QoS Exam Topics

This chapter covers the following exam topics specific to the QoS exam:

- Given a network requiring QoS, explain how to implement a QoS policy using MQC
- Explain how AutoQoS is used to implement QoS policy

CHAPTER 3

MQC, QPM, and AutoQoS

Most of the topics covered in Chapters 1, "QoS Overview," and 2, "QoS Tools and Architectures," can apply to a network that uses equipment from most any manufacturer. Sure, there were some specifics about Cisco IOS QoS tools and about Cisco IP Phones, but all the concepts about QoS architectures and traffic characteristics of voice, video, and data apply to any network, regardless of manufacturer.

This chapter is specific about discussing several tools available only for Cisco products. Most of the more modern QoS tools from Cisco use configuration commands that conform to a convention called the Modular QoS CLI (MQC), which significantly reduces the complexity of QoS configuration as compared to QoS tools that don't use MQC commands. Frankly, before MQC, QoS configuration was one of the more challenging things to configure in Cisco IOS Software. With MQC, most of the complexity has been removed.

Although easier configuration of a router or switch using MQC is indeed wonderful, MQC enables a couple of other important Cisco QoS tools. Cisco offers a management application called QoS Policy Manager (QPM), which provides a web browser interface to network engineers, allowing them to easily define QoS policies for a network, all with intuitive pointing and clicking. QPM can baseline the network's QoS performance, configure the routers and switches based on the policy, measure the ensuing performance, and monitor the configurations to make sure no one changes the QoS configuration.

Compared to the old days, MQC makes it easier to configure each device, and QPM makes it easier to configure and monitor QoS for an entire network.

In addition, Cisco offers a tool called AutoQoS in Cisco IOS Release 12.3 mainline router and Cisco IOS Release 12.1EA 2950 switch. (Go to http://www.cisco.com/go/fn to use the Cisco Feature Navigator to find more specific information about AutoQoS support on different platforms.) AutoQoS allows a network engineer to configure a single device with just a few generic commands, and the device automatically configures all the appropriate QoS tools. So, even without QPM, a network engineer can configure QoS with confidence. Also, the automatically generated configuration can be changed, if the default settings are not quite what the engineer wants.

MQC, QPM, and AutoQoS provide some fantastic advantages. In this chapter, you'll read about all three. After that, Chapters 4 through 9 take a closer look at six categories of QoS tools available in Cisco routers.

"Do I Know This Already?" Quiz Questions

The purpose of the "Do I Know This Already?" quiz is to help you decide whether you need to read the entire chapter. If you already intend to read the entire chapter, you do not necessarily need to answer these questions now.

The 12-question quiz, derived from the major sections in the "Foundation Topics" portion of the chapter, helps you determine how to spend your limited study time.

Table 3-1 outlines the major topics discussed in this chapter and the "Do I Know This Already?" quiz questions that correspond to those topics.

Table 3-1 *"Do I Know This Already?" Foundation Topics Section-to-Question Mapping*

Foundation Topics Section Covering These Questions	Questions	Score
Cisco Modular QoS CLI	1–5	
Cisco QoS Policy Manager	6	
Cisco AutoQoS Feature	7–11	
Comparisons of CLI, MQC, and AutoQoS	12	
Total Score		

CAUTION The goal of self-assessment is to gauge your mastery of the topics in this chapter. If you do not know the answer to a question or are only partially sure of the answer, mark this question wrong for purposes of the self-assessment. Giving yourself credit for an answer you correctly guess skews your self-assessment results and might provide you with a false sense of security.

You can find the answers to the "Do I Know This Already?" quiz in Appendix A, "Answers to the 'Do I Know This Already?' Quizzes and Q&A Sections." The suggested choices for your next step are as follows:

- **10 or less overall score**—Read the entire chapter. This includes the "Foundation Topics," the "Foundation Summary," and the "Q&A" sections.
- **11 or 12 overall score**—If you want more review on these topics, skip to the "Foundation Summary" section and then go to the "Q&A" section. Otherwise, move to the next chapter.

Cisco Modular QoS CLI

1. What does MQC stand for?

   a. Multiprotocol QoS Commands

   b. Multiprotocol QoS CLI

   c. Modular QoS Commands

   d. Modular QoS CLI

   e. Modular QoS Convention

2. Which of the following MQC commands is most related to the process of classifying packets into service classes?

   a. **service-policy**

   b. **route-map**

   c. **map-policy**

   d. **policy-map**

   e. **map-class**

   f. **class-map**

3. Which of the following is *not* a benefit of MQC?

   a. Reduces configuration time

   b. Provides Modular QoS Call-Admission-Control

   c. Same set of commands on all Cisco IOS platforms

   d. Separates classification from per-hop behavior (PHB) actions

4. Which of the following is not true about the mechanics of MQC?

   a. Packets are classified inside a class map.

   b. PHBs are defined inside a service policy.

   c. Matching multiple DSCPs requires multiple **match** commands.

   d. One command is used to enable a QoS policy on an interface for packets both entering and exiting an interface.

5. Examine the configuration snippet that follows. Which of the following statements is true about the configuration?

```
Router(config)#class-map fred
Router(config-cmap)#match dscp EF
Router(config-cmap)#match access-group 101
```

   a. Packets with DSCP EF and that also match ACL 101 will match the class.

   b. Packets that either have DSCP EF or that match ACL 101 will match the class.

c. Packets that match ACL 101 will match the class because the second **match** command replaces the first.

d. Packets will match only DSCP EF because the first match exits the class map.

The Cisco QoS Policy Manager

6. Which of the following is false about QPM?

   a. Provides a standard set of commands for configuring QoS

   b. Allows a user to specify QoS policies without knowing the MQC command syntax

   c. Graphs QoS performance

   d. Takes advantage of Cisco QoS MIBs

The Cisco AutoQoS Feature

7. Which option on a 2950 switch **auto qos voip** command tells the switch to trust the CoS only if a Cisco IP Phone is attached to the port?

   a. trust cos

   b. ciscophone

   c. cisco-phone

   d. ciscoipphone

8. Which option on a 6500 switch **set port qos** command tells the switch to trust the CoS only if a Cisco IP Phone is attached to the port?

   a. trust cos

   b. ciscophone

   c. cisco-phone

   d. ciscoipphone

9. Which of the following PHBs cannot be enabled using the AutoQoS VoIP feature on a router?

   a. Low Latency Queuing

   b. CB Marking

   c. Shaping

   d. MLP LFI

   e. Policing

10. Which router commands display the configuration that results from enabling AutoQoS VoIP on a router's S0/0 interface, including the details of any class maps or policy maps?

    a. **show autoqos**

    b. **show auto qos**

    c. **show auto qos interface s0/0**

    d. **show running-config**

11. Which of the following statements are true about requirements before AutoQoS can be enabled on a router interface?

    a. CEF must be enabled unless trust is to be configured.

    b. No **service-policy** commands can be on the interfaces.

    c. WFQ must be disabled using the **no fair-queue** command.

    d. HDLC encapsulation must be changed to PPP.

    e. For proper operation, bandwidth should be set to the correct value.

Comparisons of CLI, MQC, and AutoQoS

12. Comparing CLI, MQC, and AutoQoS, which is considered to require the least amount of time to implement?

    a. CLI

    b. MQC

    c. AutoQoS

    d. All take equal time

Foundation Topics

All the tools covered in this chapter provide some important advantages for companies that choose to use Cisco products. Together, these tools improve accuracy of QoS configurations, make the configurations easier to understand to everyone working on the network, and help make converged multiservice traffic run more smoothly through the network. Ultimately, these tools take advantage of the native QoS tools described in Chapters 4 through 9.

This chapter starts with MQC, then covers QPM, and ends with AutoQoS.

Cisco Modular QoS CLI

Back in the mid 1980s, Cisco Systems got its start by building and selling routers. As time went on, Cisco kept adding more and more features to its router software, called Cisco IOS Software, including some QoS features.

Each feature could be configured using the Cisco command-line interface (CLI), but in most cases, each QoS tool used a totally different set of commands than the other tools. At the same time, the networks in which the routers and switches were installed started to have more stringent QoS requirements, causing Cisco customers to need to use multiple QoS tools. As a result, the task of figuring out what to do with the various QoS tools, how to configure them, and how to monitor the success of those tools in the network was a bit daunting.

The Cisco Modular QoS CLI (MQC) helped resolve these problems by defining a common set of configuration commands to configure most QoS features in a router or switch. After hearing the term MQC for the first time, many people think that Cisco has created a totally new CLI, different from IOS configuration mode, to configure QoS. In reality, MQC defines a new set of configuration commands—commands that are typed in using the same IOS CLI, in configuration mode.

As time goes on, and as Cisco creates new IOS releases for both routers and switches, all QoS tools will use MQC. In fact, almost all the tools covered by the current QoS exam use MQC.

You can identify an MQC-based tool because the name of the tool starts with the phrase "class-based" (commonly noted as "CB"). These tools include CB Marking, CB Weighted Fair Queuing (CBWFQ), CB Policing, CB Shaping, and CB Header Compression. Most QoS tools need to

perform classification functions; all MQC supporting tools use the same commands for classification. The person configuring the router needs to learn only one set of commands for classification for all these MQC-based tools, which reduces effort and reduces mistakes.

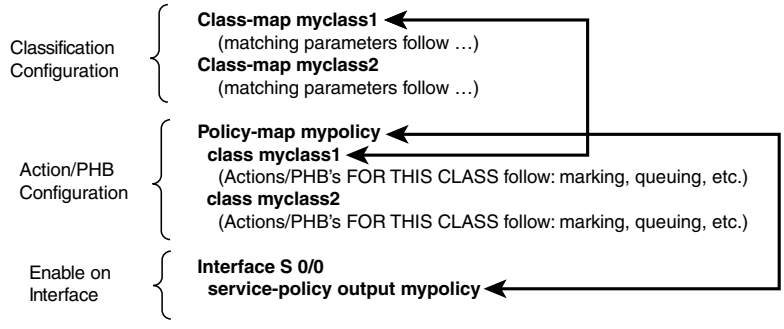
The Mechanics of MQC

MQC separates the classification function of a QoS tool from the action (the per-hop behavior, or PHB) that the QoS tool wants to perform. To do so, there are three major commands with MQC, with several subordinate commands:

- The **class-map** command defines the matching parameters for classifying packets into service classes.
- Because different tools create different PHBs, the PHB actions (marking, queuing, and so on) are configured under a **policy-map** command.
- Because MQC operates on packets that either enter or exit an interface, the policy map needs to be enabled on an interface by using a **service-policy** command.

