10

Access Control Lists (ACLs) for the Series 3400cl and Series 6400cl Switches

Contents

Introduction

Feature	Default	Menu	CLI	Web
Numbered ACLs				
Standard ACLs	None	—	10-47	—
Extended ACLs	None	—	10-52	—
Named ACLs		—	10-58	—
Enable or Disable an ACL		—	10-61	—
Display ACL Data	n/a	—	10-62	—
Delete an ACL	n/a	—	10-62	—
Configure an ACL from a TFTP Server	n/a	—	10-72	—
Enable ACL Logging	n/a	—	10-77	—
Show ACL Resources				
Access-List Resources Help				

ACL Applications on Series 3400cl and 6400cl Switches

ACLs can filter traffic from a host, a group of hosts, or from entire subnets. Where it is necessary to apply ACLs to filter traffic from outside a network or subnet, applying ACLs at the edge of the network or subnet removes unwanted traffic as soon as possible, and thus helps to improve system performance. ACLs on the 3400cl/6400cl switches filter inbound traffic only and can rapidly consume switch resources. Also, ACLs, QoS, and Rate-Limiting share the same per-port mask resources on these switches. For these reasons, the best places to apply ACLs on the 3400cl/6400cl switches are on “edge” ports where ACLs are likely to be less complex and resource-intensive than in core network applications where the per-VLAN and inbound/outbound ACL filtering offered by the Series 5300xl switches may be the best ACL solution.

General Application Options

Layer 3 IP filtering with Access Control Lists (ACLs) on the 3400cl/6400cl switches enables you to improve network performance and restrict network use by creating policies for:

- **Switch Management Access:** Permits or denies in-band management access. This includes preventing the use of certain TCP or UDP applications (such as Telnet, SSH, web browser, and SNMP) for transactions between specific source and destination IP addresses.
- **Application Access Security:** Eliminates inbound, unwanted IP, TCP, or UDP traffic by filtering packets where they enter the switch on specific physical ports or trunks.

This chapter describes how to configure, apply, and edit ACLs in ProCurve Series 3400cl and Series 6400cl switches and how to monitor the results of ACL actions.

Notes

Unlike the ProCurve Series 5300xl switches, it is not necessary to enable routing on 3400cl/6400cl switches to support ACL operation.

ACLs can enhance network security by blocking selected IP traffic, and can serve as one aspect of maintaining network security. *However, because ACLs do not provide user or device authentication, or protection from malicious manipulation of data carried in IP packet transmissions, they should not be relied upon for a complete security solution.*

ACLs in the 3400cl/6400cl switches do not screen non-IP traffic such as AppleTalk and IPX.

For ACL filtering to take effect, configure an ACL and then assign it to the inbound traffic on a statically configured port or trunk.

