



Exam Topics in This Chapter

- 35** Configure standard access lists to filter IP traffic.
- 36** Configure extended access lists to filter IP traffic.
- 37** Monitor and verify selected access list operations on the router.

Understanding Access List Security

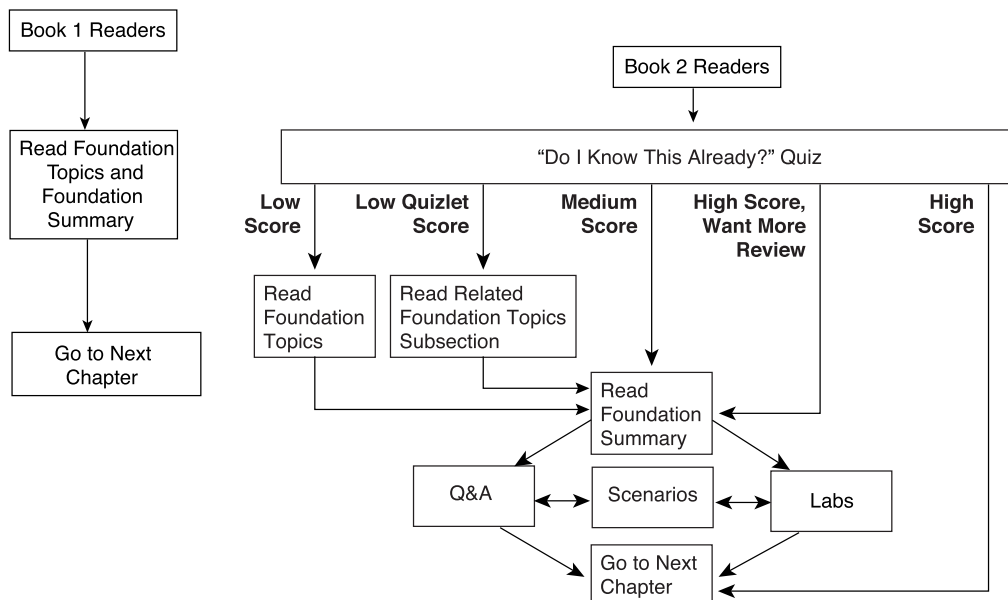
When deciding on the name of this chapter, the first title I chose was “Understanding Network Security.” Then I thought to myself (that’s what you do when you spend weeks on end in your home office writing), “You could easily write a whole book just on this topic!” So I changed the title to better reflect the scope of the security topics in this book, which of course reflects Cisco’s expectations of CCNA candidates. Cisco expects CCNAs to understand security from the perspective of filtering traffic using access lists. Cisco also expects CCNAs to master the ideas and configuration behind the Telnet, auxiliary, console, and enable passwords. These topics are covered in Chapter 2, “Cisco IOS Software Fundamentals.”

The reason that access lists are so important to CCNA candidates is that practically every network uses them. If you do more than basic filtering, access lists can become very tricky. In fact, when I was getting certified to teach Cisco classes in 1993, the Cisco Worldwide Training folks said that the TAC’s most frequent question topic area was how to configure access lists. Access lists are likely to remain a core competency issue for router support personnel for a long time. Also, several other Cisco IOS software features call on access list logic to perform packet-matching features.

When studying access lists in this book or others, keep in mind that there are usually many ways to configure an access list to achieve the same result. Focus on the syntax of the commands and the nuances of the logic. If a particular example (given a set of criteria) is configured differently than you would have configured it, do not be concerned. In this book, I attempt to point out when a particular list could have been written differently.

How to Best Use This Chapter

Two main approaches to using this book are described in Chapter 1, “All About the Cisco Certified Network Associate Certification.” They are called “Book 1” and “Book 2.” Book 1 is for readers who need a thorough foundation before their final study time, and Book 2 is intended for readers who are reviewing and filling in the missing parts of their CCNA knowledge. Using Figure 8-1 as a guide, you should either read the Foundation sections of this chapter or begin with the “Do I Know This Already?” quiz.

Figure 8-1 *How to Use This Chapter*

“Do I Know This Already?” Quiz

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

This 12-question quiz helps you determine how to spend your limited study time. The quiz is sectioned into three smaller four-question “quizlets” that correspond to the three major topic headings in this chapter. Figure 8-1 outlines suggestions on how to spend your time in this chapter based on your quiz score. Use Table 8-1 to record your scores.

Table 8-1 *Scoresheet for Quiz and Quizlets*

Quizlet Number	Foundation Topics Section Covering These Questions	Questions	Score
1	Standard IP Access Lists	1 to 4	
2	Extended IP Access Lists	5 to 8	
3	Named IP Access Lists	9 to 12	
All questions		1 to 12	

- 1** Configure a numbered IP access list that stops packets from subnet 134.141.7.0/255.255.255.0 from exiting serial 0 on a router. Allow all other packets.

- 2** How would a user who does not have the enable password find out what access lists have been configured and where they are enabled?

- 3** Name all the items that a standard IP access list can examine to make a match.

- 4** How many IP access lists of either type can be active on an interface at the same time?

- 5** Configure and enable an IP access list that allows packets from subnet 10.3.4.0/24, to any Web server, to get out serial interface S0. Also allow packets from 134.141.5.4 going to all TCP-based servers using a well-known port to enter serial 0. Deny all other traffic.

- 6** Name all the items that an extended IP access list can examine to make a match.

7 How many IP extended **access-list** commands are required to check a particular port number on all IP packets?

8 What command lists the IP extended access lists enabled on serial 1 without showing other interfaces?

9 Configure a named IP access list that allows only packets from subnet 193.7.6.0 255.255.255.0, going to hosts in network 128.1.0.0 and using a Web server in 128.1.0.0, to enter serial 0 on a router.

10 Name all the items that a named standard IP access list can examine to make a match.

11 List the types of IP access lists (numbered standard, numbered extended, named standard, named extended) that can be enabled to prevent Telnet access into a router. What commands would be used to enable this function, assuming that **access-list 2** was already configured to match the right packets?

12 Name all the items that a named extended IP access list can examine to make a match.

The answers to the “Do I Know This Already?” quiz are found 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:

- **6 or less overall score**—Read the entire chapter. This includes the “Foundation Topics” and “Foundation Summary” sections, the “Q&A” section, and the scenarios at the end of the chapter.
- **2 or less on any quizlet**—Review the subsections of the “Foundation Topics” section, based on Table 8-1. Then move to the “Foundation Summary” section, the “Q&A” section, and the scenarios at the end of the chapter.
- **7, 8, or 9 overall score**—Begin with the “Foundation Summary” section, and then go to the “Q&A” section and the scenarios at the end of the chapter.
- **10 or more overall score**—If you want more review of these topics, skip to the “Foundation Summary” section and then go to the “Q&A” section and the scenarios at the end of the chapter. Otherwise, move to the next chapter.

Foundation Topics

Standard IP Access Lists

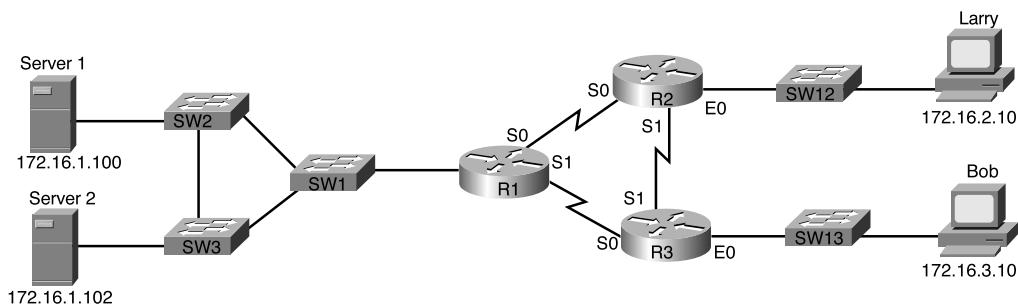
- 35** Configure standard access lists to filter IP traffic.
- 37** Monitor and verify selected access list operations on the router.

IP access lists cause a router to discard some packets based on criteria defined by the network engineer. The goal of these filters is to prevent unwanted traffic in the network—whether to prevent hackers from penetrating the network, or just to prevent employees from using systems that they should not be using. Access lists should simply be part of an organization’s security policy, but for CCNA study purposes, we do not need to consider the business goals that drive the security policy. As long as you can configure access lists to filter packets, you know what you need to know about filtering for the CCNA exam.

By the way, IP access lists can also be used to filter routing updates, to match packets for prioritization, and to match packets for implementing quality of service features, but these additional features are not covered on the CCNA exam.

As soon as you know what needs to be filtered, the next goal is to decide where to filter the traffic. Figure 8-2 serves as an example. In this case, imagine that Bob is not allowed to access Server1, but Larry is.

