QoS Hierarchical Queueing Framework

The QoS Hierarchical Queueing Framework (HQF) feature enables you to manage quality of service (QoS) at three different levels: the physical interface level, the logical interface level, and the class level for QoS queueing and shaping mechanisms by using the modular QoS command-line interface (MQC) to provide a granular and flexible overall QoS architecture. In Release 12.2(28)SB, this feature was introduced as QoS: Frame Relay QoS Hierarchical Queueing Framework Support on the Cisco 7200 Series Router.

- Finding Feature Information, page 1
- Prerequisites for QoS Hierarchical Queueing Framework, page 1
- Restrictions for QoS Hierarchical Queueing Framework, page 2
- Information About QoS Hierarchical Queueing Framework, page 2
- How to Configure QoS Hierarchical Queueing Framework, page 8
- Configuration Examples for QoS Hierarchical Queueing Framework, page 12
- Additional References for QoS Hierarchical Queueing Framework, page 14
- Feature Information for QoS Hierarchical Queueing Framework, page 14

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for QoS Hierarchical Queueing Framework

Configure MQC in your network.
Restrictions for QoS Hierarchical Queueing Framework

- Service policies with queueing features cannot simultaneously coexist on child and parent interfaces, such as tunnel and physical interfaces or subinterface and physical interfaces.
- If a queueing policy is applied on a tunnel interface, and if a queueing policy needs to be applied on the physical interface on which the tunnel is built, the pmap on tunnel needs to be removed before the pmap on the physical interface can be attached.
- Queuing based QoS policies are not supported on LAG (port-channel) interfaces.

Information About QoS Hierarchical Queueing Framework

Background of QoS Hierarchical Queueing Framework

MQC allows you to configure QoS using a generic CLI that is applicable to all types of interfaces and protocols. MQC builds configurations that depend on HQF for queueing and shaping.

For example, to support Frame Relay, extensions to the HQF mechanism were required so that fragmentation could be provided within the queueing framework. These extensions enable priority queueing (PQ) configurations to be set up to support latency-sensitive traffic.

Functions of QoS Hierarchical Queueing Framework

HQF provides queueing and shaping capabilities. HQF is a logical engine used to support QoS features. The HQF hierarchy is a tree structure that is built using policy maps.

When data passes through an interface using HQF, the data is classified so that it traverses the branches of the tree. Data arrives at the top of the tree and is classified on one of the leaves. Data then traverses down the hierarchy (tree) until it is transmitted out the interface at the root (trunk).

For example, the following configuration builds the hierarchy shown in the figure below:

```plaintext
policy-map class
  class c1
    bandwidth 14
  class c2
    bandwidth 18
policy-map map1
  class class-default
    shape average 64000
  service-policy class
policy-map map2
  class class-default
    shape average 96000
  map-class frame-relay fr1
  service-policy output map1
  map-class frame fr2
  service-policy output map2
interface serial4/1
  encapsulation frame-relay
  frame-relay interface-dlci 16
  class fr1
```
Benefits of QoS Hierarchical Queueing Framework

The QoS Hierarchical Queueing Framework feature provides the following benefits:

- Faster deployment of QoS queueing and shaping in large-scale networks.
- Consistent queueing behavior applied with common MQC CLI across all main Cisco software releases, making implementation of QoS easier and transparent regardless of the Cisco software release being used.
- Common functionality for both distributed and non-distributed implementations, providing consistency of QoS feature behavior across all software-forwarding hardware, thus making implementation of QoS easier and transparent regardless of the platform being used.
- Behavioral consistency across hardware, resulting in accelerated delivery of feature enhancements and new QoS features in different Cisco software releases.
- Multiple levels of packet scheduling.
- Support for integrated class-based shaping and queueing.
- The ability to apply fair queueing and drop policies on a per-class basis.
- Up to three levels of queueing can be configured on the egress. Queueing features are not supported in the ingress direction.
- Priority can be configured on a nonleaf node, but nonqueueing (policing or marking) classes can be configured only on its child.

New Functionality in QoS Hierarchical Queueing Framework

The QoS Hierarchical Queueing Framework feature introduces the following functionality.
Hierarchical Policy with Queueing Features at Every Level

You can apply class-based queueing to any traffic class in the parent or child level of a hierarchical policy and obtain service levels for different sessions or subscribers.

