

QoS Packet Policing

Traffic policing allows you determine whether network traffic is above or below a predetermined rate and to provide different treatment for such traffic. In its simplest form a policer (rate limiter) drops any traffic that exceeds a predetermined rate.

- About QoS Policing, on page 1
- Single-Rate, Two-Color Policer, on page 5
- Single-Rate, Three-Color Policer, on page 6
- Dual-Rate, Three-Color Policer, on page 8
- Configuring Rates and Burst Parameters, on page 9
- Color-Aware Policers, on page 16
- Hierarchical Policy Containing Policers, on page 19
- Verifying the Configuration and Operation of the Policing Feature, on page 22
- Configuration Examples for QoS Packet Policing, on page 25
- Command Reference, on page 27

About QoS Policing

Why Traffic Policing

Allowing you to control the maximum rate of traffic transmitted or received on an interface, traffic policing is typically configured on interfaces at the edge of a network to limit traffic into the network. In most traffic policing configurations, traffic that falls within the rate parameters is transmitted whereas traffic that exceeds the parameters is dropped or marked (and transmitted).

Note Unlike a shaper, a policer does not buffer packets. Rather, the specified action is taken immediately.

Typically, we use policers for admission control: queue or network.

Queue Admission Control limits the amount of data that can enter a queue. A *priority queue* is representative of this category wherein we avoid latency by limiting the rate at which packets may be enqueued.

Network Admission Control enforces a contract between the network administrator (service provider) and his customers. Generally both will agree on the rate at which the provider should accept traffic. This could be the

service-rate (max rate for all the traffic customer sends to provider) or a *per-class restriction* (e.g., the amount of priority traffic a customer may send).

- Using network admission control, you may decide to either drop excess traffic immediately or mark that traffic as 'out of contract.' If the latter, you can either provide that traffic a lesser treatment or drop it first if (and when) congestion occurs within this network.

Policer Definitions



Note The terms *Policer* and *Rate Limiter* usually refer to the same QoS mechanism. Policer (*Policing*) will be used throughout this document.

A policer is a device that allows you to define different treatments for packets within the same traffic class depending on whether packets are received above or below a specified rate(s).

In its simplest form, a policer indicates that traffic above a specified rate should be dropped:

```
policy-map police-all-traffic
  class class-default
    police 1m
```

Traffic through this class arriving at a rate less than 1 Mbps is considered *conforming* (adhering to the specified rate). The default action for conforming traffic is to forward packets.

Traffic arriving at a rate exceeding 1 Mbps is considered *exceeding* the configured rate. The default action for exceeding traffic is to drop packets.

The following definitions are relevant to understanding the sections that follow.

Policer Actions

In the previous example, copied below, we used a policer in its most basic form:

```
policy-map police-all-traffic
  class class-default
    police 1m
```

Conforming traffic was allowed to pass through the policer (transmitted traffic below 1m) whereas exceeding traffic was dropped. We took *immediate action* when we recognized that traffic had exceeded the specified rate. However, you may not want to always take immediate action. You might want to *defer action* rather than immediately drop traffic.

For example, you may decide that traffic above the predetermined rate should only be dropped if the network is congested. If so, you might choose to forward all traffic but mark something in the packet (e.g., DSCP) differently for conforming and exceeding traffic. The decision on whether or not to drop can then be made at the congestion point.

In the following example, we mark rather than drop traffic. We define a traffic class as any traffic arriving with a DSCP value of AF41 and demote traffic exceeding a specified rate to AF42:

```
policy-map ma
rk-out-of-contract
```

```
class AF41
  police 1m conform-action transmit exceed-action set-dscp-transmit AF42
```

The *conform-action* is to transmit traffic (simply forward, default action) arriving at a rate less than or equal to the specified 1 Mbps rate.

The *exceed-action* for traffic exceeding 1 Mbps is to mark the packet's DSCP value rather than drop traffic.

Transmit and drop represent *actions* specified for traffic *conforming* to or *exceeding* the specified rate. You specify an action with the **police** command. Supported actions are listed in the following table.

Note The rules for policer actions are very similar to those for the **set** command.

You can only mark Layer 2 and outer Layer 3 headers.

Multi-Action Policer

In the previous section we saw how a policer can be configured to mark some field in the packet. In fact, we can mark multiple fields in the packet.

You can apply multiple actions to traffic within each rate designation, analogous to how you configure multiple set actions within a traffic class. For example, if you know a packet will be transmitted through both a TCP/IP and a Frame Relay environment, you can change the DSCP value of the exceeding or violating packet, and also set the Frame Relay Discard Eligibility (DE) bit from 0 to 1 to indicate lower priority.

When specifying multiple policing actions, observe the following:

- You must enter policy-map class police configuration (config-pmap-c-police) submode.
- You can specify a maximum of four actions simultaneously, one line per action.
- You cannot specify contradictory actions such as **conform-action transmit** and **conform-action drop**.

Analogous to the **set** command, you can either configure multiple actions on the same packet (e.g., marking Layer 2 and Layer 3 fields) or define actions for different traffic types (e.g., marking the DSCP value in IPv4 packets and experimental (EXP) bits in MPLS packets).

In the following example, we cap RTP traffic (`rtp-traffic`) at 1 Mbps and drop traffic exceeding that rate (`exceed-action drop`). For conforming traffic, we mark both the COS and DSCP values in IPv4 packets and the COS and EXP bits in MPLS packets:

```
class rtp-traffic
  police cir 1000000
    conform-action set-cos-transmit 4
    conform-action set-dscp-transmit af41
    conform-action set-mpls-exp-topmost-transmit 4
    exceed-action drop
```

All packets in a traffic class count towards the rate seen by that class but actions are applied only to applicable traffic. For example, imagine that IPv4 and MPLS packets are classified into the same traffic class and a policer is configured to mark a specific DSCP value. Both IPv4 and MPLS packets count towards the observed rate, but only IPv4 packets can be marked.

Note Configuring multiple actions is supported for single and dual-rate policers.

A Note on CLI Variants

This section shows how multiple variants of the CLI can achieve the same result.

Context

The variations have emerged in different Cisco IOS software releases over time and as software trains have merged. Within the same software release, three equivalent variants exist. To avoid backwards compatibility issues, we decided to retain the variants. Please note, however, that the software implementing the policing is identical regardless of the CLI variant used.

Illustration

For the following examples, we set **police** to 10 Mbps, **conform action** to transmit (default), and **exceed action** to drop (default). At a "high" level we have three variants of the **police** command that achieve the same result: **police** *value*, **police cir** *value*, and **police rate** *value*. This set of variants is equivalent to: **police [cir|rate]***value*, where **cir** and **rate** are optional. With a rate of 10 Mbps, we can build the following command: **police [cir|rate] 10m**.

Using each variant to configure policing:

```
policy-map policer-cli-example
  class class-default
    police 10000000


policy-map policer-cli-example
  class class-default
    police cir 10m


policy-map policer-cli-example
  class class-default
    police rate 10m
```

To verify that the three variants yield the same result, you can use two stages of verification:

1. Issue **show policy-map interface** to display the configuration within IOS.
2. Issue **show platform hardware qfp active feature qos interface** to illustrate how we program hardware. This display is unchanged regardless of the CLI variant used.

