



CHAPTER 24

Configuring QoS

Have you ever participated in a long-distance phone call that involved a satellite connection? The conversation might be interrupted with brief, but perceptible, gaps at odd intervals. Those gaps are the time, called the latency, between the arrival of packets being transmitted over the network. Some network traffic, such as voice and video, cannot tolerate long latency times. Quality of Service (QoS) is a feature that lets you give priority to critical traffic, prevent bandwidth hogging, and manage network bottlenecks to prevent packet drops.

This chapter describes how to apply QoS policies, and includes the following sections:

- [QoS Overview, page 24-1](#)
- [Creating the Standard Priority Queue for an Interface, page 24-5](#)
- [Identifying Traffic for QoS Using Class Maps, page 24-8](#)
- [Creating a Policy for Standard Priority Queueing and/or Policing, page 24-9](#)
- [Creating a Policy for Traffic Shaping and Hierarchical Priority Queueing, page 24-11](#)
- [Viewing QoS Statistics, page 24-13](#)

QoS Overview

You should consider that in an ever-changing network environment, QoS is not a one-time deployment, but an ongoing, essential part of network design.



Note

QoS is only available in single context mode.

This section describes the QoS features supported by the security appliance, and includes the following topics:

- [Supported QoS Features, page 24-2](#)
- [What is a Token Bucket?, page 24-2](#)
- [Policing Overview, page 24-3](#)
- [Priority Queueing Overview, page 24-3](#)
- [Traffic Shaping Overview, page 24-4](#)
- [DSCP and DiffServ Preservation, page 24-5](#)

Supported QoS Features

The security appliance supports the following QoS features:

- **Policing**—To prevent individual flows from hogging the network bandwidth, you can limit the maximum bandwidth used per flow. See the [“Policing Overview” section on page 24-3](#) for more information.
- **Priority queuing**—For critical traffic that cannot tolerate latency, such as Voice over IP (VoIP), you can identify traffic for Low Latency Queuing (LLQ) so that it is always transmitted ahead of other traffic. See the [“Priority Queueing Overview” section on page 24-3](#) for more information.
- **Traffic shaping**—If you have a device that transmits packets at a high speed, such as a security appliance with Fast Ethernet, and it is connected to a low speed device such as a cable modem, then the cable modem is a bottleneck at which packets are frequently dropped. To manage networks with differing line speeds, you can configure the security appliance to transmit packets at a fixed slower rate. See the [“Traffic Shaping Overview” section on page 24-4](#) for more information.

What is a Token Bucket?

A token bucket is used to manage a device that regulates the data in a flow. For example, the regulator might be a traffic policer or a traffic shaper. A token bucket itself has no discard or priority policy. Rather, a token bucket discards tokens and leaves to the flow the problem of managing its transmission queue if the flow overdrives the regulator.

A token bucket is a formal definition of a rate of transfer. It has three components: a burst size, an average rate, and a time interval. Although the average rate is generally represented as bits per second, any two values may be derived from the third by the relation shown as follows:

average rate = burst size / time interval

Here are some definitions of these terms:

- **Average rate**—Also called the committed information rate (CIR), it specifies how much data can be sent or forwarded per unit time on average.
- **Burst size**—Also called the Committed Burst (Bc) size, it specifies in bits or bytes per burst how much traffic can be sent within a given unit of time to not create scheduling concerns. (For traffic shaping, it specifies bits per burst; for policing, it specifies bytes per burst.)
- **Time interval**—Also called the measurement interval, it specifies the time quantum in seconds per burst.

In the token bucket metaphor, tokens are put into the bucket at a certain rate. The bucket itself has a specified capacity. If the bucket fills to capacity, newly arriving tokens are discarded. Each token is permission for the source to send a certain number of bits into the network. To send a packet, the regulator must remove from the bucket a number of tokens equal in representation to the packet size.

If not enough tokens are in the bucket to send a packet, the packet either waits until the bucket has enough tokens (in the case of traffic shaping) or the packet is discarded or marked down (in the case of policing). If the bucket is already full of tokens, incoming tokens overflow and are not available to future packets. Thus, at any time, the largest burst a source can send into the network is roughly proportional to the size of the bucket.