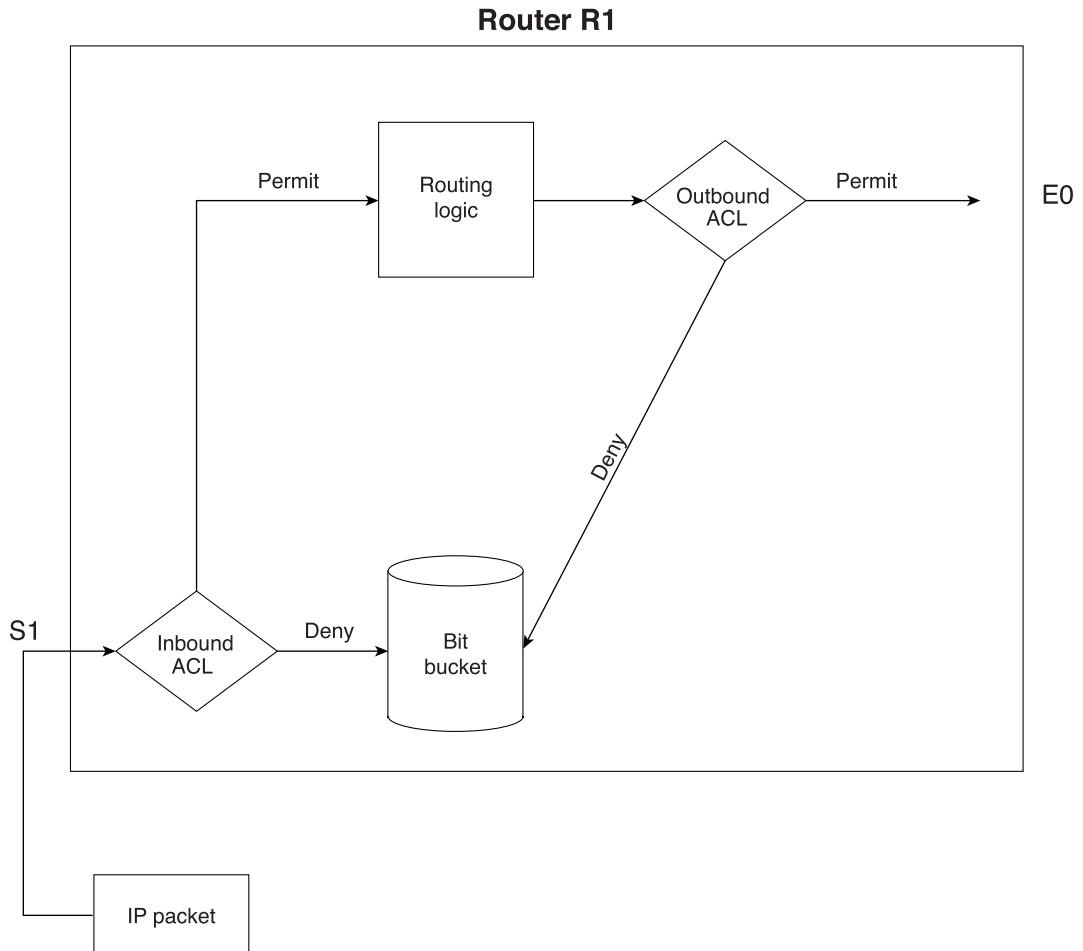
Figure 8-2 Locations Where Access List Logic Can Be Applied in the Network



Filtering logic could be configured on any of the three routers and on any of their interfaces. However, some choices simply would not work (but others would). Because Bob’s traffic is the only traffic that needs to be filtered, and the goal is to stop access to Server1, the access list could be applied at either R1 or R3. And because Bob’s attempted traffic to Server1 would not need to go through R2, R2 would not be a good place to put the access list logic. For the sake of discussion, I’ll pick R1.

As soon as you have chosen where you want to place the access list, you must choose the interface on which to apply the access logic. You must also decide whether to apply the logic for inbound or outbound packets. For instance, imagine that you wanted to filter Bob's packets sent to Server1. Figure 8-3 shows the options for filtering the packet.

Figure 8-3 Locations Where Access List Logic Can Be Applied on Router R1



Filtering logic can be applied to packets entering S1 or to packets exiting E0 on R1 in order to match the packet sent by Bob to Server1. In general, you can filter packets by creating and enabling access lists for both incoming and outgoing packets on each interface. Here are some key features of Cisco access lists:

- Packets can be filtered as they enter an interface, before the routing decision.
- Packets can be filtered before they exit an interface, after the routing decision.

- *Deny* is the term used in Cisco IOS software to imply that the packet will be filtered.
- *Permit* is the term used in Cisco IOS software to imply that the packet will not be filtered.
- The filtering logic is configured in the access list.
- At the end of every access list is an implied “deny all traffic” statement. Therefore, if a packet does not match any of your access list statements, it is blocked.

For example, you might create an access list in R1 and enable it on R1’s S1 interface. The access list would look for packets that came from Bob. Therefore, the access list would need to be enabled for inbound packets, because in this network, packets from Bob enter S1, and packets to Bob exit S1.

Access lists have two major steps in their logic: matching and action. Matching logic examines each packet and determines whether it matches the **access-list** statement. For instance, Bob’s IP address would be used for matching packets sent from Bob. As soon as an **access-list** statement is matched, there are two actions to choose from: deny and permit. Deny means to discard the packet, and permit implies that the packet should continue on its way. So the access list for preventing Bob’s traffic to the server might go something like this:

Look for packets with Bob’s source IP address and Server1’s destination IP address. When you see them, discard them.

Not surprisingly, it can get a lot harder than that in real life. Even a short list of matching criteria can create complicated access lists on a variety of interfaces in a variety of routers. I’ve even heard of a couple of large networks with a couple of full-time people who do nothing but plan and implement access lists!

Access lists are a series of statements with matching criteria and the resulting actions. When an access list has multiple entries, the first statement matched determines the action. The two diamond-shaped symbols in Figure 8-3 represent the application of access list logic. That logic can be summarized as follows:

- Step 1** The matching parameters of the first **access-list** statement are compared to the packet.
- Step 2** If a match is made, the action defined in this **access-list** statement (permit or deny) is performed.
- Step 3** If a match is not made in Step 2, Steps 1 and 2 are repeated using the next sequential **access-list** statement.
- Step 4** If no match is made with an entry in the access list, the deny action is performed.

Access list logic is applicable whether you’re using standard or extended access lists; the only difference between the two is in what constitutes a match.

The matching criteria available to access lists is based on fields inside the IP, TCP, and UDP headers. Extended access lists can check source and destination IP addresses, as well as source and destination port numbers, along with several other fields. However, standard IP access lists can examine only the source IP address.

You can configure the portion of the IP address that is checked by the **access-list** command. For instance, if you wanted to stop Bob from sending packets to Server1, you would look at the entire IP address of Bob and Server1 in the access list. But what if the criteria were to stop all hosts in Bob's subnet from getting to Server1? Because all hosts in Bob's subnet have the same numbers in their first three octets, the access list could just check the first three octets of the address in order to match all packets with a single **access-list** statement.

Cisco *wildcard masks* are access list parameters that define the portion of the IP address that should be examined. For example, suppose that one mask implies that the whole packet should be checked and another implies that only the first three octets need to be examined. To perform this matching, Cisco access lists use wildcard masks. Table 8-2 lists some of the more popular wildcard masks, as well as a few that are not quite as common.

Table 8-2 *Sample Access List Wildcard Masks*

Wildcard Mask	Binary Version of the Mask	Description
0.0.0.0	00000000.00000000.00000000.00000000	The entire IP address must match.
0.0.0.255	00000000.00000000.00000000.11111111	Just the first 24 bits must match.
0.0.255.255	00000000.00000000.11111111.11111111	Just the first 16 bits must match.
0.255.255.255	00000000.11111111.11111111.11111111	Just the first 8 bits must match.
255.255.255.255	11111111.11111111.11111111.11111111	Don't even bother to compare; it's automatically considered to match (0 bits need to match).
0.0.15.255	00000000.00000000.00001111.11111111	Just the first 20 bits must match.
0.0.3.255	00000000.00000000.00000011.11111111	Just the first 22 bits must match.
32.48.0.255	00100000.00110000.00000000.11111111	All bits except the 3rd, 11th, 12th, and last 8 must match.

The first several examples show the typical use of the wildcard mask. As you can see, it is not a subnet mask. A wildcard of 0.0.0.0 means that the entire IP address must be examined, and be equal, in order to be considered a match. 0.0.0.255 means that the last octet automatically matches, but the first three must be examined, and so on. More generally, the wildcard mask means the following:

Bit positions of binary 0 mean that the access list compares the corresponding bit position in the IP address and makes sure it is equal to the same bit position in the address configured in the **access-list** statement. Bit positions of binary 1 are wildcards—those bit positions are immediately considered to be a match.

The next two rows of Table 8-2 show two reasonable but not obvious wildcard masks. 0.0.5.255, as seen in binary, is 20 0s followed by 12 1s. This means that the first 20 bits must match. Similarly, 0.0.3.255 means that the first 22 bits must be examined to find out if they match. Why are these useful? If the subnet mask is 255.255.240.0, and you want to match all hosts in the same subnet, the 0.0.15.255 wildcard means that all network and subnet bits must be matched, and all host bits are automatically considered to match. Likewise, if you want to filter all hosts in a subnet that uses subnet mask 255.255.252.0, the wildcard mask 0.0.3.255 matches the network and subnet bits. In general, if you want a wildcard mask that helps you match all hosts in a subnet, invert the subnet mask, and you have the correct wildcard mask.

The last entry in Table 8-2 is unreasonable for real networks, but it is included to make a point. The wildcard mask just defines which bits must be compared and which are automatically assumed to match. You should not expect such strange masks on the exam! The point is that although subnet masks must use a sequential set of binary 1s followed by only binary 0s, wildcard masks do not have to follow any such rule.

Standard IP Access List Configuration

Standard IP access list configuration works much like a simple programming language. The logic is something like this:

If statement 1 is matched, carry out the action defined in that statement. If it isn't, examine the next statement. If it matches, carry out the action it defines. Continue looping through the list until a statement is matched or until the last statement in the list is not matched.

A standard access list is used to match a packet and then take the directed action. Each standard access list can match all or only part of the packet's source IP address. The only two actions taken when an **access-list** statement is matched are to either deny (discard) or permit (forward) the packet.

Table 8-3 lists the configuration commands related to standard IP access lists. Table 8-4 lists the related EXEC commands. Several examples follow the lists of commands.

