
# Configuring Quality of Service

This chapter describes the QoS features built into your ML-Series card and how to map QoS scheduling at both the system and interface levels.

This chapter contains the following major sections:



The ML-Series card employs the Cisco IOS Modular QoS CLI (MQC). For more information about general MQC configuration, refer to the following IOS documents:

- *Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.1* at this URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/qos_c/index.htm
- *Cisco IOS Quality of Service Solutions Command Reference, Release 12.1* at this URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/qos_r/index.htm

# Understanding QoS

The ML-Series card multiplexes multiple IP/Ethernet services onto the SONET/SDH circuit and dynamically allocate transmission bandwidth to data services based on data service requirements, which allow the network to operate at a significantly higher level of utilization. To support service-level agreements (SLAs), this dynamic allocation must accommodate the service elements of bandwidth, including loss and delay. The characteristics of these service elements make up QoS.

## Priority Mechanism in IP and Ethernet

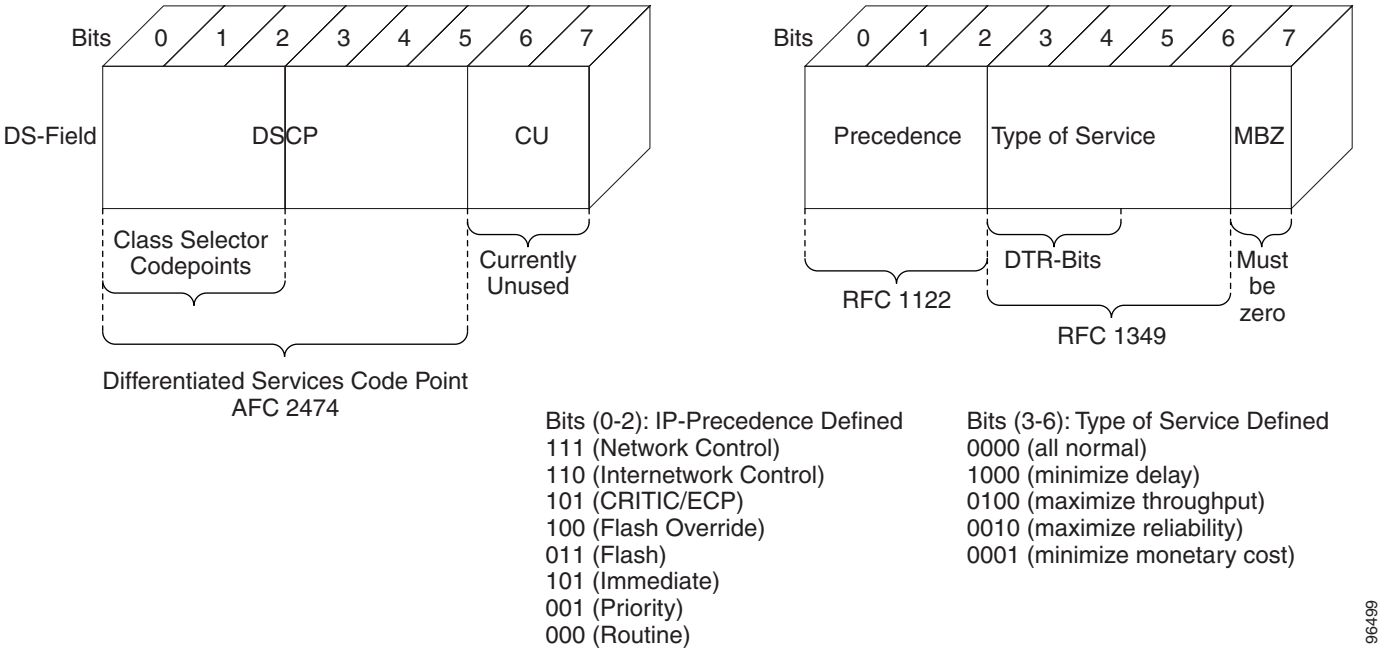
For any QoS service to be applied to data, there must be a way to mark or identify an IP packet or an Ethernet frame. When identified, a specific priority may be assigned to each individual IP packet or Ethernet frame. The IP Precedence or the IP Differentiated Services Code Point (DSCP) field prioritizes the IP packets, and the Ethernet class of service (IEEE 802.1p defined CoS) is used for the Ethernet frames. IP precedence and Ethernet CoS are further described below.

# IP Precedence and Differentiated Services Code Point

IP precedence uses the three precedence bits in the IPv4 header's ToS (Type of Service) field to specify class of service for each IP packet (RFC 1122). The most significant three bits on the IPv4 ToS field provides up to eight distinct classes, of which six are used for classifying services and the remaining two are reserved. On the edge of the network, the IP Precedence is assigned by the client device or the router, so that each subsequent network element can provide services based on the determined policy or the service level agreement (SLA).

IP Differentiated Services Code Point (DSCP) uses the six bits in the IPv4 header to specify class of service for each IP packet (RFC 2474). Figure 13-1 illustrates IP precedence and DSCP. The DSCP field classifies packets into any of the 64 possible classes. On the network edge the IP DSCP is assigned by the client device or the router, so that each subsequent network element can provide services based on the determined policy or the service level agreement.

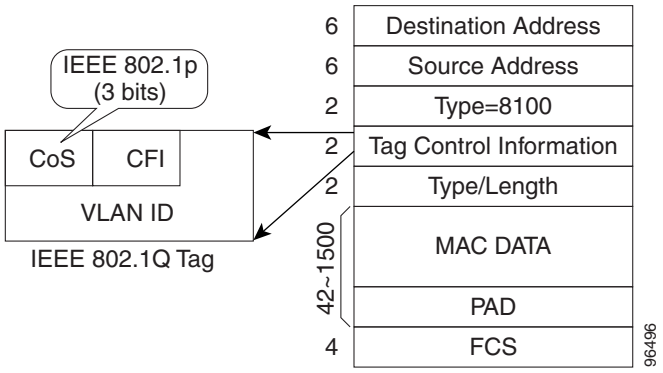
***Figure 13-1 IP Precedence and Differentiated Service Code Point (DSCP)***




# Ethernet CoS

Ethernet CoS refers to three bits within a four byte IEEE 802.1Q (VLAN) header used to indicate the priority of the Ethernet frame as it passes through a switched network. The CoS bits in the 802.1Q header are commonly referred to as the 802.1p bits. There are three CoS bits which provide eight classes, matching the number delivered by IP Precedence. In many real-world networks, a packet may traverse both Layer 2 and Layer 3 domains. To maintain QoS across the network, the IP ToS can be mapped to the Ethernet CoS and vice versa, for example in linear or one-to-one mapping, because each mechanism supports eight classes. Similarly, a set of DSCP values (64 classes) may be mapped into each of the eight individual Ethernet CoS values. Figure 13-2 is an IEEE 802.1Q Ethernet frame, which consists of a 2 byte Ethertype and a 2-byte tag (IEEE 802.1Q Tag) on the Ethernet protocol header.