Note that the token bucket mechanism used for traffic shaping has both a token bucket and a data buffer, or queue; if it did not have a data buffer, it would be a policer. For traffic shaping, packets that arrive that cannot be sent immediately are delayed in the data buffer.

For traffic shaping, a token bucket permits burstiness but bounds it. It guarantees that the burstiness is bounded so that the flow will never send faster than the token bucket capacity, divided by the time interval, plus the established rate at which tokens are placed in the token bucket. See the following formula:

$$(\text{token bucket capacity in bits} / \text{time interval in seconds}) + \text{established rate in bps} = \text{maximum flow speed in bps}$$

This method of bounding burstiness also guarantees that the long-term transmission rate will not exceed the established rate at which tokens are placed in the bucket.

Policing Overview

Policing is a way of ensuring that no traffic exceeds the maximum rate (in bits/second) that you configure, thus ensuring that no one traffic flow or class can take over the entire resource. When traffic exceeds the maximum rate, the security appliance drops the excess traffic. Policing also sets the largest single burst of traffic allowed.

Priority Queueing Overview

LLQ priority queueing lets you prioritize certain traffic flows (such as latency-sensitive traffic like voice and video) ahead of other traffic.

The security appliance supports two types of priority queueing:

- Standard priority queueing—Standard priority queueing uses an LLQ priority queue on an interface (see the [“Creating the Standard Priority Queue for an Interface”](#) section on page 24-5), while all other traffic goes into the “best effort” queue. Because queues are not of infinite size, they can fill and overflow. When a queue is full, any additional packets cannot get into the queue and are dropped. This is called *tail drop*. To avoid having the queue fill up, you can increase the queue buffer size. You can also fine-tune the maximum number of packets allowed into the transmit queue. These options let you control the latency and robustness of the priority queueing. Packets in the LLQ queue are always transmitted before packets in the best effort queue.
- Hierarchical priority queueing—Hierarchical priority queueing is used on interfaces on which you enable a traffic shaping queue. A subset of the shaped traffic can be prioritized. The standard priority queue is not used. See the following guidelines about hierarchical priority queueing:
 - Priority packets are always queued at the head of the shape queue so they are always transmitted ahead of other non-priority queued packets.
 - Priority packets are never dropped from the shape queue unless the sustained rate of priority traffic exceeds the shape rate.
 - For IPSec-encrypted packets, you can only match traffic based on the DSCP or precedence setting.
 - IPSec-over-TCP is not supported for priority traffic classification.

Traffic Shaping Overview

Traffic shaping is used to match device and link speeds, thereby controlling packet loss, variable delay, and link saturation, which can cause jitter and delay.

- Traffic shaping must be applied to all outgoing traffic on a physical interface or in the case of the ASA 5505, on a VLAN. You cannot configure traffic shaping for specific types of traffic.
- Traffic shaping is implemented when packets are ready to be transmitted on an interface, so the rate calculation is performed based on the actual size of a packet to be transmitted, including all the possible overhead such as the IPSec header and L2 header.
- The shaped traffic includes both through-the-box and from-the-box traffic.
- The shape rate calculation is based on the standard token bucket algorithm. The token bucket size is twice the Burst Size value. See the [“What is a Token Bucket?”](#) section on page 24-2.
- When bursty traffic exceeds the specified shape rate, packets are queued and transmitted later. Following are some characteristics regarding the shape queue (for information about hierarchical priority queueing, see the [“Priority Queueing Overview”](#) section on page 24-3):
 - The queue size is calculated based on the shape rate. The queue can hold the equivalent of 200-milliseconds worth of shape rate traffic, assuming a 1500-byte packet. The minimum queue size is 64.
 - When the queue limit is reached, packets are tail-dropped.
 - Certain critical keep-alive packets such as OSPF Hello packets are never dropped.
 - The time interval is derived by $time_interval = burst_size / average_rate$. The larger the time interval is, the burstier the shaped traffic might be, and the longer the link might be idle. The effect can be best understood using the following exaggerated example:

Average Rate = 1000000

Burst Size = 1000000

In the above example, the time interval is 1 second, which means, 1 Mbps of traffic can be bursted out within the first 10 milliseconds of the 1-second interval on a 100 Mbps FE link and leave the remaining 990 milliseconds idle without being able to send any packets until the next time interval. So if there is delay-sensitive traffic such as voice traffic, the Burst Size should be reduced compared to the average rate so the time interval is reduced.