Let's run Step 1:

```
show policy-map int GigabitEthernet1/0/0

  Service-policy input: policer-cli-example

    Class-map: class-default (match-any)
      162 packets, 9720 bytes
      5 minute offered rate 2000 bps, drop rate 0000 bps
      Match: any
      police:
```

```
    cir 10000000 bps, bc 312500 bytes
  conformed 212 packets, 12720 bytes; actions:
    transmit
  exceeded 0 packets, 0 bytes; actions:
    drop
  conformed 2000 bps, exceeded 0000 bps
```

Next, let's run Step 2:

```
show platform hardware qfp active feature qos int g1/0/0
```

```
Interface: GigabitEthernet1/0/0, QFP interface: 12
  Direction: Input
    Hierarchy level: 0
    Policy name: policer-cli-example
      Class name: class-default, Policy name: policer-cli-example
        Police:
            cir:  10000000 bps,  bc: 315392 bytes
            pir: 0 bps,  be: 315392 bytes
            rate mode: Single Rate Mode
            conformed: 16 packets, 960 bytes; actions:
              transmit
            exceeded: 0 packets, 0 bytes; actions:
              drop
            violated: 0 packets, 0 bytes; actions:
              drop
            color aware: No
            green_qos_group: 0, yellow_qos_group: 0
```

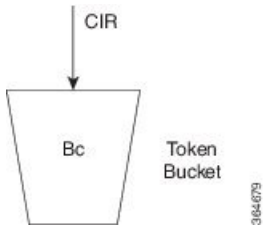
Single-Rate, Two-Color Policer

A single-rate, two-color policer (1R2C) determines whether traffic is above or below a predetermined rate (CIR in bps) and allows you to take action in either instance. The possible actions for any arriving packet are conform (packet counts as traffic falling below the CIR) and exceed (packet counts as traffic exceeding the CIR).

We need to allow for any *potential burstiness*. This behavior occurs when many packets arrive together, and the arrival rate over a short interval exceeds the CIR while the arrival rate over a longer range might conform to the CIR. To accommodate bursts yet enforce our predetermined CIR over time, we use a *token bucket* scheme.

Applying this scheme, we can represent a single-rate, two-color policer with a single-token bucket:

Figure 1: Single-Rate, Two-Color Policer



Tokens are continuously replenished at CIR and the depth of the bucket is Bc. If the bucket is full, additional tokens arriving are lost.

When a packet arrives, the policer assesses whether the bucket contains enough tokens (bytes) to *cover that incoming packet* (sufficient bytes to match the packet length). If so, the packet is regarded as conforming, the action is taken and the appropriate number of tokens (packet length) is removed from the bucket.

If the packet arrives and the bucket contains insufficient tokens to cover the packet, the exceed action is taken; the number of tokens in the bucket are unchanged. Subsequent packets may find that the bucket has replenished sufficiently to be now designated "conforming." If no packets arrive, the bucket continues to fill to the burst limit (Bc).

Specifying the bucket depth determines the allowable amount of burstiness for conforming traffic (how many bytes/packets) that may arrive closely together, assuming the bucket has had time to refill.

In this example we have specified a CIR of 10 Mbps and a burst allowance of 15000 bytes. So, a burst of 10 MTU-sized packets on an Ethernet interface could be designated conforming:

```
policy-map police-with-burst
  class class-default
    police cir 10m bc 15000
```



Note The current IOS CLI enables you to configure policing in multiple ways yet accomplish the same result.

Single-Rate, Three-Color Policer

A single-rate, three-color policer (1R3C) supports three possible output states: conform, exceed and violate. The definition of conform is analogous to that in a 1R2C policer – traffic that adheres to a predetermined rate allowing for some burst tolerance.

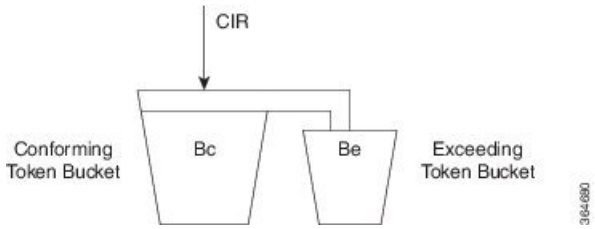
The difference stems from how we designate traffic that does not conform – traffic that a two-color policer would designate as exceed. We introduce further granularity where this traffic could be exceed or violate. Essentially, traffic that bursts ‘minimally’ above the CIR is designated as exceed but more sustained bursts above the CIR would be designated as violate.

To achieve this behavior we introduce a second token bucket. Just as the conforming token bucket is used to differentiate between traffic that conforms or exceeds, the excess token bucket enables us to differentiate between traffic that exceeds or violates.

Here are the bucket scenarios:

- The conforming token bucket is initially full (the number of bytes specified as Bc (conforming burst size).
- The exceeding token bucket is initially full (the number of bytes specified in the Be (excess burst size).
- If the conforming token bucket is full when tokens arrive (at the CIR, analogous to a 1R2C policer), they overflow into the excess token bucket.
- If both buckets are full, further tokens are lost.

Figure 2: Single-Rate, Three-Color Policer



The conforming bucket here behaves as it does in the 1R2C scenario. If the bucket contains sufficient tokens to cover the incoming packet, the packet is considered "conforming," the conforming action occurs and we remove an appropriate number of tokens from the bucket. The exceeding bucket is unaffected and we continue to replenish the conforming bucket (Bc) at CIR.

However, if the conforming bucket is full and additional tokens arrive they are not immediately lost. Instead, they overflow into the exceeding bucket. If this bucket is full, excess tokens are lost.

Similarly, when a packet arrives and the conforming bucket has insufficient tokens to cover that packet we cannot immediately declare it as exceeding; it might be exceeding or violating. If the exceeding bucket has enough tokens to cover the packet, the exceeding action is taken, and we remove the necessary number of tokens from the exceeding bucket. No bytes are removed from the conforming bucket.

If neither bucket, conforming or exceeding, has enough tokens to cover the packet, it is categorized as violating and the appropriate action is taken. Neither the conforming nor exceeding bucket is decremented:

If neither bucket, conforming or exceeding, has enough tokens to cover the packet, it is categorized as violating and the appropriate action is taken. Neither the conforming nor exceeding bucket is decremented:

```
policy-map ingress-enforcement
  class af41-metering
    police cir percent 10 bc 5 ms be 10 ms
    conform-action set-dscp-transmit af41
    exceed-action set-dscp-transmit af42
    violate-action drop
```

In this example we are policing traffic (for class af41) to 10% of the interface's bandwidth and the following apply:

- Traffic (**`conform-action set-dscp-transmit af41`**) burst up to 5 ms is forwarded and still marked as af41.
- Traffic (**`exceed-action set-dscp-transmit af42`**) burst exceeding 5 ms and up to an additional 10ms of burst is marked as af42. Elsewhere in the network, when we detect af42, we know it was received beyond the agreed contract [at the edge of the network]; under congestion, we could drop it first.
- Traffic (**`violate-action drop`**) burst beyond 15 ms above our CIR is considered violating and dropped immediately.

Note We only replenish the exceeding bucket when the conforming bucket is full. So, if you send a non-bursty stream at a rate exceeding the CIR, shortly, both the conforming and exceeding buckets will be drained; we do not replenish the exceeding bucket. All subsequent packets are considered either conforming or violating.

Dual-Rate, Three-Color Policer

Traffic rates are easier to understand than traffic burstiness. When specifying a contract for network admission control , you might have trouble describing expectations in terms of multiple burst sizes above a single rate. The dual-rate, three-color (2R3C) policer simplifies matters by primarily employing rates to differentiate conform, exceed and violate. It also introduces a second rate, PIR (Peak Information Rate)

CIR and PIR have the following characteristics:

- Traffic below the CIR is conforming.
- Traffic greater than CIR but less than PIR is exceeding.
- Traffic above PIR is violating.

You specify these rates with the **cir** and **pir** keywords of the **police** command.

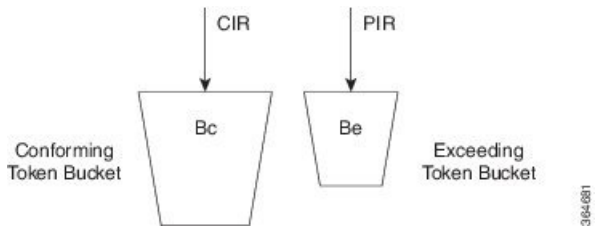
With a 2R3C policer, unlike a 1R3C, we replenish token buckets independently whenever a packet arrives at the policer. We refill conforming buckets at rate CIR; it can contain up to value Bc. ; exceeding buckets, at PIR; it can contain up to value Be.

Note PIR must exceed CIR and overflow between buckets is disallowed.