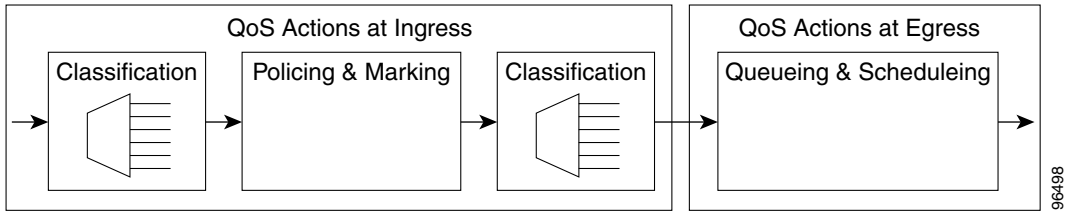
*Figure 13-2 Ethernet frame and the CoS bit (802.1p)*




# ML-Series QoS

The ML-Series QoS classifies each packet in the network based on its input interface, Bridge Group (VLAN), Ethernet CoS, IP Precedence, IP DSCP, or RPR-CoS. Once classified into class flows, further QoS functions can be applied to each packet as it traverses the card. Figure 13-3 illustrates the ML-Series QoS flow.

*Figure 13-3 ML-Series QoS Flow*




Policing provided by the ML-Series card ensures that attached equipment does not submit more than a pre-defined amount of bandwidth (Rate Limiting) into the network. The policing feature can be used to enforce the committed information rate (CIR) and the peak information rate (PIR) available to a customer at an interface. Policing also helps characterize the statistical nature of the information allowed into the network so that traffic engineering can more effectively ensure that the amount of committed bandwidth is available on the network, and the peak bandwidth is over-subscribed with an appropriate ratio. The policing action is applied per classification.

Priority marking can set the Ethernet 802.1p CoS bits or RPR-CoS bits as they exit the ML-Series card. The marking feature operates on the outer 802.1p tag, and provides a mechanism for tagging packets at the ingress of a QinQ packet. The subsequent network elements can provide QoS based only on this service-provider-created QoS indicator.

Per class flow queuing enables fair access to excess network bandwidth, allows allocation of bandwidth to support SLAs, and ensures that applications with high network resource requirements are adequately served. Buffers are allocated to queues dynamically from a shared resource pool. The allocation process incorporates the instantaneous system load as well as the allocated bandwidth to each queue to optimize buffer allocation. Congestion management on the ML-Series is performed through a tail drop mechansim along with discard eligibility on the egress scheduler.

The ML-Series uses a Weighted Deficit Round Robin (WDRR) scheduling process to provide fair access to excess bandwidth as well as guaranteed throughput to each class flow.

Admission control is a process that is invoked each time that service is configured on the ML-Series card to ensure that QoS resources are not over committed. In particular, admission control ensures that no configurations are accepted, where a sum of the committed bandwidths on an interface, exceeds total bandwidth on the interface.

# Classification

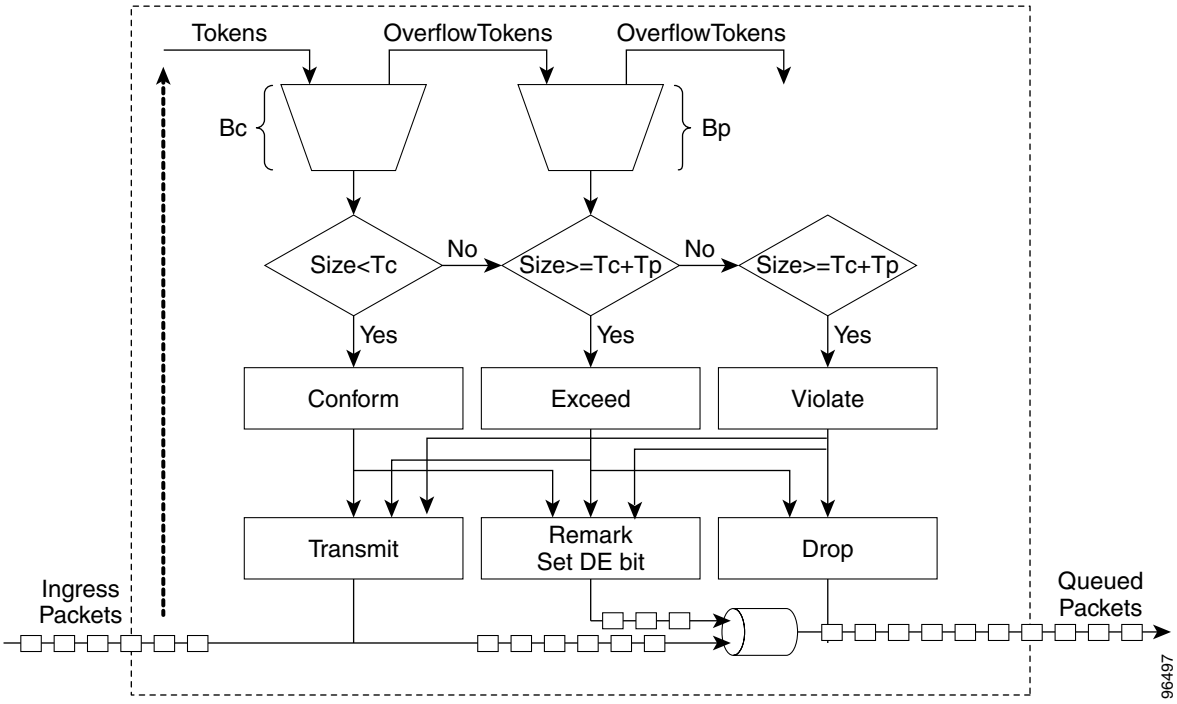
Classification can be based on any single packet classifiction criteria or a combination (logical AND and OR). A total of 254 classes, not including the class default, can be defined on the card. Classification of packets is configured using the Modular CLI **class-map** command. For traffic transiting the RPR, only the Input Interface and/or the RPR-CoS can be used as classification criteria.

