking is only supported in policies that are enabled on the input interfaces. The corresponding matching mechanism can be enabled on the output interfaces on the same switch and an appropriate QoS action can be applied.

Traffic-class can be set in the ingress policy to direct the matched traffic into a corresponding VoQ. Discard-class can be set in the ingress policy to be used later for VoQ congestion management for selecting the corresponding queue-limit or random-detect settings based on discard-class.

Table Map Marking

Table map marking enables the mapping and conversion from one field to another using a conversion table. This conversion table is called a table map.

Depending upon the table map attached to an interface, CoS, DSCP, and Precedence values of the packet are rewritten. The device allows configuring both ingress table map policies and egress table map policies. Table maps can also be used to map or set the incoming traffic to a particular VoQ using traffic-class values (0 to 7). CoS or DSCP or PREC to discard-class on the ingress and QoS group to CoS or DSCP or PREC on the egress are also supported.

Only table-map values of same type of QoS tags are supported. For example, table-map of type CoS to CoS or DSCP to DSCP or PREC to PREC is supported, and also table-map of type DSCP or CoS or PREC to QoS group and DSCP or CoS or PREC to traffic-class is supported.

A table map-based policy supports the following capabilities:

- Mutation: You can have a table map that maps from one DSCP value set to another DSCP value set, and this can be attached to noth ingress and egress ports.
- Rewrite: Packets coming in are rewritten depending upon the configured table map.
- Mapping: Table map based policies can be used instead of set policies.

The following steps are required for table map marking:

1. Define the table map: Use the **table-map** global configuration command to map the values. The table does not know of the policies or classes within which it will be used. The default command in the table map is used to indicate the value to be copied into the to field when there is no matching from field.
2. Define the policy map: You must define the policy map where the table map will be used.
3. Associate the policy to an interface.

Note

- For bridged traffic the from type should be CoS and for routed traffic the from type should be DSCP or PREC. Both can be configured for SVI member interfaces. CoS or DSCP will be selected automatically in the data plane based on how traffic is forwarded.
- DSCP to COS table-maps are not supported.
- When you map a QoS group to a DSCP table used for DSCP marking at egress, the QoS group does not add the equivalent of CoS in the packet. Configure a separate QoS group to COS table map if you want to define the QoS group to a non-zero COS value.

Traffic Conditioning

To support QoS in a network, traffic entering the service provider network needs to be policed on the network boundary routers to ensure that the traffic rate stays within the service limit. Even if a few routers at the network boundary start sending more traffic than what the network core is provisioned to handle, the increased traffic load leads to network congestion. The degraded performance in the network makes it difficult to deliver QoS for all the network traffic.

Traffic policing functions (using the police feature) and shaping functions (using the traffic shaping feature) manage the traffic rate, but differ in how they treat traffic when tokens are exhausted. The concept of tokens comes from the token bucket scheme, a traffic metering function.

Note When running QoS tests on network traffic, you may see different results for the shaper and policing data. Network traffic data from shaping provides more accurate results.

This table compares the policing and shaping functions.

Table 4: Comparison Between Policing and Shaping Functions

Policing Function	Shaping Function
Sends conforming traffic up to the configured rate and allows bursts.	Smooths traffic and sends it out at a constant rate.
When tokens are exhausted, action is taken immediately.	When tokens are exhausted, it buffers packets and sends them out later, when tokens are available. A class with shaping has a queue associated with it which will be used to buffer the packets.
Policing has multiple units of configuration – in bits per second and packets per second. **Note** Policing in packets per second (PPS) is only supported for traffic punted to the CPU, and not for network facing interfaces.	Shaping has only one unit of configuration - in bits per second.

Policing Function	Shaping Function
Policing has multiple possible actions associated with an event, marking and dropping being example of such actions.	Shaping does not have the provision to mark packets that do not meet the profile. A new policy map must be available, which can then be applied on the egress interface.
Works for input traffic only.	Implemented for output traffic only.
Transmission Control Protocol (TCP) detects the line at line speed but adapts to the configured rate when a packet drop occurs by lowering its window size.	TCP can detect that it has a lower speed line and adapt its retransmission timer accordingly. This results in less scope of retransmissions and is TCP-friendly.

Policing

The QoS policing feature is used to impose a maximum rate on a traffic class. If the rate is exceeded, then a specific action is taken as soon as the event occurs. The rate parameters (committed information rate [CIR] and peak information rate [PIR]) are configured in bits per second, and the burst parameters (conformed burst size [B_c] and extended burst size [B_e]) are configured in bytes per second.

Only the single-rate two-color policing forms or policers are supported for QoS.

Single-Rate Two-Color Policing

Single-rate two-color policer is the mode in which you configure only a CIR.

The B_c is an optional parameter, and if it is not specified it is computed by default. In this mode, when an incoming packet has enough tokens available, the packet is considered to be conforming.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm, on page 18.

Dual-Rate Three-Color Policing

With the dual rate policer, the device supports only color-blind mode. In this mode, you configure a committed information rate (CIR) and a peak information rate (PIR). As the name suggests, there are two token buckets in this case, one for the peak rate, and one for the conformed rate. The device also supports color-aware mode which can be set using **set discard class 1** action on the ingress classification.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm, on page 18.

In the color-blind mode, the incoming packet is first checked against the peak rate bucket. If there are not enough tokens available, the packet is said to violate the rate. If there are enough tokens available, then the tokens in the conformed rate buckets are checked to determine if there are enough tokens available. The tokens in the peak rate bucket are decremented by the size of the packet. If it does not have enough tokens available, the packet is said to have exceeded the configured rate. If there are enough tokens available, then the packet is said to conform, and the tokens in both the buckets are decremented by the size of the packet.

The rate at which tokens are replenished depends on the packet arrival. Assume that a packet comes in at time T1 and the next one comes in at time T2. The time interval between T1 and T2 determines the number of tokens that need to be added to the token bucket. This is calculated as:

Time interval between packets (T2-T1) * CIR)/8 bytes

Shaping

Shaping is the process of imposing a maximum rate of traffic, while regulating the traffic rate in such a way that the downstream switches and routers are not subjected to congestion. Shaping in the most common form is used to limit the traffic sent from a physical or logical interface.

Shaping has a buffer associated with it that ensures that packets which do not have enough tokens are buffered as opposed to being immediately dropped. The number of buffers available to the subset of traffic being shaped is limited and is computed based on a variety of factors. The number of buffers available can also be tuned using specific QoS commands. Packets are buffered as buffers are available, beyond which they are dropped.

Class-Based Traffic Shaping

The device uses class-based traffic shaping. This shaping feature is enabled on a class in a policy that is associated to an interface. A class that has shaping configured is allocated a number of buffers to hold the packets that do not have tokens. The buffered packets are sent out from the class using FIFO. In the most common form of usage, class-based shaping is used to impose a maximum rate for an physical interface or logical interface as a whole. The following shaping forms are supported in a class:

- Average rate shaping
- Hierarchical shaping

Shaping is implemented using a token bucket. The values of CIR, B_c and B_e determine the rate at which the packets are sent out and the rate at which the tokens are replenished.

Note For information about the token-bucket algorithm, see Token-Bucket Algorithm.

Average Rate Shaping

You use the **shape average** policy-map class command to configure average rate shaping.

This command configures a maximum bandwidth for a particular class. The queue bandwidth is restricted to this value even though the port has more bandwidth available. The device supports configuring shape average by either a percentage or by a target bit rate value.

Hierarchical Shaping

Shaping can also be configured at multiple levels in a hierarchy. This is accomplished by creating a parent policy with shaping configured, and then attaching child policies with additional shaping configurations to the parent policy.

The supported hierarchical shaping type is Port Shaper.

The port shaper uses the class-default and the only action permitted in the parent is shaping. The queuing action is in the child with the port shaper.

Queuing and Scheduling

The device uses both queuing and scheduling to help prevent traffic congestion. The device supports the following queuing and scheduling features:

- Bandwidth
- Weighted Tail Drop
- Priority queues
- Queue buffers
- Weighted Random Early Detection

When you define a queuing policy on a port, control packets are mapped to the best priority queue with the highest threshold, which is traffic-class 7. Control packets queue mapping works differently in the following scenarios:

- Without a quality of service (QoS) policy: If no QoS policy is configured, control packets with DSCP values 16, 24, 32, 40, 48, and 56 are mapped to traffic-class 7, which is the highest threshold.
- With an user-defined policy: An user-defined queuing policy configured on egress ports can affect the default priority queue setting on control packets.

  By default each QoS tag with DSPC or CoS or PREC values are matched to a specific traffic-class.

Note Queuing policy in egress direction does not support **match access-group** classification. It also does not support QoS tag based classification such as match DSCP and COS. Class-maps used in queuing policy can only match traffic-class.

Control traffic is redirected to the best queue based on the following rules:

1. The queue corresponding to traffic-class 7 is always the highest priority queue even without **priority level 1** configured.
2. Without ingress policy or if ingress policy does not set traffic-class, DSCP 16, 24, 32, 40, 48, 56 are mapped to traffic-class 7. CoS 4, 5 are mapped to traffic-class 7. The other DSCP and CoS values are mapped to traffic-class 0.
3. You can apply ingress policy and set traffic-class explicitly in order to override the above default mapping. You should make sure that voice and control plane traffic gets mapped to high priority traffic-class.
4. If traffic-class is not configured, the best queue is selected based on the default implementation logic. Cisco IOS software assigns control packets with Differentiated Services Code Point (DSCP) values 16, 24, 32, 40, 48, and 56 are mapped to traffic-class 7 and reassigns the rest of the control traffic in the best queue to traffic-class 0.

Note To provide proper QoS for Layer 3 packets, you must ensure that packets are explicitly classified into appropriate queues.

On a network interface with queueing police enabled, when both unicast and multicast traffic are directed to the same traffic class queue, traffic prioritization happens such that 80% of the available bandwidth is allocated for unicast traffic, while the remaining 20% is reserved for multicast traffic. This allocation creates a traffic distribution ratio of 80:20 between unicast and multicast traffic.

Bandwidth

The device supports the bandwidth remaining ratio configuration. You can use the **bandwidth remaining ratio** policy-map class command to configure the ratio for sharing excess bandwidth. The range is from 1 to 63.

Priority Queues

Each port supports eight ingress queues, of which seven can be given a priority.

You can use the **priority level** policy class-map command to configure the priority for seven classes, including the class-default which is the lowest priority level. Each priority level ranging from 1 to 7 can be configured to one class only, that is, there cannot be more than one class at the same priority level. Each class that does not have a priority level configured is referred to as priority normal, and there can be multiple priority normal classes.