If a steady stream of packets arrives at a rate exceeding the CIR but less than the PIR, all packets are marked either conforming or exceeding. With the 1R3C policer, this scenario would have resulted in marking a minimal number of packets as exceeding and a majority as conforming or violating.

A 2R3C policer supports three possible actions for each packet: conform, exceed, and violate. Traffic entering the interface configured with a dual-rate policer is placed into one of these action categories, which dictates how we treat a packet. For instance, in the most common configuration, you can configure to send packets that either conform or exceed (with a decreased priority), and to drop packets that violate.

Figure 3: Dual-Rate, Three-Color Policer



When a packet arrives, we assess whether ample tokens exist in the conforming and exceeding buckets to cover that packet. If so, we take the conforming action (typically, transmit or transmit and mark) and remove the necessary tokens to transmit the packet from both buckets.

If the Exceeding Token Bucket (but not the Conforming Token Bucket) contains sufficient tokens to cover the packet, we take the exceeding action (typically, transmit or transmit and marking). The appropriate number of tokens are removed from the exceeding bucket only.

If neither bucket has sufficient tokens to cover the packet, the violating action is taken (typically, transmit, transmit and marking, or drop):

```
policy-map ingress-enforcement
  class af41-metering
    police cir 100k bc 3000 pir 150k be 3000 conform-action set-dscp-transmit af41
exceed-action set-dscp-transmit af42 violate-action drop
```

Observe how code from the preceding example and the corresponding code from **`Single-Rate, Three-Color Policer`** differ:

```
cir 100k bc 3000 pir 150k be 3000
cir percent 10 bc 5 ms be 10 ms
```

In the immediate example, we handle traffic accordingly:

- Up to 100Kbps (allowing for bursts up to 3,000 bytes) as conforming and forward it with DSCP marked as af41.
- Above 100Kbps but less than 150Kbps (again allowing a 3,000 byte burst) as exceeding and forward it marked as af42.
- Above 150Kbs as violating; we drop it.

Configuring Rates and Burst Parameters

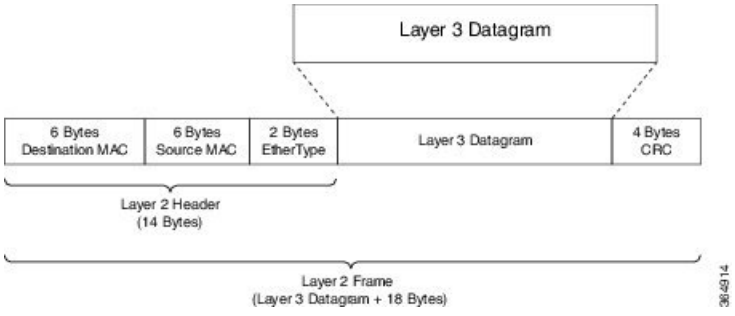
What's Included in the Policer-Rate Calculation (Overhead Accounting)

When specifying a rate or burst value, you should know how the policer assesses a packet's length (subsequently referred to as the *policing length*) when you evaluate conformance to those values. Briefly, a policer includes the Layer 3 datagram Layer 2 header lengths but neither CRC nor inter-packet overhead.

To further illustrate, consider an IP datagram transported over a GigabitEthernet link.

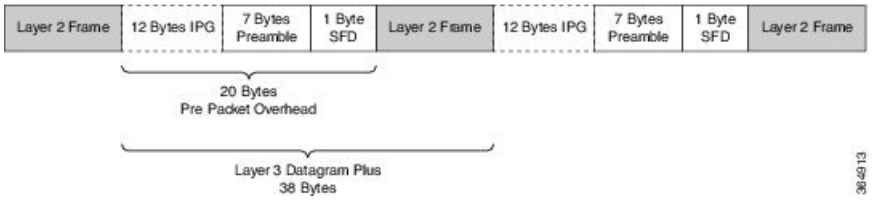
Layer 3 Datagram

First, we encapsulate it in an Ethernet frame, which adds 14 bytes of Layer 2 header and an additional 4 bytes of CRC to each datagram (18 bytes):



Ethernet Overhead

To transmit this frame over the physical medium, Ethernet requires a minimum inter-packet gap equivalent to a transmit time for 12 bytes of data. After the gap, we require seven bytes of preamble followed by a single byte start-of-frame delimiter (SFD) (Ethernet inter-packet overhead = 12 bytes IPG + 7 bytes Preamble + 1 byte SFD = 20 bytes).



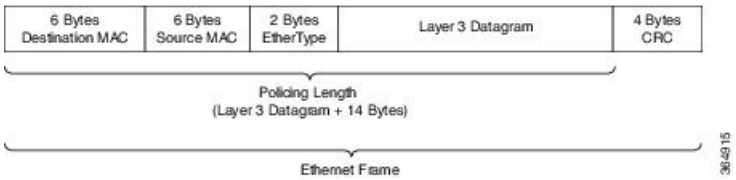
So, if you send multiple Ethernet frames sequentially, the per-packet overhead for each Layer 3 datagram is an additional 38 bytes (encapsulation [18 bytes] + Ethernet inter-packet overhead [20 bytes]). For example, if you sent 100 byte IP datagrams at line rate on a GigabitEthernet link, and used the following formula, the expected throughput in packets per second would be:

```
Linerate / Bits Per
  Byte / ( Layer 3 length + Per Packet Overhead) = Packets Per Second

1 Gbps / 8 / ( 100 + 38 ) = 905, 797 pps
```

From the policer's perspective, the packet's length is the Layer 3 datagram + Layer 2 header length (14 bytes on a GigabitEthernet interface):

Policing Length



Now consider a 500 Mbps policer configured on a GigabitEthernet interface. As in the previous example, we will send all 100 byte IP datagrams to the policer, resulting in a policing length of 100 byte datagram length + the 14 byte (Ethernet Layer 2 header). According to the following formula, the anticipated throughput would now be:

```
Policer Rate / Bits
  per Byte / (Layer 3 length + Layer 2 header length) = Packets Per Second

500 Mbps / 8 / (100 + 14) = 548,246 pps
```

Note Packets marked as *conforming* by a 500 Mbps policer will consume considerably more than 500 Mbps of physical bandwidth!

Policer on Logical Interface

On egress, a policer is unaware of the final physical interface type (tunnels can move between interfaces) and therefore the policer is unaware of the final Layer 2 overhead. So, the latter is excluded from the policing length. Similarly, because the policer cannot predict the extent of packet expansion due to overhead, if we configure encryption, we will not include encryption overhead in policer rate calculations. The egress policer will include the Layer 3 datagram and any tunnel headers (e.g., additional IP header, GRE header).

On ingress, because a policer is aware of the receiving interface type, policing on a tunnel interface includes Layer 2 overhead plus any tunnel headers.

The following table illustrates the dependencies of policer rate calculations on a ASR 1000 Series Aggregation Services Router. Be aware that we present only a subset of all permutations:

where the values are defined as follows:

- 0 - svti ('tunnel mode ipsec ipv4') has no overhead
- 14 - Layer 2 Ethernet header size
- 20 - the IP/IP header size
- 24 - the IP/GRE header size (20 + 4)

Policer on ATM Interfaces

If a policer is configured on an ATM interface, the policing length includes the Layer 3 datagram and the ATM adaption layer (AAL) header. For AAL5SNAP encapsulation length, this means that we include eight bytes of header in the policing length; for AAL5NLPID encapsulation, two bytes.

This calculation differs sharply from that applied to scheduling, where we include the complete *AAL PDU* and *cell tax*.

Changing What's Included - Overhead Accounting Adjustment

In prior sections, we described what is included by default in policer rate calculations. But what happens when you want to deviate from the default? For example, what if you want to express CIR as the physical bandwidth that would be consumed on a link? For an Ethernet interface you would include the 4 byte CRC and the 20 bytes inter-packet overhead required per packet.

Alternatively, you (a service provider) might want to police customers' traffic at Layer 3 rates. Because datagram length is unchanged as a packet traverses different interface types (or encapsulating protocols), we would not include Layer 2 header length in policer rate calculations.