In the example shown below, the traffic belonging to class parent-c2 has more scheduling time than class parent-c1:

```plaintext
policy-map child
  class child-c1
    bandwidth 400
  class child-c2
    bandwidth 400
policy-map parent
  class parent-c1
    bandwidth 1000
    service-policy child
  class parent-c2
    bandwidth 2000
    service-policy child
```

Shaping in an ATM PVC Policy

You can apply class-based shaping within an ATM PVC as shown in the following example:

```plaintext
policy-map p1
  class c1
    shape average 1000000
  class c2
    shape average 1000000
interface atm1/0.1
  pvc 1/100
  service-policy output p1
policy-map p1
  class c1
    shape average 1000000
  class c2
    shape average 1000000
interface atm1/0.1
  pvc 1/100
  service-policy output p1
```

Child Policy in a Priority Class

You can apply a child policy to a class with priority enabled as shown in the following example. The child policy can contain police or set features, but not queueing features.

```plaintext
policy-map p1
  class c1
    priority 256
    service-policy child
```

Behavioral Changes in QoS Hierarchical Queueing Framework

The QoS Hierarchical Queueing Framework feature introduces the following behavioral changes in some QoS features:
Flow-Based Fair-Queueing Support in Class-Default

The fair-queueing behavior for the class-default class is flow-based. This is a change from the weighted fair queueing (WFQ) behavior in previous releases. With flow-based fair queueing, the flow queues in the class-default class are scheduled equally instead of by weight based on the IP Precedence bits.

Default Queueing Implementation for Class-Default

When you do not explicitly configure the class-default class in a policy map, its default queueing behavior is FIFO. You can configure the bandwidth, fair-queue, or service-policy commands in the class-default class to achieve different queueing behaviors.

Class-Default and Bandwidth

The bandwidth assigned to the class-default class is the unused interface bandwidth not consumed by user-defined classes. By default, the class-default class receives a minimum of 1% of the interface bandwidth.

Default Queueing Implementation for Shape Class

When you configure the shape command in a class, the default queueing behavior for the shape queue is FIFO instead of weighted fair queueing (WFQ). You can configure the bandwidth, fair-queue, or service-policy commands in shape class to achieve different queueing behaviors.

Policy Map and Interface Bandwidth

In HQF, a policy map can reserve up to 100 percent of the interface bandwidth. If you do not assign an explicit bandwidth guarantee to the class-default class, you can assign a maximum of 99 percent of the interface bandwidth to user-defined classes, and you can reserve the other 1 percent for the class-default class by using the percent keyword from the bandwidth (policy-map class) command. If you use the kbps argument, you can assign a maximum of the entire interface bandwidth minus 1 kilobits per second (kbps) to user-defined classes and reserve the remaining 1 kbps for the class-default class.

Note

If you are migrating to Cisco IOS Release 12.4(20)T and the configured policy map allocates 100 percent of the bandwidth to the user-defined classes, an error message appears on the console after booting the HQF image. The message indicates that the allocated bandwidth exceeds the allowable amount, and the service policy is rejected. In HQF, you must reconfigure the policy to account for the minimum 1 percent of bandwidth that is guaranteed for the class-default. Then you can apply a service policy to the interface.

Per-Flow Queue Limit in Fair Queueing

In HQF, when you enable fair queueing, the per-flow queue limit is calculated in one of the following ways:

- 1/4 * n (where n = queue limit)
- Two packets (in the case of packet-based queue limits)
- One MTU size (in the case of byte-based queue limits)

These values are static even if the number of flows increase, so consider the overall buffer pool when configuring the queue limit in order to avoid exhausting the buffer pool.
The queue limit per class in packets and bytes can be configured without fair queue. Therefore, the minimum value of queue limit per class and queue limit per flow is not connected.

It is recommended to use the default value or 200 ms worth of packets/bytes for the "queue limit per class".

**Over-Subscription Support for Multiple Policies on Logical Interfaces**

When you attach a shaping policy to multiple logical interfaces including a subinterface, and the sum of the shape rate exceeds the physical interface bandwidth, congestion at the physical interface results in back pressure to each logical interface policy. This back pressure causes each policy to reduce the output rate down to its fair share of the interface bandwidth.

Here is an example: 10 subinterface policies each shaped to 2 Mbps, physical interface has 10 Mbps bandwidth (2:1 oversubscription), when all 10 subinterfaces are sending at 2 Mbps, each subinterface gets a throughput of 1 Mbps (10 Mbps/10 subinterfaces).