If a priority level is configured in a policy-map, priority level 1 must be configured. For example, policy-map with priority level 2 and priority level 3 but no priority level 1 is not allowed. If all the priority levels configured in a policy-map is sorted, they must be contiguous, that is, no priority level should be skipped, for example, the sequence of priority levels 1, 2, 4, and 5 where priority level 3 is skipped is not allowed.

Each priority level must be configured to the class that matches the corresponding traffic class as shown in the following table:

Table 5: Priority Level to Traffic Class Mapping

Priority Level	Traffic Class
1 (highest)	7
2	6
3	5
4	4
5	3
6	2
7	1
None (lowest)	0

Weighted Random Early Detection

Weighted random early detection (WRED) is a mechanism to avoid congestion in networks. WRED reduces the chances of tail drop by selectively dropping packets when the output interface begins to show signs of congestion, thus avoiding large number of packet drops at once.

For more information about WRED, see Configuring Weighted Random Early Detection.

Default Wired QoS Configuration

There are two queues configured by default on each wired interface on the device. All control traffic traverses and is processed through VoQ 7. All other traffic traverses and is processed through VoQ 0.

DSCP Maps

This section provides information about DSCP maps.

Default IP-Precedence-to-DSCP Map

You use the IP-precedence-to-DSCP map to map IP precedence values in incoming packets to a DSCP value that QoS uses internally to represent the priority of the traffic. The following table shows the default IP-precedence-to-DSCP map. If these values are not appropriate for your network, you need to modify them.

Table 6: Default IP-Precedence-to-DSCP Map

IP Precedence Value	DSCP Value
0	0
1	8
2	16
3	24
4	32
5	40
6	48
7	56

Default QoS Mapping

The following tables list default QoS mapping values. These values are applicable only if there are no policy-map applied to an interface.

Table 7: DSCP Default Mapping Table for Ingress

DSCP	DSCP	Encapsulation	QoS-group	Table Map	Drop Precedence
0	0	0	0	0	G
1	1	0	0	0	G
2	2	0	0	0	G
3	3	0	0	0	G
4	4	0	0	0	G
5	5	0	0	0	G

DSCP	DSCP	Encapsulation	QoS-group	Table Map	Drop Precedence
6	6	0	0	0	G
7	7	0	0	0	G
8	8	1	0	0	G
9	9	1	0	0	G
10	10	1	0	0	G
11	11	1	0	0	G
12	12	1	0	0	G
13	13	1	0	0	G
14	14	1	0	0	G
15	15	1	0	0	G
16	16	2	0	7	G
17	17	2	0	0	G
18	18	2	0	0	G
19	19	2	0	0	G
20	20	2	0	0	G
21	21	2	0	0	G
22	22	2	0	0	G
23	23	2	0	0	G
24	24	3	0	7	G
25	25	3	0	0	G
26	26	3	0	0	G
27	27	3	0	0	G
28	28	3	0	0	G
29	29	3	0	0	G
30	30	3	0	0	G
31	31	3	0	0	G
32	32	4	0	7	G
33	33	4	0	0	G
34	34	4	0	0	G
35	35	4	0	0	G
36	36	4	0	0	G
37	37	4	0	0	G

DSCP	DSCP	Encapsulation	QoS-group	Table Map	Drop Behavior
38	38	4	0	0	G
39	39	4	0	0	G
40	40	5	0	7	G
41	41	5	0	0	G
42	42	5	0	0	G
43	43	5	0	0	G
44	44	5	0	0	G
45	45	5	0	0	G
46	46	5	0	7	G
47	47	5	0	0	G
48	48	6	0	7	G
49	49	6	0	0	G
50	50	6	0	0	G
51	51	6	0	0	G
52	52	6	0	0	G
53	53	6	0	0	G
54	54	6	0	0	G
55	55	6	0	0	G
56	56	7	0	7	G
57	57	7	0	0	G
58	58	7	0	0	G
59	59	7	0	0	G
60	60	7	0	0	G
61	61	7	0	0	G
62	62	7	0	0	G
63	63	7	0	0	G

Table 8: CoS-Drop Eligible Indicator Mapping Table for Ingress

CoS-DEI	CoS-DEI	Encapsulation	QoS-Group	Table Map	Drop Behavior
0	0	0	0	0	G
1	1	0	0	0	G

CoS-DEI	CoS-DEI	Encapsulation	QoS-Group	Table Map	Drop Precedence
2	2	1	0	0	G
3	3	1	0	0	G
4	4	2	0	0	G
5	5	2	0	0	G
6	6	3	0	0	G
7	7	3	0	0	G
8	8	4	0	7	G
9	9	4	0	7	G
10	10	5	0	7	G
11	11	5	0	7	G
12	12	6	0	0	G
13	13	6	0	0	G
14	14	7	0	0	G
15	15	7	0	0	G

Table 9: MPLS-Experimental Mapping Table for Ingress

MPLS Experimental	MPLS Experimental	Encapsulation	QoS-Group	Table Map	Drop Precedence
0	0	0	0	0	G
1	1	1	0	0	G
2	2	2	0	0	G
3	3	3	0	0	G
4	4	4	0	0	G
5	5	5	0	0	G
6	6	6	0	0	G
7	7	7	0	0	G

Table 10: DSCP Mapping Table for Egress

DSCP	DSCP	Encapsulation Type of Service	Encapsulation Priority Code Point	Encapsulation MPLS Experimental
0	0	0	0	0
1	1	4	0	0

DSCP	DSCP	Encapsulation Type of Service	Encapsulation Priority Code Point	Encapsulation MPLS Experimental
2	2	8	0	0
3	3	12	0	0
4	4	16	0	0
5	5	20	0	0
6	6	24	0	0
7	7	28	0	0
8	8	32	1	1
9	9	36	1	1
10	10	40	1	1
11	11	44	1	1
12	12	48	1	1
13	13	52	1	1
14	14	56	1	1
15	15	60	1	1
16	16	64	2	2
17	17	68	2	2
18	18	72	2	2
19	19	76	2	2
20	20	80	2	2
21	21	84	2	2
22	22	88	2	2
23	23	92	2	2
24	24	96	3	3
25	25	100	3	3
26	26	104	3	3
27	27	108	3	3
28	28	112	3	3
29	29	116	3	3
30	30	120	3	3
31	31	124	3	3
32	32	128	4	4

DSCP	DSCP	Encapsulation Type of Service	Encapsulation Priority Code Point	Encapsulation MPLS Experimental
33	33	132	4	4
34	34	136	4	4
35	35	140	4	4
36	36	144	4	4
37	37	148	4	4
38	38	152	4	4
39	39	156	4	4
40	40	160	5	5
41	41	164	5	5
42	42	168	5	5
43	43	172	5	5
44	44	176	5	5
45	45	180	5	5
46	46	184	5	5
47	47	188	5	5
48	48	192	6	6
49	49	196	6	6
50	50	200	6	6
51	51	204	6	6
52	52	208	6	6
53	53	212	6	6
54	54	216	6	6
55	55	220	6	6
56	56	224	7	7
57	57	228	7	7
58	58	232	7	7
59	59	236	7	7
60	60	240	7	7
61	61	244	7	7
62	62	248	7	7
63	63	252	7	7

Table 11: CoS-DEI Mapping Table for Egress

CoS-DEI	CoS-DEI	Encapsulation Type of Service	Encapsulation Priority Code Point	Encapsulation MPLS Experimental
0	0	0	0	0
1	1	0	0	0
2	2	32	1	1
3	3	32	1	1
4	4	64	2	2
5	5	64	2	2
6	6	96	3	3
7	7	96	3	3
8	8	128	4	4
9	9	128	4	4
10	10	160	5	5
11	11	160	5	5
12	12	192	6	6
13	13	192	6	6
14	14	224	7	7
15	15	224	7	7

Table 12: MPLS-Experimental Mapping Table for Egress

MPLS-Experimental	MPLS-Experimental	Encapsulation Type of Service	Encapsulation Priority Code Point	Encapsulation MPLS Experimental
0	0	0	0	0
1	0	32	1	1
2	0	64	2	2
3	0	96	3	3
4	0	128	4	4
5	0	160	5	5
6	0	192	6	6
7	0	224	7	7

How to Configure QoS

How to Configure Class, Policy, and Maps

The following sections provide configuration information about class, policy, and maps.

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.

Before you begin

All match commands specified in this configuration task are considered optional, but you must configure at least one match criterion for a class.

Note Individual class-maps must be defined for different kind of QoS tag values. Matching a traffic-class with two different traffic-class values in the same class-map is not supported. Also, matching traffic-class with other QoS tag values is not supported. For example, matching DSCP or PRE or CoS cannot be added to a class-map which already has a traffic-class match.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**class-map** *class-map name* {**match-any** | **match-all**}
	Example:
	`Device(config)# class-map test_1000`
	`Device(config-cmap)#`
		Enters class map configuration mode.
		• Creates a class map to be used for matching packets to the class whose name you specify.
		• **match-any:** Any one of the match criteria must be met for traffic entering the traffic class to be classified as part of it.
		• **match-all:** All of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.
		Note

	Command or Action	Purpose
		This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**match access-group** {*index number* | *name*}
	Example:
	`Device(config-cmap)# match access-group 100`
	`Device(config-cmap)#`
		The following parameters are available for this command:
		• access-group
		• cos
		• dscp
		• ip
		• mpls
		• precedence
		• qos-group
		• traffic-class
		(Optional) For this example, enter the access-group ID:
		• Access list index (value from 1 to 2799)
		• Named access list

Step 4	**match cos** *cos value*
	Example:
	`Device(config-cmap)# match cos 2 3 4 5`
	`Device(config-cmap)#`
		(Optional) Matches IEEE 802.1Q or ISL class of service (user) priority values.
		• Enters up to 4 CoS values separated by spaces (0 to 7).

Step 5	**match dscp** *dscp value*
	Example:
	`Device(config-cmap)# match dscp af11 af12`
	`Device(config-cmap)#`
		(Optional) Matches the DSCP values in IPv4 and IPv6 packets.

Step 6	**match ip** {**dscp** *dscp value* | **precedence** *precedence value* }
	Example:
	`Device(config-cmap)# match ip dscp af11 af12`
	`Device(config-cmap)#`
		(Optional) Matches IP values including the following:
		• **dscp**: Matches IP DSCP (DiffServ codepoints).
		• **precedence**: Matches IP precedence (0 to 7).

	Command or Action	Purpose
		Note Since CPU generated packets are not marked at egress, the packet will not match the configured class-map.