Note Any interface that supports QoS policies will support overhead accounting adjustment.

Note Changing overhead accounting may impact the network. For example, if you use a policer for network admission control, you might need to configure a corresponding shaper on the equipment that connects to that network. The two views of what is included in CIR (shaper and policer) should match.

In the following example we want to include all inter-packet overhead such that a policer will allow up to 50% of the traffic on the physical link to be conforming. By adding 24 bytes per packet (**`user-defined 24`**) we address the 4 byte CRC and the 20-byte inter-packet overhead.

```
policy-map ethernet-physical-example
  class class-default
    police cir percent 50 account user-defined 24
```

Using the **atm** keyword of the **police account** command, you can direct the policer to compensate for ATM cell division and cell padding (ATM cell tax) in rate calculations.

To include cell tax and cover the AAl5 trailer, a router first adds 8 bytes to the policing length. Then, it calculates the number of ATM cells (48 bytes of data carried per 53 byte cell) required to carry the packet and multiplies this number by 53. For example, a 46 byte datagram would require 2 cells and therefore, if cell tax is included, the policing length would be considered "106 bytes."

In the following example, we show a 5 Mbps policer, which must include the cell-tax in its rate calculations:

```
policy-map include-cell-tax-example
  class class-default
    police cir 5000000 account user-defined 0 atm
```

The **atm** in the configuration dictates that we include the cell tax.

Restrictions for Overhead Accounting Adjustment

- If you enable overhead accounting on a child policy, then you must enable overhead accounting on the parent policy.
- In a policy-map, you must either enable or disable overhead accounting for all classes in a policy. Within the same policy, you cannot enable overhead accounting for some classes and disable overhead accounting for other classes.
- Overhead accounting is not reflected in any QoS counters (e.g., classification, policing, or queuing).
- You can enable overhead accounting on top-level parent policies as well as on both middle-level and bottom-level child policies. Child policies inherit overhead accounting policies configured at the "parent" or "grandparent" level.
- The overhead accounting type or value used within a policy-map and between the parent and the child policy-maps (in a hierarchical policy-map structure) must be consistent.

Overhead Accounting Adjustment (Predefined Options)

Through some predefined CLI options (based on broadband use cases), you can specify the encapsulation while the router adds or subtracts the appropriate number of bytes (see the following table).

Imagine that we send (or receive) traffic on an Ethernet interface to a DSLAM (digital subscriber line access multiplexer) elsewhere in the network. Although we are encapsulating in Ethernet frames (e.g., Dot1Q or Q-in-Q), the DSLAM encapsulates in some form of ATM encapsulation. We want the policer to execute on traffic as it would appear after the DSLAM. In all instances, we would add cell-tax to the policing length.

In the following example, we apply predefined overhead accounting values. If we receive Dot1Q-encapsulated packets on an Ethernet interface, an upstream DSLAM receives *AAL5-Mux 1483 routed encapsulated* packets, then strips the ATM and adds the Ethernet headers. On the ATM interface, the datagram would have 3 bytes of additional AAL headers but would not have the 18 bytes of Ethernet headers (including Dot1Q). So, the PDU would be 15 bytes less on the ATM interface (we subtract 15 bytes from the policing length and then add the cell-tax):

```
policy-map atm-example
  class class-default
    police 5000000 account dot1q aal5 mux-1483routed
```

Default Burst Sizes

If you don't explicitly configure a burst tolerance value (Bc or Be), IOS will configure a default. This default burst tolerance is 250 ms of data based on the appropriate rate. For example, if the CIR is 100 Mbps then 250 mS of this rate would be 100000000/8 x 0.250 = 3125000 bytes.

Bc and Be for a single rate policer are always based on the CIR. The Be for a dual-rate policer is based on the PIR:



Note When configuring a policer for queue admission control , set Bc to something suitable for applications in that queue (e.g., for a voice application, set Bc to 10 milliseconds or less).

```
policy-map policer-default
  class af41
    police cir 20000000 pir 40000000 conform-action transmit  exceed-action
                \ set-dscp-transmit af42 violate-action set-dscp-transmit af43

 show policy-map interface
 GigabitEthernet1/0/0

  Service-policy input: policer-default

    Class-map: af41 (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match:  dscp af41 (34)
      police:
          cir 20000000 bps, bc 625000 bytes           1
          pir 40000000 bps, be 1250000 bytes          2
        conformed 0 packets, 0 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          set-dscp-transmit af42
        violated 0 packets, 0 bytes; actions:
          set-dscp-transmit af43
        conformed 0000 bps, exceeded 0000 bps, violated 0000 bps
```



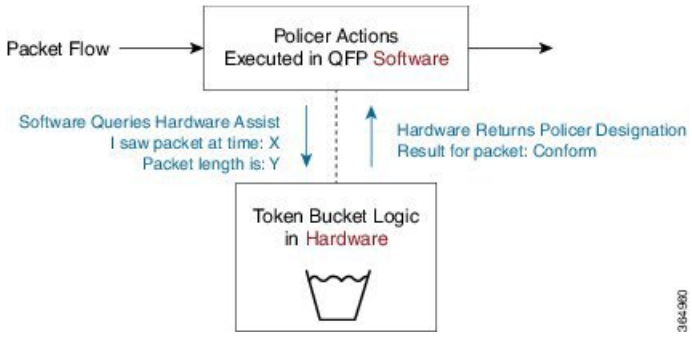
Note The dual-rate policer as well as Bc and Be default to 250ms based on the CIR (**1**) and PIR (**2**), respectively.

Rate and Burst Sizes Programmed in Hardware

On the Cisco ASR 1000 router platform, policer rate calculations are performed in *dedicated hardware*.

While *hardware assist* enables you to scale the number of policers independent of performance impact, it imposes some restrictions on the programmable rate and burst value combinations.

Figure 4:



Consider a simple policy with a single-rate, two-color policer:

```
policy-map hardware-example
  class class-default
    police cir 1m bc 3000
```

Output from the **show policy-map interface** command confirms that IOS has accepted the configured CIR and Bc values:

```
show policy-map interface g1/0/0


GigabitEthernet1/0/0

  Service-policy input: hardware-example

    Class-map: class-default (match-any)
      337 packets, 167152 bytes
      5 minute offered rate 2000 bps, drop rate 0000 bps
      Match: any
      police:
          cir 1000000 bps, bc 3000 bytes                    *
        conformed 337 packets, 167152 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          drop
        conformed 2000 bps, exceeded 0000 bps
```

***** CIR and Bc configured as expected

If you look at the dataplane, however, you can see the values actually programmed in hardware. Following is the output of the **show platform qfp active feature qos interface** command, which displays the actual policer values in hardware:

```
show platform hardware qfp active feature qos interface gig1/0/0


Interface: GigabitEthernet1/0/0, QFP interface: 9

  Direction: Input
    Hierarchy level: 0
    Policy name: hardware-example
      Class name: class-default, Policy name: hardware-example
        Police:
            cir:  1000000 bps,  bc: 3264 bytes              *
            pir: 0 bps,  be: 3008 bytes
```

```
rate mode: Single Rate Mode
conformed: 19 packets, 9424 bytes; actions:
  transmit
exceeded: 0 packets, 0 bytes; actions:
  drop
violated: 0 packets, 0 bytes; actions:
  drop
color aware: No
green_qos_group: 0, yellow_qos_group: 0
```

***** Bc as modified for hardware assist

Note Although we might slightly modify rate and burst parameters to accommodate the hardware assist, the platform always aims to retain the rates and resulting accuracy within 1% of what you request.

Percent-based Policer

The Percentage-based Policing feature enables you to configure traffic policing based on a percentage of the bandwidth available on the interface. Hence, you can use the same policy-map for multiple interface types with differing amounts of bandwidth. Recalculating the bandwidth for each interface or configuring a different policy-map for each type of interface is unnecessary.

Note If the interface is a shaped-ATM permanent-virtual circuit (PVC), we calculate the total bandwidth as follows:

- For a variable bit rate (VBR) virtual circuit (VC), the sustained cell rate (SCR) is used.
- For an available bit rate (ABR) VC, the minimum cell rate (MCR) is used.