Table 10-1. Comprehensive Command Summary

Action	Command	Page
Configuring Standard (Numbered) ACLs	ProCurve(config)# [no] access-list < 1-99 > < deny | permit > < any | host <*src-ip-addr* > | *src-ip-address/mask* > [1] [log][2]	10-47
Configuring Extended (Numbered) ACLs	ProCurve(config)# [no] access-list <100-199> < deny | permit > ip < any | host <*src-ip-addr* > | *src-ip-address*/mask > [1] [log][2]	10-52
	ProCurve(config)# [no] access-list < 100-199 > < deny | permit > < tcp | udp > < any | host <*src-ip-addr* > | *src-ip-address*/mask > [1] [eq < *src-port tcp/udp-id* >] < any | host <*dest-ip-addr* > | *dest-ip-address*/mask > [1] [eq < *dest-port tcp/udp-id* >] [log][2]	10-52
Configuring Standard (Named) ACLs	ProCurve(config)# [no] ip access-list standard < *name-str* | 1-99 >	10-58
	ProCurve(config-std-nacl)# < deny | permit > < any | host <*src-ip-addr* > | *src-ip-address/mask* > [1] [log] [2]	10-58
Configuring Extended (Named) ACLs	ProCurve(config)# [no] ip access-list extended < *name-str* | 100-199 >	10-58
	ProCurve(config-std-nacl)# < deny | permit > ip < any | host <*src-ip-addr* > | *src-ip-address*/mask > [1] < any | host <*dest-ip-addr* > | *dest-ip-address*/mask > [1] [log][2]	10-58
	ProCurve(config-std-nacl)# < deny | permit > < tcp | udp > < any | host <*src-ip-addr* > | *src-ip-address*/mask > [1] [eq < *tcp/udp-port-#* | *well-known-port-name* >] < any | host <*dest-ip-addr* > | *dest-ip-address*/mask > [1] [eq < *tcp/udp-port-#* | *well-known-port-name* >] [log][2]	10-58
Enabling or Disabling an ACL	ProCurve(config)# [no] interface < *port-list* > access-group < *name-str* | 1-99 | 100-199 > in	10-61
Deleting an ACL from the Switch	ProCurve(config)# no ip access-list < standard < *name-str* | 1-99 >> in ProCurve(config)# no ip access-list < extended < *name-str* | 100 -199 >> in	10-62

[1] The mask can be in either dotted-decimal notation (such as 0.0.15.255) or CIDR notation (such as /20).
[2] The [log] function applies only to "deny" ACLs, and generates a message only when there is a "deny" match.

Action	Command	Page
Displaying ACL Data	ProCurve(config)# show access-list	10-62
	ProCurve(config)# show access-list [*acl-name-string*]	
	ProCurve(config)# show access-list config	
	ProCurve(config)# show access-list ports < *port-list* >	
	ProCurve(config)# show access-list resources	
	ProCurve(config)# access-list resources help	
	ProCurve(config)# show config	
	ProCurve(config)# show running	

Terminology

3400cl/6400cl Switches: An all-inclusive reference to the ProCurve 3400cl and 6400cl switches.

Access Control Entry (ACE): An ACE is a policy consisting of criteria and an action to take (permit or deny) on a packet if it meets the criteria. The elements composing the criteria include:

- Source IP address and mask (standard and extended ACLs)
- Destination IP address and mask (extended ACLs only)
- TCP or UDP application port numbers (optional, extended ACLs only)

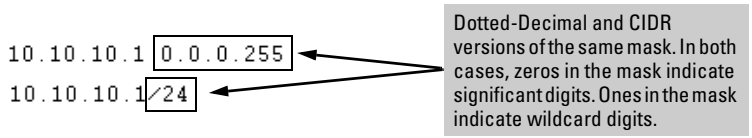
Access Control List (ACL): A list (or set) consisting of one or more explicitly configured Access Control Entries (ACEs) and terminating with an implicit "deny" default which drops any packets that do not have a match with any explicit ACE in the named ACL. The two classes of ACLs are "standard" and "extended". See "Standard ACL" and "Extended ACL".

ACE: See "Access Control Entry".

ACL: See "Access Control List".

ACL ID: A number or alphanumeric string used to identify an ACL. A *standard* ACL ID can have either a number from 1 to 99 or an alphanumeric string. An *extended* ACL ID can have either a number from 100 to 199 or an alphanumeric string.

ACL Mask: Follows an IP address (source or destination) listed in an ACE to specify either a subnet or a group of devices. Defines which bits in a packet's corresponding IP addressing must exactly match the IP addressing in the ACE, and which bits need not match (wildcards). For example:



As shown above, zeros in an ACL mask specify an exact match requirement for IP addresses, and ones specify a wildcard. In this example, a matching IP address would be any address in the range 10.10.10.1-255. (See also "How an ACE Uses a Mask To Screen Packets for Matches" on page 10-32, and Per-Port Mask on page 10-9.)

DA: The acronym for *Destination IP Address*. In an IP packet, this is the destination IP address carried in the header, and identifies the destination intended by the packet's originator. In an extended ACE, this is the second of two IP addresses required by the ACE to determine whether there is a match between a packet and the ACE. See also "SA".