# Policing

Dual leaky bucket policer is a process where the first bucket (CIR bucket) is filled with tokens at a known rate (CIR), which is a parameter that can be configured by the operator. Figure 13-4 illustrates the dual leaky bucket policer model. The tokens fill the bucket up to a maximum level, which is the amount of burstable committed (BC) traffic on the policer. The non-conforming packets of the first bucket are the overflow packets, which are passed to the second leaky bucket (the PIR bucket). The second leaky bucket is filled with these tokens at a known rate (PIR), which is a parameter that can be configured by the operator. The tokens fill the PIR bucket up to a maximum level (BP), which is the amount of peak burstable traffic on the policer. The non-conform packets of the second bucket are the overflow packets, which can be dropped or marked according to the policer definition.

On the dual leaky bucket policer, the packets conforming to the CIR are conform packets, the packets not conforming to CIR but conforming to PIR are exceed packets, and the packets not conforming to either the PIR or CIR are violate packets.

*Figure 13-4 Dual Leaky Bucket Policer Model*




# Marking and Discarding

On the ML-Series card's policer, the conform packets can be transmitted or marked and transmitted. The exceed packets can be transmitted, marked and transmitted, or dropped. The violating packets can be transmitted, marked and transmitted, or dropped. The primary application of the dual-rate or three-color policer is to mark the conform packets with CoS bit 2l, mark the exceed packet with CoS bit 1, and discard the violated packets so all the subsequent network devices can implement the proper QoS treatment per frame/packet basis based on these priority marking without knowledge of each SLA.

If a marked packet has a provider-supplied Q-Tag inserted before transmission, the marking only affects the provider Q-Tag. If a Q-Tag was received, it will be re-marked. If a marked packet is transported over the RPR ring, the marking also affects the RPR-CoS bit.

If a Q-Tag is inserted (QinQ), the marking affects the added Q-Tag. If the ingress packet contains a Q-tag and is transparently switched, the existing Q-Tag is marked. In case of a packet without any Q-Tag, the marking does not have any significance.

The local scheduler treats all non-conforming packets as discard eligible regardless of their CoS setting or the global cos commit definition. For RPR implementation, the discard eligible (DE) packets are marked using the DE bit on the RPR header. The discard eligibility based on the CoS Commit or the policing action is local to the ML-Series card scheduler, but it is global for the RPR ring.

# Queuing

ML-Series card queuing uses a shared buffer pool to allocate memory dynamically to different traffic queues. The ML-Series card uses a total of 12 MB memory for the buffer pool. Ethernet ports share 6 MB of the memory, and POS ports share the remaining 6 MBs of memory. Memory space is allocated in 1500-byte increments.

Each queue has an upper limit on the allocated number of buffers based on the class bandwidth assignment of the queue and the number of queues configured. This upper limit is typically 30% to 50% of the shared buffer capacity. Dynamic buffer allocation to each queue may be reduced based on the number of queues needing extra buffering. The dynamic allocation mechanism provides fairness in proportion to service commitments as well as optimization of system throughput over a range of system traffic loads.

The Low Latency Queue (LLQ) is defined by setting the weight to infinity or committing 100% bandwidth. When a LLQ is defined, a policer should also be defined on the ingress for that specific class to limit the maximum bandwidth consumed by the LLQ; otherwise there may be a potential risk of LLQ occupying the whole bandwidth and starving the other unicast queues.

The ML-Series includes support for 400 user-definable queues, which are assigned per the classification and bandwidth allocation definition. The classification used for scheduling classifies the frames/packet after the policing action, so if the policer is used to mark or change the CoS bits of the ingress frames/packet, the new values are applicable for the classification of traffic for queuing and scheduling. The ML-Series provides buffering for 4000 packets.

# Scheduling

Scheduling is provided by a series of schedulers that perform a weighted deficit round robin (WDRR) as well as priority scheduling mechanisms from the queued traffic associated with each egress port.

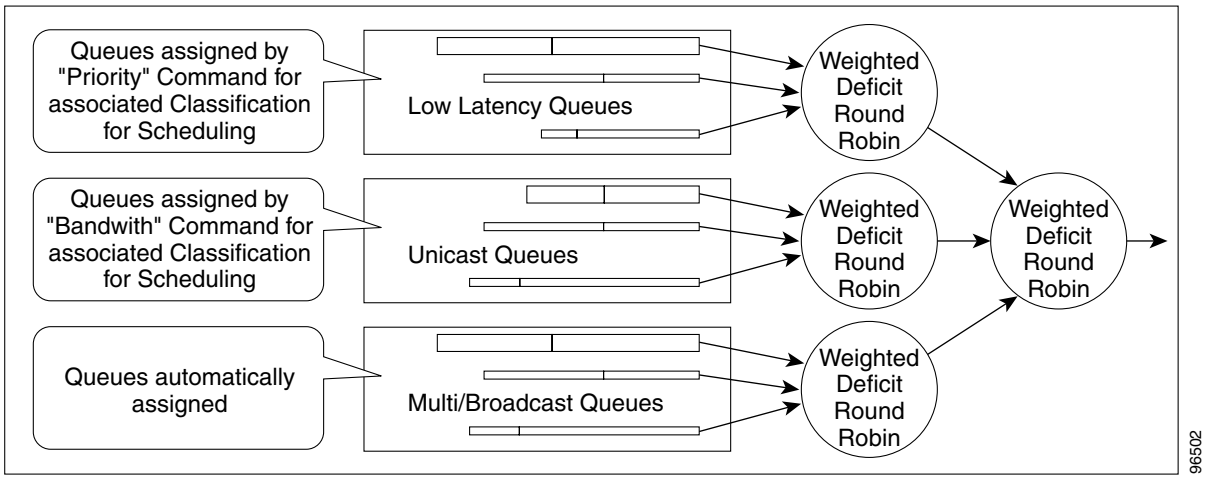
Though ordinary round robin servicing of queues can be done in constant time, the unfairness that occurs when different queues use different packet sizes. Deficit Round Robin (DRR) scheduling solves this problem. If a queue was not able to send a packet in its previous round because its packet size was too large, the remainder from the previous amount of credits a queue gets in each round (quantum) is added to the quantum for the next round.

WDRR extends the quantum idea from the DRR to provide weighted throughput for each queue. Different queues have different weights, and the quantum assigned to each queue in its round is proportional to the relative weight of the queue among all the queues serviced by that scheduler.

Weights are assigned to each queue as a result of the service provisioning process. When coupled with policing and policy mapping provisioning, these weights and the WDRR scheduling process ensure that QoS commitments are provided to each service flow.

Figure 13-5 illustrates the ML-Series card's queuing and scheduling.