Figure 3-1 shows the general flow of commands.

Figure 3-1 *MQC Commands and Their Correlation*



In Figure 3-1, the network's QoS policy calls for two service classes. (The actual types of packets that are placed into each class are not shown, just to keep the focus on the general flow of how the main commands work together.) Classifying packets into two classes calls for the use of two **class-map** commands. Each **class-map** command would be followed by a **match** subcommand, which defines the actual parameters that are compared to packet header contents to match packets for classification.

For each class, some QoS action (PHB) needs to be applied—the configuration for these actions is made under the **policy-map** command. Under a single policy map, multiple classes are referenced, in this case, the two classes myclass1 and myclass2. Inside the single policy called mypolicy, under each of the two classes myclass1 and myclass2, you can configure separate QoS actions. For instance, you could apply different marking to packets in class myclass1 and myclass2 at this point. Finally, when the **service-policy** command is applied to an interface, the QoS features are enabled.

Classification Using Class Maps

Almost every QoS tool uses classification to some degree. To put one packet into a different queue than another packet, IOS must somehow differentiate between the two packets. To perform header compression on Real-Time Transport Protocol (RTP) packets, but not on other packets, IOS must determine which packets have RTP headers. To shape data traffic going into a Frame Relay network so that the voice traffic gets enough bandwidth, IOS must differentiate between Voice over IP (VoIP) and data packets. If an IOS QoS feature needs to treat two packets differently, it must use classification.

MQC-based tools classify packets using the **match** subcommand inside an MQC class map. Several examples in this section point out some of the key features of class maps. Table 3-2 lists the **match** command options available for Cisco IOS Software Release 12.2(15)T, which is covered by the current QoS exam.

Table 3-2 **match** *Configuration Command Reference for MQC Tools*

Command	Function
match [**ip**] **precedence** *precedence-value* [*precedence-value precedence-value precedence-value*]	Matches precedence in IPv4 packets when the **ip** parameter is included; matches IPv4 and IPv6 packets when the **ip** parameter is missing.
match access-group {*access-group* | **name** *access-group-name*}	Matches an ACL by number or name.
match any	Matches all packets.
match class-map *class-map-name*	Matches based on another class map.
match cos *cos-value* [*cos-value cos-value cos-value*]	Matches a CoS value.

Table 3-2 **match** *Configuration Command Reference for MQC Tools (Continued)*

Command	Function
match destination-address mac *address*	Matches a destination MAC address.
match fr-dlci *dlci-number*	Matches a particular Frame Relay DLCI.
match input-interface *interface-name*	Matches an input interface.
match ip dscp *ip-dscp-value* [*ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value ip-dscp-value*]	Matches DSCP in IPv4 packets when the **ip** parameter is included; matches IPv4 and IPv6 packets when the **ip** parameter is missing.
match ip rtp *starting-port-number port-range*	Matches the RTP's UDP port-number range, even values only.
match mpls experimental *number*	Matches an MPLS Experimental value.
match mpls experimental topmost *value*	Matches the MPLS EXP field in the topmost label when multiple labels are in use.
match not *match-criteria*	Reverses the matching logic; in other words, things matched by the matching criteria do not match the class map.
match packet length {**max** *maximum-length-value* [**min** *minimum-length-value*] | **min** *minimum-length-value* [**max** *maximum-length-value*]}	Matches packets based on the minimum length, maximum length, or both.
match protocol citrix app *application-name-string*	Matches Network Based Application Recognition (NBAR) Citrix applications.
match protocol http [**url** *url-string* | **host** *hostname-string* | **mime** *MIME-type*]	Matches a host name and URL string.
match protocol *protocol-name*	Matches NBAR protocol types.
match protocol rtp [**audio** | **video** | **payload-type** *payload-string*]	Matches RTP audio or video payload, based on the payload type. Also allows explicitly specified payload types.
match qos-group *qos-group-value*	Matches a QoS group.
match source-address mac *address-destination*	Matches a source MAC address.

MQC Example 1: Voice and Everything Else

The first example (Example 3-1) shows the basic flow of the commands. Two class maps are used—one that matches voice packets, and one that matches everything else. Note that class map names are case sensitive, as are policy maps.

Example 3-1 *Basic Classification with Two Class Maps*

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#!
R3(config)#class-map voip-rtp
R3(config-cmap)#match ip rtp 16384 16383
R3(config-cmap)#class-map all-else
R3(config-cmap)#match any
R3(config-cmap)#policy-map voip-and-be
R3(config-pmap)#class voip-rtp
R3(config-pmap-c)#! Several options in here; CB Marking shown with the set command
R3(config-pmap-c)#set ip DSCP EF
R3(config-pmap-c)#class all-else
R3(config-pmap-c)#set ip dscp default
R3(config-pmap-c)#interface fa 0/0
R3(config-if)#service-policy input voip-and-be
R3(config-if)#end
R3#
R3#show running-config
Building configuration...
!Portions removed to save space…
ip cef
!
class-map match-all voip-rtp
  match ip rtp 16384 16383
class-map match-all all-else
  match any
!
!
policy-map voip-and-be
  class voip-rtp
   set dscp EF
  class all-else
   set dscp default
!
interface Fastethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 service-policy input voip-and-be
```

First, focus on the command prompts in Example 3-1. Note that the **class-map** command moves the CLI into class map configuration mode, with the prompt **R3(config-cmap)**. The **policy-map** command moves the CLI into policy map configuration mode, and the **class** command that follows (not **class-map**, but just **class**) moves the CLI into an additional subconfiguration mode that has no specific name.

NOTE Class map names are case-sensitive.

Next, examine the **match** commands. The **match ip rtp** command matches only the even-numbered ports in this same UDP port range and does not match the odd-numbered ports used by the Real-Time Control Protocol (RTCP) voice-signaling protocol. (VoIP payload uses only the even port numbers.) Therefore, the **match ip rtp** command matches all VoIP payload. The other **match** command in **class-map all-else**, **match any** does exactly that—it matches anything. So, one class map matches VoIP payload, and the other matches any traffic.

A little later in this chapter, you will read more about the actions (PHBs) that can be taken inside a policy map. For this example, CB Marking, using the **set** command, is shown. Continuing down the configuration, examine the **policy-map set** commands. The first command sets a Differentiated Services Code Point (DSCP) of EF (expedited forwarding) for all traffic that matches **class-map voip-rtp**. The other **set** command, which follows the **class all-else** command, sets a DSCP of Default for traffic that matches the **class-default class-map**. In other words, the policy map sets DSCP EF for packets that match one class, and DSCP Default, using the keyword **default**, for the other class.

Finally, the **service-policy** command enables CB Marking for ingress packets with the **service-policy input voip-and-be** interface subcommand. When enabled, IOS applies the policy map classes in the order they appear in the **policy-map** command. In this example, for instance, the **voip-rtp** class is used to examine the packet first; if a match appears, the packet is marked with DSCP EF. After the packet has been matched and marked, it exits the policy map. If no match occurs, only then is the next class, **all-else**, used to examine the packet.

MQC Example 2: Matching ACLs and Using class-default

Example 3-1 could have been done more efficiently using the **class-default** class, which is a class inside every policy map, at the end of the policy map. If a packet is examined by a policy map and

it does not match any of the explicitly defined classes, the packet is considered to match **class-default**. Example 3-2 shows another configuration, this time with **class-default** in use.

Example 3-2 *Class Maps Matching Voice, ACL 101, and Using* **class-default**

```
R3#show running-config
Building configuration...
!Portions removed to save space…
ip cef
!
class-map match-all voip-rtp
  match ip rtp 16384 16383
class-map match-all class2
  match access-group 101
!
!
policy-map voip-101-be
  class voip-rtp
   set dscp EF
  class class2
   set dscp AF11
  class class-default
   description this class matches everything else by default
   set dscp BE
!
interface Fastethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 service-policy input voip-101-be
!
access-list 101 permit tcp any any eq 23
access-list 101 permit tcp any eq 23 any
```

This example uses the same **class-map voip-rtp** command, which matches voice payload packets. A new **class-map class2** command defines a new class, matching packets that are permitted by ACL 101. In this case, ACL 101 matches Telnet packets.

policy-map voice-101-be refers to the two explicitly defined class maps, as well as the default class map called **class-default**. The router processes the policy map logic in the order shown in the configuration, always placing class **class-default** at the end of the policy map. With an implied **match any** included in the **class-default** class, all packets that have not already matched classes **voip-rtp** or **class2** will end up matching **class-default**.

Example 3: Matching Opposites with match not

MQC includes a feature to enable you to match packets that do not meet the specified criteria. For instance, Example 3-2 included a class (**class2**) that matched packets permitted by ACL 101. If you wanted instead to match all packets that did not match ACL 101 with a **permit** action, you could use the **match not** command. Example 3-3 duplicates Example 3-2, but with the **match not** command in use.

Example 3-3 *Class Maps Matching Voice, ACL 101, and Using* **class-default**

```
R3#show running-config
Building configuration...
!Portions removed to save space…
ip cef
!
class-map match-all voip-rtp
  match ip rtp 16384 16383
class-map match-all not-class2
  description all packets denied by ACL 101 match
  match not access-group 101
!
!
policy-map voip-101-be
  class voip-rtp
   set dscp EF
  class not-class2
   set dscp BE
  class class-default
   description this class matches everything else by default
   set dscp AF11
!
interface Fastethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 service-policy input voip-101-be
!
access-list 101 permit tcp any any eq 23
access-list 101 permit tcp any eq 23 any
```

Both Examples 3-2 and 3-3 end up doing the same thing in this case. With some MQC-based QoS tools, the **class-default** class has some special characteristics, so you might prefer to explicitly match the Telnet (ACL 101) traffic as in Example 3-2, or you might prefer to explicitly match all other traffic using **match not**, as in Example 3-3.

Example 4: Matching More Than One Thing

You might have noticed that the **class-map** commands in the **show running-config** output all have a parameter called **match-all**. The syntax of the **class-map** command is actually **class-map** [**match-all** | **match-any**] *class-name*, with **match-all** being the default setting if you do not choose either option.

The **match-all** and **match-any** commands tell the router or switch how to process the class map if multiple **match** commands are used. So far in this chapter, only one **match** command has been used in each class map. However, IOS allows you to refer to multiple fields inside each packet to match it, and to do so, you simply use multiple **match** commands.

The **match-all** and **match-any** parameters tell IOS whether to match packets that match all the **match** commands (**match-all**) or to match packets that match one or more of the **match** commands (**match-any**). If you prefer Boolean logic, **match-all** means that there is a logical AND between each **match** command with **match-all**, and a logical OR between each **match** command with **match-any**. Example 3-4 shows an example of class maps with **match-all** and **match-any**. Note that because the focus of this example is on how the matching logic works, the example does not bother showing a **policy-map** or a **service-policy** command.

Example 3-4 *Class Maps with* **match-all** *and* **match-any**