Deny: An ACE configured with this action causes the switch to drop an inbound packet for which there is a match within an applicable ACL. As an option, you can configure the switch to generate a logging output to a Syslog server and a console session.)

Extended ACL: This type of Access Control List uses layer-3 IP criteria composed of source and destination IP addresses and (optionally) TCP or UDP port criteria to determine whether there is a match with an IP packet. You can apply extended ACLs to either inbound or outbound routed traffic and to any inbound switched or routed traffic with a DA belonging to the switch itself. Extended ACLs require an identification number (ID) in the range of 100 - 199 or an alphanumeric name.

Implicit Deny: If the switch finds no matches between an inbound packet and the configured criteria in an applicable ACL, then the switch denies (drops) the packet with an implicit "deny IP any" operation. You can preempt the implicit "deny IP any" in a given ACL by configuring **permit any** (standard) or **permit IP any any** (extended) as the last explicit ACE in the ACL. Doing so permits an inbound packet that is not explicitly permitted or denied by other ACEs configured sequentially earlier in the ACL. Unless otherwise noted, "implicit deny IP any" refers to the "deny" action enforced by both standard and extended ACLs.

Inbound Traffic: For the purpose of defining where the switch applies ACLs to filter traffic, inbound traffic is any IP packet that:

- *Enters the switch through a physical port.*
- Has a destination IP address (DA) that meets either of these criteria:
  - The packet's DA is for an external device.
  - The packet's DA is for an IP address configured on the switch itself. (This increases your options for protecting the switch from unauthorized management access.)

Because ACLs are assigned to physical ports or port trunks, an ACL that filters inbound traffic on a particular port or trunk examines packets meeting the above criteria that enter the switch through that port or trunk.

Outbound Traffic: This is any traffic *leaving the switch* through a physical port or trunk. The switch does not apply ACLs to outbound traffic or internally where routed traffic moves between VLANs. That is, ACL operation is not affected by enabling or disabling routing on the switch. (Refer also to "ACL Inbound Application Points" on page 10-10.)

Permit: An ACE configured with this action allows a port or trunk to permit an inbound packet for which there is a match within an applicable ACL.

Per-Port Mask: An internally applied template for all ACL and IGMP configurations. The significance of per-port masks is that a maximum of 8 masks are available (per-port) for ACL (and IGMP) use.

```
ProCurve(config)# show access-list resources
QoS/ACL Resource Usage

 Port | Rules Available | ACL Masks Available
------+-----------------+-------------------
   1  | 120             |  8
   2  | 120             |  8
   3  | 120             |  8
   .     .                 .
   .     .                 .
   .     .                 .
```

Figure 10-1. Example of Per-Port Mask Allocation in the Default Configuration

For more information, refer to "Planning an ACL Application on a Series 3400cl or Series 6400cl Switch" on page 10-17. See also "ACL Mask" on page 10-8.

SA: The acronym for *Source IP Address*. In an IP packet, this is the source IP address carried in the IP header, and identifies the packet's sender. In an extended ACE, this is the first of two IP addresses used by the ACE to determine whether there is a match between a packet and the ACE. See also "DA".

Standard ACL: This type of Access Control List uses layer-3 IP criteria of source IP address to determine whether there is a match with an inbound IP packet. You can apply a standard ACL to inbound traffic on a port or trunk, including any inbound traffic with a DA belonging to the switch itself. Standard ACLs require an identification number (ID) in the range of 1 - 99 or an alphanumeric name.

Wildcard: The part of a mask that indicates the bits in a packet's IP addressing that do not need to match the corresponding bits specified in an ACL. See also **ACL Mask** on page 10-8.