Table 8-3 *Standard IP Access List Configuration Commands*

Command	Configuration Mode and Description
access-list <i>access-list-number</i> {deny permit} <i>source</i> [<i>source-wildcard</i>] [log]	Global command for standard numbered access lists
ip access-group { <i>number</i> <i>name</i> [in out]}	Interface subcommand to enable access lists
access-class <i>number</i> <i>name</i> [in out]	Line subcommand for standard or extended access lists

Table 8-4 *Standard IP Access List EXEC Commands*

Command	Description
show ip interface [<i>type number</i>]	Includes a reference to the access lists enabled on the interface
show access-lists [<i>access-list-number</i> <i>access-list-name</i>]	Shows details of configured access lists for all protocols
show ip access-list [<i>access-list-number</i> <i>access-list-name</i>]	Shows IP access lists

The first example is basic in order to cover the statements' syntax. As shown in Figure 8-2, Bob is not allowed to access Server1, but Larry is allowed. In Example 8-1, the access list is enabled for all packets going out R1's Ethernet0 interface. Example 8-1 shows the configuration on R1.

Example 8-1 *Standard Access List Stopping Bob from Reaching Server*

```
interface Ethernet0
ip address 172.16.1.1 255.255.255.0
ip access-group 1 out

access-list 1 deny 172.16.3.10 0.0.0.0
access-list 1 permit 0.0.0.0 255.255.255.255
```

There are several small details in this example. Standard IP access lists use a number between 1 and 99, inclusive. Number 1 is used here for no particular reason, other than it's in the right range. The **access-list** command is a global configuration command, *not* a subcommand under

the Ethernet0 interface. (The **access-list** commands do appear toward the end of the configuration file, after the interfaces.) The **ip access-group** command enables the logic on Ethernet0 for packets going out.

Access list 1 stops packets sent by Bob from exiting R1's Ethernet interface based on the matching logic of the first **access-list** statement. It forwards any other packets based on the matching logic of the second statement.

The configuration in Example 8-1 is not what shows up in the output of the **show running-config** command. Example 8-2 shows what would actually be placed in the configuration file.

Example 8-2 *Standard Access List Stopping Bob from Reaching Server1: Revised*

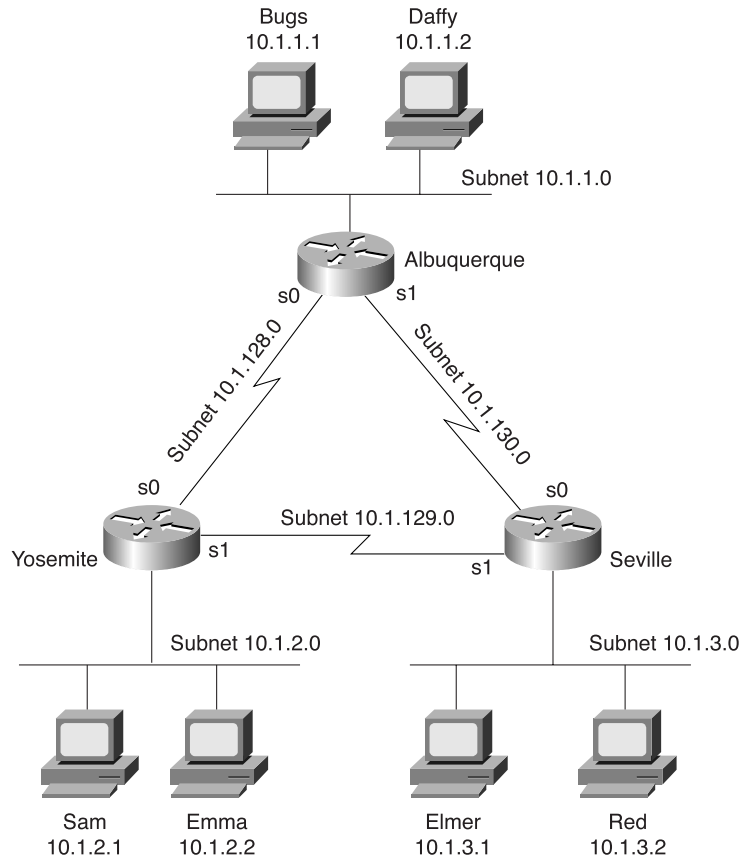
```
interface Ethernet0
ip address 172.16.1.1 255.255.255.0
ip access-group 1

access-list 1 deny host 172.16.3.10
access-list 1 permit any
```

The commands in Example 8-1 are changed based on three factors. First, “out” is the default direction for access lists, so the router would omit the **out** keyword of the **ip access-group** command. Second, the use of a wildcard mask of 0.0.0.0 is the old way to configure an access list to match a specific host's IP address. The new style is to code the **host** keyword in front of the IP address. When you type a wildcard of 0.0.0.0, the router replaces the configuration with the newer **host** keyword. Finally, when you use an IP address and a wildcard mask of 255.255.255.255, the keyword **any** is used to replace both parameters. **any** simply means that any IP address is matched.

The second example is more involved. Figure 8-4, Example 8-3, and Example 8-4 show a basic use of standard IP access lists, with two typical oversights in the first attempt at a complete answer. The criteria for the access lists are as follows:

- Sam is not allowed access to Bugs or Daffy.
- Hosts on the Seville Ethernet are not allowed access to hosts on the Yosemite Ethernet.
- All other combinations are allowed.

Figure 8-4 Network Diagram for Standard Access List Example**Example 8-3** Yosemite Configuration for Standard Access List Example

```
interface serial 0
ip access-group 3
!
access-list 3 deny host 10.1.2.1
access-list 3 permit any
```

Example 8-4 Seville Configuration for Standard Access List Example

```
interface serial 1
ip access-group 4
!
access-list 4 deny 10.1.3.0 0.0.0.255
access-list 4 permit any
```

At first glance, these two access lists seem to perform the desired function. Criterion 1 is met in Yosemite. In Yosemite, the packets from Sam are filtered before leaving S0 using access list 3. Criterion 2 is met in Seville: Packets from 10.1.3.0/24 are filtered before leaving Seville's S1 toward Yosemite, using access list 4. Both routers meet criterion 3: A wildcard **permit any** is used at the end of each access list to override the default, which is to discard all other packets. So, all the criteria appear to be met.

One subtle problem prevents this example from actually meeting the stated goals. If certain links fail, new routes are learned. For example, if the link from Albuquerque to Yosemite fails, Yosemite learns a route to 10.1.1.0/24 through Seville. Packets from Sam, forwarded by Yosemite and destined for hosts in Albuquerque, would leave Yosemite's serial1 interface without being filtered. Similarly, if the link from Albuquerque to Yosemite failed, Seville would route packets through Albuquerque, routing around the access list enabled on Seville.

Example 8-5 illustrates an alternative answer to the stated problem. The access list has been removed from Seville, and all filtering is performed on Yosemite.

Example 8-5 *Yosemite Configuration for Standard Access List Example: Alternative Solution to Example 8-3*

```
interface serial 0
ip access-group 3
!
interface serial 1
ip access-group 3
!
interface ethernet 0
ip access-group 4
!
access-list 3 deny host 10.1.2.1
access-list 3 permit any
!
access-list 4 deny 10.1.3.0 0.0.0.255
access-list 4 permit any
```

The configuration in Example 8-5 solves the problem of the earlier example, but it creates another problem. Example 8-5 denies all traffic that should be denied, but it also denies more traffic than the first of the three criteria says it should! In many cases, the meaning of the criteria for the access lists greatly affects your configuration choices. In this example, the problem of Sam's traffic going through Seville to reach Albuquerque when the link directly to Albuquerque is down is solved. The access list denies traffic from Sam (10.1.2.1) in an outbound access list on both of Yosemite's serial interfaces. However, that also prevents Sam from communicating with anyone outside Yosemite. This does not meet the spirit of the filtering goals, because it filters more than it should. An alternative would be to use the same **access-list 3** logic but use it as an inbound access list on Albuquerque's serial interfaces. However, that achieves the real goal only if there are no other servers in Albuquerque that Sam should be allowed to access. And if that were the case, criterion 1 should be rewritten to say something like "Sam is not allowed to access devices on the Albuquerque Ethernet."

The main point is this: With three simple criteria and three routers, the configuration was simple. However, it is easy to introduce problems that are not obvious.

As shown in Example 8-5, **access-list 4** does an effective job of meeting the second of the three criteria. Because the goal was to stop Seville hosts from communicating with Yosemite's hosts, and because the only LAN hosts off Yosemite are the ones on the local Ethernet, the access list is effective in stopping packets from exiting Ethernet 0.

Extended IP Access Lists

- 35 Configure extended access lists to filter IP traffic.
- 37 Monitor and verify selected access list operations on the router

Extended IP access lists are almost identical to standard IP access lists in their use. The key difference between the two is the variety of fields in the packet that can be compared for matching by extended access lists. To pass the CCNA exam, you must remember all the items that an extended IP access list can check to make a match. As with standard lists, extended access lists are enabled for packets entering or exiting an interface. The list is searched sequentially; the first statement matched stops the search through the list and defines the action to be taken. All these features are true of standard access lists as well. The matching logic, however, is different than that used with standard access lists and makes extended access lists much more complex.

Table 8-5 lists the configuration commands associated with creating extended IP access lists. Table 8-6 lists the associated EXEC commands. Several examples follow the lists of commands.