```
class-map match-all ex4-1
  match ip rtp 16384 16383
  match precedence 5
!
class-map match-any ex4-2
  match access-group 102
  match dscp AF21
!
class-map match-all ex4-3
  match dscp 0
  match dscp 1
!
class-map match-any ex4-4
  match dscp 0
  match dscp 1
!
class-map match-any ex4-5
  match dscp 0 1
!
```

First, examine **class-map match-all ex4-1**. The packet must be an RTP packet within the stated UDP port number range, plus it must also already have been marked as precedence 5. If either match condition is not true about a particular packet, the packet is not considered to match the class map. So, in this case, voice packets that had not been marked yet would not be part of this class.

Next, examine **class-map match-any ex4-2**. With match-any logic, packets match this class if they are permitted by ACL 102 (not shown), or if they are marked with DSCP AF21. If your QoS policy defined that packets matching ACL 102 should be treated as if they were marked with AF21, this class map makes sure that any packets that were not correctly marked already are still treated as AF21 traffic.

You can use multiple **match** commands to match multiple criteria; however, there might be cases in which you want to match multiple marked values, for instance, multiple DSCP values in one class map. The **class-map ex4-3**, **class-map ex4-4**, and **class-map ex4-5** commands show some of the dangers and possibilities with matching multiple DSCP values; **class-map match-all ex4-3** uses match-all logic and lists **match** commands for two DSCP values (0 and 1). No one single packet can be marked with both DSCP 0 and 1, so no packets could possibly match **class-map match-all ex4-3**. To correctly match packets that have either DSCP 0 or 1, **class-map match-any ex4-4** could be used, because it matches if only one of the **match** commands conditions are met.

You can, however, use a more convenient method for matching packets based on multiple DSCP, precedence, or CoS values. The **class-map ex4-5** command shows how a single **match** command can be used to match multiple DSCPs. Note the syntax of the **match dscp**, **match precedence**, and **match cos** commands in Table 3-2. With precedence and CoS, you can supply four different values in a single **match** command; with DSCP, you can supply eight different values. When you supply more than one DSCP, precedence, or CoS value in one **match** command, IOS uses logic such as **match-any**, meaning that a packet with any of the stated values matches the condition.

NOTE The earliest Cisco IOS releases that supported MQC commands used a syntax such as **match ip dscp af11** to match a DSCP value. Later releases support that syntax, as well as syntax that does not include the **ip** keyword—for instance, **match dscp af11** is also valid. Similarly, the **match ip precedence 1** command was originally specified, and now the **match precedence 1** command can also be used.

Example 5: Complex Matching with match-class

In most networks, you already have seen enough examples so that you can configure **class-map** commands effectively. This final example in this section points out one less-obvious way to use class maps by referring to other class maps using the **match class** command.

Imagine that you want to match based on several things in the headers of a packet; however, the logic you want to use runs something like this:

> If condition A and B are true, or if condition C is true, then place packet in this class.

The **match-all** and **match-any** keywords do not allow you to just code three **match** commands and achieve this logic. With the **match class** *class-map-name* command, you can achieve this logic, as shown in Example 3-5.

Example 3-5 *Complex Matching with* **match class** *Command*

```
class-map match-all ex5-1
  match ip rtp 16384 16383
  match precedence 5
!
class-map match-any ex5-2
  match class ex5-1
  match cos 5
```

In **class-map match-any ex5-1**, the class map looks for VoIP RTP traffic that has been marked with precedence 5. Both the RTP designation and precedence 5 must be true to match the conditions in this class map. The **class-map match-any ex5-2** command uses match-any logic, so either **match** command's conditions can be met to classify a packet into **class-map ex5-2**. Interestingly, **class-map ex5-2** uses the **match class ex5-1** command, which of course refers to the first class map, which uses match-all logic. In effect, the logic is the following:

> Match packets with RTP protocol, UDP ports between 16384 and 32767, AND marked with precedence 5
>
> or
>
> Packets with CoS 5

You will see many more examples of how to use the MQC **class-map** and **match** commands in later chapters. Next, you will read a little more about the **policy-map** and **service-policy** commands.

Performing QoS Actions (PHBs) Using policy-map Commands

MQC uses a three-step approach:

1. **class-map** commands classify packets into service classes.
2. **policy-map** commands define PHB actions.
3. **service-policy** interface subcommands enable the logic of a policy map on an interface.

As shown in Example 3-1, the **policy-map** command refers to class maps using the **class** command. Example 3-6 repeats an excerpt from Example 3-1, with specific commands highlighted.

Example 3-6 *Basic Flow of* **policy-map** *Commands*