How QoS Features Interact

You can configure each of the QoS features alone if desired for the security appliance. Often, though, you configure multiple QoS features on the security appliance so you can prioritize some traffic, for example, and prevent other traffic from causing bandwidth problems.

See the following supported feature combinations per interface:

- Standard priority queuing (for specific traffic) + Policing (for the rest of the traffic).
You cannot configure priority queueing and policing for the same set of traffic.
- Traffic shaping (for all traffic on an interface) + Hierarchical priority queueing (for a subset of traffic).

You cannot configure traffic shaping and standard priority queueing for the same interface; only hierarchical priority queueing is allowed. For example, if you configure standard priority queueing for the global policy, and then configure traffic shaping for a specific interface, the feature you configured last is rejected because the global policy overlaps the interface policy.

Typically, if you enable traffic shaping, you do not also enable policing for the same traffic, although the security appliance does not restrict you from configuring this.

DSCP and DiffServ Preservation

- DSCP markings are preserved on all traffic passing through the security appliance.
- The security appliance does not locally mark/remark any classified traffic, but it honors the Expedited Forwarding (EF) DSCP bits of every packet to determine if it requires “priority” handling and will direct those packets to the LLQ.
- DiffServ marking is preserved on packets when they traverse the service provider backbone so that QoS can be applied in transit (QoS tunnel pre-classification).

Creating the Standard Priority Queue for an Interface

If you enable standard priority queueing for traffic on a physical interface, then you need to also create the priority queue on each interface. Each physical interface uses two queues: one for priority traffic, and the other for all other traffic. For the other traffic, you can optionally configure policing.



Note

The standard priority queue is not required for hierarchical priority queueing with traffic shaping; see the [“Priority Queueing Overview”](#) section on page 24-3 for more information.

This section includes the following topics:

- [Determining the Queue and TX Ring Limits](#), page 24-6
- [Configuring the Priority Queue](#), page 24-7

Determining the Queue and TX Ring Limits

To determine the priority queue and TX ring limits, use the worksheets below.

Table 24-1 shows how to calculate the priority queue size. Because queues are not of infinite size, they can fill and overflow. When a queue is full, any additional packets cannot get into the queue and are dropped (called *tail drop*). To avoid having the queue fill up, you can adjust the queue buffer size according to the “Configuring the Priority Queue” section on page 24-7.

Table 24-1 Queue Limit Worksheet

Step 1	_____ Mbps	x	125	=	_____
	<i>Outbound bandwidth (Mbps or Kbps)¹</i>				<i># of bytes/ms</i>
	_____ Kbps	x	.125	=	_____
					<i># of bytes/ms</i>
Step 2	_____	÷	_____	x	_____
	<i># of bytes/ms from Step 1</i>		<i>Average packet size (bytes)²</i>		<i>Delay (ms)³</i>
				=	_____
					<i>Queue limit (# of packets)</i>

- For example, DSL might have an uplink speed of 768 Kbps. Check with your provider.
- Determine this value from a codec or sampling size. For example, for VoIP over VPN, you might use 160 bytes. We recommend 256 bytes if you do not know what size to use.
- The delay depends on your application. For example, for VoIP, you might use 200 ms. We recommend 500 ms if you do not know what delay to use.

Table 24-2 shows how to calculate the TX ring limit. This limit determines the maximum number of packets allowed into the Ethernet transmit driver before the driver pushes back to the queues on the interface to let them buffer packets until the congestion clears. This setting guarantees that the hardware-based transmit ring imposes a limited amount of extra latency for a high-priority packet.

Table 24-2 TX Ring Limit Worksheet

Step 1	_____ Mbps	x	125	=	_____
	<i>Outbound bandwidth (Mbps or Kbps)¹</i>				<i># of bytes/ms</i>
	_____ Kbps	x	0.125	=	_____
					<i># of bytes/ms</i>
Step 2	_____	÷	_____	x	_____
	<i># of bytes/ms from Step 1</i>		<i>Maximum packet size (bytes)²</i>		<i>Delay (ms)³</i>
				=	_____
					<i>TX ring limit (# of packets)</i>

- For example, DSL might have an uplink speed of 768 Kbps. Check with your provider.
- Typically, the maximum size is 1550 bytes, but if you allow jumbo frames (if supported for your platform), then the packet size might be larger.
- The delay depends on your application. For example, for VoIP, you might use 20 ms.