You can use percentage-based policers for both CIR and PIR, calculating either from a specified percentage of either the interface bandwidth or parent shaper (if one exists).

With percent-based policing, if you choose to specify burst parameters (Bc and Be), they must be in ms rather than bytes. Given the speed of the target interface, IOS converts the value to bytes in two steps:

1. Using the speed of a target interface, IOS converts percentage to bps CIR
2. With bps CIR and *burst in time*, burst is converted to bytes.

Let's configure the Bc to 10 ms (relative to the police rate) and the CIR to 10% of the available interface bandwidth:

```
policy-map police-percent
  class class-default
    police cir percent 10 bc 10 ms
```

If we apply police-percent to a GigabitEthernet interface (1Gbps nominal bandwidth), IOS converts the CIR to 100 Mbps and the Bc to 125,000 bytes (100 Mbps x 10msec / 8):

```
show policy-map interface GigabitEthernet1/0/0

  Service-policy input: police-percent
```

```
    Class-map: class-default (match-any)
      834 packets, 413664 bytes
      5 minute offered rate 13000 bps, drop rate 0000 bps
      Match: any
      police:
        cir 10 %, bc 10
        cir 100000000 bps, bc 125000 bytes  Configured CIR and Bc converted to bps and
bytes, respectively.
```

Now, if we attach police-percent to a POS OC3 interface, the rate will be based on a nominal bandwidth of 155 Mbps. CIR will be calculated as 15.5 Mbps; the Bc, 19375 bytes (15.5 Mbps x 10msec / 8):

```
show policy-map interface POS1/1/0

  Service-policy input: police-percent

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: any
      police:
          cir 10 %, bc 10
          cir 15500000 bps, bc 19375 bytes  Configured CIR and Bc converted to bps and
bytes, respectively.
```

Color-Aware Policers

A *color-aware policer* accounts for any preexisting markings that were determined by a previous node's policer as *in-contract* or *out-of-contract* (the previous node is typically at the edge of the network). Where color-aware policer is configured, we use such markings to determine the appropriate policing action for the packet. Traffic that was designated out-of-contract will always remain out-of-contract. Traffic that was designated in-contract may be demoted to out-of-contract by the new policer.

The ASR 1000 provides a limited implementation of color-aware policing; we restrict the contents of the class-maps used to determine the existing color of traffic:

- Only QoS group matching is supported in color-aware class-maps (only classification based on qos-group is supported.)
- Only one filter (one **match qos-group** *value* statement) is supported per color-aware class.

  You can use a child policy to set the qos-group based on a field you want in the received packet.
- Color-aware "specific" statistics are not supported.
- You cannot use the **no class-map** command to remove a color-aware map provided it is referenced in a color-aware policer. You must first remove all color-aware policers (using either the **no conform-color** or the **no exceed-color** command).

The "color" in color aware policing refers to how we educate the policer on how to interpret pre-existing markings in a received packet. Typically, we use Green to represent traffic that was pre-marked as *conforming* or in-contract. Similarly, Yellow represents traffic that was pre-marked as *exceeding* or out-of-contract.

Note that Green or Yellow are representative only; the CLI uses *conform-color* and *exceed-color* instead. Through the **police** command, these keywords specify class-maps that are used to determine the pre-existing color of that packet.

The following example shows how a child policy-map enables you to specify pre-existing color based on any field in the received packet. The color-aware policer is configured in a class that matches all packets from one of the DSCP assured forwarding traffic classes, AF4.

For this example, a packet marked AF41 is in-contract (conform or green), AF42 is out-of-contract (exceed or yellow) and AF43 is violating. The child policy mark-existing-color classifies packets based on the received DSCP, internally marking AF41 packets as qos-group 1 and AF42 packets as qos-group 2.

The color-aware policer will use the pre-conform (classify green packets) and pre-exceed (classify yellow packets) class-maps to determine the existing color of an arriving packet. Although these class-maps only support the qos-group filter, use of the child policy allows us to determine the pre-existing color based on the DSCP value in the received packet:

```
class-map af4
  match dscp af41 af42 af43
!
class-map af41
  match dscp af41
class-map af42
  match dscp af42
!
class-map pre-conform               !These are policer
  match qos-group 1                 !class-maps that
class-map pre-exceed                !only support qos-group
  match qos-group 2
!
policy-map mark-existing-color      !We use a child policy
  class af41                        !to set qos-group
    set qos-group 1                 !based on DSCP in the
  class af42                        !received packet
    set qos-group 2
!
policy-map dual-rate-color-aware
  class af4
    police cir 1m bc 5000 pir 2m be 5000
      conform-action set-dscp-transmit af41
      exceed-action set-dscp-transmit af42
      violate-action drop
      conform-color pre-conform exceed-color pre-exceed
    service-policy mark-existing-color
```

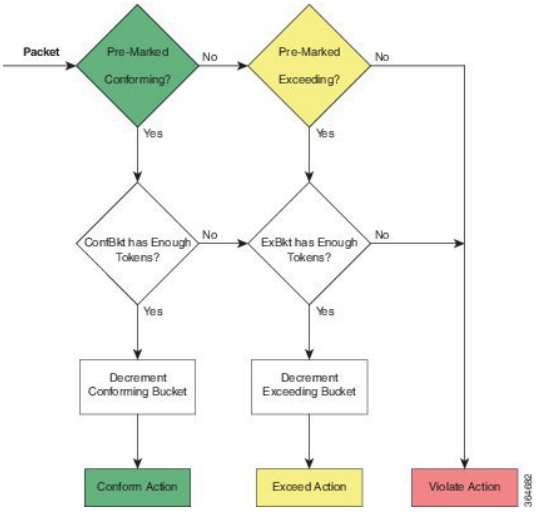
Single-Rate, Color-Aware, Three-Color Policer

The *color-aware mode* of a single-rate, three-color policer extends the standard single-rate, three-color policer.

Similar to the "color-blind" version of this type of policer, we maintain and replenish two distinct token buckets for the "color-aware" mode. The difference stems from how a packet is evaluated against these buckets. Recall that a color-aware policer honors any decision made by a previous router (the current designation of a packet) and ensures that the decision of a previous router is not undone (an exceeding or violating packet can never be promoted to conforming).

The following flowchart illustrates the algorithm used for handling traffic in single-rate, color-aware traffic policing. ConfBkt represents the conforming token bucket and ExBkt the exceeding token bucket.

Figure 5: Single-Rate, Color-Aware, Three-Color Policer



When a packet arrives, the policer uses its color-aware class-maps to determine the pre-existing color of that packet. This color may be conforming (matches the conform-color class-map), exceeding (matches the exceed-color class-map) or violating (matches neither of these class-maps).

If a packet is pre-marked as conforming it might end up as conforming, exceeding or violating. Evaluation proceeds as though the policer was operating in a color-blind mode.

- If the conforming token bucket has enough tokens the packet will take the conform action and the bucket will be decremented by the size of the packet.
- If the conforming token bucket has insufficient tokens but the exceeding token bucket does, the packet will take the exceed action and the exceeding token bucket is decremented by the size of the packet.
- If neither the conforming nor exceeding token bucket has sufficient tokens the packet will take the violate Action.

If a packet is pre-marked as exceeding it can never be promoted to conforming so evaluating the conforming token bucket is unnecessary.

- If the exceeding token bucket has sufficient tokens the packet will take the exceeding action and the bucket is decremented by the size of the packet.

If a packet is pre-marked as violating

- The violating action is taken and either token bucket is unchanged.