```
class-map match-all voip-rtp
  match ip rtp 16384 16383
class-map match-all all-else
  match any
!
!
policy-map voip-and-be
  class voip-rtp
!  (any action can be configured here; CB Marking is shown)
   set dscp EF
  class all-else
!  (any action can be configured here; CB Marking is shown)
   set dscp default
!
interface Fastethernet0/0
 description connected to SW2, where Server1 is connected
 ip address 192.168.3.253 255.255.255.0
 service-policy input voip-and-be
```

Policy maps rely on the classification logic in **class-map** commands to treat packets differently. In Example 3-6, the **policy-map voip-and-be** command includes two **class** subcommands—**class voip-rtp** and **class all-else**. Underneath the **class** commands, you can configure many different commands that define a PHB or action to be taken against packets in that class. (Upcoming Table 3-3 lists the action or PHB subcommands available underneath a **class** command.) In this example, the **set** command is used, which means that marking is the action taken on packets in each class.

The **service-policy input voip-and-be** interface subcommand enables the policy map on the Fastethernet0/0 interface for incoming packets.

NOTE Policy map names are case-sensitive. Because class map names are also case-sensitive, be careful when configuring **class** commands that refer to class map names.

Table 3-3 lists the various MQC subcommands available in Cisco IOS Software for defining actions to be taken. Chapters 4 through 9 describe each of these options in more detail, including the meaning of the parameters of the commands.

Table 3-3 *Action (PHB) Subcommands Inside a Policy Map*

Command	Function
set	CB Marking action, with options to mark several fields inside headers
bandwidth	Reserves bandwidth for the class for CBWFQ
priority	Reserves bandwidth and provides Low Latency Queuing (LLQ) with CBWFQ
shape	Shapes traffic in the class with a defined bandwidth and burst sizes
police	Polices traffic in the class with a defined bandwidth and burst sizes
compress	Performs TCP and RTP header compression on packets in the class

Enabling a Policy Map Using service-policy

You have already seen the **service-policy** command in use in several examples. This short section points out a few of the important features of the **service-policy** command.

The full syntax of the command is **service-policy** {**input** | **output**} *policy-map-name*. Note that the curly brackets mean that you must choose either input or output—you cannot leave that parameter off the command. In effect, **service-policy** tells the router or switch to perform the logic in the policy map for packets either entering (input) or exiting (output) the interface.

Some actions might not be supported in both the input and output directions. For instance, in a router, CBWFQ can be performed only on packets exiting the interface. So, you can configure the policy map with **bandwidth** subcommands, with no problems. When you try to enable the policy map with the **service-policy input** command, the router will give you an error message and not add the **service-policy** command to the configuration.

Finally, some features require that Cisco Express Forwarding (CEF) switching be enabled before the action can work. If CEF has not been enabled with the **ip cef** global command, you can still configure a policy map for an action. However, when you try to enable it with the **service-policy** command, the router will tell you that CEF switching is required.

Each interface can have at most two **service-policy** commands—one for input packets and one for output.

show Commands for MQC

MQC configuration commands provide the wonderful advantage of a set of standard commands for configuring QoS. Cisco also standardized **show** commands for MQC-based tools as well, with three commands providing all the information for any MQC configuration.

The **show class-map** *class-map-name* command lists configuration information about the class map listed in the command, and the **show class-map** command lists information about all class maps. Similarly, the **show policy-map** *policy-map-name* command lists configuration information about a specific policy map, and the **show policy-map** command lists information about all policy maps. The same general information can be seen with a **show running-config** command.

The one command that shows counters and performance information is the **show policy-map interface** *interface-name* [**input** | **output**] command. The output differs based on the PHBs that have been configured; for instance, it shows queuing statistics when CBWFQ has been configured, shows marking statistics when CB Marking has been configured, and so on. In some of the examples in this chapter, a policy map was enabled on interface Fastethernet 0/0. So, the **show policy-map interface fastethernet0/0** command shows statistics for any policy maps enabled with the **service-policy** command in either direction. Alternatively, the **show policy-map interface fastethernet0/0 input** command lists statistics only for any policy maps enabled for input packets on interface Fastethernet0/0.

MQC helps Cisco router and switch engineers easily configure and monitor QoS features and with better results. Next, you will read about QoS Policy Manager, which is a network management tool that also helps engineers configure and monitor QoS tools.

QoS Policy Manager (QPM)

QPM provides many of the features that you need when you get serious about deploying QoS across an enterprise. The following list summarizes some of the more important features:

- Enables you to define a QoS policy based on business rules.
- Automatically configures some or all network devices with QoS features, based on the QoS policy described to QPM. The features that QPM enables include marking, queuing, shaping, policing, and Link Fragmentation and Interleaving (LFI) tools.
- Loads the correct configurations automatically.
- Enables you to monitor the device configurations to make sure no one has made changes to them. If the configurations have been changed, you can use QPM to restore the original configuration.

To get a sense of how QPM eases QoS configuration, imagine that you want to create a policy to mark all VoIP traffic with DSCP EF as near to the edge of the network as possible. You simply point and click to tell QPM what fields to look for in the packet or frame header. QPM creates the CB Marking configuration and loads it into all the appropriate devices. So, to use QPM, you still need to know what the base QoS tools can do, but you do not have to know the configuration syntax of

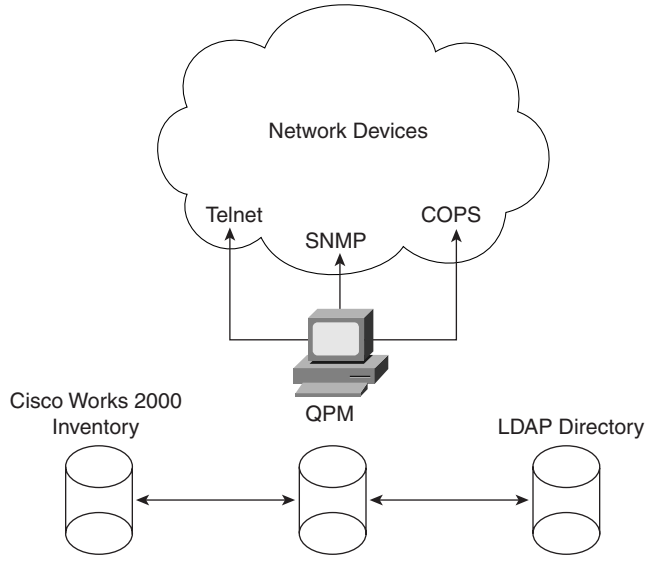
all the different QoS tools, and you do not have to repeat the configuration task on all the devices—QPM takes care of that for you.

QPM runs on Microsoft Windows 2000 Professional or Server with Service Packs 3 or 4, or with Microsoft Windows Advanced Server (without terminal services). (See http://www.cisco.com/en/US/partner/products/sw/cscowork/ps2064/products_user_guide_chapter09186a0080080808.html#4048 for the hardware and software requirements of the latest release of QPM, version 3.2, as of press time.)

To configure a common QoS policy and push this policy to the network devices, QPM needs to be able to learn which devices are present in the network and communicate with those devices. QPM can use the CiscoWorks database to discover the location of the devices in the network.

Figure 3-2 outlines the overall location and functions between the QPM server and the rest of the network.

Figure 3-2 *QPM Server and Communication with Other Devices*



For QPM to create configurations, load the configurations, and monitor the configurations for changes, QPM must know which devices it should manage. The most convenient way to define the devices for QPM to manage is to use the device list from the CiscoWorks2000 database. Cisco requires that QPM be installed on a machine that also has CiscoWorks Common Services 2.2, with Service Pack 2, which allows QPM to automatically discover the network devices; however, you can statically define devices to QPM as well.

QPM is an important tool for networks that deploy QoS extensively. The following list outlines some of the more popular features:

- Supports a wide variety of routers and switches
- Allows network-wide QoS policy definition, followed by automatic deployment of appropriate configurations
- Creates graphs of real-time performance
- Creates graphs of historical performance
- Allows end-user viewing of reports and configuration using a web browser
- Manages only a single device from the browser
- Manages the entire network from one browser window
- Implements the actual probes and responses when necessary for measuring network performance

SNMP Support for QoS

QPM uses Telnet and the Simple Network Management Protocol (SNMP) to configure and monitor devices. Cisco IOS includes a couple of important proprietary SNMP Management Information Bases (MIBs) that provide a lot of good information about QoS performance in a router. When QPM displays performance data for MQC-based QoS tools in a router, the performance data comes mostly from these specialized QoS MIBs.

First, the Class-Based QoS MIB (CBQoSMIB) contains variables that describe the MQC configuration commands in a router. This MIB also includes statistical variables, which are essentially the same kinds of stats seen with the **show policy-map** interface command.

More interestingly, CBQoSMIB goes beyond those statistics, providing statistics for packets before a policy map has been processed, and afterward. In other words, you can see statistics about packets before the PHBs have been applied and after they are applied. QPM, of course, knows about this MIB, so it is ready to show graphs of packet statistics comparing the pre- and post-policy map.

Network Based Application Recognition (NBAR) can be used by MQC **class-map** commands to help classify traffic. NBAR can also be used to simply recognize and count traffic based on different protocol types. Of particular interest, NBAR can recognize hard-to-recognize protocols that do not use well-known ports, or that use dynamically allocated port numbers, by looking past the TCP and UDP headers into the application layer protocol. So, NBAR provides some interesting statistics about what protocols might be running through a network. (Chapter 4, "Classification and Marking," covers NBAR in much more detail.)

You can look at NBAR statistics for different protocols with the **show nbar protocol-discovery** command from a router; however, IOS includes the Cisco NBAR Protocol Discovery (CNPD) MIB, making the same statistics available from CiscoWorks and QPM. In particular, when planning a QoS implementation, NBAR can be a useful tool for figuring out what protocols are actually being sent through the network. The CNPD MIB can be used to easily graph and track the counters to decide how much bandwidth is needed for each particular type of application.

An engineer has a great set of tools for configuring and managing QoS with MQC at the command line, and QPM and these specialized MIBs from the management station. However, Cisco has one other significant tool that aids in QoS implementations, called AutoQoS, which will be covered in the last section of this chapter.

Cisco AutoQoS Feature

AutoQoS is a feature on some Cisco routers and switches that enables you to configure QoS on a device with only a few **auto qos** commands. By following some restrictions, and using the **auto qos** command, you can cause the router or switch to have a valid, working QoS configuration. Best of all, you do not have to know anything about how the QoS features work to use AutoQoS!

Because routers and switches do different tasks, QoS differs slightly between them. To get an idea about what AutoQoS does, consider that AutoQoS on routers works to classify packets into three service classes:

- Voice payload
- Voice signaling
- All other traffic

If you look in the router IOS documentation, AutoQoS is actually called AutoQoS VoIP, in reference to the design goal behind AutoQoS that is to provide good performance for voice traffic. That is true on both routers and switches.

NOTE This section covers AutoQoS VoIP, which is supported in IOS 12.3 mainline. Note that Cisco has also announced AutoQoS Enterprise, which is part of IOS 12.3T.

AutoQoS provides some great advantages. AutoQoS automatically classifies traffic, generating the MQC QoS commands, as well as QoS commands for a couple of other QoS features. The configurations are indeed consistent if AutoQoS is used throughout the network. You can modify the automatically generated configurations as well, so you can have AutoQoS do the hard work, and then come behind it and tweak the configuration. AutoQoS also re-marks all non-voice traffic that has DSCP EF, AF31, or CS3 to DSCP 0, which helps prevent data applications from attempting to get the same treatment as voice. And AutoQoS conforms to the Cisco current *Best Practices for Voice QoS*, so you can be confident the voice traffic will be treated well.

AutoQoS is supported on routers, on IOS-based switches, and in Cat-OS on 6500 switches. The next three sections take a look at AutoQoS in each of these three contexts.

AutoQoS VoIP for Routers

The beauty of AutoQoS is that you need to configure only a simple command or two, and AutoQoS does the rest. This section starts with some explanation of how to configure AutoQoS on a router, followed by an explanation of what AutoQoS automatically configured on the router.

AutoQoS VoIP Default Configuration

Before configuring AutoQoS VoIP, you should refer to the *IOS 12.3 QoS Configuration Guide,* which lists several considerations and conditions for the right environment for enabling this feature. For QOS exam purposes, repeating the full list here is not helpful; however, considering a couple of the most common considerations can help. For instance

- AutoQoS VoIP requires that CEF be enabled first.
- AutoQoS VoIP cannot be used if the interface already has a **service-policy** command configured.
- Because AutoQoS VoIP relies on the bandwidth settings configured in the **bandwidth** command, the routers should be configured with correct bandwidth settings on each interface before enabling AutoQoS VoIP. (If you change the bandwidth after enabling AutoQoS VoIP, AutoQoS VoIP does not react and does not change the QoS configuration.)
- Supports only point-to-point subinterfaces on Frame Relay interfaces.
- Supports HDLC, PPP, Frame Relay, and ATM data link protocols.

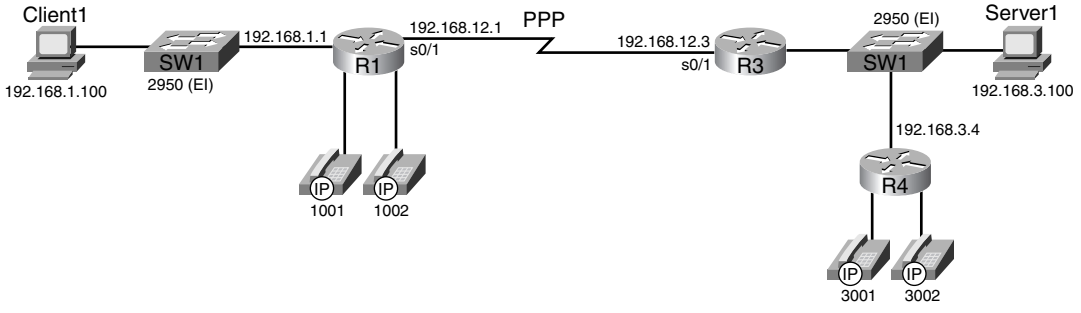
None of these considerations poses a big problem in most networks. Having met those requirements, configuring AutoQoS VoIP is quite easy. For reference, Table 3-4 lists the commands related to AutoQoS VoIP, followed by an example configuration.

Table 3-4 *Command Reference for AutoQoS VoIP (for Routers)*

Command	Function
auto qos voip [**trust**] [**fr-atm**]	Configuration command that enables AutoQoS VoIP on an interface (PPP or HDLC) or VC (FR and ATM)
no auto qos	Disables AutoQoS VoIP on an interface (PPP or HDLC) or VC (FR and ATM)
show auto qos [**interface** *interface-type*]	Displays what AutoQoS actually created
show policy-map interface *interface-name* [**input** | **output**]	Displays actual configuration of MQC-based parts of the configuration, including any later changes made by an engineer

To begin, look at Figure 3-3 and Example 3-7. Figure 3-3 shows R1 and R3 connected over a serial link, with 2950 switches attached to each router. As noted earlier, AutoQoS VoIP on routers supports serial link protocols (such as PPP), but it does not support LAN interfaces. This example begins with the pertinent configuration before enabling AutoQoS VoIP and then shows the configuration command.

Figure 3-3 *AutoQoS VoIP: Example Network with PPP*



Example 3-7 *AutoQoS Configuration with PPP*