Configuring the Priority Queue

To create the priority queue, perform the following steps.

Step 1 To create the priority queue, enter the following command:

```
hostname(config)# priority-queue interface_name
```

Where the *interface_name* argument specifies the physical interface name on which you want to enable the priority queue, or for the ASA 5505, the VLAN interface name.

Step 2 (Optional) To change the size of the priority queues, enter the following command:

```
hostname(config-priority-queue)# queue-limit number_of_packets
```

The default queue limit is 1024 packets. Because queues are not of infinite size, they can fill and overflow. When a queue is full, any additional packets cannot get into the queue and are dropped (called *tail drop*). To avoid having the queue fill up, you can use the **queue-limit** command to increase the queue buffer size.

See the “[Determining the Queue and TX Ring Limits](#)” section on page 24-6 to determine the *number_of_packets* value.

The upper limit of the range of values for the **queue-limit** command is determined dynamically at run time. To view this limit, enter **queue-limit ?** on the command line. The key determinants are the memory needed to support the queues and the memory available on the device.

The **queue-limit** that you specify affects both the higher priority low-latency queue and the best effort queue.

Step 3 (Optional) To specify the depth of the priority queues, enter the following command:

```
hostname(config-priority-queue)# tx-ring-limit number_of_packets
```

The default tx-ring-limit is 128 packets. This command sets the maximum number of low-latency or normal priority packets allowed into the Ethernet transmit driver before the driver pushes back to the queues on the interface to let them buffer packets until the congestion clears. This setting guarantees that the hardware-based transmit ring imposes a limited amount of extra latency for a high-priority packet.

See the “[Determining the Queue and TX Ring Limits](#)” section on page 24-6 to determine the *number_of_packets* value.

The upper limit of the range of values for the **tx-ring-limit** command is determined dynamically at run time. To view this limit, enter **tx-ring-limit ?** on the command line. The key determinants are the memory needed to support the queues and the memory available on the device.

The **tx-ring-limit** that you specify affects both the higher priority low-latency queue and the best-effort queue.

The following example establishes a priority queue on interface “outside” (the GigabitEthernet0/1 interface), with the default queue-limit and tx-ring-limit.

```
hostname(config)# priority-queue outside
```

The following example establishes a priority queue on the interface “outside” (the GigabitEthernet0/1 interface), sets the queue-limit to 260 packets, and sets the tx-ring-limit to 3:

```
hostname(config)# priority-queue outside  
hostname(config-priority-queue)# queue-limit 260  
hostname(config-priority-queue)# tx-ring-limit 3
```

Identifying Traffic for QoS Using Class Maps

QoS is part of the Modular Policy Framework. See the [Chapter 21, “Using Modular Policy Framework,”](#) for more information. In Modular Policy Framework, you identify the traffic on which you want to enable QoS in a class map. This section includes the following topics:

- [Creating a QoS Class Map, page 24-8](#)
- [QoS Class Map Examples, page 24-8](#)

Creating a QoS Class Map

For priority traffic, identify only latency-sensitive traffic. For policing traffic, you can choose to police all other traffic, or you can limit the traffic to certain types. For traffic shaping, all traffic on an interface must be shaped.

To create the class maps for QoS traffic, see the **class-map** command in the [“Identifying Traffic \(Layer 3/4 Class Map\)”](#) section on page 21-4.

You can match traffic based on many characteristics, including access lists, tunnel groups, DSCP, precedence, and more. See the following guidelines for configuring class maps for QoS:

- For traffic shaping, you can only use the **class-default** class map, which is automatically created by the security appliance, and which matches all traffic.
- You cannot use the **class-default** class map for priority traffic.
- For hierarchical priority queueing, for IPSec-encrypted packets, you can only match traffic based on the DSCP or precedence setting.
- For hierarchical priority queueing, IPSec-over-TCP traffic is not supported.