Dual-Rate, Color-Aware, Three-Color Policer

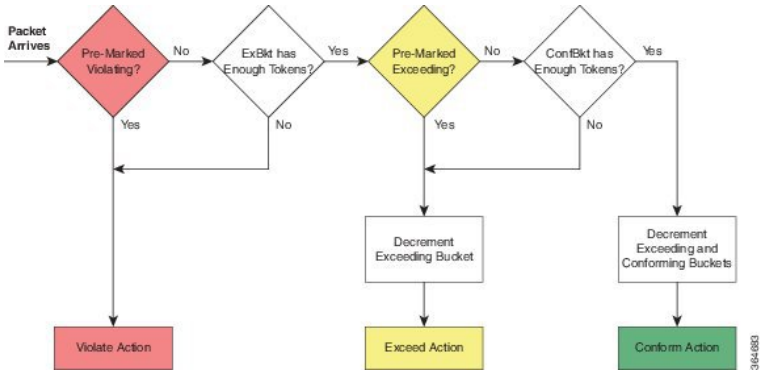
The color-aware mode of a dual-rate, three color policer extends the standard dual-rate, three-color policer.)

Similar to the "color-blind" version of this type of policer, we maintain and replenish two distinct token buckets for the "color-aware" mode. The difference arises from how a packet is evaluated against these buckets. Recall that a color-aware policer honors any decision made by a previous router (the current designation of a packet)

and ensures that the decision of a previous router is not undone (an exceeding or violating packet can never be promoted to conforming).

The following diagram illustrates the algorithm used for handling traffic in dual-rate, color-aware policing. ConfBkt represents the conforming token bucket and ExBkt the exceeding token bucket.

Figure 6: Dual-Rate, Color-Aware, Three-Color Policer



When a packet arrives, the policer uses its color-aware class-maps to determine the pre-existing color of that packet. This color may be conforming (matches the conform-color class-map), exceeding (matches the exceed-color class-map) or violating (matches neither of these class-maps).

If a packet is pre-marked as violating

- we take the violating action and neither bucket is changed (decremented).

If a packet is pre-marked as exceeding

- and the exceeding bucket has sufficient tokens, the packet will remain as exceeding and we decrement the exceeding bucket by the size of the packet.
- and the exceeding bucket has insufficient tokens, the packet will take the violate action and neither bucket is changed.

If a packet is pre-marked as conforming

- and the exceeding bucket has insufficient tokens the packet will take the violate action and neither bucket is changed.
- and the exceeding bucket has sufficient tokens but the conforming bucket does not the packet will take the exceed action and we decrement the exceeding bucket by the size of the packet.
- and both the exceeding and conforming buckets have sufficient tokens the conform action will be taken and we decrement both buckets by the size of the packet.

Hierarchical Policy Containing Policers

In hierarchical traffic policing, we introduced hierarchical policies as a way to offer more granular control over traffic classes and to have some QoS actions operate on the aggregate of a number of those classes.

The ASR 1000 Series supports at most three levels in a hierarchical policy and the policing feature (one particular QoS action) can be configured at any level of that policy.

When describing hierarchical policies we often use different language to describe the distinct levels within that hierarchy (e.g., Top/Middle/Bottom, Parent/Child/Grandchild, Root/Leaf, Child/Parent/Grandparent). Because this can lead to ambiguity, we will always refer to the levels as Parent/Child/Grandchild where meanings are defined as follows:

Parent policy is a policy-map that will be attached to an interface using the **service-policy** command.
Child policy is a policy embedded directly in a class of the parent policy (using the **service-policy** command within a class).
Grandchild policy is a policy embedded directly in a class of the child policy.

Occasionally, we will refer to a policy's child or parent. They represent more relative terms (e.g., the parent of the *grandchild policy* references the child policy when we communicate in absolute terms).

Ingress Hierarchical Policy Containing only Policers

One of the simplest and perhaps most typical use of policers in hierarchical policies is an ingress policy containing only policers. We have already described how a policer is often used for network admission control (defined inWhy Traffic Policing, on page 1). Replacing a simple policer with hierarchical policers allows the network operator to not only set an aggregate rate for network admission but also to specify rates for the individual classes of traffic that will be carried over the network.

For example, consider the following policy:

```
policy-map child
  class voice
    police cir percent 10 bc 5 ms
 !
policy-map parent
  class class-default
    police cir 50000000
    service-policy child
```

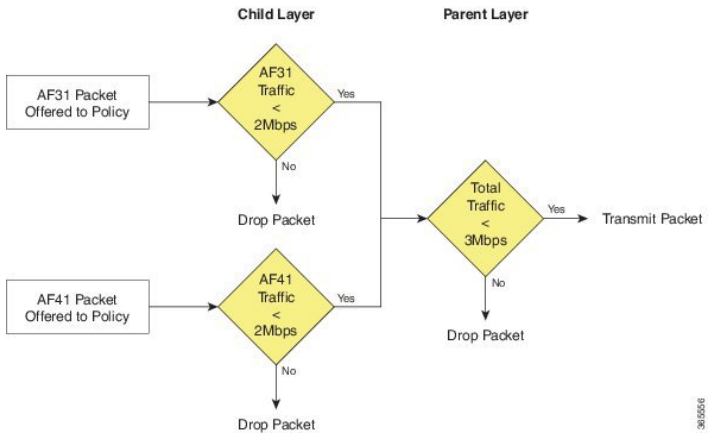
The policy-map parent, which is attached to the interface, defines the aggregate network admission rate (or service rate) for a customer so connected. In this example, the customer has contracted for 50 Mbps of network service. Within that network rate, the child policy limits individual classes of traffic. For example, the voice class specifies that traffic arriving at a rate exceeding 5 Mbps (10% of the parent) would simply be dropped.

Hierarchical Policers Order of Operation

On the ASR 1000 Series Appregation Services Router, we evaluate hierarchical policers for the child first then the parent. Although this scheme differs from IOS classic, it provides a much more meaningful construct. The following example should clarify this notion:

```
policy-map child
  class AF41
    police 2m
  class AF31
    police 2m
!
policy-map parent
  class AF41_or_AF31
    police 3m
    service-policy child
```

Figure 7: Hierarchical Policing



If a packet arrives with a marking of AF31 it must first pass through the AF31 policer in the child policy, which allows such packets to pass through up to a rate of 2 Mbps. The packet must then pass through the parent policer, which observes both AF31 and AF41 traffic.

The combined rate of AF31 and AF41 traffic from the child policy could be up to 4 Mbps as each has a 2 Mbps policer configured. Although a packet passed through the child policer, it may be dropped by the parent policer if the rate arriving at that policer is above the configured 3 Mbps rate.

When policers are used in an egress policy-map with scheduling semantics (bandwidth/shape/priority) all policers will be evaluated before a packet is enqueued. Furthermore, a policer in the parent level would be enforced before a shape value in the child level (scheduling happens).

Percent-Based Policer in Hierarchical Polices

If a percent-based policer is used in the parent level of a policy-map the meaning of the percent is fairly intuitive - it is a percent of the bandwidth available in the interface where the policy-map is attached. When we use a percent-based policer in the child or grandchild level, the meaning can be a bit more ambiguous.

If the percent-based policer is configured in the child level it examines the parent level class to assess whether the bandwidth of that class has been constrained by either a shaper or policer. If so, the child policer CIR is a percent of the shape or police rate configured in the parent level. If not, the percent is interpreted as percent of the bandwidth available in the interface where the policy-map is attached.

If the percent-based policer is configured in the grandchild level it first looks at the child level class for a shaper or policer. If it finds one, it uses that rate in the child level. If none exists, the grandchild policer looks at its class in the parent level. It either finds a rate there or uses the rate of the interface to which the policy is attached.

If the percent policer is configured in the grandchild level and a rate-limiting feature (e.g., shaper or policer) is configured in both the child and parent levels, the grandchild always uses the rate configured in the child level. This is crucial as the sum of shapers or policers at any level in a policy can be greater than the physical bandwidth available.

If both a shaper and a policer are configured in the parent of a class with a percent-based policer, the percent-based policer is based on the lower rate configured (shaper or policer).

The following hierarchical policy-map illustrates these considerations:

```
policy-map grandchild
  class AF11
    police cir percent 60
```

```
  class AF12
    police cir percent 40
!
policy-map child
  class AF1
    bandwidth percent 50
    service-policy grandchild
!
policy-map parent
  class class-default
    shape average 50000000
    service-policy child
```

The policers in the grandchild policy are percent-based policers. They defer to their parent class (class AF1 in the parent policy) for rate-limiting features. Because none exist here, the policers "step up" to the parent class (**class class-default**, in the parent policy).