*Figure 13-5 Queuing and Scheduling Model*




The weighting structure allows traffic to be scheduled at 1/2048 of the port rate. This equates to approximately 488 kbps for traffic exiting a GigabitEthernet port, approximately 293 kbps for traffic exiting an OC-12c port, and approximately 49 kbps for traffic exiting a FastEthernet port.

The multicast/broadcast queue is automatically created on every egress port of the ML-Series card with a committed bandwidth of 10%. This queue is used for multicast/broadcast data traffic, control traffic, L2 protocol tunnelling, and flooding traffic of the unknown MAC during MAC learning. If the aggregate of multicast/broadcast traffic at any egress port exceeds 10% of the bandwidth those frames beyond 10% of the bandwidth, are treated as best effort by the scheduler.

The unicast queues are created as the output service policy implementation on the egress ports. Each unicast queue is assigned with a committed bandwidth and the weight of the queue is determined by the normalization of committed bandwidth of all defined unicast queues for that port. The traffic beyond the committed bandwidth on any queue is treated by the scheduler according to the relative weight of the queue.

The LLQ is created as the output service policy implementation on the egress ports. Each LLQ queue is assigned with a committed bandwidth of 100% and is served with lower latency. To limit the bandwidth usage by the LLQ, a strict policer needs to be implemented on the ingress for the LLQ traffic classes.

The discard eligibility (DE) allows some packets to be treated as committed and some as discard-eligible on the scheduler. For the Ethernet frames the CoS (802.1p) bits are used to identify committed and discard eligible packets, where the RPR-CoS and the DE bits are used for RPR traffic. When congestion occurs and a queue begins to fill, the discard-eligible packets hit a lower tail-drop threshold than the committed packets. Committed packets are not dropped until the total committed load exceeds the interface output. The tail-drop thresholds adjust dynamically in the card to maximize use of the shared buffer pool while guaranteeing fairness under all conditions.

# Multicast QoS

On the ML-Series cards, multicast (including IP-multicast) and broadcast traffic forwarding is supported at line-rate; however the QoS implementation on multicast traffic varies from the unicast QoS. The difference is in the priority handling for the multicast traffic on the scheduler.

For unicast packets, the priority is defined by the **bandwidth** command, which creates a CIR for the unicast packets in a particular class.

The priority handling of multicast packets is not based on the **bandwidth** command. Instead, multicast frames are assigned to a queue that has a committed bandwidth of 10% of the port bandwidth. If the multicast and broadcast traffic exceeds 10% of the port bandwidth, frames exceeding 10% are given low priority (best effort). The10% committed bandwidth for multicast is applied to the aggregate traffic and does not allow the multicast traffic of one customer to be given higher priority than another customer, unlike the QoS model for unicast traffic.

The scheduler allocates 10% of the bandwidth for multicast and broadcast traffic. Any other QoS implementation is not applicable for multicast and broadcast traffic except the allocation of 10% bandwidth for all multicast/broadcast traffics. Buffers are allocated to queues dynamically from a shared resource pool.

# Control Packets and L2 Tunneled Protocols

The control packets originated by the ML-Series card have a higher priority than data packets. The external Layer 2 and Layer 3 control packets are handled as data packets and assigned to broadcast queues. Bridge protocol data unit (BPDU) prioritization in the ML-Series card gives Layer 2-tunneled BPDU sent out the multicast/broadcast queue a higher discard value and therefore a higher priority than than other packets in the multicast/broadcast queue. The Ethernet CoS (802.1p) for Layer 2-tunneled protocols can be assigned by the ML-Series card.

# Priority Marking

Priority marking allows the operator to assign the 802.1p CoS bits of packets that exit the card. This marking allows the operator to use the CoS bits as a mechanism for signaling to downstream nodes the QoS treatment the packet should be given. This feature operates on the outer-most 802.1p CoS field. When used with the QinQ feature, priority marking allows the user traffic (inner Q-Tag) to traverse the network transparently, while providing a means for the network to internally signal QoS treatment at Layer 2.

Priority marking follows the classification process, and therefore any of the classification criteria identified earlier can be used as the basis to set the outgoing 802.1p CoS field. For example, a specific CoS value can be mapped to a specific Bridge Group.

Priority marking is configured using the MQC **set-cos** command. If packets would otherwise leave the card without an 802.1q tag, then the set cos will have no effect on that packet. If an 802.1q tag is inserted in the packet (either a normal tag or a Q-in-Q tag), the inserted tag will have the set cos priority. If an 802.1q tag is present on packet ingress and retained on packet egress, the priority of that tag will be modified.If the ingress interface is an QinQ Access port, and the **set cos** policy-map classifies based on ingress tag priority, this will classify based on the user priority. This is a way to allow the user-tag priority to determine the Service Provider tag priority. When a packet does not match any **set cos** policy-map, the priority of any preserved tag is unchanged and the priority of any inserted 802.1q tag is set to 0.

The **set-cos** command on the output service policy is only applied to unicast traffic. Priority marking for multicast/broadcast traffic can only be achieved by the set cos action of the policing process on the input service policy.

## QinQ Implementation

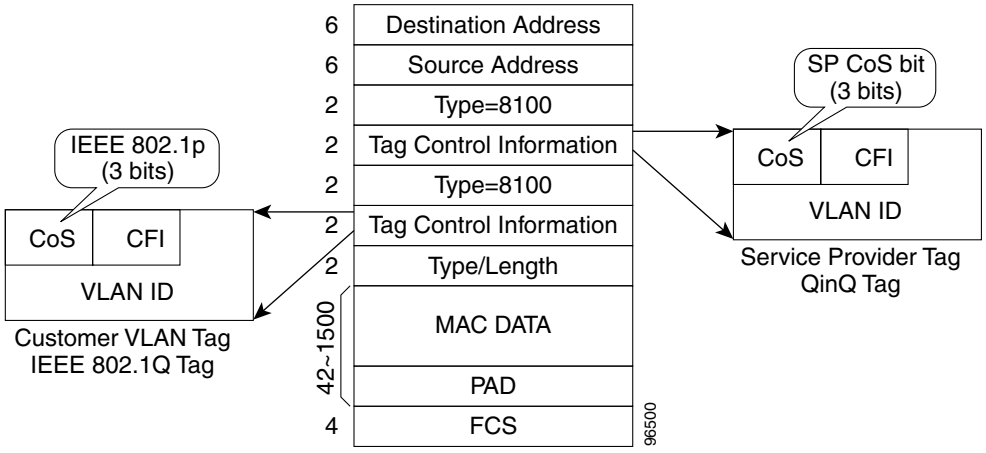
The hierarchical VLAN or 802.1Q tunneling feature enables the service provider to transparently carry the customer VLANs coming from any specific port (UNI) and transport them over the service provider network. This feature is also known as QinQ, which is performed by adding an additional 802.1Q tag on every customer frame.