```
R3#show running-config
! portions omitted for brevity
ip cef
!
interface Serial0/1
 ip address 192.168.12.3 255.255.255.0
 encapsulation ppp
!
r3#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#int s 0/1
R3(config-if)#auto qos voip
R3(config-if)#end
R3#
R3#show int s 0/1
Serial0/1 is up, line protocol is up
  Hardware is PowerQUICC Serial
  Internet address is 192.168.12.3/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
```

In this example, the configuration already uses CEF, as shown in the **ip cef** command. Also, interface Serial0/1 does not have a **service-policy** command. Next, the **auto qos voip** command enables AutoQoS VoIP on the interface. AutoQoS VoIP is supposed to be easy to configure—and it is. That is all there is to the typical configuration.

Of course, if you are willing to dig into QoS enough to pass the QOS exam, you are probably curious as to what the router just did for you (or to you, depending on your perspective). Example 3-8 lists the output of two **show** commands that list the new configuration generated by AutoQoS VoIP.

Example 3-8 **show auto qos** *Commands*

```
R3#show auto qos
!
! First ACL matches Voice payload, second match Voice signaling
!
  ip access-list extended AutoQoS-VoIP-RTCP
   permit udp any any range 16384 32767
  ip access-list extended AutoQoS-VoIP-Control
   permit tcp any any eq 1720
   permit tcp any any range 11000 11999
   permit udp any any eq 2427
   permit tcp any any eq 2428
   permit tcp any any range 2000 2002
   permit udp any any eq 1719
   permit udp any any eq 5060
!
! Next class-map matches voice payload
!
  class-map match-any AutoQoS-VoIP-RTP-UnTrust
   match protocol rtp audio
   match access-group name AutoQoS-VoIP-RTCP
!
! Next class-map matches voice control traffic
!
  class-map match-any AutoQoS-VoIP-Control-UnTrust
   match access-group name AutoQoS-VoIP-Control
!
! This one matches all non-voice that was already marked like voice
!
  class-map match-any AutoQoS-VoIP-Remark
   match ip dscp ef
   match ip dscp cs3
   match ip dscp af31
!
! Policy-map performs low latency queuing for voice payload, CBWFQ for
! Voice signaling, re-marks non-voice packets that are marked like voice,
! and queues all non-voice in class-default.
!
```

continues

Example 3-8 **show auto qos** *Commands (Continued)*

```
   policy-map AutoQoS-Policy-UnTrust
    class AutoQoS-VoIP-RTP-UnTrust
     priority percent 70
     set dscp ef
    class AutoQoS-VoIP-Control-UnTrust
     bandwidth percent 5
     set dscp af31
    class AutoQoS-VoIP-Remark
     set dscp default
    class class-default
     fair-queue

 Serial0/1 -
 !
  interface Serial0/1
   service-policy output AutoQoS-Policy-UnTrust
!
! Sends rmon events for packet drops for voice to the SNMP manager
!
  rmon event 33333 log trap AutoQoS description "AutoQoS SNMP traps for Voice Dr
ops" owner AutoQoS
  rmon alarm 33333 cbQosCMDropBitRate.1161.1163 30 absolute rising-threshold 1 3
3333 falling-threshold 0 owner AutoQoS
R3#show auto qos int s 0/1

 Serial0/1 -
 !
  interface Serial0/1
   service-policy output AutoQoS-Policy-UnTrust
```

Of the two commands shown, **show auto qos** shows all the configuration generated by AutoQoS VoIP. Had any changes been made to the configuration, they would not be shown in the output of this command—a **show running-config** command would be required to see those changes. The **show auto qos interface serial0/1** command simply shows the policy map enabled on the interface with the **service-policy** command.

You have not read enough of the book to appreciate the details of the configuration, but by the end of the book, you should be able to easily understand the configuration shown in Example 3-8.

More AutoQoS Configuration Options

To remove the AutoQoS configuration, you simply use the **no** form of the command—in other words, **no auto qos voip**. However, if you happened to change the QoS configuration manually, this

command will not necessarily remove all the configuration. In that case, you have to remove the configuration manually.

The **auto qos voip** command comes with two optional parameters, **trust** and **fr-atm**. The **trust** parameter tells AutoQoS VoIP to trust the incoming DSCP markings. So, the automatically configured QoS commands will not look for voice payload and voice signaling. Instead, following best practices, the commands look for packets with DSCP EF as voice payload, and DSCPs AF31 and CS3 as voice signaling. This option makes sense if you know that the packets have already been marked close to the edge of the network. By leaving off the **trust** option, you are essentially telling the router that you do not trust the markings, so AutoQoS VoIP generates configurations that classify based on other packet headers besides DSCP.

The **fr-atm** option is used with Frame Relay permanent virtual circuits (PVCs) when the PVC is using ATM service interworking. *Service interworking* means that one end of the VC is Frame Relay and the other end is ATM. AutoQoS VoIP configures Frame Relay Fragmentation by default, which is not supported with service interworking VCs. The **fr-atm** parameter makes AutoQoS VoIP use another fragmentation tool, called Multilink Point-to-Point Protocol (MLP) over Frame Relay and ATM, to perform fragmentation.

Ignoring the details of the configuration that AutoQoS VoIP generates, it is easy to choose the right parameters. If all packets have not been marked by the time they arrive at the router, do not use the **trust** option; if they have already been marked, use the **trust** option. If using Frame Relay and the VC uses service interworking, use the **fr-atm** option. Make sure you configure bandwidth correctly and look at the list of considerations in the QoS configuration guide.

Although making AutoQoS VoIP work is easy, it is important to understand what it is trying to do. The next section takes a closer look at what AutoQoS is actually trying to accomplish on a router.

AutoQoS VoIP for Router PHBs

The key to appreciating what AutoQoS VoIP configures is to focus on two facts:

- It is oriented toward making VoIP traffic work well.
- It follows Cisco best practices for the various QoS tools in regard to VoIP.

By reading the rest of the chapters in this book, including the material in Chapter 10, "Cisco QoS Best Practices," you will learn more about what is best to use for QoS. However, any time someone suggests the best way to do something with technology, there can always be some points of

disagreement. Table 3-5 shows the PHBs that can be configured by AutoQoS VoIP on a router and some comments about its choices.

Table 3-5 *PHBs Generated by AutoQoS VoIP Configuration (for Routers)*

PHB	Comments
Class and Mark	If the **trust** option is omitted, AutoQoS VoIP configures CB Marking, using NBAR, to classify traffic into voice payload (marked DSCP EF), voice signaling (marked DSCP AF31), and all else (marked DSCP BE).
Queuing	Voice payload is placed into an LLQ. Voice signaling is in another queue with a low-bandwidth CBWFQ queue. All other traffic defaults into the **class-default** queue, which by default gets 25 percent of link bandwidth.
Compression	If the link has a bandwidth of 768 kbps or less, cRTP is enabled.
LFI	If the link has a bandwidth of 768 kbps or less, AutoQoS enables LFI. For interfaces originally configured for HDLC or PPP, AutoQoS reconfigures MLP with LFI on those interfaces. For Frame Relay, AutoQoS configures FR Fragmentation, unless the **fr-atm** option is configured on the **auto qos** command. In that case, AutoQoS configures MLP over Frame Relay LFI. In each case, the fragment size is tuned for a 10-ms fragment.
Shaping	On Frame Relay interfaces, Frame Relay Traffic Shaping (FRTS) is configured, tuned for a Shaping interval of 10 ms.

This table actually summarizes a lot of the best practices for QoS in an enterprise network. It is difficult to appreciate the details until you read the remaining chapters, so as a suggestion, go ahead and read through the rest of the chapters of the book. Chapter 10 reminds you to come back to this section of Chapter 3 and review what AutoQoS VoIP thinks of as best practices. You will have a much better appreciation for what it does for you at that point.

If you do take this suggestion, you might enjoy looking through a Frame Relay configuration as well. Example 3-9 shows how to configure AutoQoS VoIP on a Frame Relay interface, along with the resulting QoS configuration. After reading the rest of the book, working through this

configuration can be an interesting exercise, both for reviewing what you learned and for solidifying your understanding of the PHBs created by AutoQoS VoIP.

Example 3-9 *Low-Bandwidth (256 kbps) Frame Relay AutoQoS Configuration*

```
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#int s 0/0.1
R3(config-subif)#bandwidth 256
R3(config-subif)#frame-relay interface-dlci 143
R3(config-fr-dlci)#auto qos voip
R3(config-fr-dlci)#^Z
R3#
R3#sh auto qos
 !
  ip access-list extended AutoQoS-VoIP-RTCP
   permit udp any any range 16384 32767
 !
  ip access-list extended AutoQoS-VoIP-Control
   ended IP access list AutoQoS-VoIP-Control
   permit tcp any any eq 1720
   permit tcp any any range 11000 11999
   permit udp any any eq 2427
   permit tcp any any eq 2428
   permit tcp any any range 2000 2002
   permit udp any any eq 1719
   permit udp any any eq 5060
 !
  class-map match-any AutoQoS-VoIP-RTP-UnTrust
   match protocol rtp audio
   match access-group name AutoQoS-VoIP-RTCP
 !
  class-map match-any AutoQoS-VoIP-Control-UnTrust
   match access-group name AutoQoS-VoIP-Control
 !
  class-map match-any AutoQoS-VoIP-Remark
   match ip dscp ef
   match ip dscp cs3
   match ip dscp af31
 !
    policy-map AutoQoS-Policy-UnTrust
     class AutoQoS-VoIP-RTP-UnTrust
      priority percent 70
      set dscp ef
     class AutoQoS-VoIP-Control-UnTrust
      bandwidth percent 5
      set dscp af31
     class AutoQoS-VoIP-Remark
```

continues

Example 3-9 *Low-Bandwidth (256 kbps) Frame Relay AutoQoS Configuration (Continued)*