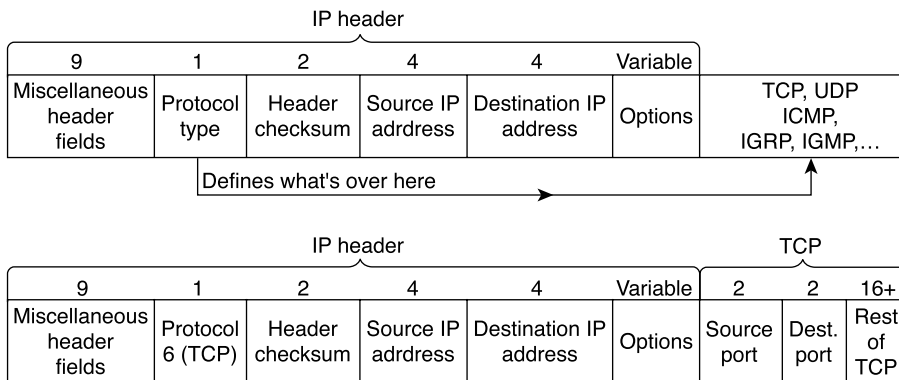
Table 8-5 *Extended IP Access List Configuration Commands*

Command	Configuration Mode and Description
access-list <i>access-list-number</i> [dynamic <i>dynamic-name</i> [timeout <i>minutes</i>]] { deny permit } <i>protocol source source-wildcard destination destination-wildcard</i> [precedence <i>precedence</i>] [tos <i>tos</i>] [log log-input] [time-range <i>time-range-name</i>]	Global command for extended numbered access lists
ip access-group { <i>number</i> <i>name</i> [in out]}	Interface subcommand to enable access lists
access-class <i>number</i> <i>name</i> [in out]	Line subcommand for standard or extended access lists

Table 8-6 *Extended IP Access List EXEC Commands*

Command	Description
show ip interface [type number]	Includes a reference to the access lists enabled on the interface
show access-lists [access-list-number access-list-name]	Shows details of configured access lists for all protocols
show ip access-list [access-list-number access-list-name]	Shows IP access lists

Extended access lists create powerful matching logic by examining many parts of a packet. Figure 8-5 shows several of the fields in the packet headers that can be matched. The top set of headers shows the IP protocol type, which identifies what header follows the IP header. The source and destination IP addresses are also shown. In the second set of headers, an example with a TCP header following the IP header is shown. The TCP source and destination port numbers are listed in the abbreviated TCP header. Table 8-7 provides the complete list of items that can be matched with an IP extended access list.

Figure 8-5 *Extended Access List Matching Options***Table 8-7** *Standard and Extended IP Access Lists: Matching*

Type of Access List	What Can Be Matched
IP standard	Source IP address
	Portions of the source IP address using a wildcard mask

Table 8-7 *Standard and Extended IP Access Lists: Matching (Continued)*

Type of Access List	What Can Be Matched
IP extended	Source IP address
	Portions of the source IP address using a wildcard mask
	Destination IP address
	Portions of the destination IP address using a wildcard mask
	Protocol type (TCP, UDP, ICMP, IGRP, IGMP, and others)
	Source port
	Destination port
	Established—matches all TCP flows except the first
	IP TOS
	IP precedence

A statement is considered to match if all options in the statement match. If one option does not match, the statement is skipped, and the next entry in the list is examined. Table 8-8 provides several sample **access-list** statements.

Table 8-8 *Standard access-list Commands and Logic Explanations*

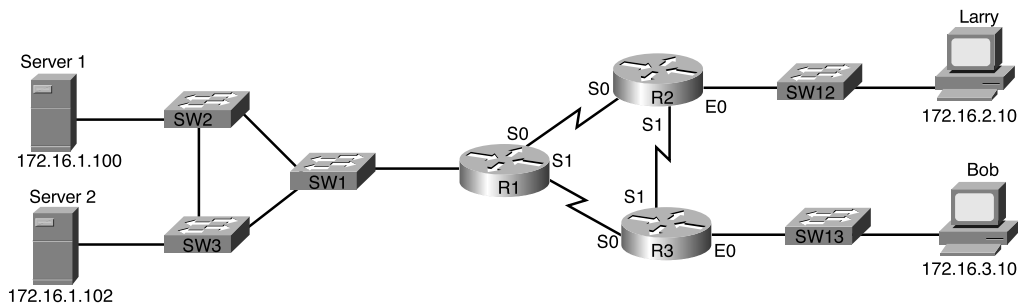
access-list Statement	What It Matches
access-list 101 deny tcp any host 10.1.1.1 eq 23	A packet with any source address. The destination must be 10.1.1.1, with a TCP header and a destination of port 23.
access-list 101 deny tcp any host 10.1.1.1 eq telnet	The same as the preceding function. The telnet keyword is used instead of port 23.
access-list 101 deny udp 1.0.0.0 0.255.255.255 lt 1023 any	A packet with a source in network 1.0.0.0 to any destination, using UDP with a source port less than 1023.
access-list 101 deny udp 1.0.0.0 0.255.255.255 lt 1023 44.1.2.3 0.0.255.255	A packet with a source in network 1.0.0.0 to destinations beginning with 44.1 using UDP with a source port less than 1023.
access-list 101 deny ip 33.1.2.0 0.0.0.255 44.1.2.3 0.0.255.255	A packet with a source in 33.1.2.0/24 to destinations beginning with 44.1.
access-list 101 deny icmp 33.1.2.0 0.0.0.255 44.1.2.3 0.0.255.255 echo	A packet with a source in 33.1.2.0/24 to destinations beginning with 44.1 that are ICMP echo requests and replies.

The sequence of the parameters is very important—and very tricky, in some cases. When checking port numbers, the parameter on the **access-list** command checking the port checks the source port number when placed immediately after the check of the source IP address. Likewise, if the port parameter follows the check of the destination address, the logic matches the destination port. For example, the command **access-list 101 deny tcp any eq telnet any** matches all packets that use TCP and whose source TCP port is 23 (Telnet). Likewise, the command **access-list 101 deny tcp any any eq telnet** matches all packets that use TCP and whose destination TCP port is 23 (Telnet).

Extended IP Access Lists: Example 1

The first example is basic in order to cover the statements’ syntax. In this case, Bob is denied access to all FTP servers on R1’s Ethernet, and Larry is denied access to Server1’s Web server. Figure 8-6 is a reminder of the network topology. In Example 8-6, an access list is created on R1. Example 8-6 shows the configuration on R1.

Figure 8-6 Network Diagram for Extended Access List Example 1



Example 8-6 R1’s Extended Access List: Example 1

```
interface Serial0
ip address 172.16.12.1 255.255.255.0
ip access-group 101 in

interface Serial1
ip address 172.16.13.1 255.255.255.0
ip access-group 101 in

access-list 101 deny tcp host 172.16.3.10 172.16.1.0 0.0.0.255 eq ftp
access-list 101 deny tcp host 172.16.2.10 host 172.16.1.100 eq http
access-list 101 permit ip any any
```

Focusing on the syntax for a moment, there are several new items to review. First, the access list number for extended access lists is from 100 to 199, inclusive. A protocol parameter is the first option after the permit or deny action. When checking for TCP or UDP port numbers, the TCP or UDP protocol must be specified. The **eq** parameter means “equals.” It implies that you are checking the port numbers—in this case, the destination port numbers. You can use the numeric values—or, for the more popular options, a more obvious text version is valid. (If you were to type **eq 80**, the config would show **eq http**.)

The single access list, checking inbound traffic on both serial interfaces of R1, overcomes the rerouting issue that was covered with standard access lists. Because extended access lists can check the packets more exactly, they can perform the exact function much more easily.

An important question can be raised with this first example—and it’s probably covered on the exam: Where should you put access lists? For instance, Example 8-6 is implemented on R1. And, unless a link is down, the access list is checking for packets that will never be matched. The three strategies that Cisco has advanced for quite some time are as follows:

- Place access lists as close as possible to the packet’s source.
- Place more frequently matched statements at the top of the access list to improve performance.
- Achieve both goals without changing what actually gets denied.

So, in this example, the access lists should have been placed on R2 and R3, respectively. And because the goal is to put the most frequently matched statements first, the **permit any** should be first in the list, right? Of course not! The first entry in the list that is matched determines the action. So, changing the **permit any** action at the beginning changes what is actually denied, which goes against the strategy and also goes against what Example 1 is trying to achieve. Example 8-7 defines an access list on R3 that prevents Bob from reaching all FTP servers off R1’s Ethernet. The requirement to prevent Larry from reaching Server1’s Web server is left as an exercise.

Example 8-7 *R3’s Extended Access List Stopping Bob from Reaching FTP Servers Near R1*

```
interface Serial0
ip address 172.16.13.3 255.255.255.0
ip access-group 101 out

interface Serial1
ip address 172.16.12.3 255.255.255.0
ip access-group 101 out

access-list 101 deny tcp host 172.16.3.10 172.16.1.0 0.0.0.255 eq ftp
access-list 101 permit ip any any
```

The access list in Example 8-7 conforms to Cisco's design goals. It is close to the source, being in R3. It does not try to prevent Larry from getting to Server1, because that will presumably be done close to the source, at R2. Omitting checks for Larry should reduce the number of comparisons made by the access list. The **permit any any** at the end of the list needs to be at the end, even if it is matched more than the other statement in the list, because moving it would change the access list's behavior.

Extended IP Access Lists: Example 2

Example 8-8, based on the network shown in Figure 8-4, shows the use of extended IP access lists. Extended access list Example 2 uses the same criteria as standard access list Example 2:

- Sam is not allowed access to Bugs or Daffy.
- Hosts on the Seville Ethernet are not allowed access to hosts on the Yosemite Ethernet.
- All other combinations are allowed.

Example 8-8 Yosemite Configuration for Extended Access List Example 2

```
interface serial 0
ip access-group 110
!
interface serial 1
ip access-group 110
!
access-list 110 deny ip host 10.1.2.1 10.1.1.0 0.0.0.255
access-list 110 deny ip 10.1.2.0 0.0.0.255 10.1.3.0 0.0.0.255
access-list 110 permit ip any any
```

Two important side effects occur with the configuration shown in Example 8-8, compared to the standard access list configuration shown in Examples 8-3 and 8-4. The issue of having packets routed around the access list is taken care of, because the access lists are enabled for output packets on both serial interfaces. Also, most of the packets are filtered at the router nearest the source of the packets, which reduces network overhead. Access lists could have been added at Seville as well, to deny the packets originating from Seville's Ethernet.