Overview

Types of IP ACLs

Standard ACL: Use a standard ACL when you need to permit or deny traffic based on source IP address. Standard ACLs are also useful when you need to quickly control a performance problem by limiting traffic from a subnet, group of devices, or a single device. (This can block all inbound IP traffic from the configured source, but does not block traffic from other sources within the network.) This ACL type uses a numeric ID of 1 through 99 or an alphanumeric ID string. You can specify a single host, a finite group of hosts, or any host.

Extended ACL: Use extended ACLs whenever simple IP source address restrictions do not provide the breadth of traffic selection criteria you want for a port or trunk. Extended ACLs allow use of the following criteria:

- Source and destination IP addresses
- TCP application criteria
- UDP application criteria

ACL Inbound Application Points

You can apply ACL filtering to IP traffic inbound on a physical port or static trunk with a destination (DA):

- On another device. (ACLs are not supported on dynamic LACP trunks.)
- On the switch itself. In figure 10-2, below, this would be any of the IP addresses shown in VLANs "A", "B", and "C" on the switch. (IP routing need not be enabled.)

The switch can apply ACL filtering to traffic *entering the switch* on ports and/or trunks configured to apply ACL filters. For example, in figure 10-2 you would assign an inbound ACL on port 1 to filter a packet from the workstation 10.28.10.5 to the server at 10.28.20.99. Note that all ACL filtering is performed on the inbound port or trunk. Routing may be enabled or disabled on the switch, and any permitted inbound traffic may have any valid destination.

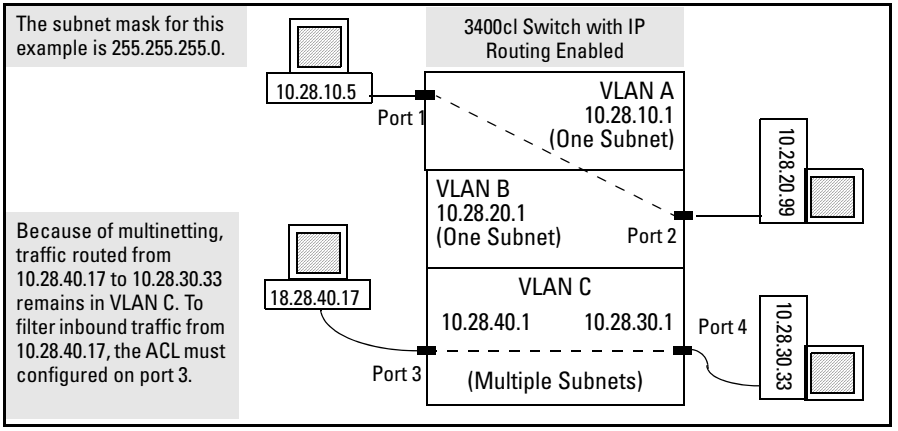


Figure 10-2. Example of Filter Applications

Features Common to All ACLs

- On any port or static trunk you can apply one ACL to inbound traffic.
- Any ACL can have multiple entries (ACEs).
- You can apply any one ACL to multiple ports and trunks.
- A source or destination IP address and a mask, together, can define a single host, a range of hosts, or all hosts.
- Before changing the content of an ACL assigned to one or more ports or trunks, you must first remove the ACL from those ports or trunks.
- Every standard ACL includes an implied "**deny any**" as the last entry, and every extended ACL includes an implied "**deny IP any any**" as the last entry. The switch applies this action to any packets that do not match other criteria in the ACL.
- In any ACL, you can apply an ACL log function to ACEs that have a "deny" action. The logging occurs when there is a match on a "deny" ACE. (The switch sends ACL logging output to Syslog and, optionally, to a console session.)
- Standard and Extended ACL features cannot be combined in one ACL.

You can configure ACLs using either the CLI or a text editor. The text-editor method is recommended when you plan to create or modify an ACL that has more entries than you can easily enter or edit using the CLI alone. Refer to "Editing ACLs and Creating an ACL Offline" on page 10-69.