Step 7	**match mpls experimental topmost** *experimental value*
	Example:
	`Device(config-cmap)# match mpls experimental topmost 3`
	`Device(config-cmap)#`
		(Optional) Match MPLS experimental value on topmost label (from 0 to 7).

Step 8	**match qos-group** *qos group value*
	Example:
	`Device(config-cmap)# match qos-group 10`
	`Device(config-cmap)#`
		(Optional) Matches QoS group value (from 0 to 31).

Step 9	**match traffic-class** *traffic class value*
	Example:
	`Device(config-cmap)# match traffic-class 7`
	`Device(config-cmap)#`
		(Optional) Matches QoS traffic class value (from 1 to 7).

Step 10	**end** **Example:** `Device(config-cmap)# end`	Saves the configuration changes.

What to do next

Configure the policy map.

Creating a Traffic Policy

To create a traffic policy, use the **policy-map** global configuration 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 entered after you enter the policy map configuration mode. After entering the **class** command, the device is automatically in policy map class configuration mode, which is where the QoS policies for the traffic policy are defined.

The following policy map class-actions are supported:

- exit: Exits from the QoS class action configuration mode.
- no: Negates or sets default values for the command.
- police: Policer configuration options.
- service-policy: Configures the QoS service policy.
- set: Sets QoS values using the following options:
  - CoS values
  - Discard-class values
  - DSCP values
  - IP values
  - MPLS exp values
  - Precedence values
  - QoS group values
  - Traffic class values

Note If the **set** option is not used in the egress policy to remark egress traffic, the statistics of the policy will not be displayed when you use the **show policy-map interface** command.

Before you begin

You should have first created a class map.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy-map name*
	Example:
	`Device(config)# policy-map test_2000`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

	Command or Action	Purpose

Step 3	**class** {*class-name* | **class-default**}
	Example:
	`Device(config-pmap)# class test_1000`
	`Device(config-pmap-c)#`
		Specifies the name of the class whose policy you want to create or change.
		You can also create a system default class for unclassified packets.

Step 4	**police** {*target_bit_rate* | **cir** | **rate**}
	Example:
	`Device(config-pmap-c)# police 100000`
	`Device(config-pmap-c)#`
		(Optional) Configures the policer:
		• *target_bit_rate*: Enter the bit rate per second, enter a value between 8000 and 400000000000.
		• **cir**: Committed Information Rate.
		• **rate**: Specify police rate, PCR for hierarchical policies or SCR for single-level ATM 4.0 policer policies.

Step 5	**service-policy** *policy-map name*
	Example:
	`Device(config-pmap-c)# service-policy test_2000`
	`Device(config-pmap-c)#`
		(Optional) Configures the QoS service policy.

Step 6	**set** {**cos** | **discard-class** | **dscp** | **ip** | **mpls** | **precedence** | **qos-group** | **traffic-class**}
	Example:
	`Device(config-pmap-c)# set cos 7`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		• **cos**: Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **discard-class**: Sets discard behavior identifier.
		• **dscp**: Sets DSCP in IP(v4) and IPv6 packets.
		• **ip**: Sets IP specific values.
		• **mpls**: Sets MPLS specific values.
		• **precedence**: Sets precedence in IP(v4) and IPv6 packet.
		• **qos-group**: Sets the QoS Group.
		• **traffic-class**: Sets the traffic class.

Step 7	**end** **Example:**	Saves the configuration changes.

	Command or Action	Purpose
	`Device(config-pmap-c) #end` `Device(config-pmap-c) #`	

What to do next

Configure the interface.

Creating a Traffic Queueing Policy

To create a traffic queueing policy for QoS, peform this procedure.

The following queueing policy map class-actions are supported:

- bandwidth: Bandwidth configuration options.
- exit: Exits from the QoS class action configuration mode.
- no: Negates or sets default values for the command.
- priority: Configures strict scheduling priority for the class.
- queue-limit: Queue threshold for tail drop configuration options.
- random-detect: Enables Random Early Detection as drop policy.
- service-policy: Configures QoS service policy.
- shape: Traffic shaping configuration options.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queueing** *policy name*
	Example:
	`Device(config)# policy-map type queueing test`
	`Device(config-pmap)#`
		Specifies the name of the queueing profile policy and enters policy map configuration mode.

Step 3	**class** *class name* **Example:** `Device(config-pmap)# class`	Specifies the name of the class to be associated with the policy and enters policy class map configuration mode. Command options for policy class map configuration mode include the following:

	Command or Action	Purpose
	traffic-class7 Device(config-pmap-c)#	• *word*: Class map name. • **class-default**: System default class matching any otherwise unclassified packets.
Step 4	**bandwidth remaining ratio** *ratio* **Example:** Device(config-pmap)# **bandwidth remaining ratio 10** Device(config-pmap-c)#	(Optional) Configures ratio for sharing excess bandwidth. The range is from 1 to 100. **Note** **bandwidth remaining ratio** and **priority level** cannot be configured simultaneously for the same class, even though they can be in the same policy.
Step 5	**priority level** *level* **Example:** Device(config-pmap)# **priority level 1** Device(config-pmap-c)#	(Optional) Configures multi-level priority queue. The range is from 1 to 7. **Note** Priority level 1 can be configured only for traffic-class 7, priority level 2 can be configured only for traffic-class 6, and so on. Each traffic-class value can be mapped only to its respective priority level only.
Step 6	**queue-limit** *value* [**bytes**] **Example:** Device(config-pmap-c)# **queue-limit 100000 bytes** Device(config-pmap-c)#	(Optional) Sets the queue limit threshold percentage value in bytes. The range is from 1000000 to 396000000.
Step 7	**random-detect** {**discard-class** | **discard-class-based** | **exponential-weighting-constant**} **Example:** Device(config-pmap)# **random-detect discard-class-based** Device(config-pmap-c)#	(Optional) Enables Random Early Detection as the drop policy. • **discard-class**: Parameters for each discard-class value (0 or 1). **Note** **discard-class** and **priority** cannnot be configured simultaneously for the same class, even though they can be in the same policy. • **discard-class-based**: Enables discard-class-based WRED as the drop policy. • **exponential-weighting-constant**: Weight for mean queue depth calculation (0 to 15).

	Command or Action	Purpose

Step 8	**service-policy** *name*
	Example:
	`Device(config-pmap)# service-policy policy1`
	`Device(config-pmap-c)#`
		(Optional) Configures QoS Service Policy.
		Note policy1 used in this example should be a queueing policy. It can be configured only under class-default.

Step 9	**policy rate** {*Kb/s* | **percent**}
	Example:
	`Device(config-pmap-c)# policy rate 1000000000`
	`Device(config-pmap-c)#`
		Configures the traffic shaping average. The parameters include:
		• *Kb/s*: Use this command to configure a specific value. The range is 1.2 Mbps to 400 Gbps.
		• **percent**: Allocates a minimum bandwidth to a particular class. The queue can oversubscribe bandwidth in case other queues do not utilize the entire port bandwidth. The total sum cannot exceed 100 percent, and in case it is less than 100 percent, the rest of the bandwidth is divided along all non priority queues based on bandwidth remaining ratio.

Step 10	**end** **Example:** `Device(config-cmap)# end`	Exits class map configuration mode and returns to privileged EXEC mode.

Configuring Class-Based Packet Marking for Ingress Policy-map

This procedure explains how to configure the following class-based packet marking features on your device:

- CoS value
- DSCP value
- IP value
- Precedence value
- QoS group value
- Traffic-class value
- Discard-class value
- MPLS experimental value

Before you begin

You should have created a class map and a policy map before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map** *policy name*
	Example:
	`Device(config)# policy-map policy1`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class-default`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change.
		Command options for policy class map configuration mode include the following:
		• **exit**: Exits from the QoS class action configuration mode.
		• **no**: Negates or sets default values for the command.
		• **police**: Policer configuration options.
		• **service-policy**: Configures the QoS service policy.
		• **set**: Sets QoS values using the following options:
		  • CoS values
		  • DSCP values
		  • Precedence values
		  • QoS group values
		Note
		This procedure describes the supported configurations using **set** command options. The other command options (**bandwidth**) are described in other sections of this guide. Only

	Command or Action	Purpose
		one **set** command for QoS tag (CoS/DSCP/PREC/EXP) is supported per class. The same class can also have set traffic-class and/or set discard-class if it is a ingress policy.

Step 4	**set cos** {*cos value* | **cos table** *table-map name* | **qos-group table** *table-map name*}
	Example:
	`Device(config-pmap)# set cos cos table cos-mapping`
	`Device(config-pmap)#`
		(Optional) Sets the specific IEEE 802.1Q Layer 2 CoS value of an outgoing packet. Values are from 0 to7.
		You can also set the CoS values using a table map. The following is a list of from types supported when using a table map. For example, **set cos qos-group table** *table-map name* means use the table-map to map from QoS-group to CoS.
		• **cos table**: Sets the CoS value based on a table map.
		• **qos-group table**: Sets the CoS value from QoS group based on a table map.
		Note **set cos qos-group table** is only supported in egress policy.
		Note Table-map is only supported in class-default.

Step 5	**set dscp** {*dscp value* | **default** | **dscp table** *table-map name* | **ef** | **precedence table** *table-map name* | **qos-group table** *table-map name* | **traffic-class table** *table-map name*}
	Example:
	`Device(config-pmap)# set dscp af11`
	`Device(config-pmap)#`
		(Optional) Sets the DSCP value.
		You can also set the DSCP values using a table map. The following is a list of from types supported when using a table map. For example, **set dscp qos-group table** *table-map name* means use the table-map to map from QoS-group to DSCP.
		• **dscp table**: Sets the DSCP value from DSCP based on a table map.
		• **precedence table**: Sets the DSCP value from precedence based on a table map.
		• **qos-group table**: Sets the DSCP value from a QoS group based upon a table map.
		Note Table-map is only supported in class-default.

	Command or Action	Purpose

Step 6	**set ip** {**dscp** | **precedence**}
	Example:
	`Device(config-pmap)# set ip dscp c3`
	`Device(config-pmap)#`
		(Optional) Sets IP specific values. These values are either IP DSCP or IP precedence values.
		You can also set the IP values using a table map. The following is a list of from types supported when using a table map. For example, **set ip dscp qos-group table** *table-map name* means use the table-map to map from QoS-group to IP DSCP.
		Note
		Table-map is only supported in class-default.
		You can set the following values using the **set ip dscp** command:
		• **dscp table**: Sets the DSCP value from DSCP based on a table map.
		• **precedence table**: Sets the DSCP value from precedence based on a table map.
		• **qos-group table**: Sets the DSCP value from a QoS group based upon a table map.
		The following is a list of from types supported when using a table map. For example, **set ip precedence qos-group table** *table-map name* means use the table-map to map from QoS-group to IP precedence.
		• *precedence value*: Sets the precedence value (from 0 to 7) .
		• **dscp table**: Sets the precedence value from DSCP value based on a table map.
		• **precedence table**: Sets the precedence value from precedence based on a table map
		• **qos-group table**: Sets the precedence value from a QoS group based upon a table map.
		Note
		set ip precedence qos-group table *table-map name* is only supported in egress.