Extended IP Access Lists: Example 3

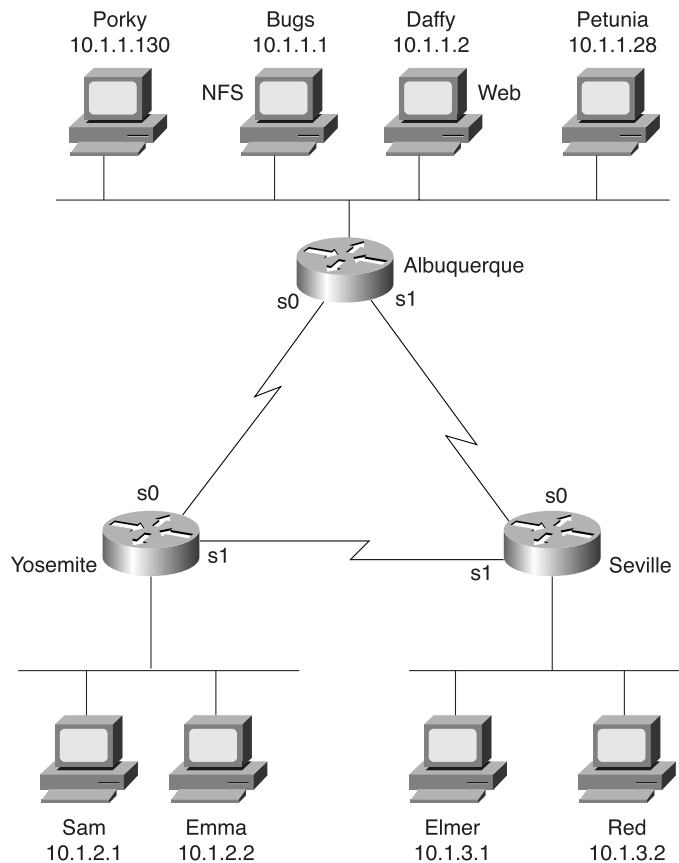
Figure 8-7 shows the network for another example of extended IP access lists.

The filtering criteria for this extended access list example are more complicated:

- The Web server (Daffy) is available to all users.

- UDP-based clients and servers on Bugs are unavailable to hosts whose IP addresses are in the upper half of the valid IP addresses in each subnet. (The subnet mask used is 255.255.255.0.)
- Packets between hosts on the Yosemite Ethernet and the Seville Ethernet are allowed only if packets are routed across the direct serial link.
- Clients Porky and Petunia can connect to all hosts except Red.
- Any other connections are permitted.

Figure 8-7 Network Diagram for Extended Access List Example 3



Examples 8-9, 8-10, and 8-11 show one solution for this third extended access list example.

Example 8-9 Yosemite Configuration for Extended Access List Example 3

```
interface serial 0
ip access-group 110
!
interface serial 1
ip access-group 111
!
! Criterion 1 met with next statement
access-list 110 permit tcp any host 10.1.1.2 eq www
! Criterion 2 met with next statement
access-list 110 deny udp 0.0.0.128 255.255.255.127 host 10.1.1.1
! Criterion 3 met with next statement
access-list 110 deny ip 10.1.2.0 0.0.0.255 10.1.3.0 0.0.0.255
! Criterion 5 met with next statement
access-list 110 permit ip any any
!
! Criterion 1 met with next statement
access-list 111 permit tcp any host 10.1.1.2 eq www
! Criterion 2 met with next statement
access-list 111 deny udp 0.0.0.128 255.255.255.127 host 10.1.1.1
! Criterion 5 met with next statement
access-list 111 permit ip any any
```

Example 8-10 Seville Configuration for Extended Access List Example 3

```
interface serial 0
ip access-group 110
!
interface serial 1
ip access-group 111
!
! Criterion 1 met with next statement
access-list 110 permit tcp any host 10.1.1.2 eq www
! Criterion 2 met with next statement
access-list 110 deny udp 0.0.0.128 255.255.255.127 host 10.1.1.1
! Criterion 3 met with next statement
access-list 110 deny ip 10.1.3.0 0.0.0.255 10.1.2.0 0.0.0.255
! Criterion 5 met with next statement
access-list 110 permit ip any any
!
! Criterion 1 met with next statement
access-list 111 permit tcp any host 10.1.1.2 eq www
! Criterion 2 met with next statement
access-list 111 deny udp 0.0.0.128 255.255.255.127 host 10.1.1.1
! Criterion 5 met with next statement
access-list 111 permit ip any any
```

Example 8-11 *Albuquerque Configuration for Extended Access List Example 3*

```

interface serial 0
ip access-group 112
!
interface serial 1
ip access-group 112
!
! Criterion 4 met with next four statements
access-list 112 deny ip host 10.1.1.130 host 10.1.3.2
access-list 112 deny ip host 10.1.1.28 host 10.1.3.2
access-list 112 permit ip host 10.1.1.130 any
access-list 112 permit ip host 10.1.1.28 any
! Criterion 5 met with next statement
access-list 112 permit ip any any

```

The access lists on Yosemite and Seville are almost identical; each is focused on the first three criteria. List 110 is used as outbound access lists on the Yosemite and Seville links connected to Albuquerque. The first three statements in list 110 in each router complete the first three criteria for this example; the only difference is in the source and destination addresses used in the third statement, which checks for the respective subnet numbers at each site.

Both Yosemite and Seville have a list 111 that is used on the link between the two. Each list 111 on Yosemite and Seville is identical to list 110, except that list 111 is missing one statement. This missing statement (relative to list 110) is the one that meets criterion 3, which says to not filter this traffic from going across the direct serial link. Because list 111 is used on that link, there is no need for the extra statement. The final statement in lists 110 and 111 in Seville and Yosemite provides coverage for the fifth criterion for this example—allowing all other packets to flow.

The second **access-list** statement in lists 110 and 111 on Seville and Yosemite is trickier than you will see on the CCNA exam. This example is representative of the types of nuances you might see on the CCNP and CCIE exams. The mask has only one binary 0 in it, in bit 25 (the first bit in the last byte). The corresponding bit in the address has value 1; in decimal, the address and mask imply addresses whose fourth byte is between 128 and 255, inclusive. Regardless of subnet number, hosts in the upper half of the assignable addresses in each subnet are matched with this combination. (Because the subnet mask is 255.255.255.0, all host addresses in the upper half of the address range are between 128 and 254 in the last octet.)

Two major problems exist when you use extensive detailed criteria for access lists. First, the criteria are open to interpretation. Many people tend to create the lists to match the order in which each point of the criteria are written; no attempt at optimization is made. Finally, it is easy to create the lists in such a way that the criteria are not actually met, as in extended IP access list Example 2.

Example 8-12 shows an alternative solution to the extended access list Example 3 solution that was shown in Examples 8-9, 8-10, and 8-11. All access lists have been removed from Seville and Yosemite, as compared to that earlier solution.

Example 8-12 *Albuquerque Configuration for Extended Access List Example 3: Second Solution*

```
interface serial 0
ip access-group 112
!
interface serial 1
ip access-group 112
!
! Next statement meets criterion 1
access-list 112 permit tcp host 10.1.1.2 eq www any
! Next statement meets criterion 2
access-list 112 deny udp host 10.1.1.1 0.0.0.128 255.255.255.127
! Next statements meet criterion 3
access-list 112 deny ip 10.1.3.0 0.0.0.255 10.1.2.0 0.0.0.255
access-list 112 deny ip 10.1.2.0 0.0.0.255 10.1.3.0 0.0.0.255
! Next statement meets criterion 4
access-list 112 deny ip host 10.1.1.130 host 10.1.3.2
access-list 112 deny ip host 10.1.1.28 host 10.1.3.2

! Next statement meets criterion 5
access-list 112 permit ip any any
```

Several differences exist between the first solution in Examples 8-9, 8-10, and 8-11, and the second solution in Example 8-12. First, all the filtering is performed in Albuquerque. Criterion point 4 is completed more concisely, allowing the **permit all** final statement to let Porky and Petunia talk to other hosts besides Red. Packets are sent by Yosemite and Seville to Albuquerque hosts, as well as packets sent back from servers in Albuquerque to the Albuquerque router, before being filtered. However, the number of these packets will be small, because the filter prevents the client from sending more than the first packet used to connect to the service.

Named IP Access Lists

- 35 Configure standard access lists to filter IP traffic.
- 36 Configure extended access lists to filter IP traffic.
- 37 Monitor and verify selected access list operations on the router.

Named IP access lists allow the same logic to be configured as with numbered standard and extended access lists. As a CCNA, you will need to remember the configuration commands'

syntax differences and also be able to create both numbered and named lists with the same logic. The key differences between numbered and named IP access lists are as follows:

- A name is a more intuitive reminder of a list's function.
- Names allow for more access lists than 99 standard and 100 extended, which is the restriction with numbered access lists.
- Named access lists allow individual statements to be deleted. Numbered lists allow only for the deletion of the entire list. Insertion of the new statement into a named list requires the deletion and re-addition of all statements that should be later in the list than the newly added statement.
- The actual names used must be unique across all named access lists of all protocols and types on an individual router. Names can be duplicated on different routers.

The configuration syntax is very similar between named and numbered IP access lists. The items that can be matched with a numbered standard IP access list are identical to the items that can be matched with a named standard IP access list. Likewise, the items are identical with both numbered and named extended IP access lists.

Two important differences exist between numbered and named access lists. One key difference is that named access lists use a global command, which moves the user into a named IP access list submode under which the matching and permit/deny logic is configured. The other key difference is that when a named matching statement is deleted, only that one statement is deleted. With numbered lists, the deletion of any statement in the list deletes all the statements in the list. (This feature is demonstrated in more detail in an upcoming example.)