General Steps for Planning and Configuring ACLs

1. Identify the traffic type to filter. Options include:
   - Any inbound IP traffic
   - Inbound TCP traffic only
   - Inbound UDP traffic only
2. The SA and/or the DA of inbound traffic you want to permit or deny.
3. Determine the best points at which to apply specific ACL controls. For example, you can improve network performance by filtering unwanted traffic at the edge of the network instead of in the core.
4. Design the ACLs for the selected control points. Where you are using explicit "deny" ACEs, you can optionally use the ACL logging feature to help verify that the switch is denying unwanted packets where intended. Remember that excessive ACL logging activity can degrade the switch's performance. (Refer to "Enable ACL "Deny" Logging" on page 10-75.)
5. Create the ACLs in the selected switches.
6. Assign the ACLs to filter the inbound traffic on ports and/or static trunk interfaces configured on the switch.
7. Test for desired results.

For more details on ACL planning considerations, refer to "Planning an ACL Application on a Series 3400cl or Series 6400cl Switch" on page 10-17.

Caution Regarding the Use of Source Routing

Source routing is enabled by default on the switch and can be used to override ACLs. For this reason, if you are using ACLs to enhance network security, the recommended action is to use the **no ip source-route** command to disable source routing on the switch. (If source routing is disabled in the running-config file, the **show running** command includes "**no ip source-route**" in the running-config file listing.)

ACL Operation

Introduction

An ACL is a list of one or more Access Control Entries (ACEs), where each ACE consists of a matching criteria and an action (permit or deny). An ACL applies only to the switch in which it is configured. ACLs operate on assigned ports and static trunks, and filter these traffic types:

- Traffic entering the switch. (Note that ACLs do not screen traffic at any internal point.where traffic moves between VLANs or subnets within the switch; only on inbound ports and static trunks. Refer to "ACL Inbound Application Points" on page 10-10.)
- Switched or routed traffic entering the switch and having an IP address on the switch as the destination

You can apply one inbound ACL to each port and static trunk configured on the switch. The complete range of options per interface includes:

- **No ACL** assigned. (In this case, all traffic entering the switch on the interface does so without any ACL filtering, which is the default.)
- **One ACL** assigned to filter the inbound traffic entering the switch on the interface.
- **Multiple Assignments for the same ACL.** (The switch allows one ACL assignment to an interface, but you can assign the same ACL to multiple interfaces.)

Note

On a given port or trunk, after you assign an ACL, the default action is to deny any traffic that is not specifically permitted by the ACL. (This applies only to the inbound traffic flow filtered by the ACL.)

The Packet-Filtering Process

Sequential Comparison and Action. When the switch uses an ACL to filter a packet, it sequentially compares each ACE's filtering criteria to the corresponding data in the packet until it finds a match.

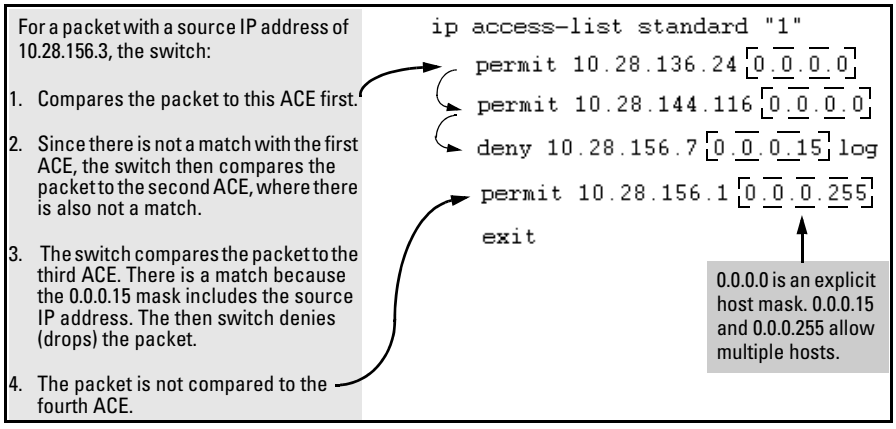


Figure 10-3. Example of Sequential Comparison

That is, the switch tries the first ACE in the list. If there is not a match, it tries the second ACE, and so on. When a match is found, the switch invokes the configured action for that entry (permit or drop the packet) and no further comparisons of the packet are made with the remaining ACEs in the ACL. This means that when the switch finds an ACE whose criteria matches a packet, it invokes the action configured for that ACE, and any remaining ACEs in the ACL are ignored. *Because of this sequential processing, successfully implementing an ACL depends in part on configuring ACEs in the correct order for the overall policy you want the ACL to enforce.*