Step 7	**set precedence** {*precedence value* | **dscp table** *table-map name* | **precedence table** *table-map*	(Optional) Sets precedence values in IPv4 and IPv6 packets.

	Command or Action	Purpose

	name | **qos-group table** *table-map name* | **traffic-class table** *table-map name*}
	Example:
	`Device(config-pmap)# set precedence 5`
	`Device(config-pmap)#`
		You can also set the precedence values using a table map. The following is a list of from types supported when using a table map. For example, **set precedence qos-group table** *table-map name* means use the table-map to map from QoS-group to precedence.
		• *precedence value*: Sets the precedence value (from 0 to 7) .
		• **dscp table**: Sets the precedence value from DSCP value based on a table map.
		• **precedence table**: Sets the precedence value from precedence based on a table map.
		• **qos-group table**: Sets the precedence value from a QoS group based upon a table map.
		Note
		Table-map is only supported in class-default.

Step 8	**set discard-class** {*discard-class value* | **cos table** *table-map name* | **dscp table** *table-map name* | **mpls experimental table** *table-map name* | **precedence table** *table-map name*}
	Example:
	`Device(config-pmap)# set discard-class 0`
	`Device(config-pmap)#`
		(Optional) Sets discard-class values. You can also set the discard-class values using a table map.
		Note
		Table-map is only supported in class-default.
		• **cos table**: Sets the discard-class value from CoS value based on a table map.
		• **dscp table**: Sets the discard-class value from DSCP value based on a table map.
		• **mpls experimental table**: Sets the discard-class value from MPLS experimental based on a table map.
		• **precedence table**: Sets the discard-class value from precedence value based on a table map.

Step 9	**set qos-group** {*qos-group value* | **dscp table** *table-map name* | **mpls experimental table** *table-map name* | **precedence table** *table-map name*} **Example:** `Device(config-pmap)# set qos-group 10`	(Optional) Sets QoS group values. You can also set the QoS group using a table map. • **cos table**: Sets QoS group from CoS value based on a table map. • **dscp table**: Sets QoS group from DSCP value based on a table map.

	Command or Action	Purpose
	`Device(config-pmap)#`	• **mpls experimental table**: Sets QoS group from MPLS experimental value based on a table map. • **precedence table**: Sets QoS group from precedence based on a table map.

Step 10	**set traffic-class** {*traffic-class value* | **cos table** *table-map name* | **dscp table** *table-map name* | **mpls experimental table** *table-map name* | **precedence table** *table-map name*}
	Example:
	`Device(config-pmap)# set traffic-class 3`
	`Device(config-pmap)#`
		(Optional) Sets traffic-class values from 0 to 7, or from the following table-maps:
		• **cos table**: Sets the traffic-class value from COS value based on a table map.
		• **dscp table**: Sets the traffic-class value from DSCP value based on a table map.
		• **mpls experimental table**: Sets the traffic-class value from MPLS experimental value based on a table map.
		• **precedence table**: Sets the traffic-class value from precedence value based on a table map.

Step 11	**end**
	Example:
	`Device(config-pmap)# end`
	`Device#`
		Exits policy map configuration mode and returns to privileged EXEC mode.

Step 12	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Attach the traffic policy to an interface using the **service-policy** command.

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).

Before you begin

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

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**interface** *type*
	Example:
	`Device(config)# interface FortyGigE1/0/1`
	`Device(config-if)#`
		Enters interface configuration mode and configures an interface.
		Command parameters for the interface configuration include:
		• **TenGigabitEthernet**: 10-Gigabit Ethernet
		• **FiftyGigabitEthernet**: 25-Gigabit Ethernet
		• **FiftyGigabitEthernet**: 50-Gigabit Ethernet
		• **FortyGigabitEthernet**: 40-Gigabit Ethernet
		• **HundredGigabitEthernet**: 100-Gigabit Ethernet
		• **FourHundredGigabitEthernet**: 400-Gigabit Ethernet
		Cisco Catalyst 9500 Series Switches support breakout as 4x10 G and 4x25 G. On Cisco Catalyst 9500X series switches, breakout is supported only on odd 100-G ports and on all 400-G ports. These switches support breakout as 2x100 G and 4x100 G.
		Note
		• Tunnel and Vlan interface are not supported.
		• Policing, queuing, and marking are not supported on the management interface.

Step 3	**service-policy** [**type queueing**] {**input** *policy-map* | **output** *policy-map*} **Example:** `Device(config-if)# service-policy output policy_map_01`	Attaches a policy map to an input or output interface. This policy map is then used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface (egress). **Note**

	Command or Action	Purpose
	`Device(config-if)#`	• Non-queuing policy with policer is supported only in ingress. • On egress, only non-queueing policy with marking is supported. • Queuing policy (**type queueing**) is supported only in egress.

Step 4	**end**
	Example:
	`Device(config-if)# end`
	`Device#`
		Exits interface configuration mode and returns to privileged EXEC mode.

Step 5	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays statistics for the policy on the specified interface.

What to do next

Proceed to attach any other traffic policy to an interface, and to specify the direction in which the policy should be applied.

Classifying, Policing, and Marking Traffic on Physical Ports by Using Policy Maps

You can configure a nonhierarchical policy map on a physical port that specifies which traffic class to act on. Actions supported are remarking and policing.

Before you begin

You should have already decided upon the classification, policing, and marking of your network traffic by policy maps prior to beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**class-map** {*class-map name* | **match-any** | **match-all**}	Enters class map configuration mode.

	Command or Action	Purpose

	Example:
	`Device(config)# class-map ipclass1`
	`Device(config-cmap)# exit`
	`Device(config)#`
		• Creates a class map to be used for matching packets to the class whose name you specify.
		• 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.
		• If you specify **match-all**, all of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.
		Note This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**match access-group** { *access list index* | *access list name* }
	Example:
	`Device(config-cmap)# match access-group 1000`
	`Device(config-cmap)# exit`
	`Device(config)#`
		The following parameters are available for this command:
		• access-group
		• cos
		• dscp
		• ip
		• precedence
		• qos-group
		(Optional) For this example, enter the access-group ID:
		• Access list index (value from 1 to 2799)
		• Named access list

Step 4	**policy-map** *policy-map-name*
	Example:
	`Device(config)# policy-map ipclass1`
	`Device(config-pmap)#`
		Creates a policy map by entering the policy map name, and enters policy-map configuration mode.
		By default, no policy maps are defined.

Step 5	**class** {*class-map-name* | **class-default**} **Example:** `Device(config-pmap)# class ipclass1`	Defines a traffic classification, and enter policy-map class configuration mode. By default, no policy map class-maps are defined.

	Command or Action	Purpose
	`Device(config-pmap-c)#`	If a traffic class has already been defined by using the **class-map** global configuration command, specify its name for *class-map-name* in this command. A **class-default** traffic class is predefined and can be added to any policy. It is always placed at the end of a policy map. With an implied **match any** included in the **class-default** class, all packets that have not already matched the other traffic classes will match **class-default**.

Step 6	**set** {**cos** | **discard-class** | **dscp** | **ip** | **mpls** | **precedence** | **qos-group** | **traffic-class**}
	
	Example:
	
	`Device(config-pmap-c)# set dscp 45`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		
		• **cos**: Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **discard-class**: Sets discard behavior identifier.
		• **dscp**: Sets DSCP in IP(v4) and IPv6 packets.
		• **ip**: Sets IP specific values.
		• **mpls**: Sets MPLS specific values.
		• **precedence**: Sets precedence in IP(v4) and IPv6 packet.
		• **qos-group**: Sets the QoS Group.
		• **traffic-class**: Sets the traffic class.
		
		In this example, the **set dscp** command classifies the IP traffic by setting a new DSCP value in the packet.

Step 7	**police** {*target_bit_rate* | **cir** | **rate** }
	
	Example:
	
	`Device(config-pmap-c)# police 100000 conform-action transmit exceed-action drop`
	`Device(config-pmap-c)#`
		(Optional) Configures the policer:
		
		• *target_bit_rate*: Specifies the bit rate per second, enter a value between 8000 and 400000000000.
		• **cir**: Committed Information Rate.
		• **rate**: Specifies the police rate PCR for hierarchical policies.
		
		In this example, the **police** command adds a policer to the class where any traffic beyond the 100000 set target bit rate is dropped.

	Command or Action	Purpose
Step 8	**exit** **Example:** `Device(config-pmap-c)# exit`	Returns to policy map configuration mode.
Step 9	**exit** **Example:** `Device(config-pmap)# exit`	Returns to global configuration mode.
Step 10	**interface** *interface-id* **Example:** `Device(config)# interface HundredGigE1/0/2`	Specifies the port to attach to the policy map, and enters interface configuration mode. Valid interfaces include physical ports.
Step 11	**service-policy input** *policy-map-name* **Example:** `Device(config-if)# service-policy input ipclass1`	Specifies the policy-map name, and applies it to an ingress port. Only one policy map per ingress port is supported.
Step 12	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 13	**show policy-map** [*policy-map-name* [**class** *class-map-name*]] **Example:** `Device# show policy-map ipclass1`	(Optional) Verifies your entries.
Step 14	**copy running-config startup-config** **Example:** `Device# copy-running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

If applicable to your QoS configuration, configure classification, policing, and marking of traffic on SVIs by using policy maps.

Classifying and Marking Traffic by Using Policy Maps

Before you begin

You should have already decided upon the classification, policing, and marking of your network traffic by using policy maps prior to beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**class-map** {*class-map name* | **match-any** | **match-all**}
	Example:
	`Device(config)# class-map class_cos2`
	`Device(config-cmap)# match cos 2`
		Enters class map configuration mode.
		• Creates a class map to be used for matching packets to the class whose name you specify.
		• 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.
		• If you specify **match-all**, all of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.
		Note This is the default. If **match-any** or **match-all** is not explicitly defined, **match-all** is chosen by default.