Table 8-9 lists the key configuration commands and shows their differences and similarities.

Table 8-9 *Comparison of Named and Numbered IP Access List Configuration Commands*

	Numbered	Named
Commands for matching packets: standard IP ACLs	access-list 1-99 permit deny ...	ip access-list standard name permit deny ... *
Commands for matching packets: extended IP ACLs	access-list 100-199 permit deny ...	ip access-list extended name permit deny ... *
Commands for enabling ACLs	ip access-group 1-99 in out	ip access-group name in out
Commands for enabling ACLs	ip access-group 100-199 in out	ip access-group name in out

*This command is a subcommand of the preceding command.

The word *name* represents a name created by the administrator. This name must be unique among all named access lists of all types in this router. Also, note that because the named list does not imply standard or extended by the value of the list's number, the command explicitly states the type of access list. Also, the ... represents all the matching parameters, which are identical in meaning and syntax when comparing the respective numbered and named IP access lists. Also note that the same command is used to enable the list on an interface for both numbered and named lists.

One difference between the two types of lists is that individual matching statements can be removed from named lists. Example 8-13 shows the configuration mode output when entering the access list used on Albuquerque in access list 112 of Example 8-12, but this time as a named access list instead of a numbered access list. One typo is shown in the original creation of the access list in Example 8-13, with changes made to delete and add the statement shown later in this same example. (The statement that is a typo is **deny ip 10.1.2.0 0.0.0.255 10.2.3.0 0.0.0.255**. It is a typo because there is no subnet 10.2.3.0; the intent was to configure 10.1.3.0 instead.)

Example 8-13 *Named Access List Configuration*

```

conf t
Enter configuration commands, one per line. End with Ctrl-Z.
Router(config)#ip access-list extended barney
Router(config-ext-nacl)#permit tcp host 10.1.1.2 eq www any
Router(config-ext-nacl)#deny udp host 10.1.1.1 0.0.0.128 255.255.255.127
Router(config-ext-nacl)#deny ip 10.1.3.0 0.0.0.255 10.1.2.0 0.0.0.255
! The next statement is purposefully wrong so that the process of changing
! the list can be seen.
Router(config-ext-nacl)#deny ip 10.1.2.0 0.0.0.255 10.2.3.0 0.0.0.255

Router(config-ext-nacl)#deny ip host 10.1.1.130 host 10.1.3.2
Router(config-ext-nacl)#deny ip host 10.1.1.28 host 10.1.3.2
Router(config-ext-nacl)#permit ip any any
Router(config-ext-nacl)#^Z
Router#show running-config
Building configuration...

Current configuration:

.
. (unimportant statements omitted)
.
!
ip access-list extended barney
 permit tcp host 10.1.1.2 eq www any
 deny  udp host 10.1.1.1 0.0.0.128 255.255.255.127
 deny  ip 10.1.3.0 0.0.0.255 10.1.2.0 0.0.0.255
 deny  ip 10.1.2.0 0.0.0.255 10.2.3.0 0.0.0.255
 deny  ip host 10.1.1.130 host 10.1.3.2
 deny  ip host 10.1.1.28 host 10.1.3.2
 permit ip any any

Router#conf t

```

Example 8-13 *Named Access List Configuration (Continued)*

```

Enter configuration commands, one per line. End with Ctrl-Z.
Router(config)#ip access-list extended barney
Router(config-ext-nacl)#no deny ip 10.1.2.0 0.0.0.255 10.2.3.0 0.0.0.255
Router(config-ext-nacl)#^Z
Router#show access-list

Extended IP access list barney
  permit tcp host 10.1.1.2 eq www any
  deny  udp host 10.1.1.1 0.0.0.128 255.255.255.127
  deny  ip 10.1.3.0 0.0.0.255 10.1.2.0 0.0.0.255
  deny  ip host 10.1.1.130 host 10.1.3.2
  deny  ip host 10.1.1.28 host 10.1.3.2
  permit ip any any
Router#conf t
Enter configuration commands, one per line. End with Ctrl-Z.
Router(config)#ip access-list extended barney
Router(config-ext-nacl)#no permit ip any any
Router(config-ext-nacl)#no deny ip host 10.1.1.130 host 10.1.3.2
Router(config-ext-nacl)#no deny ip host 10.1.1.28 host 10.1.3.2
Router(config-ext-nacl)#deny ip 10.1.2.0 0.0.0.255 10.1.3.0 0.0.0.255
Router(config-ext-nacl)#deny ip host 10.1.1.130 host 10.1.3.2
Router(config-ext-nacl)#deny ip host 10.1.1.28 host 10.1.3.2
Router(config-ext-nacl)#permit ip any any
Router(config-ext-nacl)#^Z
Router#show ip access-list

Extended IP access list barney
  permit tcp host 10.1.1.2 eq www any
  deny  udp host 10.1.1.1 0.0.0.128 255.255.255.127
  deny  ip 10.1.3.0 0.0.0.255 10.1.2.0 0.0.0.255
  deny  ip 10.1.2.0 0.0.0.255 10.1.3.0 0.0.0.255
  deny  ip host 10.1.1.130 host 10.1.3.2
  deny  ip host 10.1.1.28 host 10.1.3.2
  permit ip any any

```

If an access list is not configured but is enabled on an interface with the **ip access-group** command, no packets are filtered because of this command. After the access list's first command is configured, Cisco IOS software implements the access list's logic. This is true of IP standard access lists as well as extended and named access lists. Access lists that filter other types of packets follow this same logic.

Controlling vty Access with IP Access Lists

Access into and out of the virtual terminal line (vty) ports of the Cisco IOS software can be controlled by IP access lists. (vty is used for Telnet access to and from the Cisco IOS software.) The inbound case is the more obvious case. For instance, imagine that only hosts in subnet 10.1.1.0/24 are supposed to be capable of Telnetting into any of the Cisco routers in a network.

In such a case, the configuration in Example 8-14 could be used on each router to deny access from IP addresses not in that one subnet.

Example 8-14 vty Access Control Using the **access-class** Command

```
line vty 0 4
 login
 password cisco
 access-class 3 in
!  
! Next command is a global command
access-list 3 permit 10.1.1.0 0.0.0.255
```

The **access-class** command refers to the matching logic in **access-list 3**. The keyword **in** refers to packets that are entering the router when you are trying to Telnet to that router's vtys. The **out** keyword is used both with outbound Telnet from a router and when using the reverse Telnet feature of the Cisco IOS software (which is unlikely to be on the exam). The **out** keyword implies that the packets originated by the Telnet client in the router are checked using the packets' destination address.

IP Access List Summary

To pass the CCNA exam, you must be proficient in using IP access lists. Here are the most important details to recall:

- The order of the list is important.
- All matching parameters must be true before a statement is "matched."
- An implied **deny all** is at the end of the list.

The strategy of choosing the location for access lists is covered in more depth on the CCNP exam than on the CCNA exam. However, it's generally better to filter packets closer to their source, because soon-to-be discarded packets waste less bandwidth than if they are allowed to flow over additional links before being denied.

Be particularly careful of questions relating to existing lists. For example, suppose a question suggests that one more **access-list** command should be added. Simply adding that command places the statement at the end of the list. However, the statement might need to be earlier in the list to accomplish the goal described in the question. Also focus on the differences between named and numbered IP access lists.

Foundation Summary

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

The logic for any access list can be summarized as follows:

- Step 1** The matching parameters of the first **access-list** statement are compared to the packet.
- Step 2** If a match is made, the action defined in this **access-list** statement (permit or deny) is performed.
- Step 3** If a match is not made in Step 2, Steps 1 and 2 are repeated using the next sequential **access-list** statement.
- Step 4** If no match is made with an entry in the access list, the deny action is performed.

Here are some key features of Cisco access lists:

- Packets can be filtered as they enter an interface, before the routing decision.
- Packets can be filtered before they exit an interface, after the routing decision.
- *Deny* is the term used in the Cisco IOS software to imply that the packet will be filtered.
- *Permit* is the term used in the Cisco IOS software to imply that the packet will not be filtered.
- The filtering logic is configured in the access list.
- At the end of every access list is an implied “deny all traffic” statement. Therefore, if a packet does not match any of your **access-list** statements, it is blocked.

Table 8-10 shows several examples of masks, packet source addresses, and addresses in **access-list** commands.

Table 8-10 *Sample Access List Wildcard Masks*

Wildcard Mask	Binary Version of Mask	Description
0.0.0.0	00000000.00000000.00000000.00000000	The entire IP address must match.
0.0.0.255	00000000.00000000.00000000.11111111	Just the first 24 bits must match.
0.0.255.255	00000000.00000000.11111111.11111111	Just the first 16 bits must match.
0.255.255.255	00000000.11111111.11111111.11111111	Just the first 8 bits must match.
255.255.255.255	11111111.11111111.11111111.11111111	Don't even bother to compare; it's automatically considered to match (0 bits need to match).
0.0.15.255	00000000.00000000.00001111.11111111	Just the first 20 bits must match.
0.0.3.255	00000000.00000000.00000011.11111111	Just the first 22 bits must match.
32.48.0.255	00100000.00110000.00000000.11111111	All bits except the 3rd, 11th, 12th, and last 8 must match.

Table 8-11 lists the configuration commands related to standard IP access lists. Table 8-12 lists the related exec commands.