```
     set dscp default
    class class-default
     fair-queue

!  Serial0/0.1: DLCI 143 -
 !
  interface Serial0/0
   frame-relay traffic-shaping
 !
  interface Serial0/0.1 point-to-point
   frame-relay interface-dlci 143
    class AutoQoS-VoIP-FR-Serial0/0-143
   frame-relay ip rtp header-compression
 !
  map-class frame-relay AutoQoS-VoIP-FR-Serial0/0-143
   frame-relay cir 256000
   frame-relay bc 2560
   frame-relay be 0
   frame-relay mincir 256000
   service-policy output AutoQoS-Policy-UnTrust
   frame-relay fragment 320
 !
  rmon event 33333 log trap AutoQoS description "AutoQoS SNMP traps for Voice Dr
ops" owner AutoQoS
  rmon alarm 33338 cbQosCMDropBitRate.1619.1621 30 absolute rising-threshold 1 3
3333 falling-threshold 0 owner AutoQoS
```

NOTE The current QOS exam does not cover FRTS as a topic unto itself. Chapter 7, "Congestion Avoidance Through Drop Policies," covers some basics of FRTS so that you understand some of the Frame Relay Fragmentation coverage. However, AutoQoS creates FRTS configuration in some cases. Appendix B, "Additional QoS Reference Materials" (found on the book's accompanying CD-ROM), includes coverage of FRTS from the previous edition of this book, if you want more background information.

AutoQoS VoIP for Cisco IOS Switches

Cisco supports AutoQoS VoIP on 2950 (Enhanced Image), 3550, 4500, and 6500 Series switches. It is similar to AutoQoS VoIP for routers in philosophy but different in the details. For instance, for both switches and routers, AutoQoS VoIP does the following:

- Automatically configures QoS settings based on simple configuration commands
- Classifies based on voice payload, voice signaling, and other traffic
- Follows Cisco Best Practices for QoS

AutoQoS VoIP for Cisco IOS switches differs from routers in terms of both what PHBs are configured and the configuration commands created by AutoQoS. Even though IOS-based Cisco switches support MQC-style commands, many of the core QoS features are not configured with MQC commands. Also, different QoS tools are used on the switches (as compared to routers) but the ultimate goal is still the same—to ensure VoIP traffic works well, with minimal configuration.

As with AutoQoS VoIP for routers, this section begins with some discussion of the configuration and then moves on to discuss the PHBs created by the automatically configured QOS commands.

AutoQoS VoIP Configuration for IOS Switches

IOS switches use the same basic interface subcommand as routers, but with a couple of different options. The full syntax is as follows:

```
auto qos voip {cisco-phone | trust}
```

Note that you must choose either **cisco-phone** or **trust**. To appreciate which one to use, consider a typical campus switch design, as shown in Figure 3-4.

Figure 3-4 *Typical Campus Switch Design*

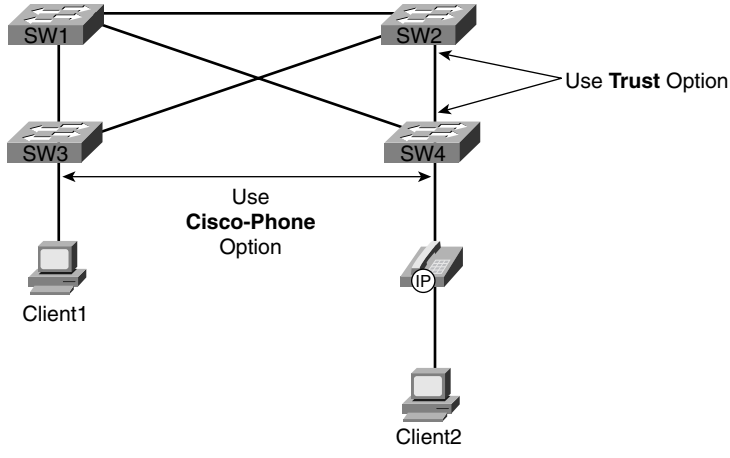
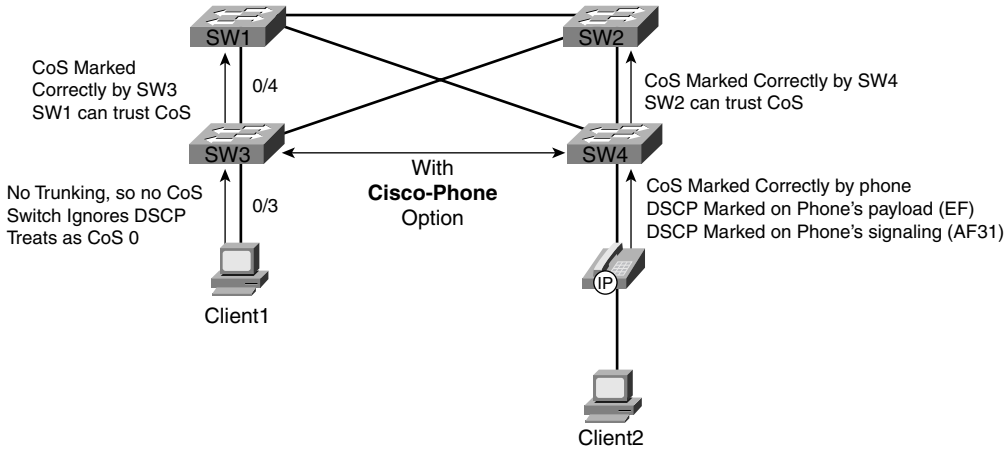


Figure 3-4 shows that for switch ports connected to end-user devices, the **cisco-phone** option is suggested. Alternatively, for trunks between two switches, the **trust** option is suggested. But why? Figure 3-5 shows the explanation against the same diagram, with the explanation following.

Figure 3-5 *Two Settings, Three Actions, for Classification and Marking*



The **cisco-phone** parameter tells the switch to use CDP version 2 to recognize whether a phone is currently attached to the port. If a Cisco IP Phone is there, the switch can trust the QoS markings from the phone. If a phone is not there, the switch simply does not trust the QoS markings on the frames entering that interface and treats all traffic as BE traffic.

In effect, the **cisco-phone** option tells the switch to extend the *trust boundary* down to the Cisco IP Phone.

As a result of the actions at SW3 and SW4, SW1 and SW2 can simply trust the CoS values of the incoming frames. For frames that come in from an untrusted device or port, as with Client1 and SW3, SW3 forwards the frame to SW1 with a CoS of 0, because SW3 did not trust the device off that port. However, SW4 trusted the IP Phone to mark the CoS value correctly, so frames forwarded by SW4 to SW2 have CoS 5 (voice payload), CoS 3 (voice signaling), or CoS 0 (traffic from PC Client2).

To summarize, configure the **cisco-phone** setting on ports connected to end users, and the **trust** setting when connecting to other switches, assuming the CoS values will be marked correctly.

Example 3-10 shows a configuration for SW3, with both **cisco-phone** and **trust** configured.

Example 3-10 *2950 (EI) Configuration for AutoQoS VoIP*

```
SW3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
SW3(config)#int fa 0/3
SW3(config-if)#auto qos voip cisco-phone
SW3(config-if)#int fa 0/4
SW3(config-if)#auto qos voip trust
SW3(config-if)#^Z
SW3#show auto qos
```

Example 3-10 *2950 (EI) Configuration for AutoQoS VoIP (Continued)*