Implicit Deny. If a packet does not have a match with the criteria in any of the ACEs in the ACL, the switch denies (drops) the packet. (This is termed *implicit deny.*) If you need to override the implicit deny so that any packet that does not have a match will be permitted, then you can enter **permit any** as the last ACE in the ACL. This directs the switch to permit (forward) any packets that do not have a match with any earlier ACE listed in the ACL, and prevents these packets from being filtered by the implicit deny.

Note on Implicit Deny

For ACLs configured to filter inbound packets, note that Implicit Deny filters *any packets, including those with a DA specifying the switch itself*. This operation helps to prevent management access from unauthorized IP sources.

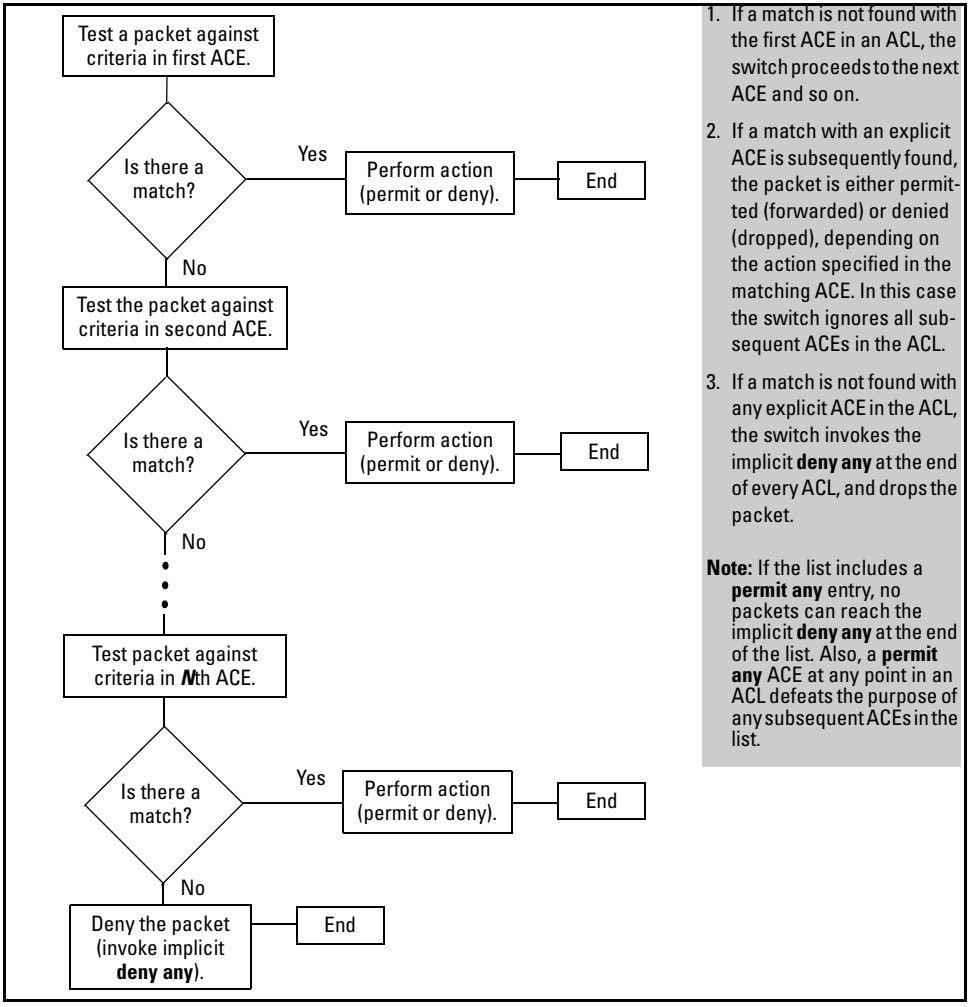


Figure 10-4. The Packet-Filtering Process in an ACL with *N* Entries (ACEs)