QoS Class Map Examples

For example, in the following sequence, the **class-map** command classifies all non-tunneled TCP traffic, using an access list named `tcp_traffic`:

```
hostname(config)# access-list tcp_traffic permit tcp any any
hostname(config)# class-map tcp_traffic
hostname(config-cmap)# match access-list tcp_traffic
```

In the following example, other, more specific match criteria are used for classifying traffic for specific, security-related tunnel groups. These specific match criteria stipulate that a match on tunnel-group (in this case, the previously-defined Tunnel-Group-1) is required as the first match characteristic to classify traffic for a specific tunnel, and it allows for an additional match line to classify the traffic (IP differential services code point, expedited forwarding).

```
hostname(config)# class-map TG1-voice
hostname(config-cmap)# match tunnel-group tunnel-grp1
hostname(config-cmap)# match dscp ef
```

In the following example, the **class-map** command classifies both tunneled and non-tunneled traffic according to the traffic type:

```
hostname(config)# access-list tunneled extended permit ip 10.10.34.0 255.255.255.0
20.20.10.0 255.255.255.0
hostname(config)# access-list non-tunneled extended permit tcp any any
hostname(config)# tunnel-group tunnel-grp1 type IPSec_L2L
```

```

hostname(config)# class-map browse
hostname(config-cmap)# description "This class-map matches all non-tunneled tcp traffic."
hostname(config-cmap)# match access-list non-tunneled

hostname(config-cmap)# class-map TG1-voice
hostname(config-cmap)# description "This class-map matches all dscp ef traffic for
tunnel-grp 1."
hostname(config-cmap)# match dscp ef
hostname(config-cmap)# match tunnel-group tunnel-grp1

hostname(config-cmap)# class-map TG1-BestEffort
hostname(config-cmap)# description "This class-map matches all best-effort traffic for
tunnel-grp1."
hostname(config-cmap)# match tunnel-group tunnel-grp1
hostname(config-cmap)# match flow ip destination-address

```

The following example shows a way of policing a flow within a tunnel, provided the classed traffic is not specified as a tunnel, but does go *through* the tunnel. In this example, 192.168.10.10 is the address of the host machine on the private side of the remote tunnel, and the access list is named “host-over-l2l”. By creating a class-map (named “host-specific”), you can then police the “host-specific” class before the LAN-to-LAN connection polices the tunnel. In this example, the “host-specific” traffic is rate-limited before the tunnel, then the tunnel is rate-limited:

```

hostname(config)# access-list host-over-l2l extended permit ip any host 192.168.10.10
hostname(config)# class-map host-specific
hostname(config-cmap)# match access-list host-over-l2l

```

The following example builds on the configuration developed in the previous section. As in the previous example, there are two named class-maps: tcp_traffic and TG1-voice.

```

hostname(config)# class-map TG1-best-effort
hostname(config-cmap)# match tunnel-group Tunnel-Group-1
hostname(config-cmap)# match flow ip destination-address

```

Adding a third class map provides a basis for defining a tunneled and non-tunneled QoS policy, as follows, which creates a simple QoS policy for tunneled and non-tunneled traffic, assigning packets of the class TG1-voice to the low latency queue and setting rate limits on the tcp_traffic and TG1-best-effort traffic flows.

Creating a Policy for Standard Priority Queueing and/or Policing

After you identify the traffic in “[Identifying Traffic for QoS Using Class Maps](#)” section on page 24-8, you can create a policy map for an interface or globally for all interfaces that assigns QoS actions (and other feature actions) to the traffic in the class map. (See the [Chapter 21, “Using Modular Policy Framework,”](#) for information about other features. This chapter only discusses QoS.)

You can configure standard priority queueing and policing for different class maps within the same policy map. See the “[How QoS Features Interact](#)” section on page 24-4 for information about valid QoS configurations.

To create a policy map, perform the following steps:

Step 1 To add or edit a policy map, enter the following command:

```
hostname(config)# policy-map name
```

For example:

```
hostname(config)# policy-map QoS_policy
```

Step 2 To configure priority queueing, enter the following commands:

```
hostname(config-pmap)# class priority_map_name
hostname(config-pmap-c)# priority
```

where the *priority_map_name* is the class map you created for prioritized traffic in “[Identifying Traffic for QoS Using Class Maps](#)” section on page 24-8.