**Shaping on a GRE Tunnel**

In HQF, you can apply the shaping to a generic routing encapsulation (GRE) tunnel by using a hierarchical service policy after encapsulation. This means that the shape rate is based on packets with tunnel encapsulation and L2 encapsulation.

When configuring the shape feature in the parent policy applied to the tunnel interface, you can use the class-default class only. You cannot configure a user-defined class in the parent policy.

A typical hierarchical policy applied to a GRE tunnel interface is shown below:

```
interface tunnel0
  service-policy output parent
  policy-map parent
    class class-default
      shape average 10000000
      service-policy child
      policy-map child
      class voice
        priority 512
      class video
        bandwidth 6000
      class data
        bandwidth 3000
```

Some QoS deployments include a service policy with queueing features applied at the tunnel or a virtual interface and a service policy with queueing features applied at the physical interface. In Cisco IOS Release 12.4(20)T, you can apply a service policy with queueing features only at one of these interfaces. When migrating to Cisco IOS Release 12.4(20)T, a router configuration containing service policies at both interfaces will keep only the one applied to the physical interface.

**FRF.12 and FRF.9**

With HQF implementation, when you enable Frame Relay fragmentation (FRF.12) on an FR PVC or FR main interface, priority class packets are no longer subject to fragmentation. Priority packets, regardless of the packet size, always interleave among data fragments.
When you enable Frame Relay payload compression (FRF.9) on an FR PVC or main interface, priority class packets are no longer compressed. When you enable both FRF.12 and FRF.9, priority class packets are neither fragmented nor compressed.

**User-Defined Classes Added to Policy Maps Attached to Logical Interfaces**

A policy map may be configured with multiple user-defined classes and may contain a default class, called class-default. Optionally, a policy map may contain just the class-default, as illustrated below:

```
policy-map parent  
class class-default  
service-policy child
```

Typically, at this point, you would attach the policy map to the interface. After the policy map has been attached the interface, the HQF would allow you to add a user-defined class to the policy map.

However, HQF behavior has now changed so that this kind of modification is no longer permitted on a logical interface. If you want to add a user-defined class to a policy map (and that policy map has already been attached to a logical interface), you must first remove the policy map from the logical interface. Then add the user-defined class to the policy map and reattach the policy map to the logical interface.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>This behavior change applies only to logical interfaces. It does not apply to physical interfaces.</td>
</tr>
</tbody>
</table>

**Nested Policy and Reference Bandwidth for Child Policy**

In HQF when you configure a nested policy with a child queueing policy under a parent shaping class, the reference bandwidth for the child queueing policy is taken from the following: minimum (parent shaper rate, parent class's implicit/explicit bandwidth guarantee). When you do not define bandwidth for the parent class, the interface bandwidth divides equally among all parent classes as the implicit bandwidth guarantee.

The example below shows a nested policy applied on a serial interface of 1536 kbps. The 1536 kbps is equally shared, as the implicit bandwidth, among parent classes parent-c1 and class-default. For the parent class, the shaping rate of 1200 kbps is the maximum, while the implicit guarantee of 768 kbps is the minimum.

```
interface serial 0/0  
service-policy parent  
policy-map child  
class child-c1  
bandwidth percent 10  
policy-map parent  
class parent-c1  
shape average 1200000  
service-policy child
```

For the child policy child-c1 to take the parent shaping rate as the reference bandwidth, configure parent class parent-c1 with an explicit guarantee greater than the shaping rate. For example,

```
policy-map parent  
class parent-c1  
bandwidth 1300  
shape average 1200000  
service-policy child
```

When configuring explicit bandwidth for parent classes with oversubscription, the restrictions in the "Policy Map Bandwidth" section applies.
Handling Traffic Congestion on an Interface Configured with a Policy Map

In Cisco IOS Release 12.4(20)T, if an interface configured with a policy map is congested, the implicitly defined queue allows the traffic as defined in the bandwidth statement of each traffic class. The queueing is activated whenever there is traffic congestion on an interface.