Using the QinQ feature, service providers can use a single VLAN to support customers with multiple VLANs. QinQ preserves customer VLAN IDs and segregates traffic from different customers within the service-provider infrastructure, even when traffic from different customers originally shared the same VLAN ID. The QinQ also expands VLAN space by using a VLAN-in-VLAN hierarchy and tagging the tagged packets. When the service provider tag is added, the QinQ network typically loses any visibility to the IP header or the customer Ethernet 802.1Q tag on the QinQ encapsulated frames.

On the ML-Series cards, the QinQ access ports (802.1Q tunnel ports or QinQ UNI ports) have visibility to the customer CoS and the IP Precedence or IP DSCP values; therefore, the SP tag can be assigned with proper CoS bit which would reflect the customer IP Precedence, IP DSCP, or CoS bits. In the QinQ network, the QoS is then implemented based on the 802.1p bit of the service provider tag. The ML-Series cards do not have visibility into the customer CoS or IP Precedence or DSCP values after the packet is double-tagged (that is beyond the entry point of the QinQ service).

Figure 13-6 illustrates the QinQ implementation on the ML-Series card.

***Figure 13-6 QinQ***




The ML-Series cards can be used as the 802.1Q tunneling device for the QinQ network and also provide the option to copy the customer frame's CoS bit into the CoS bit of the added QinQ tag. This way the service provider QinQ network can be fully aware of the necessary QoS treatment for each individual customer frames.

## Flow Control Pause and QoS

If flow control and port-based policing are both enabled for an interface, flow control handles the bandwidth. If the policer gets non-compliant flow, then the policer drops or demarks the packets using the policer definition of the interface.

**Note** QoS and policing are not supported on the ML-Series card interface when link aggregation is used.

**Note** Egress shaping is not supported on the ML-Series cards.

# QoS on RPR

For VLAN bridging over RPR, all ML-Series cards on the ring must be configured with the base RPR and RPR QOS configuration. SLA and bridging configurations are only needed at customer RPR access points, where 802.1q VLAN CoS is copied to the RPR CoS. This 802.1q VLAN CoS copying can be overwritten with a **set cos** <*action*> command. The cos commit rule applies at RPR ring ingress. Transit RPR ring traffic is classified on CoS only.

If the packet does not have a VLAN header, the RPR CoS for non-VLAN traffic is set using the following rules:

1. The default CoS is 0.
2. If the packet comes in with an assigned CoS. The assigned CoS replaces the default, or if an IP packet originates locally, the IP Precedence setting replaces the CoS setting.
3. The input policy map has a **set cos** action.
4. The output policy map has a **set cos** action (except for broadcast or multicast packets).

The RPR header contains a CoS value and DE indicator. The RPR DE is set for non-committed traffic.

# Configuring QoS

This section describes the tasks for configuring the ML-Series card QoS functions using the Modular Quality of Service Command-Line Interface (MQC). The ML-Series card does not support the full set of MQC functionality.

To configure and enable class-based QoS features, perform the procedures described in the following sections:

- Creating a Traffic Class, page 13-11
- Creating a Traffic Policy, page 13-12
- Attaching a Traffic Policy to an Interface, page 13-15
- Configuring CoS-based QoS, page 13-16
- Monitoring and Verifying QoS, page 13-16 (Optional)

For QoS configuration examples, see the "Configuration Examples" section on page 13-17.

# Creating a Traffic Class

The **class-map** global configuration command is used to create a traffic class. The syntax of the **class-map** command is as follows:

**class-map** [**match-any** | **match-all**] *class-name*
**no class-map** [**match-any** | **match-all**] *class-name*

The match-all and match-any options need to be specified only if more than one match criterion is configured in the traffic class. The **class-map match-all** command is used when all of the match criteria in the traffic class must be met for a packet to match the specified traffic class. The **class-map match-any** command is used when only one of the match criterion in the traffic class must be met for a packet to match the specified traffic class. If neither the **match-all** nor **match-any** keyword is specified, the traffic class will behave in a manner consistent with **class-map match-all** command.

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

| Command | Purpose |
|---|---|
| `Router(config)# class-map class-map-name` | Specifies the user-defined name of the traffic class. Names can be a maximum of 40 alphanumeric characters. If match-all or match-any are not specified, traffic must match all the match criteria to be classified as part of the traffic class.<br>There is no default-match criteria.<br>Multiple match criteria are supported. The command matches either ALL or ANY of the criteria, as controlled by the **match-all** and **match-any** subcommands of the class-map command. |
| `Router(config)# class-map match-all class-map-name` | Specifies that all match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. |
| `Router(config)# class-map match-any class-map-name` | Specifies that one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. |
| `Router(config-cmap)# match any` | Specifies that all packets will be matched. |
| `Router(config-cmap)# match bridge-group bridge-group-number` | Specifies the bridge-group-number against whose contents packets are checked to determine if they belong to the class. |
| `Router(config-cmap)# match cos cos-number` | Specifies the CoS value against whose contents packets are checked to determine if they belong to the class. |

| Command | Purpose |
|---|---|
| `Router(config-cmap)# match input-interface interface-name` | Specifies the name of the input interface used as a match criterion against which packets are checked to determine if they belong to the class.<br><br>The shared packet ring (SPR) interface, **spr1**, used in resilient packet ring (RPR) is a valid interface-name for the ML-Series card. For more information on the SPR interface, see Chapter 16, "Configuring Resilient Packet Ring."<br><br>The **input-interface** choice is not valid when applied to the INPUT of an interface (redundant). |
| `Router(config-cmap)# match ip dscp ip-dscp-value` | Specifies up to eight differentiated services code point (DSCP) values used as match criteria. The value of each service code point is from 0 to 63. |
| `Router (config-cmap)# match ip precedence ip-precedence-value` | Specifies up to eight IP Precedence values used as match criteria. |

# Creating a Traffic Policy

To configure a traffic policy, use the **policy-map** global configuration command to specify the traffic policy name, and use the following configuration commands to associate a traffic class, which was configured with the **class-map** command and one or more QoS features. The traffic class is associated with the traffic policy when the **class** command is used. The **class** command must be issued after entering policy-map configuration mode. After entering the **class** command, you are automatically in policy-map class configuration mode, which is where the QoS policies for the traffic policy are defined.