For example:

```
hostname(config)# class-map priority-class
hostname(config-cmap)# match tunnel-group Tunnel-Group-1
hostname(config-cmap)# match dscp ef

hostname(config-cmap)# policy-map QoS_policy

hostname(config-pmap)# class priority_class
hostname(config-pmap-c)# priority
```

Step 3 To configure policing, enter the following commands:

```
hostname(config-pmap)# class policing_map_name
hostname(config-pmap-c)# police {output | input} conform-rate [conform-burst]
[conform-action [drop | transmit]] [exceed-action [drop | transmit]]
```

where the *policing_map_name* is the class map you created for prioritized traffic in “[Identifying Traffic for QoS Using Class Maps](#)” section on page 24-8.

The *conform-burst* argument specifies the maximum number of instantaneous bytes allowed in a sustained burst before throttling to the conforming rate value, between 1000 and 512000000 bytes.

The **conform-action** keyword sets the action to take when the rate is less than the *conform_burst* value.

The *conform-rate* argument sets the rate limit for this traffic flow; between 8000 and 2000000000 bits per second.

The **drop** keyword drops the packet.

The **exceed-action** keyword sets the action to take when the rate is between the *conform-rate* value and the *conform-burst* value.

The **input** keyword enables policing of traffic flowing in the input direction.

The **output** keyword enables policing of traffic flowing in the output direction.

The **transmit** keyword transmits the packet.

For example:

```
hostname(config)# class-map policing-class
hostname(config-cmap)# match any

hostname(config-cmap)# policy-map QoS_policy

hostname(config-pmap)# class police_class
hostname(config-pmap-c)# police output 56000 10500
```

Step 4 To activate the policy map on one or more interfaces, enter the following command:

```
hostname(config)# service-policy policymap_name {global | interface interface_name}
```

Where **global** applies the policy map to all interfaces, and **interface** applies the policy to one interface. Only one global policy is allowed. Interface service policies take precedence over the global service policy for a given feature. For example, if you have a global policy with inspections, and an interface

policy with TCP normalization, then both inspections and TCP normalization are applied to the interface. However, if you have a global policy with inspections, and an interface policy with inspections, then only the interface policy inspections are applied to that interface.

In this example, the maximum rate for traffic of the `tcp_traffic` class is 56,000 bits/second and a maximum burst size of 10,500 bytes per second. For the `TC1-BestEffort` class, the maximum rate is 200,000 bits/second, with a maximum burst of 37,500 bytes/second. Traffic in the `TC1-voice` class has no policed maximum speed or burst rate because it belongs to a priority class.

```
hostname(config)# access-list tcp_traffic permit tcp any any
hostname(config)# class-map tcp_traffic
hostname(config-cmap)# match access-list tcp_traffic

hostname(config)# class-map TG1-voice
hostname(config-cmap)# match tunnel-group tunnel-grp1
hostname(config-cmap)# match dscp ef

hostname(config-cmap)# class-map TG1-BestEffort
hostname(config-cmap)# match tunnel-group tunnel-grp1
hostname(config-cmap)# match flow ip destination-address

hostname(config)# policy-map qos
hostname(config-pmap)# class tcp_traffic
hostname(config-pmap-c)# police output 56000 10500

hostname(config-pmap-c)# class TG1-voice
hostname(config-pmap-c)# priority

hostname(config-pmap-c)# class TG1-best-effort
hostname(config-pmap-c)# police output 200000 37500

hostname(config-pmap-c)# class class-default
hostname(config-pmap-c)# police output 1000000 37500

hostname(config-pmap-c)# service-policy qos global
```

Creating a Policy for Traffic Shaping and Hierarchical Priority Queueing

You can create a policy map for an interface or globally for all interfaces that assigns QoS actions (and other feature actions) to the traffic in the class map. (See the [Chapter 21, “Using Modular Policy Framework,”](#) for information about other features. This chapter only discusses QoS.)

You can configure traffic shaping for all traffic on an interface, and optionally hierarchical priority queueing for a subset of latency-sensitive traffic. See the [“How QoS Features Interact”](#) section on [page 24-4](#) for information about valid QoS configurations.