Step 3	**policy-map** *policy-map-name*
	Example:
	`Device(config-cmap)# policy-map policy_cos2`
	`Device(config-pmap)#`
		Creates a policy map by entering the policy map name, and enters policy-map configuration mode.
		By default, no policy maps are defined.

	Command or Action	Purpose
Step 4	**description** *description* **Example:** `Device(config-pmap)# description cos 2`	(Optional) Enters a description of the policy map.

Step 5	**class** {*class-map-name* | **class-default**}
	Example:
	`Device(config-pmap)# class class_cos2`
	`Device(config-pmap-c)#`
		Defines a traffic classification, and enters the policy-map class configuration mode.
		By default, no policy map class-maps are defined.
		If a traffic class has already been defined by using the **class-map** global configuration command, specify its name for *class-map-name* in this command.
		A **class-default** traffic class is predefined and can be added to any policy. It is always placed at the end of a policy map. With an implied **match any** included in the **class-default** class, all packets that have not already matched the other traffic classes will match **class-default**.

Step 6	**set** {**cos** | **qos-group** | **traffic-class** | **discard-class**}
	Example:
	`Device(config-pmap-c)# set cos 7`
	`Device(config-pmap-c)#`
		(Optional) Sets the QoS values. Possible QoS configuration values include:
		• **cos**: Sets the IEEE 802.1Q/ISL class of service/user priority.
		• **qos-group**: Sets QoS group.
		• **traffic-class**: Sets traffic class.
		• **discard-class**: Sets discard class.
		In this example, the **set cos** command classifies the traffic by matching the packets with CoS values of 2, and sets the CoS value to a new value of 7.

Step 7	**exit** **Example:** `Device(config-pmap-c)# exit`	Returns to policy map configuration mode.
Step 8	**exit** **Example:** `Device(config-pmap)# exit`	Returns to global configuration mode.

	Command or Action	Purpose
Step 9	**interface** *interface-id* **Example:** `Device(config)# interface HundredGigE 1/0/3`	Specifies the port to attach to the policy map, and enters interface configuration mode. Valid interfaces include physical ports.
Step 10	**service-policy input** *policy-map-name* **Example:** `Device(config-if)# service-policy input policy_cos2`	Specifies the policy-map name, and applies it to an ingress port. Only one policy map per ingress port is supported.
Step 11	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 12	**show policy-map** [*policy-map-name* [**class** *class-map-name*]] **Example:** `Device# show policy-map`	(Optional) Verifies your entries.
Step 13	**copy running-config startup-config** **Example:** `Device# copy-running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Table Maps

Table maps are a form of marking, and also enable the mapping and conversion of one field to another using a table. For example, a table map can be used to map and convert a Layer 2 CoS setting to a new or same CoS value, and also convert a Layer 2 CoS setting to a traffic class or a QoS group. Similar settings apply to Layer 3 DSCP and Precedence.

When setting a table-map in a policy, the **from** and **to** values should be of the same type, example DCSP to DSCP, COS to COS, and so on. This exception is not applicable for QoS-group and traffic-class where **from** and **to** values are of different types, for example, DSCP to traffic-class, COS to traffic-class, and so on.

Note

- A table map can be referenced in multiple policies or multiple times in the same policy.
- The examples in this procedure uses table maps to map DSCP value to traffic-class.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**table-map** *name* {**default** {*default value* | **copy** | **ignore**} | **exit** | **map** {**from** *from value* **to** *to value* } | **no**}
	Example:
	`Device(config)# table-map table01`
	`Device(config-tablemap)#`
		Creates a table map and enters the table map configuration mode. In table map configuration mode, you can perform the following tasks:
		• **default**: Configures the table-map. The default behaviour is 'default copy', which can be set or modified to default ignore or default x, where x can be CoS or DSCP or traffic-class or QoS-group or EXP values.
		• **exit**: Exits from the table map configuration mode.
		• **map**: Maps a *from* to a *to* value in the table map.
		• **no**: Negates or sets the default values of the command.

Step 3	**map from** *value* **to** *value*
	Example:
	`Device(config-tablemap)# map from 0 to 2`
	`Device(config-tablemap)# map from 1 to 4`
	`Device(config-tablemap)# map from 24 to 3`
	`Device(config-tablemap)# map from 40 to 6`
	`Device(config-tablemap)# default 0`
	`Device(config-tablemap)#`
		In this step, packets with DSCP value 0 are marked to the traffic-class 2, DSCP value 1 to the traffic-class 4, DSCP value 24 to the traffic-class 3, DSCP value 40 to the traffic-class 6 and all others to the traffic-class 0.
		Note
		The mapping from DSCP to traffic-class values in this example is configured by using the **set** policy map class configuration command as described in a later step in this procedure.

Step 4	**exit**
	Example:
	`Device(config-tablemap)# exit`
	`Device(config)#`
		Returns to global configuration mode.

Step 5	**exit**	Returns to privileged EXEC mode.

	Command or Action	Purpose

	Example:
	`Device(config) exit`
	`Device#`	

Step 6	**show table-map**
	Example:
	`Device# show table-map`
	`Table Map table01`
	`from 0 to 2`
	`from 1 to 4`
	`from 24 to 3`
	`from 40 to 6`
	`default 0`
		Displays the table map configuration.

Step 7	**configure terminal**
	Example:
	`Device# configure terminal`
	`Device(config)#`
		Enters global configuration mode.

Step 8	**policy-map**
	Example:
	`Device(config)# policy-map table-policy`
	`Device(config-pmap)#`
		Configures the policy map for the table map.

Step 9	**class class-default**
	Example:
	`Device(config-pmap)# class class-default`
	`Device(config-pmap-c)#`
		Matches the class to the system default.

Step 10	**set traffic-class dscp table** *table map name*
	Example:
	`Device(config-pmap-c)# set traffic-class dscp table table01`
	`Device(config-pmap-c)#`
		Incoming packets with DSCP value are mapped to hit a particular VOQ which is defined by the traffic-class values in a table-map.

Step 11	**end** **Example:** `Device(config-pmap-c)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device#`	

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

How to Configure QoS Features and Functionality

The following sections provide configurational information about QoS features and functionality.

Configuring Bandwidth

This procedure explains how to configure bandwidth on your device.

Before you begin

You should have created a created a class-map for bandwidth based on traffic-class classification before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queueing** *policy name*
	Example:
	`Device(config)# policy-map type queueing policy_bandwidth01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class traffic-class7`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: In type queueing policy, class-default always matches traffic-class 0.

	Command or Action	Purpose

Step 4	**bandwidth remaining ratio** *ratio*
	Example:
	`Device(config-pmap-c)# bandwidth remaining ratio 10`
	`Device(config-pmap-c)#`
		Configures the bandwidth for the policy map.
		• *ratio*: Ratios can range from 1 to 63.

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating the policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Police

This procedure explains how to configure policing on your device.

Before you begin

You should have created a class map for policing before beginning this procedure.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`Device# configure terminal`
	`Device(config)#`
		Enters global configuration mode.

Step 2	**policy-map** *policy name* **Example:** `Device(config)# policy-map`	Enters policy map configuration mode. Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

	Command or Action	Purpose
	policy_police01 Device(config-pmap)#	
Step 3	**class** *class name* **Example:** Device(config-pmap)# **class class_police01** Device(config-pmap-c)#	Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following: • *word*: Class map name. • **class-default**: System default class matching any otherwise unclassified packets.
Step 4	**police** {*target_bit_rate* [**conform-action** | **pir**] | **cir** {*target_bit_rate* | **percent** *percentage*} | **rate** {*target_bit_rate* | **percent** *percentage*} **conform-action transmit exceed-action** {**drop** [**violate action**] | **set-cos-transmit** | **set-dscp-transmit** | **set-prec-transmit** | **transmit** [**violate action**] }} **Example:** Device(config-pmap-c)# **police 1200000 conform-action transmit exceed-action drop** Device(config-pmap-c)#	The following **police** subcommand options are available: • *target_bit_rate*: Bits per second (from 1200000 to 400000000000). • **conform-action**: Action taken when rate is less than conform burst. • **pir**: Peak Information Rate. • **cir**: Committed Information Rate. • *target_bit_rate*: Target bit rate (from 1200000 to 400000000000). • **percent**: Percentage of interface bandwidth for CIR. • **rate**: Specifies the police rate, PCR for hierarchical policies, or SCR for single-level ATM 4.0 policer policies. • *target_bit_rate*: Target Bit Rate (from from 1200000 to 400000000000). • **percent**: Percentage of interface bandwidth for rate. The following **police conform-action transmit exceed-action** subcommand options are available: • **drop**: Drops the packet. • **set-cos-transmit**: Sets the CoS value and sends it.

	Command or Action	Purpose
		• **set-discard-class-transmit**: Sets the discard-class value and sends it. • **set-dscp-transmit**: Sets the DSCP value and sends it. • **set-prec-transmit**: Rewrites the packet precedence and sends it. • **transmit**: Transmits the packet. **Note** Policer-based markdown actions are only supported using table maps. Only one markdown table map is allowed for each marking field in the device.

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies. **Note** The **show policy-map** command output does not display counters for conformed bytes and exceeded bytes

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Priority

This procedure explains how to configure priority on your device.

Note The device supports giving priority to specified queues. There are seven priority levels available (1 to 7).

Before you begin

You should have created a class map for priority before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queueing** *policy name*
	Example:
	`Device(config)# policy-map type queueing policy_priority01`
	`Device(config-pmap)#`
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy, and enters policy map configuration mode.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class traffic-class7`
	`Device(config-pmap-c)#`
		Specifies the name of the class whose policy you want to create or change, and enters policy class map configuration mode. Command options for policy class map configuration mode include the following:
		• *class name*: Class map name.
		• **class-default**: In type queueing policy, class-default always matches traffic-class 0.

Step 4	**priority level** *level_value*
	Example:
	`Device(config-pmap-c)# priority level 1`
	`Device(config-pmap-c)#`
		(Optional) This command assigns a strict scheduling priority for the class.
		• *level_value*: Specifies the multilevel (1-7) priority queue.
		Note
		• Priority level 1 is the highest priority level, followed by priority levels 2, 3, 4, 5, 6, and 7 respectively. Queues mapped to high priority classes are serviced before queues mapped to lower priority classes.
		• Shaping or policing a priority queue limits the traffic to the configured rate regardless of queue utilization of non-priority queues.

Step 5	**end** **Example:**	Saves configuration changes.

	Command or Action	Purpose
	`Device(config-pmap-c)# end` `Device#`	
Step 6	**show policy-map type queueing** **Example:** `Device# show policy-map type queueing`	(Optional) Displays the runtime representation and statistics of all the queueing policies configured on the device.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Queues and Shaping

The following sections provide configurational information about queueing and shaping.