When the bandwidth or priority action is used on any class in a policy map, then there must be a class defined by the **match-any** command, which has a bandwidth or priority action in that policy map. This is to ensure that all traffic can be classified into a default class which has some assigned bandwidth. A minimum bandwidth can be assigned if the class is not expected to be used or no reserved bandwidth is desired for default traffic.

The QoS policies that can be applied in the traffic policy in policy-map class configuration mode are detailed in the following example:

The syntax of the **policy-map** command is:

**policy-map** *policy-name*
**no policy-map** *policy-name*

The syntax of the **class** command is:

**class** *class-name*
**no class** *class-name*

All traffic that fails to meet the matching criteria belongs to the default traffic class. The default traffic class can be configured by the user, but cannot be deleted.

To create a traffic policy, use the following commands as needed, beginning in global configuration mode:

| Command | Purpose |
|---|---|
| `Router (config)# policy-map policy-name` | Specifies the name of the traffic policy to configure. Names can be a maximum of 40 alphanumeric characters. |
| `Router (config-pmap)# class class-name` | Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy. |
| `Router (config-pmap)# class class-default` | Specifies the default class to be created as part of the traffic policy. |
| `Router (config-pmap-c)# bandwidth {bandwidth-kbps \| percent percent}` | Specifies a minimum bandwidth guarantee to a traffic class in periods of congestion. A minimum bandwidth guarantee can be specified in kbps or by a percentage of the overall available bandwidth.<br>Valid choices for the ML-Series card are:<br>• Rate in kilobits per second (8 to 2000000)<br>• Percent of total available bandwidth (1 to 100)<br>If multiple classes and bandwidth actions are specified in a single policy map, they must use the same choice in specifying bandwidth (kilobits or percent).<br>**Note** When using the **bandwidth** command, excess traffic (beyond the configured commit) will be allocated any available bandwidth in proportion to the relative bandwidth commitment of it's traffic class compared to other traffic classes. Excess traffic from two classes with equal commits will have equal access to available bandwidth. Excess traffic from a class with a minimum commit might receive only a minimum share of available bandwidth compared to excess bandwidth from a class with a high commit. |

| Command | Purpose |
|---|---|
| `Router (config-pmap-c)# police cir-rate-bps normal-burst-byte [max-burst-byte] [pir pir-rate-bps] [conform-action {set-cos-transmit \| transmit}] [exceed-action {set-cos-transmit \| drop}] [violate-action {set-cos-transmit \| transmit}]` | Defines a policer for the currently selected class when the policy map is applied to input. Policing is supported only on ingress, not on egress.<br>• For *cir-rate-bps,* specify the average committed information rate (cir) in bits per second (bps). The range is 96000 to 800000000.<br>• For *normal-burst-byte,* specify the cir burst size in bytes. The range is 8000 to 64000.<br>• (Optional) For *maximum-burst-byte*, specify the peak information rate (pir) burst in bytes. The range is 8000 to 64000.<br>• (Optional) For *pir-rate-bps*, specify the average pir traffic rate in bps where the range is 96000 to 800000000.<br>• (Optional) Conform action options are:<br>– Set a CoS priority value and transmit<br>– Transmit packet (default)<br>• (Optional) Exceed action options are:<br>– Set a CoS value and transmit<br>– Drop packet (default)<br>• (Optional) The violate action is only valid if pir is configured. Violate action options are:<br>– Set a CoS value and transmit<br>– Drop packet (default) |
| `Router (config-pmap-c)# priority kbps` | Specifies low latency queuing for the currently selected class. This command can only be applied to an output. When the policy-map is applied to an output, an output queue with strict priority is created for this class. The only valid rate choice is in kilobits per second (8 to 2000000).<br>**Note** When using the priority action, the traffic in that class is given a 100% commit (CIR), regardless of the rate entered as the priority rate. To ensure that CIR commitments are met for the interface, a policer must be configured on the input of all interfaces that might deliver traffic to this output class, limiting the peak rate to the priority rate entered. |

| Command | Purpose |
|---|---|
| `Router (config-pmap-c)# set-cos cos-value` | Specifies a class of service (CoS) value or values to associate with the packet. The number is in the range from 0 to 7.<br><br>This command may only be used in a policy-map applied to an output. It specifies the VLAN CoS priority to set for the outbound packets in the currently selected class. If QinQ is used, the top-level VLAN tag is marked. If outbound packets have no VLAN tag, the action has no effect. This action is applied to the packet after any set cos action done by a policer, and will therefore override the CoS set by a policer action.<br><br>If a packet is marked by the policer and forwarded out an interface that also has a set cos action assigned for the traffic class, the value specified by the police action will take precedence in setting the 802.1p CoS field.<br><br>This command also sets the CoS value in the RPR header for packets exiting the ML-Series on the RPR interface. |
| `Router (config-pmap-c)# set-cos-tranmit cos-value` | Sets (marks) a VLAN CoS priority associated with the packets under/over the policed rate.<br><br>The classification of any output policy map will be based on the new CoS value set (even if the packet were to enter and/or exit without a VLAN tag). An output policy map may override the VLAN CoS priority set for the packet. When the packet is transmitted out an interface, the associated VLAN CoS priority will be written into the VLAN tag. If QinQ is used, the top-level VLAN tag at the time of transmit will be marked. If the packet is transmitted without a VLAN tag, no marking will occur.<br><br>This command also sets the CoS value in the RPR header for packets exiting the ML-Series on the RPR interface. |

# Attaching a Traffic Policy to an Interface

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). Only one traffic policy can be applied to an interface in a given direction.

Use the **no** form of the command to detach a traffic policy from an interface. The **service-policy** command syntax is as follows:

**service-policy** {**input** | **output**} *policy-map-name*
**no service-policy** {**input** | **output**} *policy-map-name*

To attach a traffic policy to an interface, use the following commands in interface configuration mode, as needed:

| Command | Purpose |
|---|---|
| `Router(config)# interface interface-id` | Enters interface configuration mode, and specifies the interface to apply the policy map.<br>Valid interfaces are limited to physical Ethernet and POS interfaces.<br>**Note** Policy maps cannot be applied to SPR interfaces, subinterfaces, port channel interfaces, or BVIs. |
| `Router(config-if)# service-policy output policy-map-name` | Specifies the name of the traffic policy to be attached to the output direction of an interface. The traffic policy evaluates all traffic leaving that interface. |
| `Router(config-if)# service-policy input policy-map-name` | Specifies the name of the traffic policy to be attached to the input direction of an interface. The traffic policy evaluates all traffic entering that interface. |

# Configuring CoS-based QoS

Cisco ONS 15454/ONS 15454 SDH Software Release 4.1 introduces the global **cos commit** <*cos value*> command. This command allows the ML-Series card to base the QoS treatment for a packet coming in on a network interface on the attached CoS value, rather than on a per-customer-queue policer.