There, they find a shaper that limits the throughput to 50Mbps. So, the policer in class AF11 would be configured with a CIR of 30 Mbps (60% of 50Mbps); the policer in class AF12, a CIR of 20 Mbps.

Verifying the Configuration and Operation of the Policing Feature

As with all MQC QoS features, you have three ways to verify the configuration and performance of the policing feature:

- **show policy-map** *policy-name*

  Displays the user-entered configuration. Analogous to contents of the running configuration on the router but displays default values and actions not explicitly called out in the configuration.

- **show policy-map interface** *interface-name*

  Displays statistics for all features within that policy-map. Primary means of verifying that a QoS policy is operating as expected..

- **show platform hardware qfp active feature qos interface** *interface-name*

  Displays real-time information from the dataplane. Shows the exact rates and burst sizes that are programmed in hardware.

Example 1: show policy-map *policy-name* Command

If we configure the policy-map simple_policer as follows:

```
policy-map simple_policer
  class AF1
    police cir 20000000
```

show policy-map command output looks like this:

```
show policy-map simple_policer

Policy Map simple_policer
  Class AF1
```

```
    police cir 20000000 bc 625000
      conform-action transmit
      exceed-action drop
```

Besides the explicit conforming burst and conform (or exceed) actions, notice the lack of statistics or interface information. We merely define a policy, an action applicable to multiple interfaces.

Example 2: show policy-map interface *interface-name* Command

Here is a sample output of the **show policy-map interface** command for an instance of a policy-map attached to a particular interface:

```
show policy-map interface GigabitEthernet1/0/0

GigabitEthernet1/0/0

 Service-policy input: simpler_policer

  Class-map: AF1 (match-any)                              --+
   1000 packets, 1496000 bytes                              |Classification
   5 minute offered rate 0000bps, drop rate 0000bps         |Section
   Match: :dscp af11 (10) af12 (12) af13 (14)               |
   police:                                                 --+
     cir 20000000 bps, bc 625000 bytes                      |
    conformed 447 packets, 668712 bytes; actions:           |Policing
     transmit                                               |Section
    exceeded 553 packets, 827288 bytes; actions:            |
     drop                                                   |
    conformed 0000 bps, exceeded 0000 bps                  --+

  Class-map: class-default (match-any)
    0 packets, 0 bytes
    5 minute offered rate 0000 bps, drop rate 0000 bps
    Match: any
```

The organization of the output reflects the policy-map definition combined with a hierarchical output that represents the policy-map and class hierarchy. Within each class, a classification section displays the classification counters (statistics of packets that were determined to belong to this class) and the classification criteria (a summary of the class-map that defines what packets belong to this class). Following the classification section, you observe a block that represents each QoS action configured within that class. Because policing is the sole action in this example, only a block of policing statistics displays.

The following table summarizes the meaning of different fields in the **show** command output.

SUMMARY STEPS

1. As you will observe, the output provides a summary of the configuration along with statistics. A router uses the statistics (over time) to calculate rates and display them. Rates in these formulations represent a *decayed average (rate)*. The frequency (default, 300 seconds) of the calculation hinges on the load-interval for that interface. Statistics in the **show policy-map interface** output persist until you issue a **clear counters** command. The dataplane updates the statistics every 10 seconds.
2. Notice that Classification and Policer Action statistics arise from different entities in the dataplane. Consequently, they might update at slightly different times (briefly, the action counters might exceed the classification counters).

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	As you will observe, the output provides a summary of the configuration along with statistics. A router uses the statistics (over time) to calculate rates and display them. Rates in these formulations represent a *decayed average (rate)*. The frequency (default, 300 seconds) of the calculation hinges on the load-interval for that interface. Statistics in the **show policy-map interface** output persist until you issue a **clear counters** command. The dataplane updates the statistics every 10 seconds.	
Step 2	Notice that Classification and Policer Action statistics arise from different entities in the dataplane. Consequently, they might update at slightly different times (briefly, the action counters might exceed the classification counters).	

Example 3: show platform hardware qfp active feature qos interface Command

This command should only be necessary if you believe the router is configured correctly but is behaving unexpectedly. Viewing information directly from the dataplane can help you assess whether any quantization of rates or burst parameters were necessary to accommodate the hardware.

The following example corresponds to the **show policy-map interface** output from the previous example:

```
show platform hardware qfp active feature qos interface g1/0/0

Interface: GigabitEthernet1/0/0, QFP interface: 9
  Direction: Input
    Hierarchy level: 0
    Policy name: simple_policer
      Class name: AF1, Policy name: simple_policer
        Police:
            cir:  20000000 bps,  bc: 638976 bytes
            pir: 0 bps,  be: 638976 bytes
            rate mode: Single Rate Mode
            conformed: 447 packets, 668712 bytes; actions:
              transmit
            exceeded: 427 packets, 638792 bytes; actions:
              drop
            violated: 126 packets, 188496 bytes; actions:
              drop
            color aware: No
            green_qos_group: 0, yellow_qos_group: 0
      Class name: class-default, Policy name: simple_policer
```

If you understand the output of the previous two commands (Example 1 and Example 2), this output should be pretty self-explanatory. However, you should be aware of the following points related to using this command:

Although we configured a single-rate two-color policer, the output of the dataplane command corresponds to a single-rate, three-color policer. The hardware always operates in a three-color mode. To achieve two-color functionality it simply matches the Violate and Exceed actions. When we push statistics to the

control plane, IOS aggregates the Exceed and Violate statistics to generate the expected appearance of a two-color policer.

Statistics in the dataplane are transitory. Every 10 seconds the dataplane pushes statistics to IOS and then clears its local counters. Essentially, all statistics observed through the dataplane command are counts of what transpired since the last push. This means that dataplane commands help you view hardware behavior in real time. For meaningful (persistent) statistics, however, you should always use the regular IOS **show policy-map interface** command.

Configuration Examples for QoS Packet Policing

Example 1: Simple Network Admission Control

In its simplest form a policer can be used to rate-limit all traffic entering an interface (and thereby a network). We assume that the network sending the traffic will "shape" what exits its egress interface and only send traffic that will conform to the contracted rate. We can use *egress scheduling* on the senders' network to apportion the contracted rate to different classes of traffic.

With this simplest example of policing no classification is required as the policer is intended to cap all traffic. We will consider all traffic as belonging to class-default in the absence of any user-defined classes.

In the following example, we have a GigabitEthernet connection but the customer has only contracted for a 100 Mbps service rate. The configuration could look something like this:

```
policy-map ingress_cap_all_100m
  class class-default
    police cir 100000000
!
interface GigabitEthernet1/0/0
  service-policy ingress_cap_all_100m
```

Example 2: Network Admission Control - Hierarchical Policers

In Example 1: Simple Network Admission Control, on page 25 we policed all traffic to a contracted service-rate, assuming that the sender would apportion bandwidth within that contracted rate. However, we may not always trust the sender to limit the traffic within an individual class. For example, say we offer a priority service (traffic guaranteed low latency through the network) but charge the user for different levels of priority access. By simply applying the simple policer in Example 1 we could not guarantee that the sender doesn't forward us more priority traffic than contracted. We can expand the example to enforce also a cap on an individual class of traffic.

In the following example, we limit the total admission to 100 Mbps AND ensure that voice traffic caps at 5 Mbps of traffic:

```
class-map match-all voice
  match dscp ef
!
! child policy to enforce 5Mbps Voice Traffic
!
policy-map ingress_police_child
  class voice
    police cir percent 5 bc 5 ms
!
policy-map police_ingress_parent
```

```
  class class-default
    police cir 100000000
  service-policy ingress_police_child
!
interface GigabitEthernet1/0/0
  service-policy in police_ingress_parent
```

Example 3: Network Admission Control - Color-Aware Policer

In Example 2: Network Admission Control - Hierarchical Policers, on page 25, we introduced the scheme of capping a particular class of traffic within the contracted service-rate. This scheme hinges on customer shaping of traffic to the service-rate.