Configuring Egress Queue Characteristics

Depending on the complexity of your network and your QoS solution, you may need to perform all of the procedures in this section. You need to make decisions about these characteristics:

- Which packets are mapped by DSCP, CoS, or QoS group value to each queue and threshold ID?
- Which traffic class based classification apply to the queues, and if incoming packets are mapped to any of the traffic classes from 0 to 7, so that traffic gets queued in specific VoQ queues?
- How much of the fixed buffer space is allocated to the queues?
- Does the bandwidth of the port need to be rate limited?
- How often should the egress queues be serviced and which technique (shaped, shared, or both) should be used?

Note You can only configure the egress queues on the device, and only traffic-class based classification is supported.

Configuring Queue Limits

Queue-limit can be only be configured in unit of bytes, and total number of thresholds are limited.

Before you begin

The following are prerequisites for this procedure:

- You should have created a class map for the queue limits before beginning this procedure.

- You must have configured either bandwidth, shape, or priority on the policy map prior to configuring the queue limits.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queuing** *policy name*
	Example:
	`Device(config)# policy-map type queueing test`
	`Device(config-pmap)#`
		Specifies the name of the queueing profile policy and enters policy map configuration mode.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class traffic-class7`
	`Device(config-pmap-c)#`
		Specifies the name of the class to be associated with the policy and enters policy class map configuration mode. Command options for policy class map configuration mode include the following:
		• *word*: Class map name.
		• **class-default**: In type queueing policy, class-default always matches traffic-class 0.

Step 4	**shape average** {*bits/s* | **percent**}
	Example:
	`Device(config-pmap-c)# shape average 1000000000`
	`Device(config-pmap-c)#`
		Configures the traffic shaping average. The parameters include:
		• *bits/s*: Use this command to configure a specific value. The range is 1200000 to 400000000000.
		• **percent**: Allocates a maximum bandwidth to a particular class. The value can be from 1 to 100. Shaping is done on the percentage value, and traffic will not exceed the shape percentage value.

Step 5	**queue-limit** *value* [**bytes**] **Example:** `Device(config-pmap-c)# queue-limit 100000 bytes`	Sets the queue limit threshold percentage value in bytes. The range is from 1000000 to 396000000.

	Command or Action	Purpose

Step 6	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Exits policy class map configuration mode and enters privileged EXEC mode.

Configuring Shaping

You use the **shape** command to configure shaping (maximum bandwidth) for a particular class. The queue's bandwidth is restricted to this value even though the port has additional bandwidth left. You can configure shaping as an average percent, as well as a shape average value in bits per second.

Before you begin

You should have created a class map for shaping before beginning this procedure.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queuing** *policy name*
	Example:
	`Device(config)# policy-map type queuing policy_shaping01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.

Step 3	**class** *class name*
	Example:
	`Device(config-pmap)# class class traffic-class7`
	`Device(config-pmap-c)#`
		Enters policy class map configuration mode. Specifies the name of the class whose policy you want to create or change. Command options for policy class map configuration mode include the following:
		• *class name*: Class map name.
		• **class-default**: In type queueing policy, class-default always matches traffic-class 0.

Step 4	**shape average** {*target bit rate* | **percent** *percentage*}	Configures the average shape rate. You can configure the average shape rate by target bit rates (bits per second) or by percentage of

	Command or Action	Purpose

	Example:
	`Device(config-pmap-c)# shape average percent 50`
	`Device(config-pmap-c)#`
		interface bandwidth for the Committed Information Rate (CIR).

Step 5	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Saves configuration changes.

Step 6	**show policy-map** **Example:** `Device# show policy-map`	(Optional) Displays policy configuration information for all classes configured for all service policies.

What to do next

Configure any additional policy maps for QoS for your network. After creating your policy maps, attach the traffic policy or polices to an interface using the **service-policy** command.

Configuring Sharped Profile Queuing

This procedure explains how to configure sharped profile queuing on your switch:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**policy-map type queueing** *policy name*
	Example:
	`Device(config)# policy-map type queueing policy_shaping01`
	`Device(config-pmap)#`
		Enters policy map configuration mode.
		Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
		policy name is the name of the child policy map. The name can be a maximum of 40 alphanumeric characters.

Step 3	**class** class name **Example:**	Enters policy class map configuration mode. Specifies the name of the class whose policy

	Command or Action	Purpose
	`Device(config-pmap)# class traffic-class1` `Device(config-pmap-c)#`	you want to create or change. Command options for policy class map configuration mode include the following: • *class name*: Class map name. • **class-default**: System default class matching any otherwise unclassified packets.

Step 4	**bandwidth remaining ratio** *ratio*
	Example:
	`Device(config-pmap)# bandwidth remaining ratio 10`
	`Device(config-pmap-c)#`
		(Optional) Configures ratio for sharing excess bandwidth. The range is from 1 to 63.

Step 5	**shape average** {*target bit rate* | **percent** *percentage*}
	Example:
	`Device(config-pmap-c)# shape average percent 50`
	`Device(config-pmap-c)#`
		Configures the average shape rate. You can configure the average shape rate by target bit rates (bits per second) or by percentage of interface bandwidth for the Committed Information Rate (CIR).

Step 6	**end**
	Example:
	`Device(config-pmap-c)# end`
	`Device#`
		Exits policy class map configuration mode and returns to privileged EXEC mode.

Sharped Profile Queuing Configuration

The following is a sample output for sharped queuing:

```
Device# show policy-map type queueing parent class class-default

   Class class-default
      Average Rate Traffic Shaping
      cir 5%
  Policy Map type queueing child
    Class tc7
      priority level 1
      Average Rate Traffic Shaping
      cir 1500000000 (bps)
    Class class-default
```

Monitoring QoS

The following commands can be used to monitor QoS on the device:

Table 13: Monitoring QoS

Command	Description
show class-map [*class_map_name*]	Displays a list of all class maps configured.
show policy-map [**type queueing**] [*policy_map_name*]	Displays a list of all policy maps configured. Command parameters include: • policy map name • type queueing
show policy-map interface {**TenGigE** | | **HundredGigE** | }	Displays the runtime representation and statistics of all the policies configured on the device. Command parameters include: • TenGigabitEthernet: 10-Gigabit Ethernet • FortyGigabitEthernet: 40-Gigabit Ethernet • FiftyGigabitEthernet: 25-Gigabit Ethernet • FiftyGigabitEthernet: 50-Gigabit Ethernet • HundredGigabitEthernet: 100-Gigabit Ethernet • FourHundredGigabitEthernet: 400-Gigabit Ethernet Cisco Catalyst 9500 Series Switches support breakout as 4x10 G and 4x25 G. On Cisco Catalyst 9500X series switches, breakout is supported only on odd 100-G ports and on all 400-G ports. These switches support breakout as 2x100 G and 4x100 G. **Note** Though **wireless** option is visible on the CLI, it is not supported.

Command	Description
show policy-map type queueing interface {**TenGigE** | | **HundredGigE** | }	Displays the runtime representation and statistics of all the queuing policies configured on the device. Command parameters include: • TenGigabitEthernet: 10-Gigabit Ethernet • FortyGigabitEthernet: 40-Gigabit Ethernet • FiftyGigabitEthernet: 25-Gigabit Ethernet • FiftyGigabitEthernet: 50-Gigabit Ethernet • HundredGigabitEthernet: 100-Gigabit Ethernet • FourHundredGigabitEthernet: 400-Gigabit Ethernet Cisco Catalyst 9500 Series Switches support breakout as 4x10 G and 4x25 G. On Cisco Catalyst 9500X series switches, breakout is supported only on odd 100-G ports and on all 400-G ports. These switches support breakout as 2x100 G and 4x100 G. **Note** Though **wireless** option is visible on the CLI, it is not supported.
show table-map	Displays all the table maps and their configurations.

Configuration Examples for QoS

The following sections provide configuration examples for QoS.

Examples: TCP Protocol Classification

TCP packets can be classified based on port numbers. The configuration for TCP protocol is as follows:

```
Device# show ip acce tcp

Extended IP access list tcp
    10 permit tcp any any eq 80
Device #
```

```
Device #show run class-map tcp

Current configuration : 63 bytes
!
class-map match-all tcp
 match access-group name tcp
!
end

Device# show run policy-map tcp

Current configuration : 56 bytes
!
policy-map tcp
 class tcp
  police 1000000000
!
end

Device# show run int fif 1/0/1

Current configuration : 93 bytes
!
interface FiftyGigE1/0/1
 no ip address
 no keepalive
 service-policy input tcp
end

Device #
```

Examples: UDP Protocol Classification

UDP packets can be classified based on port numbers. The configuration example for UDP protocol is as follows:

```
Device# show ip acce udp

Extended IP access list udp
    10 permit udp any any eq ntp
Device #

Device# show run class-map udp
Building configuration...

Current configuration : 63 bytes
!
class-map match-all udp
 match access-group name udp
!
end

Device# show run policy-map udp
Building configuration...

Current configuration : 56 bytes
!
policy-map udp
 class udp
  police 1000000000
!
end
```

```
Device# show run int fif 1/0/1

Current configuration : 93 bytes
!
interface FiftyGigE1/0/1
 no ip address
 no keepalive
 service-policy input udp
end

Device #
```

Examples: RTP Protocol Classification

RTP packets can be classified based on port numbers. The configuration example for RTP protocol is as follows:

```
Device# show ip access-list rtp

Extended IP access list rtp
    10 permit udp any any eq 554
    11 permit tcp any any eq 554
Device #

Device# show run class-map rtp

Current configuration : 63 bytes
!
class-map match-all rtp
 match access-group name rtp
!
end

Device# show run policy-map rtp

Current configuration : 56 bytes
!
policy-map rtp
 class rtp
  police 1000000000
!
end

Device# show run int FiftyGigE1/0/1

Current configuration : 93 bytes
!
interface FiftyGigE1/0/1
 no ip address
 no keepalive
 service-policy input rtp
end

Device #
```

Examples: Classification by Access Control Lists

This example shows how to classify packets for QoS by using access control lists (ACLs):

```
Device# configure terminal
Device(config)# access-list 101 permit ip host 12.4.1.1 host 15.2.1.1
Device(config)# class-map acl-101
Device(config-cmap)# description match on access-list 101
Device(config-cmap)# match access-group 101
Device(config-cmap)#
```

After creating a class map by using an ACL, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Class of Service Layer 2 Classification

This example shows how to classify packets for QoS using a class of service Layer 2 classification:

```
Device# configure terminal
Device(config)# class-map match-any cos
Device(config-cmap)# match cos ?
  <0-7>  Enter up to 4 class-of-service values separated by white-spaces
Device(config-cmap)# match cos 3 4 5
Device(config-cmap)#
```

After creating a class map by using a CoS Layer 2 classification, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Class of Service DSCP Classification

This example shows how to classify packets for QoS using a class of service DSCP classification:

```
Device# configure terminal
Device(config)# class-map match-any dscp
Device(config-cmap)# match dscp af21 af22 af23
Device(config-cmap)#
```

After creating a class map by using a DSCP classification, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Classification by DSCP or Precedence Values

This example shows how to classify packets by using DSCP or precedence values:

```
Device# configure terminal
Device(config)# class-map prec2
Device(config-cmap)# description matching precedence 2 packets
Device(config-cmap)# match ip precedence 2
Device(config-cmap)# exit
Device(config)# class-map ef
Device(config-cmap)# description EF traffic
Device(config-cmap)# match ip dscp ef
Device(config-cmap)# exit

Device(config)# class-map prec5
```

```
Device(config-cmap)# match precedence 5
Device(config-cmap)# exit

Device(config)# class-map cs1
Device(config-cmap)# match dscp cs1
Device(config-cmap)#
```

After creating a class map by using a DSCP or precedence values, you then create a policy map for the class, and apply the policy map to an interface for QoS.