How to Configure QoS Hierarchical Queueing Framework

Configuring a Service Policy

**SUMMARY STEPS**

1. enable
2. configure terminal
3. policy-map [type access-control] policy-map-name
4. class [class-name | class-default]
5. shape [average | peak] cir [bc] [be]
6. interface type number
7. encapsulation frame-relay [cisco | ietf]
8. service-policy [type access-control] {input | output} policy-map-name
9. end

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> policy-map [type access-control] policy-map-name</td>
<td>Specifies the name of the policy map to be created. Enters policy-map configuration mode.</td>
</tr>
<tr>
<td>Example: Device(config)# policy-map shape</td>
<td>• The optional type access-control keywords determine the exact pattern to look for in the protocol stack of interest.</td>
</tr>
<tr>
<td></td>
<td>• Enter the policy-map name.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>**class [class-name</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>**shape [average</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>interface type number</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Device(config-pmap-c)# interface serial4/3&lt;br&gt;<strong>Purpose:</strong> Configures the interface type specified and enters interface configuration mode.&lt;br&gt;• Enter the interface type and number.</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>**encapsulation frame-relay [cisco</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>**service-policy [type access-control] {input</td>
</tr>
</tbody>
</table>
| **Step 9**         | **end**<br>**Example:**<br>Device(config-if)# end<br>**Purpose:** (Optional) Exits interface configuration mode.
# Attaching an MQC Policy to a Map Class

## SUMMARY STEPS

1. `enable`
2. `configure terminal`
3. `map-class frame-relay map-class-name`
4. `service-policy [type access-control] {input | output} policy-map-name`
5. `interface type number`
6. `frame-relay class name`
7. `frame-relay interface-dlci dlci [cisco | ietf] [voice-cir cir] [ppp virtual-template-name]`
8. `end`

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> map-class frame-relay map-class-name</td>
<td>Specifies the name of a Frame Relay map class that is to be created or modified and enters map-class configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config)# map-class frame-relay shape</td>
<td>• Enter the map-class name.</td>
</tr>
<tr>
<td><strong>Step 4</strong> service-policy [type access-control] {input</td>
<td>output} policy-map-name</td>
</tr>
<tr>
<td><strong>Note</strong> You can configure policy maps on ingress or egress devices and attach them in the input or output direction of an interface. The direction (input or output) and the device (ingress or egress) to which the policy map should be attached varies according to your network configuration.</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong> Device(config-map-class)# service-policy output shape</td>
<td>• The optional type access-control keywords determine the exact pattern to look for in the protocol stack of interest.</td>
</tr>
<tr>
<td></td>
<td>• Enter the input or output keyword followed by the policy-map name.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| **Step 5**       | interface *type number*  
**Example:**  
Device(config-map-class)# interface serial4/3  
- Configures the interface type specified and enters interface configuration mode.  
  - Enter the interface type and number. |
| **Step 6**       | frame-relay class *name*  
**Example:**  
Device(config-if)# frame-relay class shape  
- Associates a map class with an interface or subinterface.  
  - Enter the name of the map class. |
| **Step 7**       | frame-relay interface-dlci *dlci [cisco | ietf] [voice-cir *cir* [ppp virtual-template-name]]*  
**Example:**  
Device(config-if)# frame-relay interface-dlci 16  
- Assigns a data-link connection identifier (DLCI) to a specified Frame Relay subinterface on a device and enters Frame Relay DLCI interface configuration mode.  
  - Enter the DLCI number.  
  - (Optional) Enter cisco or ietf for the encapsulation type.  
  - (Optional; supported on the Cisco MC3810 only) Enter voice-cir and cir to specify the upper limit on the voice bandwidth that may be reserved for this DLCI. The default is the committed information rate (CIR) configured for the Frame Relay map class.  
  - (Optional) Enter ppp to enable the circuit to use PPP in Frame Relay encapsulation.  
  - (Optional) Enter the virtual template name to specify to which virtual template interface to apply the PPP connection. |
| **Step 8**       | end  
**Example:**  
Device(config-fr-dlci)# end  
- (Optional) Exits Frame Relay DLCI interface configuration mode. |

**Verifying the HQF Configuration**

**SUMMARY STEPS**

1. enable  
2. show policy-map interface [type access-control] interface-name [ve[y/p] vci] [dlci dlci] [input | output]  
3. exit
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>(Optional) Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Skip this step if you are using the <code>show</code> command in user EXEC mode.</td>
</tr>
<tr>
<td>Device&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Displays the packet statistics of all classes that are configured for all service policies either on the specified interface or subinterface or on a specific PVC on the interface.</td>
</tr>
<tr>
<td>show policy-map interface [type access-control] interface-name [vc[vpi] vci] [dlci dlci] [input</td>
<td>output]</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# show policy-map interface serial4/3</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>(Optional) Exits privileged EXEC mode.</td>
</tr>
<tr>
<td>exit</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device# exit</td>
<td></td>
</tr>
</tbody>
</table>