If we received traffic at a rate exceeding the parent policer (the contracted service-rate), no guarantee exists that it would not drop some of the voice traffic admitted by the child policer. To ensure that any traffic admitted by the child policer is also admitted by the parent, you could employ a color-aware policer for the parent policer.

The following example shows how complex outcomes can be achieved with combinations of policers. Here, we mark all voice traffic admitted by the child policer as Green (qos-group1) and all traffic other than voice as Yellow (qos-group2). The parent policer is configured with a CIR that ensures that we forward all the Green traffic and a PIR that ensures that we enforce the contracted service-rate:

```
class-map match-all voice
  match dscp ef
!
!child policy to enforce 5Mbps Voice Traffic
!
policy-map ingress_police_child
  class voice
    !conforming voice marked Green, Excess Dropped
    police cir 5m bc 3125 conform-action set-qos-transmit 1
  class class-default
   !all traffic other than voice marked Yellow
   set qos-group2
!
class maps needed for color-aware policer
!
class-map policer-green
  match qos-group1
class-map policer-yellow
  match qos-group2
!
!parent policy to enforce 100Mbps service rate
!
policy-map ingress_police_parent
  class class-default
    police cir 5m bc 3125 pir 100m be 625000
      conform-action transmit
      exceed-action transmit
      violate-action drop
      conform-color policer-green exceed-color policer-yellow
    service-policy ingress_police_child
!
interface GigabitEthernet1/0/0
  service-policy in ingress_police_parent
```

Command Reference

police

As discussed within chapter there are three variants of the police command that achieve the same result, namely:

[no] police *cir* [**bc** *conform-burst*] [**pir** *pir*] [**be** *peak-burst*] [**conform-action** *action* [**exceed-action** *action* [**violate-action** *action*]]]

[no police cir *cir* [**bc** *conform-burst*] [**pir** *pir*] [**be** *peak-burst*] [**conform-action** *action* [**exceed-action** *action* [**violate-action** *action*]]]

[no] police rate*cir* [**bc** *conform-burst*] [**pir** *pir*] [**be** *peak-burst*] [**conform-action** *action* [**exceed-action** *action* [**violate-action** *action*]]]

Henceforth we shall denote this as:

[no] police [cir | rate]*cir* [**bc** *conform-burst*] [**pir** *pir*] [**be** *peak-burst*] [**conform-action** *action* [**exceed-action** *action* [**violate-action** *action*]]]

We have already seen how the same command may be used to configure different types of policers.

Rather than present a single CLI which would be confusing and option combinations which might not be correct, we will present a subset of the options depending on the policer type you wish to configure.

Single-Rate, Two-Color Policer

This policer type can be expressed on a single line if you require only one action per designated conformance level:

[no] police [cir | rate]*cir***[percent** *percent***][[bc]** *conform-burst* **[ms]**] [**account** *options*]**[conform-action** *action* **[exceed-action** *action***]]**

Multiple lines (using sub-modes) are necessary if we requiremore than one action:

[no] police [cir | rate]*cir***[percent** *percent***][[bc]** *conform-burst* **[ms]**][**account** *options*] **<return>**
[no] conform-action *action* **<return>**
[no] conform-action *action* **<return>**
[no] exceed-action *action* **<return>**
[no] exceed-action *action* **<return>**

Single-Rate, Three-Color Policer

This policer type can be expressed on a single line if you require only one action per designated conformance level:

[no] police [cir | rate]*cir***[percent** *percent***][[bc]** *conform-burst* **[ms]**] [**[be]** *exceed-burst* **[ms]**][**account** *options*]**conform-action** *action* **exceed-action** *action* **[violate-action** *action***]**

Multiple lines (using sub-modes) are necessary if we require more than one action:

[no] police [cir | rate]*cir***[percent** *percent***][[bc]** *conform-burst* **[ms]**] [**[be]** *exceed-burst* **[ms]**][**account** *options*] **<return>**
[no] conform-action *action* **<return>**
[no] conform-action *action* **<return>**
[no] exceed-action *action* **<return>**
[no] exceed-action *action* **<return>**
[no] violate-action *action* **<return>**
[no] violate-action *action* **<return>**

Dual-Rate, Three Color Policer

This policer type can be expressed on a single line if you require only one action per designated conformance level:

[no] police [cir | rate]*cir***[percent** *percent***][[bc]** *conform-burst* **[ms]**] **[pir]** *peak-rate* **[ms][[be]** *exceed-burst* **[ms]**][**account** *options*]**conform-action** *action* **exceed-action** *action* **[violate-action** *action***]**

Multiple lines (using sub-modes) are necessary if we require more than one action:

[no] police [cir | rate]*cir***[percent** *percent***][[bc]** *conform-burst* **[ms]**] [**[pir]** *peak-rate* **[ms]**][**[be]** *exceed-burst* **[ms]**][**account** *options*] **<return>**
[no] conform-action *action* **<return>**
[no] conform-action *action* **<return>**
[no] exceed-action *action* **<return>**
[no] exceed-action *action* **<return>**
[no] violate-action *action* **<return>**
[no] violate-action *action* **<return>**

Single-Rate, Three-Color, Color-Aware Policer

[no] police [cir | rate]*cir***[percent** *percent***][[bc]** *conform-burst* **[ms]**] [**[be]** *exceed-burst* **[ms]**][**account** *options*] **<return>**
[no] conform-action *action* **<return>**
[no] conform-action *action* **<return>**
[no] exceed-action *action* **<return>**
[no] exceed-action *action* **<return>**
[no] violate-action *action* **<return>**
[no] conform-color *conform-color* **exceed-color** *exceed-color***<return>**

Dual-Rate, Three-Color, Color-Aware Policer

[no] police [cir | rate]*cir***[percent** *percent***][[bc]** *conform-burst* **[ms]**][**[pir]** *peak-rate* **[ms]**] [**[be]** *exceed-burst* **[ms]**][**account** *options*] **<return>**
[no] conform-action *action* **<return>**
[no] conform-action *action* **<return>**
[no] exceed-action *action* **<return>**
[no] exceed-action *action* **<return>**
[no] violate-action *action* **<return>**
[no] conform-color *conform-color* **exceed-color** *exceed-color***<return>**

police Command Default and Modes; Keyword/Argument Descriptions

Command Default

Disabled

Command Modes

Policy-map class configuration (config-pmap-c) when specifying a single action to be applied to a marked packet

Policy-map class police configuration (config-pmap-c-police) submode when specifying multiple actions to be applied to a marked packet

Syntax Description

The following table list the keywords/arguments for the **police** command and their purpose.

Keyword/Argument	Definition
bc	Specifies the Conforming Burst Size(Bc).
be	Specifies the Exceeding Burst Size(Be).
cir	Specifies the Committed Information Rate(CIR).
Conform-Action	Specifies the action to take on traffic that is determined to be "conforming."
Exceed-Action	Specifies the action to take on traffic that is determined to be "exceeding."
pir	Specifies the Peak Information Rate (PIR).
Violate-Action	Specifies the action to take on traffic that is determined to be "violating."

The following table lists the options for the Account keyword.

Table 1: Account keyword options

Option	Purpose
qinq	Specifies queue-in-queue encapsulation as the BRAS-DSLAM encapsulation type
dot1q	Specifies IEEE 802.1Q VLAN encapsulation as the BRAS-DSLAM encapsulation type
aal5	Specifies the ATM Adaptation Layer 5 that supports connection-oriented variable bit rate (VBR) services
aal3	Specifies the ATM Adaptation Layer 3 that supports both connectionless and connection-oriented links

Option	Purpose
subscriber-encapsulation	Specifies the encapsulation type at the subscriber line
user-defined	Indicates that the router is to use the offset value that you specify when adjusting policing length
offset	Specifies the number of bytes that the router is to use when calculating overhead. Valid values are from -63 to 63 bytes
atm	Applies the ATM cell tax in the ATM overhead calculation