If you want to configure hierarchical priority queueing, then first identify the traffic in [“Identifying Traffic for QoS Using Class Maps”](#) section on [page 24-8](#); traffic shaping always uses the `class-default` class map, which is automatically available.

**Note**

One side-effect of priority queueing is packet re-ordering. For IPSec packets, out-of-order packets that are not within the anti-replay window generate warning syslog messages. These warnings are false alarms in the case of priority queueing. You can configure the IPSec anti-replay window size to avoid possible false alarms. See the **crypto ipsec security-association replay** command in the *Cisco Security Appliance Command Reference*.

To create a policy map, perform the following steps:

- Step 1** (Optional) For hierarchical priority queueing, create a policy map that applies the priority queueing action to a class map by entering the following commands:

```
hostname(config)# policy-map name
hostname(config-pmap)# class priority_map_name
hostname(config-pmap-c)# priority
```

where the *priority_map_name* is the class map you created for prioritized traffic in “[Identifying Traffic for QoS Using Class Maps](#)” section on page 24-8.

For example:

```
hostname(config)# policy-map priority-sub-policy
hostname(config-pmap)# class priority-sub-map
hostname(config-pmap-c)# priority
```

- Step 2** To add or edit a policy map for traffic shaping, enter the following command:

```
hostname(config)# policy-map name
```

For example:

```
hostname(config)# policy-map shape_policy
```

- Step 3** To configure traffic shaping, enter the following commands:

```
hostname(config-pmap)# class class-default
hostname(config-pmap-c)# shape average rate [burst_size]
```

where the **average rate** argument sets the average rate of traffic in bits per second over a given fixed time period, between 64000 and 154400000. Specify a value that is a multiple of 8000. See the “[Traffic Shaping Overview](#)” section on page 24-4 for more information about how the time period is calculated.

The *burst_size* argument sets the average burst size in bits that can be transmitted over a given fixed time period, between 2048 and 154400000. Specify a value that is a multiple of 128. If you do not specify the *burst_size*, the default value is equivalent to 4-milliseconds of traffic at the specified average rate. For example, if the average rate is 1000000 bits per second, 4 ms worth = $1000000 * 4/1000 = 4000$.

You can only identify the **class-default** class map, which is defined as **match any**, because the security appliance requires all traffic to be matched for traffic shaping.

- Step 4** (Optional) To configure hierarchical priority queueing, enter the following command:

```
hostname(config-pmap-c)# service-policy priority_policy_map_name
```

where the *priority_policy_map_name* is the policy map you created for prioritized traffic in [Step 1](#).

For example:

```
hostname(config)# policy-map priority-sub-policy
hostname(config-pmap)# class priority-sub-map
hostname(config-pmap-c)# priority
```

```
hostname(config-pmap-c) # policy-map shape_policy
hostname(config-pmap) # class class-default
hostname(config-pmap-c) # shape
hostname(config-pmap-c) # service-policy priority-sub-policy
```

Step 5 To activate the policy map on an interface, enter the following command:

```
hostname(config) # service-policy polycymap_name interface interface_name
```



Note

You cannot configure traffic shaping in the global policy.

The following example enables traffic shaping on the outside interface, and limits traffic to 2 Mbps; priority queuing is enabled for VoIP traffic that is tagged with DSCP EF and AF13 and for IKE traffic:

```
hostname(config) # access-list ike permit udp any any eq 500
hostname(config) # class-map ike
hostname(config-cmap) # match access-list ike

hostname(config-cmap) # class-map voice_traffic
hostname(config-cmap) # match dscp EF AF13

hostname(config-cmap) # policy-map qos_class_policy
hostname(config-pmap) # class voice_traffic
hostname(config-pmap-c) # priority
hostname(config-pmap-c) # class ike
hostname(config-pmap-c) # priority

hostname(config-pmap-c) # policy-map qos_outside_policy
hostname(config-pmap) # class class-default
hostname(config-pmap-c) # shape average 2000000 16000
hostname(config-pmap-c) # service-policy qos_class_policy

hostname(config-pmap-c) # service-policy qos_outside_policy interface outside
```

Viewing QoS Statistics

This section includes the following topics:

- [Viewing QoS Police Statistics, page 24-13](#)
- [Viewing QoS Standard Priority Statistics, page 24-14](#)
- [Viewing QoS Shaping Statistics, page 24-14](#)
- [Viewing QoS Standard Priority Queue Statistics, page 24-15](#)

Viewing QoS Police Statistics

To view the QoS statistics for traffic policing, use the **show service-policy** command with the **police** keyword:

```
hostname# show service-policy police
```

The following is sample output for the **show service-policy police** command:

```

hostname# show service-policy police

Global policy:
  Service-policy: global_fw_policy

Interface outside:
  Service-policy: qos
  Class-map: browse
    police Interface outside:
      cir 56000 bps, bc 10500 bytes
      conformed 10065 packets, 12621510 bytes; actions: transmit
      exceeded 499 packets, 625146 bytes; actions: drop
      conformed 5600 bps, exceed 5016 bps
  Class-map: cmap2
    police Interface outside:
      cir 200000 bps, bc 37500 bytes
      conformed 17179 packets, 20614800 bytes; actions: transmit
      exceeded 617 packets, 770718 bytes; actions: drop
      conformed 198785 bps, exceed 2303 bps

```

Viewing QoS Standard Priority Statistics

To view statistics for service policies implementing the **priority** command, use the **show service-policy** command with the **priority** keyword:

```
hostname# show service-policy priority
```

The following is sample output for the **show service-policy priority** command:

```

hostname# show service-policy priority
Global policy:
  Service-policy: global_fw_policy
Interface outside:
  Service-policy: qos
  Class-map: TG1-voice
  Priority:
    Interface outside: aggregate drop 0, aggregate transmit 9383

```



Note

“Aggregate drop” denotes the aggregated drop in this interface; “aggregate transmit” denotes the aggregated number of transmitted packets in this interface.

Viewing QoS Shaping Statistics

To view statistics for service policies implementing the **shape** command, use the **show service-policy** command with the **shape** keyword:

```
hostname# show service-policy shape
```

The following is sample output for the **show service-policy shape** command:

```

hostname# show service-policy shape
Interface outside
  Service-policy: shape
  Class-map: class-default

  Queueing
    queue limit 64 packets

```

```
(queue depth/total drops/no-buffer drops) 0/0/0
(pkts output/bytes output) 0/0

shape (average) cir 2000000, bc 8000, be 8000
```

The following is sample output of the **show service policy shape** command, which includes service policies that include the **shape** command and the **service-policy** command that calls the hierarchical priority policy and the related statistics:

```
hostname# show service-policy shape

Interface outside:
  Service-policy: shape
    Class-map: class-default

      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0

      shape (average) cir 2000000, bc 16000, be 16000

    Service-policy: voip
      Class-map: voip

        Queueing
        queue limit 64 packets
        (queue depth/total drops/no-buffer drops) 0/0/0
        (pkts output/bytes output) 0/0
        Class-map: class-default

          queue limit 64 packets
          (queue depth/total drops/no-buffer drops) 0/0/0
          (pkts output/bytes output) 0/0
```

Viewing QoS Standard Priority Queue Statistics

To display the priority-queue statistics for an interface, use the **show priority-queue statistics** command in privileged EXEC mode. The results show the statistics for both the best-effort (BE) queue and the low-latency queue (LLQ). The following example shows the use of the **show priority-queue statistics** command for the interface named test, and the command output.

```
hostname# show priority-queue statistics test

Priority-Queue Statistics interface test

Queue Type      = BE
Packets Dropped = 0
Packets Transmit = 0
Packets Enqueued = 0
Current Q Length = 0
Max Q Length    = 0

Queue Type      = LLQ
Packets Dropped = 0
Packets Transmit = 0
Packets Enqueued = 0
Current Q Length = 0
Max Q Length    = 0
hostname#
```

In this statistical report, the meaning of the line items is as follows:

- “Packets Dropped” denotes the overall number of packets that have been dropped in this queue.
- “Packets Transmit” denotes the overall number of packets that have been transmitted in this queue.
- “Packets Enqueued” denotes the overall number of packets that have been queued in this queue.
- “Current Q Length” denotes the current depth of this queue.
- “Max Q Length” denotes the maximum depth that ever occurred in this queue.