```
Initial configuration applied by AutoQoS:
wrr-queue bandwidth 20 1 80 0
no wrr-queue cos-map
wrr-queue cos-map 1 0 1 2 4
wrr-queue cos-map 3 3 6 7
wrr-queue cos-map 4 5
mls qos map cos-dscp 0 8 16 26 32 46 48 56
!
interface FastEthernet0/3
 mls qos trust device cisco-phone
 mls qos trust cos
!
interface FastEthernet0/4
 mls qos trust cos
SW3#
```

The configuration itself is easy. On ports known to be end-user ports, such as FastEthernet0/3, the **auto qos voip cisco-phone** interface subcommand is used. On trunk ports, assuming the frames will be marked with correct CoS values, the **auto qos voip trust** interface subcommand is used.

The resulting configuration shown in the **show auto qos voip** command is difficult to understand until you have learned more about QoS configuration on a 2950 switch. You might want to return to this section of Chapter 3 after reading Chapter 9, "LAN QoS," after which the output of the **show auto qos voip** command should be more meaningful.

AutoQoS VoIP for IOS Switch PHBs

It is important to understand what the AutoQoS VoIP configured, at least in terms of the meaning behind the commands. As with routers, two key points help in understanding the PHBs created on the switches:

- It is oriented toward making VoIP traffic work well.
- It follows Cisco Best Practices for the various QoS tools in regard to VoIP.

The list of PHBs created by AutoQoS VoIP on a 2950 switch differs from routers and is summarized in Table 3-6.

Table 3-6 *PHBs Generated by AutoQoS VoIP Configuration (for 2950 EI Switches)*

PHB	Comments
Class and Mark	Classifies based on CoS on trusted ports, or on cisco-phone ports on which an actual phone is attached. Assumes CoS 0 on ports with **cisco-phone** configured when no phone is detected with CDP 2.0. Re-marks DSCP of packets based on CoS (CoS 5 – DSCP EF, CoS 3 – DSCP AF31, CoS 0 – DSCP BE).
Queuing	Creates an LLQ for voice payload (CoS 5) and assigns 20 percent of remaining bandwidth to voice signaling (CoS 3) and 80 percent to all other (CoS 0) traffic.

Chapter 9 provides details of the underlying QoS commands created by AutoQoS.

Next, this chapter ends with a short description of AutoQoS for Cat-OS on the 6500 Series switches.

AutoQoS VoIP for 6500 Cat-OS

The Cisco 6500 series switches support Cat-OS when running the Catalyst Operating System (Cat-OS). Like AutoQoS on routers and IOS-based switches, AutoQoS uses simple commands to initiate the automatic configuration of QoS commands, with settings that help ensure good performance of VoIP traffic.

The Cisco QoS course and exam tend to focus on switch examples using IOS-based switches and mostly ignores commands on Cat-OS. The one exception to that rule in the Cisco QoS course is the coverage of the AutoQoS commands in Cat-OS. So, for the purposes of this book, this short section explains the commands and their meaning.

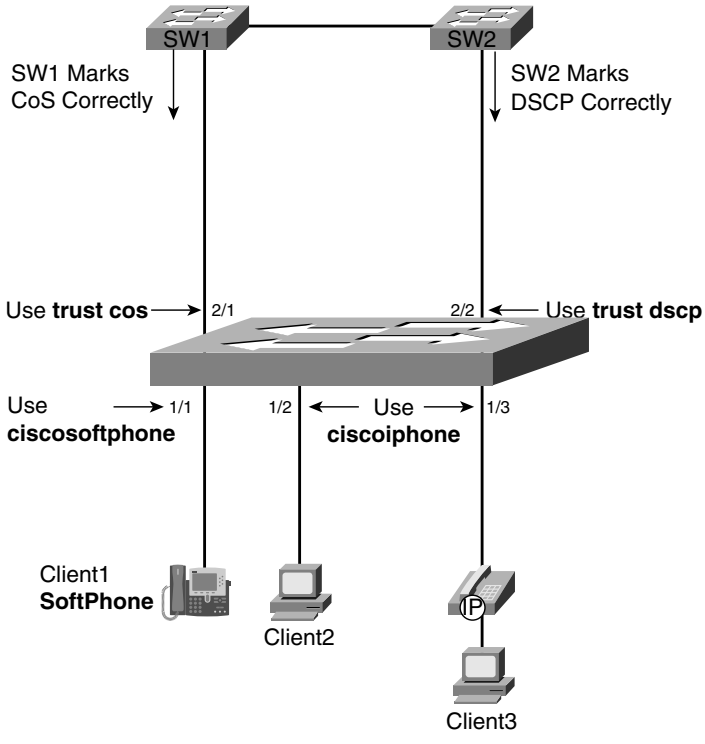
Table 3-7 lists the Cat-OS commands related to AutoQoS VoIP.

Table 3-7 *Command Reference for AutoQoS VoIP (for 6500 Cat-OS)*

Command	Function
set qos autoqos	Sets global settings, such as for a CoS-DSCP map
set port qos autoqos *mod/port* **trust** [**cos** | **dscp**]	For the specified ports, tells the switch to trust the DSCP or CoS setting of incoming frames
set port qos autoqos *mod/port* **voip** [**ciscosoftphone** | **ciscoipphone**]	For the specified ports, tells the switch to use CDP V2 to sense the presence of a phone and to trust CoS if a phone is found (**ciscoipphone**), or to trust DSCP but police traffic so a PC running Cisco SoftPhone does not abuse the privilege of having its DSCP trusted (**ciscosoftphone**)

To configure AutoQoS on a 6500 switch, you first need to use the **set qos autoqos** command. Afterward, you need to use a **set port qos autoqos** command covering each port, choosing either **trust cos**, **trust dscp**, **voip ciscoipphone**, or **voip ciscosoftphone** as options. To appreciate each of the four options, consider Figure 3-6, which demonstrates the implementation of all four options.

Figure 3-6 *AutoQoS Options on 6500 Cat-OS*



Some of the options for the 6500 switch are the same as for IOS-based switches such as the 2950. For instance, if frames have already been marked with the correct CoS value on a trunk, as is the case on port 2/1 in Figure 3-6, the command **set port qos autoqos 2/1 trust cos** tells the switch to trust CoS on that port. That is the same action as with the switch IOS command **auto qos voip trust**.

Another option that works like AutoQoS on an IOS switch is the **ciscoipphone** option. If a port might have an IP Phone, or it might not, such as on ports 1/2 and 1/3 in Figure 3-6, the command **set port qos autoqos 1/2 voip ciscoipphone** is the correct setting. As with the switch IOS command

auto qos voip cisco-phone, this **set** command tells the switch to use CDP V2 to discover if a Cisco IP Phone is on the port. If a Cisco IP Phone is on the port, trust CoS and extend the trust boundary down to the Cisco IP Phone, marking CoS 0 for all traffic sent by the PC attached to the phone. If no Cisco IP Phone is on the port, treat all traffic as CoS 0.

Of the other two options, using the command **set port qos autoqos 2/2 trust dscp** is the most obvious. If the DSCP has already been set correctly, you should use this command to tell the switch to trust the incoming DSCP setting.

Finally, the command **set port qos autoqos 1/1 voip ciscosoftphone** is recommended on port 1/2 in Figure 3-6. The PC attached to that port is running the Cisco SoftPhone application, which in effect creates an IP Phone via software running on the PC. However, the Cisco SoftPhone marks DSCP correctly (EF for payload, AF31 for signaling), but it does not mark CoS—in fact, trunking is not even needed on this port. You could just trust DSCP on this port, but the danger is that the PC could set DSCP EF or AF31 for other IP packets, hoping to get good performance from the network. To prevent such abuse, the **set port qos autoqos 1/1 voip ciscosoftphone** command also enables policing at a rate of 320 kbps for DSCP EF traffic and 32 kbps for DSCP AF31 traffic. Cisco SoftPhone can use up to 256 kbps for a single call, although it typically uses less. So, rather than just trusting DSCP, the **ciscoipphone** option lets you trust DSCP, but it reduces the impact if the end user is marking other packets as DSCP EF or AF31.

Table 3-8 lists the four competing options for AutoQoS settings on each port.

Table 3-8 *Comparing Options for* **set port** *Command with AutoQoS*

Command	Function
trust cos	Accept the CoS of incoming frames.
trust dscp	Accept the DSCP of incoming packets.
voip ciscoipphone	Use CDPv2 to discover the absence or presence of an IP Phone. If one is there, trust CoS and extend the trust boundary to the IP Phone, causing the PC's frames to be marked CoS 0. If no phone is there, treat all incoming frames as CoS 0.
voip ciscosoftphone	Trust DSCP, but police DSCP EF at 320 kbps and DSCP AF31 at 32 kbps.

Comparisons of CLI, MQC, and AutoQoS

In the past, QoS tools each had a different set of CLI configuration commands. Over time, with the addition of more and more tools, configuring QoS became a challenge. With the advent of MQC, and with the more recently developed and most useful QoS tools using MQC commands, the configuration complexity was reduced tremendously. Going a step further, by creating AutoQoS VoIP, Cisco provided an even simpler way to configure QoS for VoIP traffic, with no requirement to understand QoS to make the network work well.

The QoS course includes a wonderful table comparing these three options, included here as Table 3-9, summarizing the comparison points between the three options.

Table 3-9 *Comparisons of CLI, MQC, and AutoQoS*

	CLI	MQC	AutoQoS
Ease of Use	Poor	Easier	Simple
Ability to Fine Tune	OK	Very Good	Very Good
Time to Implement	Longest	Average	Shortest
Modularity	Poor	Excellent	Excellent

Foundation Summary

The "Foundation Summary" is a collection of tables and figures that provide a convenient review of many key concepts in this chapter. For those of you already comfortable with the topics in this chapter, this summary can help you recall a few details. For those of you who just read this chapter, this review should help solidify some key facts. For any of you doing your final preparation before the exam, these tables and figures are a convenient way to review the day before the exam.

Figure 3-7 shows the general flow of MQC commands.

Figure 3-7 *MQC Commands and Their Correlation*

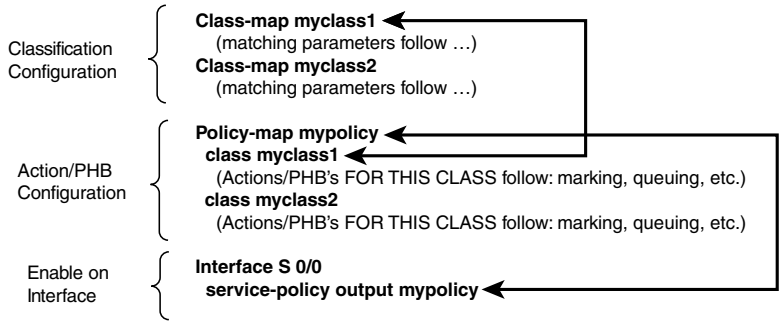


Table 3-10 lists the **match** command options available for the IOS revision covered by the current QOS exam, 12.2(15)T.

Table 3-10 **match** *Configuration Command Reference for MQC Tools*

Command	Function
match [**ip**] **precedence** *precedence-value* [*precedence-value precedence-value precedence-value*]	Matches precedence in IPv4 packets when the **ip** parameter is included; matches IPv4 and IPv6 packets when the **ip** parameter is missing.
match access-group {*access-group* | **name** *access- group-name*}	Matches an ACL by number or name.
match any	Matches all packets.
match class-map *class-map-name*	Matches based on another class map.
match cos *cos-value* [*cos-value cos-value cos-value*]	Matches a CoS value.
match destination-address mac *address*	Matches a destination MAC address.

Table 3-10 **match** *Configuration Command Reference for MQC Tools (Continued)*

Command	Function
match fr-dlci *dlci-number*	Matches a particular Frame Relay DLCI.
match input-interface *interface-name*	Matches an input interface.
match ip dscp *ip-dscp-value* [*ip-dscp-value ip-dscp- value ip-dscp-value ip-dscp-value ip-dscp-value ip- dscp-value ip-dscp-value*]	Matches DSCP in IPv4 packets when the **ip** parameter is included; matches IPv4 and IPv6 packets when the **ip** parameter is missing.
match ip rtp *starting-port-number port-range*	Matches the RTP's UDP port-number range, even values only.
match mpls experimental *number*	Matches an MPLS Experimental value.
match mpls experimental topmost *value*	When multiple labels are in use, this command matches the MPLS EXP field in the topmost label.
match not *match-criteria*	Reverses the matching logic; in other words, things matched by the matching criteria do not match the class map.
match packet length {**max** *maximum-length-value* [**min** *minimum-length-value*] | **min** *minimum-length-value* [**max** *maximum-length-value*]}	Matches packets based on the minimum length, maximum length, or both.
match protocol citrix app *application-name-string*	Matches NBAR Citrix applications.
match protocol http [**url** *url-string* | **host** *hostname- string* | **mime** *MIME-type*]	Matches a host name and URL string.
match protocol *protocol-name*	Matches NBAR protocol types.
match protocol rtp [**audio** | **video** | **payload-type** *payload-string*]	Matches RTP audio or video payload, based on the payload type. Also allows explicitly specified payload types.
match qos-group *qos-group-value*	Matches a QoS group.
match source-address mac *address-destination*	Matches a source MAC address.

Table 3-11 lists the various MQC subcommands available in Cisco IOS Software for defining actions to be taken.

Table 3-11 *Action (PHB) Subcommands Inside a Policy Map*

Command	Function
set	CB Marking action, with options to mark several fields inside headers
bandwidth	Reserves bandwidth for the class for CBWFQ
priority	Reserves bandwidth, and provides LLQ with CBWFQ
shape	Shapes traffic in the class with a defined bandwidth and burst sizes
police	Polices traffic in the class with a defined bandwidth and burst sizes
compress	Performs TCP and RTP header compression on packets in the class

Because routers and switches do different tasks, QoS differs slightly between them, but to get an idea about what AutoQoS does, consider that AutoQoS on routers works to classify packets into three service classes:

- Voice payload
- Voice signaling
- All other traffic

Table 3-12 lists the commands related to AutoQoS VoIP, followed by an example configuration.

Table 3-12 *Command Reference for AutoQoS VoIP (for Routers)*

Command	Function
auto qos voip [**trust**] [**fr-atm**]	Configuration command that enables AutoQoS VoIP on an interface (PPP or HDLC) or VC (FR and ATM)
no auto qos	Disables AutoQoS VoIP on an interface (PPP or HDLC) or VC (FR and ATM)
show auto qos [**interface** *interface-type*]	Displays what AutoQoS actually created
show policy-map interface *interface-name* [**input** | **output**]	Displays actual configuration of MQC-based parts of the configuration, including any later changes made by an engineer

Table 3-13 shows the PHBs that can be configured by AutoQoS VoIP on a router and some comments about its choices.

Table 3-13 *PHBs Generated by AutoQoS VoIP Configuration (for Routers)*

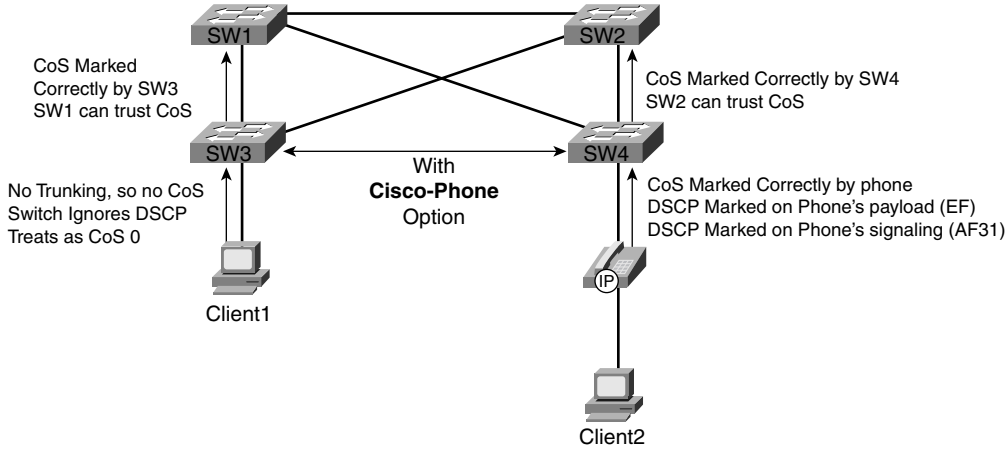
PHB	Comments
Class and Mark	If the **trust** parameter is omitted, AutoQoS VoIP configures CB Marking, using NBAR, to classify traffic into voice payload (marked DSCP EF), voice signaling (marked DSCP AF31), and all else (marked DSCP BE).
Queuing	Voice payload is placed into an LLQ. Voice signaling is in another queue with a low-bandwidth CBWFQ queue. All other traffic defaults into the **class-default** queue, which by default gets 25 percent of link bandwidth.
Compression	If the link has a bandwidth of 768 kbps or less, cRTP is enabled.
LFI	If the link has a bandwidth of 768 kbps or less, AutoQoS enables LFI. For interfaces originally configured for HDLC or PPP, AutoQoS reconfigures MLP with LFI on those interfaces. For Frame Relay, AutoQoS configures FR Fragmentation, unless the **fr-atm** option is configured on the **auto qos** command. In that case, AutoQoS configures MLP over Frame Relay LFI. In each case, the fragment size is tuned for a 10-ms fragment.
Shaping	On Frame Relay interfaces, FRTS is configured, tuned for a Shaping interval of 10 ms.

Cisco IOS switches use the same interface subcommand as routers, but with a couple of different options. The full syntax is as follows:

```
auto qos voip {cisco-phone ¦ trust}
```

Figure 3-8 shows the locations where the **cisco-phone** and **trust** options should be used, along with some notes about how the **cisco-phone** option works.

Figure 3-8 *Two Settings, Three Actions, for Classification and Marking*



The list of PHBs created by AutoQoS VoIP differs from routers and is summarized in Table 3-14.

Table 3-14 *PHBs Generated by AutoQoS VoIP Configuration (for 2950 EI Switches)*

PHB	Comments
Class and Mark	Classifies based on CoS on trusted ports, or on cisco-phone ports on which an actual phone is attached. Assumes CoS 0 on ports with **cisco-phone** configured when no phone is detected with CDP 2.0. Re-marks DSCP of packets based on CoS (CoS 5 – DSCP EF, CoS 3 – DSCP AF31, CoS 0 – DSCP BE).
Queuing	Creates an LLQ for voice payload (CoS 5), and assigns 20 percent of remaining bandwidth to voice signaling (CoS 3), and 80 percent to all other (CoS 0) traffic.

Table 3-15 lists the four competing options for AutoQoS settings on each port when configuring AutoQoS on Cat-OS. The QoS course includes a wonderful table comparing these three options, included here as Table 3-16, summarizing the comparison points between the three options.

Table 3-15 *Comparing Options for* **set port** *Command with AutoQoS*

Command	Function
trust cos	Accept the CoS of incoming frames.
trust dscp	Accept the DSCP of incoming packets.
voip ciscoipphone	Use CDPv2 to discover the absence or presence of an IP Phone. If one is there, trust CoS, and extend the trust boundary to the IP Phone, causing the PC's frames to be marked CoS 0. If no phone is there, treat all incoming frames as CoS 0.
voip ciscosoftphone	Trust DSCP, but police DSCP EF at 320 kbps and DSCP AF31 at 32 kbps.

Table 3-16 *Comparisons of CLI, MQC, and AutoQoS*

	CLI	MQC	AutoQoS
Ease of Use	Poor	Easier	Simple
Ability to Fine Tune	OK	Very Good	Very Good
Time to Implement	Longest	Average	Shortest
Modularity	Poor	Excellent	Excellent

For Further Reading

This book attempts to cover the breadth and depth of QoS as covered on the QOS exam (642-642). However, you might want to read more about topics in this chapter, or other classification and marking topics.

For more on the topics in this chapter:

- Cisco IOS 12.2(15)T AutoQoS Configuration Guide (http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t15/ftautoq1.htm)
- Cisco 2950 QoS Configuration Guide (http://www.cisco.com/univercd/cc/td/doc/product/lan/cat2950/12119ea1/2950scg/swqos.htm)
- 6500 Catalyst OS Auto QoS Configuration Guide (http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/sw_8_2/confg_gd/autoqos.htm)
- "Cisco AutoQoS White Paper" (http://cisco.com/en/US/tech/tk543/tk759/technologies_white_paper09186a00801348bc.shtml)

For design-related guidance:

- "Cisco AVVID Network Infrastructure Enterprise Quality of Service Design" (http://cisco.com/application/pdf/en/us/guest/netsol/ns17/c649/ccmigration_09186a00800d67ed.pdf)

Q&A

As mentioned in the Introduction, you have two choices for review questions. The following questions give you a more difficult challenge than the exam itself by using an open-ended question format. By reviewing now with this more difficult question format, you can exercise your memory better, and prove your conceptual and factual knowledge of this chapter. You can find the answers to these questions in Appendix A.

The second option for practice questions is to use the CD-ROM included with this book. It includes a testing engine and more than 200 multiple-choice questions. You should use this CD-ROM nearer to the end of your preparation for practice with the actual exam format:

1. Configure two class maps, one that matches the packets permitted by ACL 101, and one that matches packets denied by ACL 101. Do not use **class-default**, and do not bother configuring a policy map.

2. Configure a policy map that refers to predefined classes c1, C2, and c3, with the action for each class map being to set the DSCP value to AF11, AF21, and AF22, respectively. Assume that the class maps are already defined.
3. List the three major configuration steps and the main command used in each step for the configuration of a QoS feature using MQC.
4. Describe two different ways with which you could classify packets with DSCP AF31, AF32, and AF33 into a single class using MQC commands.
5. List three benefits of MQC as compared with non-MQC-based QoS features.
6. Consider the configuration snippet that follows. What commands would list statistics for the QoS policy implemented on fastethernet 0/0?

```
class-map fred
 match dscp ef
policy-map barney
 class fred
  set dscp af11
 class class-default
  set dscp be
interface fastethernet0/0
 service-policy input barney
```

7. List the two SNMP MIBs included in Cisco router IOS that can be used by QPM to improve the statistics presented to a QPM user. List the long version of the names and the acronyms.
8. What information can be seen by using the CBQoSMIB that cannot be seen with **show** commands on the device being managed?
9. How many classes can be associated with a single policy map in Cisco IOS Software Release 12.2(15)T?
10. On a router using AutoQoS, what command enables the feature for Frame Relay VCs that use Frame Relay-to-ATM service interworking?
11. On a router using AutoQoS, what command enables the feature on a serial interface when the router can trust the DSCP settings of incoming packets?
12. Describe the classification configuration created by a router when enabling AutoQoS on a serial interface, with all default values chosen on the **auto qos** command.
13. Describe the marking actions created by a router when enabling AutoQoS on a serial interface, with all default values chosen on the **auto qos** command.
14. List three of the requirements on router AutoQoS that need to be true before actually configuring AutoQoS.
15. List the data link protocols on a router that support AutoQoS.
16. List the PHBs created by a router when the **auto qos voip** command is used on a PPP serial interface with the default bandwidth setting.

17. List the PHBs created by a router when the **auto qos voip** command is used on a PPP serial interface with **bandwidth 768** configured.

18. List the PHBs created by a router when the **auto qos voip** command is used on a Frame Relay PVC with **bandwidth 832** configured.

19. When configuring AutoQoS on a router, with a Frame Relay interface, what configuration mode must you be in before using the **auto qos** command? What command gets you into that configuration mode?

20. When configuring a 2950 switch with the **auto qos voip trust** command, what PHBs are configured on the interface?

21. When configuring a 2950 switch with the **auto qos voip cisco-phone** command, what PHBs are configured on the interface?

22. When configuring a 2950 switch with the **auto qos voip cisco-phone** command, what version of CDP is required in order for AutoQoS to work?

23. When planning to use AutoQoS on a 2950 switch, what types of ports are generally configured with the **trust** option, and what type are generally configured with the **cisco-phone** option?

24. When using AutoQoS on a 6500 running Cat-OS, describe the difference in using the **ciscosoftphone** setting and the **trust dscp** setting.

25. When using AutoQoS on a 6500 running Cat-OS, describe when you might choose to use the **ciscosoftphone** option versus the **trust dscp** option.

26. When using AutoQoS on a 6500 running Cat-OS, describe when you might choose to use the **ciscoipphone** setting versus the **trust cos** setting.

27. When using AutoQoS on a 6500 running Cat-OS, the **set port qos autoqos 3/1 voip ciscoipphone** command has been configured. Describe what else must be true before AutoQoS will trust incoming CoS values for frames on port 3/1.

28. Comparing the CLI of older QoS options in a Cisco router, MQC, and AutoQoS, which takes the least time to implement?

29. Comparing the CLI of older QoS options in a Cisco router, MQC, and AutoQoS, which is considered to be the most modular?

30. Comparing the CLI of older QoS options in a Cisco router, MQC, and AutoQoS, which is considered to be the most difficult to use?