Table 8-11 *Standard IP Access List Configuration Commands*

Command	Configuration Mode and Purpose
access-list <i>access-list-number</i> {deny permit} <i>source</i> [<i>source-wildcard</i>] [log]	Global command for standard numbered access lists
ip access-group { <i>number</i> <i>name</i> [in out]}	Interface subcommand to enable access lists
access-class <i>number</i> <i>name</i> [in out]	Line subcommand for standard or extended access lists

Table 8-12 *Standard IP Access List exec Commands*

Command	Function
show ip interface [<i>type number</i>]	Includes a reference to the access lists enabled on the interface
show access-lists [<i>access-list-number</i> <i>access-list-name</i>]	Shows details of configured access lists for all protocols
show ip access-list [<i>access-list-number</i> <i>access-list-name</i>]	Shows IP access lists

Table 8-13 lists the configuration commands associated with creating extended IP access lists. Table 8-14 lists the associated EXEC commands.

Table 8-13 *Extended IP Access List Configuration Commands*

Command	Configuration Mode and Purpose
access-list <i>access-list-number</i> [dynamic <i>dynamic-name</i> [timeout <i>minutes</i>]] { deny permit } <i>protocol source source-wildcard destination destination-wildcard</i> [precedence <i>precedence</i>] [tos <i>tos</i>] [log log-input] [time-range <i>time-range-name</i>]	Global command for extended numbered access lists
ip access-group { <i>number</i> <i>name</i> [in out]}	Interface subcommand to enable access lists
access-class <i>number</i> <i>name</i> [in out]	Line subcommand for standard or extended access lists

Table 8-14 *Extended IP Access List Configuration Commands*

Command	Function
show ip interface [<i>type number</i>]	Includes a reference to the access lists enabled on the interface
show access-lists [<i>access-list-number</i> <i>access-list-name</i>]	Shows details of configured access lists for all protocols
show ip access-list [<i>access-list-number</i> <i>access-list-name</i>]	Shows IP access lists

The three strategies that Cisco has advanced for quite some time are as follows:

- Place access lists as close as possible to the packet's source.
- Place more frequently matched statements at the top of the access list to improve performance.
- Achieve both goals without changing what actually gets denied.

The key differences between numbered and named IP access lists are as follows:

- A name is a more intuitive reminder of a list's function.
- Names allow for more access lists than 99 standard and 100 extended, which is the restriction using numbered access lists.
- Named access lists allow individual statements to be deleted. Numbered lists only allow for the deletion of the entire list. Insertion of the new statement into a named list requires the deletion and re-addition of all statements that should be later in the list than the newly added statement.
- The actual names used must be unique across all named access lists of all protocols and types on an individual router. Names can be duplicated on different routers.

Table 8-15 lists the key configuration commands and shows their differences and similarities.

Table 8-15 *Comparison of Named and Numbered IP Access List Configuration Commands*

	Numbered	Named
Commands for matching packets: standard IP ACLs	access-list 1-99 permit deny ...	ip access-list standard <i>name</i> permit deny ...*
Commands for matching packets: extended IP ACLs	access-list 100-199 permit deny ...	ip access-list extended <i>name</i> permit deny ...*
Commands for enabling ACLs	ip access-group 1-99 in out	ip access-group <i>name</i> in out
Commands for enabling ACLs	ip access-group 100-199 in out	ip access-group <i>name</i> in out

*This command is a subcommand of the preceding command.

Q&A

As mentioned in Chapter 1, the questions and scenarios in this book are more difficult than what you should experience on the exam. The questions do not attempt to cover more breadth or depth than the exam, but they are designed to make sure that you know the answer. Rather than allowing you to derive the answer from clues hidden in the question, the questions challenge your understanding and recall of the subject. Questions from the “Do I Know This Already?” quiz at the beginning of this chapter are repeated here to ensure that you have mastered this chapter’s topics. Hopefully these questions will help limit the number of exam questions on which you narrow your choices to two options and then guess. Also be sure to use the CD and take the simulated exams.

The answers to these questions can be found in Appendix A.

- 1 Configure a numbered IP access list that stops packets from subnet 134.141.7.0 255.255.255.0 from exiting serial 0 on a router. Allow all other packets.

- 2 Configure an IP access list that allows only packets from subnet 193.7.6.0 255.255.255.0, going to hosts in network 128.1.0.0 and using a Web server in 128.1.0.0, to enter serial 0 on a router.

- 3 How would a user who does not have the enable password find out what access lists have been configured and where they are enabled?

- 4 Configure and enable an IP access list that stops packets from subnet 10.3.4.0/24 from getting out serial interface S0 and that stops packets from 134.141.5.4 from entering S0. Permit all other traffic.

- 5 Configure and enable an IP access list that allows packets from subnet 10.3.4.0/24, to any Web server, to get out serial interface S0. Also allow packets from 134.141.5.4 going to all TCP-based servers using a well-known port to enter serial 0. Deny all other traffic.

- 6 Can standard IP access lists be used to check the source IP address when enabled with the **ip access-group 1 in** command, and can they check the destination IP addresses when using the **ip access-group 1 out** command?

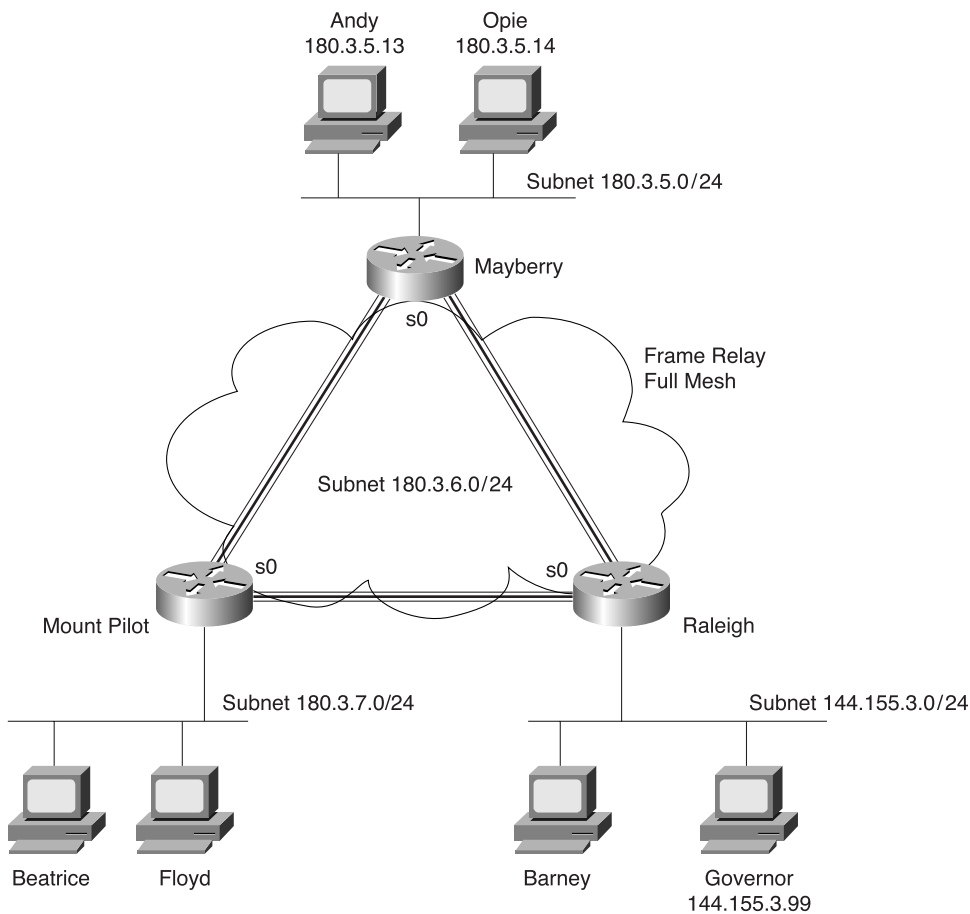
- 7 How many IP extended **access-list** commands are required to check a particular port number on all IP packets?

- 8 True or false: If all IP or IPX **access-list** statements in a particular list define the deny action, the default action is to permit all other packets.

- 9 How many IP access lists of either type can be active on an interface at the same time?

For questions 10 through 12, assume that all parts of the network shown in Figure 8-8 are up and working. IGRP is the IP routing protocol in use. Answer the questions following Example 8-15, which contains an additional configuration in the Mayberry router.

Figure 8-8 Network Diagram for Questions 10 Through 12



Example 8-15 Access List at Mayberry

```
access-list 44 permit 180.3.5.13 0.0.0.0
!
interface serial 0
ip access-group 44
```

10 Describe the types of packets that this filter would discard, and specify at what point they would be discarded.

11 Does the access list in Example 8-15 stop packets from getting to Web server Governor? Why or why not?

12 Referring to Figure 8-8, create and enable access lists so that access to Web server Governor is allowed from hosts at any site and so that no other access to hosts in Raleigh is allowed.

13 Name all the items that a standard IP access list can examine to make a match.

14 Name all the items that an extended IP access list can examine to make a match.

15 True or false: When you use extended IP access lists to restrict vty access, the matching logic is a best match of the list rather than a first match in the list.

-
- 16** In a standard numbered IP access list with three statements, a **no** version of the first statement is issued in configuration mode. Immediately following, another access list configuration command is added for the same access list. How many statements are in the list now, and in what position is the newly added statement?

- 17** In a standard named IP access list with three statements, a **no** version of the first statement is issued in configuration mode. Immediately following, another access list configuration command is added for the same access list. How many statements are in the list now, and in what position is the newly added statement?

- 18** Name all the items that a named standard IP access list can examine to make a match.

- 19** Configure a named IP access list that stops packets from subnet 134.141.7.0 255.255.255.0 from exiting serial 0 on a router. Allow all other packets.

- 20** Configure a named IP access list that allows only packets from subnet 193.7.6.0 255.255.255.0, going to hosts in network 128.1.0.0 and using a Web server in 128.1.0.0, to enter serial 0 on a router.

- 21** List the types of IP access lists (numbered standard, numbered extended, named standard, named extended) that can be enabled to prevent Telnet access into a router. What commands would be used to enable this function, assuming that **access-list 2** was already configured to match the right packets?