CoS-based QoS is applied with a single global **cos commit** <*cos value*> command:

| Command | Purpose |
|---|---|
| `Router(config)# cos-commit <cos value>` | Labels packets that come in with a CoS equal to or higher than the *cos value* as CIR and packets with a lower CoS as DE. |

# Monitoring and Verifying QoS

To display the information relating to a traffic class or traffic policy, use one of the following commands in EXEC mode, as needed.

***Table 13-1 Commands for QoS Status***

| Command | Purpose |
|---|---|
| `Router# show class-map name` | Displays the traffic class information of the user-specified traffic class. |
| `Router# show policy-map` | Displays all configured traffic policies. |
| `Router# show policy-map name` | Displays the user-specified policy map. |
| `Router# show policy-map interface interface` | Displays configurations of all input and output policies attached to an interface. Statistics displayed with this command are unsupported and show zero. |

Examples of these commands are shown here:

```
Router# show class-map
 Class Map match-any class-default (id 0)
   Match any
 Class Map match-all policer (id 2)
   Match ip precedence 0

Router# show policy-map
 Policy Map police_f0
  class  policer
   police 1000000 10000 conform-action transmit exceed-action drop

Router# show policy-map interface

 FastEthernet0

  service-policy input: police_f0

    class-map: policer (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      match: ip precedence 0

    class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      match: any
        0 packets, 0 bytes
        5 minute rate 0 bps
```

# Configuration Examples

This section provides the specific command and network configuration examples:

- Traffic Classes Defined Example
- Traffic Policy Created Example
- class-map match-any and class-map match-all Commands Example
- class-map match-any and class-map match-all Commands Example
- match spr1 Interface Example
- ML-Series VoiP Example
- ML-Series Policing Example
- ML-Series CoS-based QoS Example

## Traffic Classes Defined Example

This example shows how to create a class map called class1 that matches incoming traffic entering interface fastethernet0:

```
Router(config)# class-map class1
Router(config-cmap)# match input-interface fastethernet0
```

This example shows how to create a class map called class2 that matches incoming traffic with IP-precedence values of 5, 6, and 7:

```
Router(config)# class-map match-any class2
Router(config-cmap)# match ip precedence 5 6 7
```

**Note** If a class-map contains a match rule which specifies multiple values, such as 5 6 7 in this example, then the class-map must be match-any, not the default match-all. Without the match-any an error message is printed and the class is ignored. The supported commands which allow multiple values are **match cos**, **match ip precedence** and **match ip dscp**.

This example shows how to create a class map called class3 that matches incoming traffic based on bridge group 1:

```
Router(config)# class-map class3
Router(config-cmap)# match bridge-group 1
```

# Traffic Policy Created Example

In the following example, a traffic policy called policy1 is defined to contain policy specifications, including a bandwidth allocation request, for the default class and two additional classes—class1 and class2. The match criteria for these classes were defined in the traffic classes, see the "Creating a Traffic Class" section on page 13-11.

```
Router(config)# policy-map policy1
Router(config-pmap)# class class-default
Router(config-pmap-c)# bandwidth 1000
Router(config-pmap)# exit

Router(config-pmap)# class class1
Router(config-pmap-c)# bandwidth 3000
Router(config-pmap)# exit

Router(config-pmap)# class class2
Router(config-pmap-c)# bandwidth 2000
Router(config-pmap)# exit
```

# class-map match-any and class-map match-all Commands Example

This section illustrates the difference between the **class-map match-any** command and the **class-map match-all** command. The **match-any** and **match-all** options determine how packets are evaluated when multiple match criteria exist. packets must either meet all of the match criteria (**match-all**) or one of the match criteria (**match-any**) in order to be considered a member of the traffic class.

The following example shows a traffic class configured with the **class-map match-all** command:

```
Router(config)# class-map match-all cisco1
Router(config-cmap)# match cos 1
Router(config-cmap)# match bridge-group 10
```

If a packet arrives with a traffic class called cisco1 configured on the interface, the packet is evaluated to determine if it matches the cos 1 and bridge group 10. If both of these match criteria are met, the packet matches traffic class cisco1.

The following example shows a traffic class configured with the **class-map match-any** command:

```
Router(config)# class-map match-any cisco2
Router(config-cmap)# match cos 1
Router(config-cmap)# match bridge-group 10
Router(config-cmap)# match ip dscp 5
```

In traffic class called cisco2, the match criteria are evaluated consecutively until a successful match criterion is located. The packet is first evaluated to the determine whether cos 1 can be used as a match criterion. If cos 1 can be used as a match criterion, the packet is matched to traffic class cisco2. If cos 1 is not a successful match criterion, then bridge-group 10 is evaluated as a match criterion. Each matching criterion is evaluated to see if the packet matches that criterion. Once a successful match occurs, the packet is classified as a member of traffic class cisco2. If the packet matches none of the specified criteria, the packet is classified as a member of the traffic class.

Note that the **class-map match-all** command requires that all of the match criteria must be met in order for the packet to be considered a member of the specified traffic class (a logical AND operator). In the example, cos 1 AND bridge group 10 have to be successful match criteria. However, only one match criterion must be met for the packet in the **class-map match-any** command to be classified as a member of the traffic class (a logical OR operator). In the example, cos 1 OR bridge group 10 OR ip dscp 5 have to be successful match criteria.

# match spr1 Interface Example

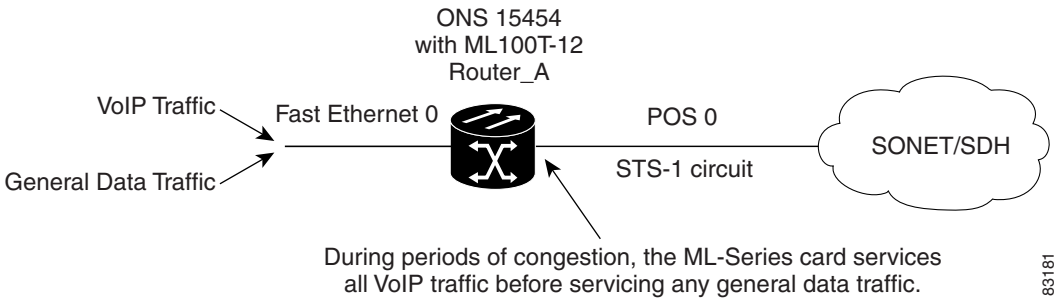
In the following example, the SPR interface is specified as a parameter to the <match input-interface> CLI when defining a class-map.

```
Router(config)#class-map spr1-cos1
Router(config-cmap)#match input-interface spr1
Router(config-cmap)#match cos 1
Router(config-cmap)#end
Router#sh class-map spr1-cos1
 Class Map match-all spr1-cos1 (id 3)
   Match input-interface SPR1
   Match cos  1
```

# ML-Series VoiP Example

Figure 13-7 shows an example of ML-Series QoS. The associated commads are provided in the sections that follow the figure.