Examples: Hierarchical Policy Configuration

The following is an example of non-queueing hierarchical policy configuration:

Note For hierarchical policy, overlapping actions are not supported. For example, a policer in both parent and child are not supported, and marking in both parent and child are not supported.

```
Device(config)# class-map dscp8
Device(config-cmap)# match dscp 8
Device(config-cmap)# exit

Device(config)# class-map pre2
Device(config-cmap)# match pre 2
Device(config-cmap)# end

Device(config)# policy-map
Device(config)# policy-map child
Device(config-pmap)# class dscp8
Device(config-pmap-c)# set dscp af11
Device(config-pmap-c)# class pre2
Device(config-pmap-c)# set precedence 5
Device(config-pmap-c)# !
Device(config-pmap-c)# end

Device(config)# policy-map parent
Device(config-pmap)# class class-default
Device(config-pmap-c)# police rate percent 20
Device(config-pmap-c-police)# service-policy child
Device(config-pmap-c)# !
Device(config-pmap-c)# end
```

The following is an example of queueing hierarchical policy configuration:

```
Device(config)# policy-map type queueing child_queue
Device(config-pmap)# class traffic-class7
Device(config-pmap-c)# shape average percent 10
Device(config-pmap-c)# priority level 1
Device(config-pmap-c)# class traffic-class6
Device(config-pmap-c)# shape average percent 20
Device(config-pmap-c)# bandwidth remaining ratio 10
Device(config-pmap-c)# !
Device(config-pmap-c)# end

Device(config)# policy-map type queueing parent_queue
Device(config-pmap)# class class-default
Device(config-pmap-c)# shape average percent 80
Device(config-pmap-c)# service-policy child_queue
```

```
Device(config-pmap-c)# !
Device(config-pmap-c)# end
```

The following is an example of a non-queueing hierarchical policy configuration using table maps:

```
Device(config)# table-map dscp2dscp
Device(config-tablemap)# default copy
Device(config)# exit

Device(config)# policy-map child_policy
Device(config-pmap)# class dscp8
Device(config-pmap-c)# police rate 1000000000
Device(config-pmap-c-police)# class pre2
Device(config-pmap-c)# police rate 2000000000
Device(config-pmap-c-police)# !
Device(config-pmap-c-police)# end

Device(config)# policy-map parent_policy
Device(config-pmap)# class class-default
Device(config-pmap-c)# set dscp dscp table dscp2dscp
Device(config-pmap-c)# service-policy child_policy
Device(config-pmap-c)# !
Device(config-pmap-c)# end
```

Examples: Classification for Voice and Video

This example describes how to classify packet streams for voice and video using device specific information.

In this example, voice and video are coming in from end-point A into HundredGigabitEthernet1/0/1on the device and have precedence values of 5 and 6, respectively. Additionally, voice and video are also coming from end-point B into FortyGigabitEthernet1/0/2 on the device with DSCP values of EF and AF11, respectively.

Assume that all the packets from the both the interfaces are sent on the uplink interface, and there is a requirement to police voice to 100 Mbps and video to 150 Mbps.

To classify per the above requirements, a class to match voice packets coming in on HundredGigabitEthernet1/0/1 is created, named voice-interface-1, which matches precedence 5. Similarly another class for voice is created, named voice-interface-2, which will match voice packets in HundredGigabitEthernet1/0/3. These classes are associated to two separate policies named input-interface-1, which is attached to HundredGigabitEthernet1/0/1, and input-interface-2, which is attached to HundredGigabitEthernet1/0/3. The action for this class is to mark the qos-group to 10. To match packets with QoS-group 10 on the output interface, a class named voice is created which matches on QoS-group 10. This is then associated to another policy named output-interface, which is associated to the uplink interface. Video is handled in the same way, but matches on QoS-group 20.

The following example shows how classify using the above device specific information:

```
Device(config-cmap)# class-map voice-interface-1
Device(config-cmap)# match ip precedence 5
Device(config-cmap)# exit

Device(config)# class-map video-interface-1
Device(config-cmap)# match ip precedence 6
Device(config-cmap)# exit

Device(config)# class-map voice-interface-2
Device(config-cmap)# match ip dscp ef
Device(config-cmap)# exit
```

```
Device(config)# class-map video-interface-2
Device(config-cmap)# match ip dscp af11
Device(config-cmap)# exit

Device(config)# policy-map input-interface-1
Device(config-pmap)# class voice-interface-1
Device(config-pmap-c)# set qos-group 10
Device(config-pmap-c)# set traffic-class 7

Device(config-pmap-c)# class video-interface-1
Device(config-pmap-c)# set qos-group 20
Device(config-pmap-c)# set traffic-class 6

Device(config-pmap-c)# policy-map input-interface-2
Device(config-pmap)# class voice-interface-2
Device(config-pmap-c)# set qos-group 10
Device(config-pmap-c)# set traffic-class 7
Device(config-pmap-c)# class video-interface-2
Device(config-pmap-c)# set qos-group 20
Device(config-pmap-c)# set traffic-class 6

Device(config)# class-map match-all traffic-class7
Device(config-cmap)# match traffic-class 7
Device(config-cmap)# exit

Device(config)# class-map match-all traffic-class6
Device(config-cmap)# match traffic-class 6

Device(config)# policy-map type queueing output-interface
Device(config-pmap)# class traffic-class7
Device(config-pmap-c)# shape average 2g
Device(config-pmap-c)# class traffic-class6
Device(config-pmap-c)# shape average 1g
Device(config-pmap-c)# end
```

Examples: Queue-limit Configuration

The following example shows how to configure a queue-limit policy based on per traffic-class:

```
Device# configure terminal
Device#(config)# policy-map type queueing test
Device#(config-pmap)# class tc7
Device#(config-pmap-c)# shape average 1000000000
Device#(config-pmap-c)# queue-limit 100000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class tc6
Device#(config-pmap-c)# shape average 2000000000
Device#(config-pmap-c)# queue-limit 2000000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class tc5
Device#(config-pmap-c)# shape average 3000000000
Device#(config-pmap-c)# queue-limit 3000000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class tc4
Device#(config-pmap-c)# shape average 4000000000
Device#(config-pmap-c)# queue-limit 100000000 bytes
Device#(config-pmap-c)# exit
```

```
Device#(config-pmap)# class tc3
Device#(config-pmap-c)# shape average 5000000000
Device#(config-pmap-c)# queue-limit 200000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class tc2
Device#(config-pmap-c)# shape average 4000000000
Device#(config-pmap-c)# queue-limit 300000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class tc1
Device#(config-pmap-c)# shape average 3000000000
Device#(config-pmap-c)# queue-limit 100000000 bytes
Device#(config-pmap-c)# exit

Device#(config-pmap)# class class-default
Device#(config-pmap-c)# shape average 2000000000
Device#(config-pmap-c)# queue-limit 100000000 bytes
Device#(config-pmap-c)# end
Device#
```

Examples: Policing Action Configuration

The following example displays the various policing actions that can be associated to the policer. These actions are accomplished using the conforming, exceeding, or violating packet configurations. You have the flexibility to drop, mark and transmit, or transmit packets that have exceeded or violated a traffic profile.

For example, a common deployment scenario is one where the enterprise customer polices traffic exiting the network towards the service provider and marks the conforming, exceeding and violating packets with different DSCP values. The service provider could then choose to drop the packets marked with the exceeded and violated DSCP values under cases of congestion, but may choose to transmit them when bandwidth is available.

Note The Layer 2 fields can be marked to include the CoS fields, and the Layer 3 fields can be marked to include the precedence and the DSCP fields.

One useful feature is the ability to associate multiple actions with an event. For example, you could set the precedence bit and the CoS for all conforming packets. A submode for an action configuration could then be provided by the policing feature.

This is an example of a policing action configuration:

```
Device# configure terminal
Device(config)# policy-map police
Device(config-pmap)# class class-default
Device(config-pmap-c)# police cir 1000000 pir 2000000
Device(config-pmap-c-police)# conform-action transmit
Device(config-pmap-c-police)# exceed-action set-dscp-transmit dscp table exceed-markdown-table
Device(config-pmap-c-police)# violate-action set-dscp-transmit dscp table
violate-markdown-table
Device(config-pmap-c-police)# end
```

In this example, the exceed-markdown-table and violate-mark-down-table are table maps.

Note Policer-based markdown actions are only supported using table maps. Only one markdown table map is allowed for each marking field in the device.

Examples: Policing Units

The policing unit is the basis on which the token bucket works. CIR and PIR are specified in bits per second. The burst parameters are specified in bytes. This is the default mode; it is the unit that is assumed when no units are specified. The CIR and PIR can also be configured in percent, in which case the burst parameters have to be configured in milliseconds.