### Configuration Examples for QoS Hierarchical Queueing Framework

#### Example: Configuring QoS Hierarchical Queueing Framework

There are two main tasks for configuring this feature:

- Configuring a policy map
- Attaching the policy map to a map class

In the following example, a policy map called `shape` is configured on serial interface 4/3 and attached in the output direction. Its parameters include a class class-default, a traffic shaping average of 256000 bps, and Frame Relay encapsulation.

```
Device# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Device(config)# policy-map shape
Device(config-pmap)# class class-default
Device(config-pmap-c)# shape average 256000
Device(config-pmap-c)# exit
Device(config-pmap-c)# exit
```
Device(config)# interface serial4/3
Device(config-if)# encapsulation frame-relay
Device(config-if)# service-policy output shape
Device(config-if)# end

In the following example, the policy map called shape is attached to serial interface 4/3 in the output direction and is associated with a map class called shape. There is also a PVC being associated with DLCI 16.

Device# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Device(config)# map-class frame-relay shape
Device(config-map-class)# service-policy output shape
Device(config-map-class)# exit
Device(config)# interface serial4/3
Device(config-if)# frame-relay class shape
Device(config-if)# frame interface-dlci 16
Device(config-fr-dlci)# end

Example: Verifying the HQF Configuration

In the following example, shaping is active with HQF installed on serial interface 4/3. All traffic is classified to the class-default queue.

Device# show policy-map interface serial4/3
Serial4/3
  Service-policy output: shape
    Class-map: class-default (match-any)
      2203 packets, 404709 bytes
      30 second offered rate 74000 bps, drop rate 14000 bps
      Match: any
      Queueing
      queue limit 64 packets
        (queue depth/total drops/no-buffer drops) 64/354/0
        (pkts output/bytes output) 1836/337280
      shape (average) cir 128000, bc 1000, be 1000
      target shape rate 128000
      lower bound cir 0, adapt to fecn 0
      Service-policy: LLQ
      queue stats for all priority classes:
        queue limit 64 packets
        (queue depth/total drops/no-buffer drops) 0/0/0
        (pkts output/bytes output) 0/0
    Class-map: c1 (match-all)
      0 packets, 0 bytes
      30 second offered rate 0 bps, drop rate 0 bps
      Match: ip precedence 1
      Priority: 32 kbps, burst bytes 1500, b/w exceed drops: 0
    Class-map: class-default (match-any)
      2190 packets, 404540 bytes
      30 second offered rate 74000 bps, drop rate 14000 bps
      Match: any
      queue limit 64 packets
        (queue depth/total drops/no-buffer drops) 63/417/0
        (pkts output/bytes output) 2094/386300
Additional References for QoS Hierarchical Queueing Framework

Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame Relay commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td>Cisco IOS Wide-Area Networking Command Reference</td>
</tr>
<tr>
<td>QoS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td>Cisco IOS Quality of Service Solutions Command Reference</td>
</tr>
<tr>
<td>MQC</td>
<td>“Applying QoS Features Using the MQC” module</td>
</tr>
</tbody>
</table>

Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.</td>
<td><a href="http://www.cisco.com/cisco/web/support/index.html">http://www.cisco.com/cisco/web/support/index.html</a></td>
</tr>
</tbody>
</table>

Feature Information for QoS Hierarchical Queueing Framework

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Table 1: Feature Information for QoS Hierarchical Queueing Framework (HQF)

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoS Hierarchical Queueing Framework (HQF)</td>
<td>12.2(28)SB</td>
<td>The QoS Hierarchical Queueing Framework (HQF) feature enables you to manage quality of service (QoS) at three different levels: the physical interface level, the logical interface level, and the class level for QoS queueing and shaping mechanisms by using the modular QoS command-line interface (MQC) to provide a granular and flexible overall QoS architecture. In Release 12.2(28)SB, this feature was introduced as QoS: Frame Relay QoS Hierarchical Queueing Framework Support on the Cisco 7200 Series Router. The following commands were introduced or modified: bandwidth (policy-map class), fair-queue (WFQ), max-reserved-bandwidth, police (two rates), queue-limit, random-detect, random-detect atm-clp-based, random-detect cos-based, random-detect prec-based, random-detect precedence, shape-max buffers, show policy-map, show policy-map interface, show queue, show queueing.</td>
</tr>
<tr>
<td></td>
<td>12.4(20)T</td>
<td></td>
</tr>
</tbody>
</table>