***Figure 13-7 ML-Series VoiP Example***




```
class-map match-all voip
```

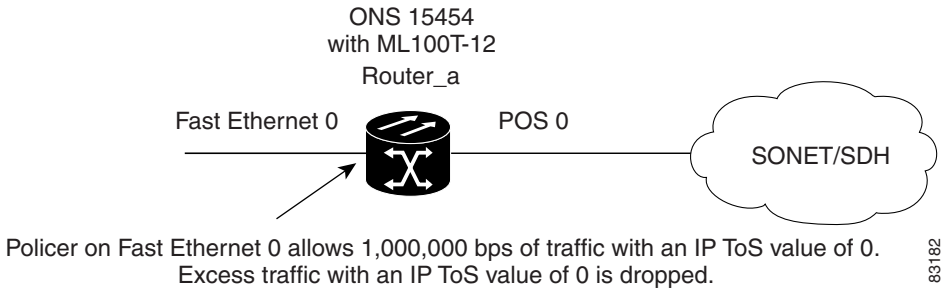
```
  match ip precedence 5
class-map match-any default
  match any
!
!
policy-map pos0
  class default
    bandwidth   1000

  class voip
    priority 1000
!
interface FastEthernet0
 ip address 1.1.1.1 255.255.255.0
!
interface POS0
 ip address 2.1.1.1 255.255.255.0
service-policy output pos0
crc 32
 no cdp enable
 pos flag c2 1
```

# ML-Series Policing Example

Figure 13-8 shows an example of ML-Series policing. The example shows how to configure a policer that will restrict traffic with an IP precedence of 0 to 1,000,000 bps. The associated code is provided in the sections that follow the figure.

***Figure 13-8 ML-Series Policing Example***

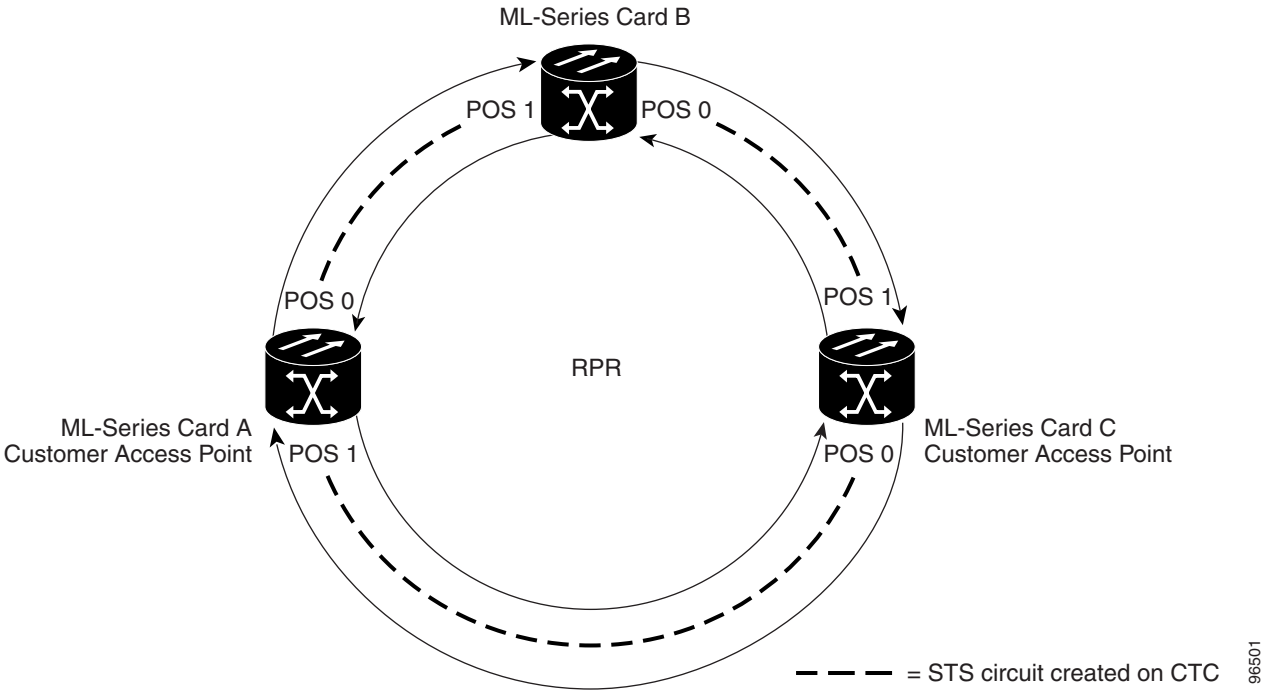



```
!
class-map match-all policer
  match ip precedence 0
!
policy-map police_f0
  class policer
    police 1000000 10000 conform-action transmit exceed-action drop
!
interface FastEthernet0
service-policy input police_f0
!
```

# ML-Series CoS-based QoS Example

Figure 13-9 shows an example of ML-Series CoS-based QoS. The associated code is provided in the sections following the figure. The CoS example assumes that the ML-Series cards are configured into an RPR and the ML-Series card POS ports are linked by point-to-point SONET circuits. For more information on configuring RPR, see Chapter 16, "Configuring Resilient Packet Ring."

***Figure 13-9 ML-Series CoS Example***




## ML-Series A Configuration (Customer Access Point)

```
hostname ML-Series A
Cos commit 2

Policy-map Fast5_in
    class class-default
    police 5000 8000 8000 pir 10000 conform-action
    set-cos-transmit 2 exceed-action set-cos-transmit
    1 violate-action drop]
```

## ML-Series B Configuration

```
hostname ML-Series B
Cos commit 2
```

## ML-Series C Configuration (Customer Access Point)

```
hostname ML-Series C
```

```
Cos commit 2

Policy-map Fast5_in
    class class-default
    police 5000 8000 8000 pir 10000 conform-action
    set-cos-transmit 2 exceed-action set-cos-transmit
    1 violate-action drop
```