The following is an example of a policer configuration in bits per second. In this configuration, a dual-rate three-color policer is configured where the units of measurement is bits. The burst and peak burst are all specified in bits.

```
Device(config)# policy-map bps-policer
Device(config-pmap)# class class-default
Device(config-pmap-c)# police rate 1200000 peak-rate 12000000
conform-action transmit exceed-action set-dscp-transmit dscp table
DSCP_EXCE violate-action drop
```

Examples: Single-Rate Two-Color Policing Configuration

The following example shows how to configure a single-rate two-color policer:

```
Device(config)# class-map match-any prec1
Device(config-cmap)# match ip precedence 1
Device(config-cmap)# exit
Device(config)# policy-map policer
Device(config-pmap)# class prec1
Device(config-pmap-c)# police cir 256000 conform-action transmit exceed-action drop
Device(config-pmap-c-police)# exit
Device(config-pmap-c)#
```

Examples: Dual-Rate Three-Color Policing Configuration

The following example shows how to configure a dual-rate three-color policer:

```
Device# configure terminal
Device(config)# policy-map test
Device(config-pmap)# class class-default
Device(config-pmap-c)# police cir 1000000000 pir 2000000000
Device(config-pmap-c-police)# conform-action transmit
Device(config-pmap-c-police)# exceed-action set-dscp-transmit dscp table exceed-markdown-table
Device(config-pmap-c-police)# violate-action set-dscp-transmit dscp table
violate-markdown-table
Device(config-pmap-c-police)# exit
Device(config-pmap-c)#
```

In this example, the exceed-markdown-table and violate-mark-down-table are table maps.

Note Policer based markdown actions are only supported using table maps.

Examples: Table Map Marking Configuration

The following steps and examples show how to use table map marking for your QoS configuration:

1. Define the table-map using the **table-map** command and indicate the mapping of the values. This table does not know of the policies or classes within which it will be used. The default command in the table map indicates the value to be copied into the 'to' field when there is no matching 'from' field. In the example, a table map named table-map1 is created. The mapping defined is to convert the value from 0 to 1 and from 2 to 3, while setting the default value to 4.

Note Only table-map values of same QoS tags are supported. For example, a table-map of type CoS to CoS or DSCP to DSCP or Prec to Prec are supported, and also a table-map of type DSCP/CoS/PREC to a QoS group and a table-map of type DSCP/CoS/PREC to a traffic-class.

```
Device(config)# table-map table-map1
Device(config-tablemap)# map from 0 to 1
Device(config-tablemap)# map from 2 to 3
Device(config-tablemap)# default 4
Device(config-tablemap)# exit
```

2. Define the policy map where the table map will be used.

   In the example, the incoming CoS is mapped to the CoS based on the mapping specified in the table table-map1. For this example, if the incoming packet has a CoS of 0, the CoS in the packet is set 1. If no table map name is specified the command assumes a default behavior where the value is copied as is from the 'from' field to the 'to' field.

```
Device(config)# policy-map policy1
Device(config-pmap)# class class-default
Device(config-pmap-c)# set cos cos table table-map1
Device(config-pmap-c)# exit
```

3. Associate the policy to an interface.

```
Device(config)# interface HundredGigE1/0/2
Device(config-if)# service-policy output policy1
Device(config-if)# exit
```

Displaying QoS Configuration

The following is a sample output of the **show policy-map** command:

```
Device# show policy-map mul4

  Policy Map mul4
    Class cs2
     police cir 1000000000 bc 1024000
       conform-action transmit
       exceed-action drop
      set dscp cs7
    Class cs3
      set traffic-class 7
     police cir 40000000000 bc 1024000
       conform-action transmit
       exceed-action drop
    Class cs1
     police cir 1000000000 bc 1024000
       conform-action transmit
       exceed-action drop
      set traffic-class 5
    Class cs5
     police cir 20000000000 bc 1024000
       conform-action transmit
       exceed-action drop
      set traffic-class 5
    Class cs6
     police cir 50000000 bc 10240
       conform-action transmit
       exceed-action drop
    Class dscp1
     police cir 5500000000 bc 1024000
       conform-action transmit
       exceed-action drop
      set traffic-class 2
      set dscp 45
    Class ttl
     police cir 1000000000 bc 1024000
       conform-action transmit
       exceed-action drop
    Class cs4
     police cir 10000000000 bc 1024000
       conform-action transmit
       exceed-action drop
      set traffic-class 5
    Class class-default
     police cir 1000000000 bc 1024000
       conform-action transmit
       exceed-action drop
```

The following is a sample output from the **show policy-map interface** command:

```
Device# show policy-map interface HundredGigE1/0/14

 HundredGigE1/0/14

  Service-policy input: mul4

    Class-map: cs2 (match-all)
      0 packets
      Match:  dscp cs2 (16)
      police:
          cir 1000000000 bps, bc 1024000 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          drop
        conformed 0000 bps, exceeded 0000 bps
```

```
  QoS Set
    dscp cs7

Class-map: cs3 (match-all)
  224757946122 packets
  Match:  dscp cs3 (24)
  QoS Set
    traffic-class 7
  police:
      cir 40000000000 bps, bc 1024000 bytes
    conformed 335092644777000 bytes; actions:
      transmit
    exceeded 2044274406000 bytes; actions:
      drop
    conformed 0000 bps, exceeded 0000 bps

Class-map: cs1 (match-all)
  0 packets
  Match:  dscp cs1 (8)
  police:
      cir 1000000000 bps, bc 1024000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      drop
    conformed 0000 bps, exceeded 0000 bps
  QoS Set
    traffic-class 5

Class-map: cs5 (match-all)
  0 packets
  Match:  dscp cs5 (40)
  police:
      cir 20000000000 bps, bc 1024000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      drop
    conformed 0000 bps, exceeded 0000 bps
  QoS Set
    traffic-class 5

Class-map: cs6 (match-all)
  0 packets
  Match:  dscp cs6 (48)
  police:
      cir 50000000 bps, bc 10240 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      drop
    conformed 0000 bps, exceeded 0000 bps

Class-map: dscp1 (match-all)
  0 packets
  Match:  dscp 5
  police:
      cir 5500000000 bps, bc 1024000 bytes
    conformed 0 bytes; actions:
      transmit
    exceeded 0 bytes; actions:
      drop
    conformed 0000 bps, exceeded 0000 bps
  QoS Set
```

```
      traffic-class 2
      dscp 45

  Class-map: ttl (match-all)
    0 packets
    Match: access-group name ttl
    police:
        cir 1000000000 bps, bc 1024000 bytes
      conformed 0 bytes; actions:
        transmit
      exceeded 0 bytes; actions:
        drop
      conformed 0000 bps, exceeded 0000 bps

  Class-map: cs4 (match-all)
    0 packets
    Match:  dscp cs4 (32)
    police:
        cir 10000000000 bps, bc 1024000 bytes
      conformed 0 bytes; actions:
        transmit
      exceeded 0 bytes; actions:
        drop
      conformed 0000 bps, exceeded 0000 bps
    QoS Set
      traffic-class 5

  Class-map: class-default (match-any)
    5073 packets
    Match: any
    police:
        cir 1000000000 bps, bc 1024000 bytes
      conformed 1215000 bytes; actions:
        transmit
      exceeded 6394500 bytes; actions:
        drop
      conformed 0000 bps, exceeded 0000 bps
```

The following is a sample output from the **show policy-map type queue interface** command. This show command displays information about which ingress packets are hitting which VoQs.

```
Device# show policy-map type queue interface HundredGigE1/0/14

 HundredGigE1/0/14

  Service-policy queueing output: llq

    queue stats for all priority classes:
      Queueing
      priority level 1
      queue limit 96000 bytes
      (total drops) 0
      (bytes output) 59924103953524

    queue stats for all priority classes:
      Queueing
      priority level 2
      queue limit 96000 bytes
      (total drops) 0
      (bytes output) 0

    queue stats for all priority classes:
      Queueing
      priority level 3
```

```
  queue limit 96000 bytes
  (total drops) 0
  (bytes output) 0

queue stats for all priority classes:
  Queueing
  priority level 4
  queue limit 96000 bytes
  (total drops) 0
  (bytes output) 0

queue stats for all priority classes:
  Queueing
  priority level 5
  queue limit 96000 bytes
  (total drops) 0
  (bytes output) 0

queue stats for all priority classes:
  Queueing
  priority level 6
  queue limit 96000 bytes
  (total drops) 0
  (bytes output) 0

queue stats for all priority classes:
  Queueing
  priority level 7
  queue limit 96000 bytes
  (total drops) 0
  (bytes output) 0

Class-map: tc7 (match-all)
  67482284685 packets
  Match: traffic-class 7
  shape (average) cir 5000000000, bc 20000000, be 20000000
  target shape rate 5000000000
  Priority: Strict,

  Priority Level: 1

Class-map: tc6 (match-all)
  0 packets
  Match: traffic-class 6
  shape (average) cir 1500000000, bc 6000000, be 6000000
  target shape rate 1500000000
  Priority: Strict,

  Priority Level: 2

Class-map: tc5 (match-all)
  0 packets
  Match: traffic-class 5
  Priority: Strict,

  Priority Level: 3
  shape (average) cir 2000000000, bc 8000000, be 8000000
  target shape rate 2000000000

Class-map: tc4 (match-all)
  0 packets
  Match: traffic-class 4
  Priority: Strict,
```

```
  Priority Level: 4
  shape (average) cir 1500000000, bc 6000000, be 6000000
  target shape rate 1500000000

Class-map: tc3 (match-all)
  0 packets
  Match: traffic-class 3
  Priority: Strict,

  Priority Level: 5
  shape (average) cir 1500000000, bc 6000000, be 6000000
  target shape rate 1500000000

Class-map: tc2 (match-all)
  0 packets
  Match: traffic-class 2
  Priority: Strict,

  Priority Level: 6
  shape (average) cir 1500000000, bc 6000000, be 6000000
  target shape rate 1500000000

Class-map: tc1 (match-all)
  0 packets
  Match: traffic-class 1
  shape (average) cir 40000000000, bc 400000000, be 400000000
  target shape rate 40000000000
  Priority: Strict,

  Priority Level: 7

Class-map: class-default (match-any)
  35230 packets
  Match: any
  Queueing
  queue limit 7500000 bytes
  (total drops) 0
  (bytes output) 0
  shape (average) cir 100000000, bc 400000, be 400000
  target shape rate 100000000
```

Where to Go Next

Review the auto-QoS documentation to see if you can use these automated capabilities for your QoS configuration.

Additional References for QoS

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	*Command Reference (Ca* *Cisco IOS Quality of Serv*

Feature History for QoS

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Cupertino 17.7.1	QoS Functionality	QoS provides preferential treatment to specific types of traffic at the expense of other traffic types. Without QoS, the device offers best-effort service for each packet, regardless of the packet contents or size. This feature was implemented on the C9500X-28C8D model.
Cisco IOS XE Dublin 17.10.1b	QoS Functionality	This feature was implemented on the C9500X-60L4D model.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/.