- 22** What command lists the IP extended access lists enabled on serial 1 without showing other interfaces?

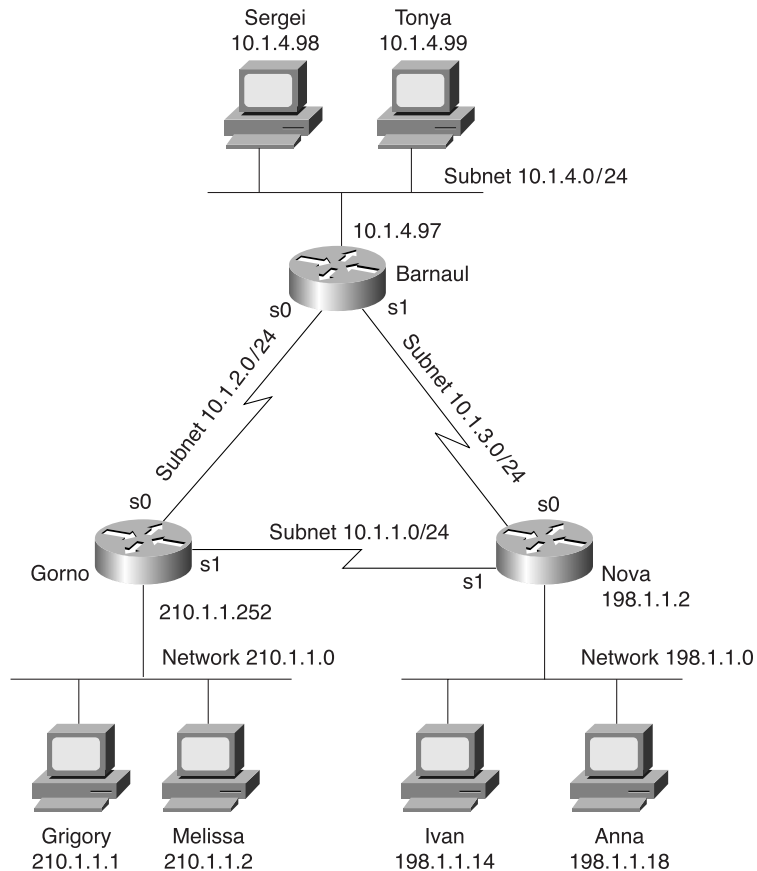
- 23** Name all the items that a named extended IP access list can examine to make a match.

Scenarios

Scenario 8-1: IP Filtering Sample 1

Scenarios 8-1 through 8-3 use Figure 8-9, each with a different set of requirements for filtering. In each case, configure a correct access list for the routers and enable the access list. Place the access list in the router that filters the unneeded packets as quickly as possible—that is, before the packets are sent far from the originator.

Figure 8-9 Network Diagram for IP Filtering Scenarios 8-1, 8-2, and 8-3



The filtering criteria for Scenario 8-1 are as follows:

- 1 Grigory can use the hosts on Nova's Ethernet.
- 2 All other hosts on Gorno (besides Grigory) cannot use the hosts on Nova's Ethernet.
- 3 All other communications are allowed.

Scenario 8-2: IP Filtering Sample 2

Again using the network diagram shown in Figure 8-9, create and enable access lists for a totally different set of requirements. Place the access list in the routers to filter the unneeded packets as quickly as possible—that is, before the packets are sent far from the originator.

The filtering criteria for Scenario 8-2 are as follows:

- 1 Hosts on the Barnaul Ethernet cannot communicate with hosts on the Gorno Ethernet.
- 2 Grigory and Melissa cannot communicate with hosts on the Nova Ethernet.
- 3 Other communications between the Nova Ethernet and the Gorno Ethernet are allowed.
- 4 Sergei (in Barnaul) can communicate only with other hosts in Barnaul.
- 5 Any communication paths not specified are allowed.

Scenario 8-3: IP Filtering Sample 3

Again using the network diagram shown in Figure 8-9, create and enable access lists for a totally different set of requirements. Place the access list in the router that filters the unneeded packets as quickly as possible—that is, before the packets are sent far from the originator.

The filtering criteria for Scenario 8-3 are as follows:

- 1 Grigory and Melissa can access any Web server in Nova.
- 2 Grigory and Melissa cannot access any other servers in Nova using TCP.
- 3 Sergei (Barnaul) can use only the Web services—and no other services—in Nova.
- 4 Hosts in Gorno can communicate with hosts in Nova unless otherwise stated.
- 5 Web clients in Barnaul are not allowed to connect to the Web server in Nova unless specifically mentioned elsewhere in these criteria.
- 6 Any unspecified communication should be disallowed.

Answers to Scenarios

Answers to Scenario 8-1: IP Filtering Sample 1

The solution to fulfilling the criteria stipulated for this access list is straightforward. Simply matching Grigory to permit his traffic and denying packets from 210.1.1.0 is all that is needed for the first two criteria. A **permit all** needs to be explicitly configured at the end of the list.

Example 8-16 provides the solution for this scenario. The access list is enabled on Nova. The problem with list 43 is that if the link from Barnaul to Gorno goes down, and if Gorno learns a route to Barnaul's subnets via Nova, Nova filters all inbound packets from (non-Grigory) Gorno hosts. A better list would be to use an extended access list that matches both the source and the destination addresses. **access-list 143** also is shown in Example 8-16, which avoids the problem seen with **access-list 43**. (**access-list 43** is enabled in the example.)

Example 8-16 *Solution to Scenario 8-1: Nova*

```
access-list 43 permit host 210.1.1.1
access-list 43 deny 210.1.1.0 0.0.0.255
access-list 43 permit any
!
access-list 143 permit ip host 210.1.1.1 198.1.1.0 0.0.0.255
access-list 143 deny ip 210.1.1.0 0.0.0.255 198.1.1.0 0.0.0.255
access-list 143 permit ip any any
!
interface serial 0
ip access-group 43 in
!
interface serial 1
ip access-group 43 in
```

Answers to Scenario 8-2: IP Filtering Sample 2

Many solutions could fulfill the criteria stipulated for this scenario. The solutions provided in Examples 8-17 and 8-18 attempt to filter packets as close to the source of the packet as possible. It is impossible to determine whether your correct solution is better than the one given here without more information about traffic loads and business needs in the network. The comments included in Examples 8-17 and 8-18 provide most of the detailed commentary.

Example 8-17 *Solution to Scenario 8-2: Barnaul Access List*

```
! Next statement meets Criterion 1
access-list 101 deny ip 10.1.4.0 0.0.0.255 210.1.1.0 0.0.0.255
! Next statement meets Criterion 4
access-list 101 deny ip host 10.1.4.98 any
! Criterion 5 met in the next statement
```

continues

Example 8-17 *Solution to Scenario 8-2: Barnaul Access List (Continued)*

```

access-list 101 permit ip any any
interface serial 0
ip access-group 101
!
interface serial 1
ip access-group 101

```

Example 8-18 *Solution to Scenario 8-2: Gorno Access List*

```

! Next statements meet Criterion 2
access-list 101 deny ip host 210.1.1.1 198.1.1.0 0.0.0.255
access-list 101 deny ip host 210.1.1.2 198.1.1.0 0.0.0.255
! Next statement meets Criterion 3, but it's not required, due to the final statement
access-list 101 permit ip 210.1.1.0 0.0.0.255 198.1.1.0 0.0.0.255
access-list 101 permit ip any any
!
interface serial 0
ip access-group 101
!
interface serial 1
ip access-group 101

```

Answers to Scenario 8-3: IP Filtering Sample 3

Many solutions could fulfill the criteria stipulated for this scenario. The solutions provided in Examples 8-19 and 8-20 attempt to filter packets as close to the source of the packet as possible. It is impossible to determine whether your correct solution is better than the one given here without more information about traffic loads and business needs in the network. The comments included in Examples 8-19 and 8-20 provide most of the detailed commentary.

Example 8-19 *Solution to Scenario 8-3: Barnaul Access List*

```

! Next statements meet Criterion 3
access-list 101 permit tcp host 10.1.4.98 198.1.1.0 0.0.0.255 eq www
access-list 101 deny tcp host 10.1.4.98 198.1.1.0.0.0.0.25 lt 1023
! Next statement meets Criterion 5, but it's not really needed
access-list 101 deny ip 10.1.4.0 0.0.0.255 198.1.1.0 0.0.0.255 eq www
! Criterion 6 is met in the default
!
interface serial 0
ip access-group 101
!
interface serial 1
ip access-group 101

```

Example 8-20 *Solution to Scenario 8-3: Gorno Access List*

```
! Next statements meet Criterion 1
access-list 101 permit tcp host 210.1.1.1 198.1.1.0 0.0.0.255 eq www
access-list 101 permit tcp host 210.1.1.2 198.1.1.0 0.0.0.255 eq www
! Next statements meet Criterion 2
access-list 101 deny tcp host 210.1.1.1 198.1.1.0 0.0.0.255 lt 1023
access-list 101 deny tcp host 210.1.1.2 198.1.1.0 0.0.0.255 lt 1023
! Next statement meets Criterion 4
access-list 101 permit ip 210.1.1.0 0.0.0.255 198.1.1.0 0.0.0.255
!Default meets Criterion 6
!
interface serial 0
ip access-group 101
!
interface serial 1
ip access-group 101
```

The default action can be used to shorten the list. For example, in Example 8-19, the commands **access-list 101 deny tcp host 10.1.4.98 198.1.1.0 0.0.0.255 lt 1023** and **access-list 101 deny ip 10.1.4.0 0.0.0.255 198.1.1.0 0.0.0.255 eq www** in access list 101 are not really needed, because the default is to deny these anyway. So, list 101 would perform the same function if it had only one statement in it (**access-list 101 permit tcp host 10.1.4.98 198.1.1.0 0.0.0.255 eq www**).