Note

The order in which an ACE occurs in an ACL is significant. For example, if an ACL contains six ACEs, but the first ACE is a "permit IP any", then the ACL permits all IP traffic, and the remaining ACEs in the list do not apply, even if they specify criteria that would make a match with any of the traffic permitted by the first ACE.

For example, suppose you want to configure an ACL on the switch (with an ID of "100") to invoke these policies:

1. Permit all inbound traffic on port 12 sent from IP address 11.11.11.42.
2. Deny *only* the inbound Telnet traffic sent from IP address 11.11.11.101.
3. Permit *only* inbound Telnet traffic sent from IP address 11.11.11.33.
4. Deny *all other* inbound traffic on port 12.

The following ACL model, when assigned to inbound filtering on port 12, supports the above case:

```
ProCurve(config)# show access-list config

 ip access-list extended "100"
1 permit ip 11.11.11.42 0.0.0.0 0.0.0.0 255.255.255.255
2 deny tcp 11.11.11.101 0.0.0.0 0.0.0.0 255.255.255.255 eq 23
3 permit ip 11.11.11.101 0.0.0.0 0.0.0.0 255.255.255.255
4 permit tcp 11.11.11.33 0.0.0.0 0.0.0.0 255.255.255.255 eq 23
5  < implicit deny IP any >

ProCurve(config)# access-group 100 in
```

1. **Permits** IP traffic inbound from source address 11.11.11.42. Packets matching this criterion are permitted and will not be compared to any later ACE in the list. Packets not matching this criterion will be compared to the next entry in the list.
2. **Denies** Telnet traffic from source address 11.11.11.101. Packets matching this criterion are dropped and are not compared to later criteria in the list. Packets not matching this criterion are compared to the next entry in the list.
3. **Permits** any IP traffic from source address 11.11.11.101. Any packets matching this criterion will be permitted and will not be compared to any later criteria in the list. Because this entry comes after the entry blocking Telnet traffic from this same address, there will not be any Telnet packets to compare with this entry; they have already been dropped as a result of matching the preceding entry.
4. **Permits** Telnet traffic from source address 11.11.11.33. Packets matching this criterion are permitted and are not compared to any later criteria in the list. Packets not matching this criterion are compared to the next entry in the list.
5. This entry does not appear in an actual ACL, but is implicit as the last entry in every ACL. Any inbound packets on port 12 that do not match any of the criteria in the ACL's preceding entries will be denied (dropped).

Figure 10-5. Example of How an ACL Filters Packets

It is important to remember that this ACL (and all ACLs) include an implicit **deny any**. That is, inbound IP packets (including switched packets having the switch as the destination IP address) that the ACL does not *explicitly* permit or deny will be *implicitly* denied, and therefore dropped. You can preempt the implicit deny by inserting a "permit IP any" at the end of an ACL, but this solution does not apply in the preceding example, where the intention is for the switch to allow only explicitly permitted packets inbound on port 12.

Overriding the Implicit "Deny Any". If you want an ACL to permit any inbound packets that are not explicitly denied by other entries in the ACL, you can do so by configuring a **permit any** entry as the last entry in the ACL. Doing so permits any packet not explicitly denied by earlier entries. (On extended ACLs, you must configure **permit ip any any**.)

Planning an ACL Application on a Series 3400cl or Series 6400cl Switch

Before creating and implementing ACLs, you should understand the Series 3400cl and Series 6400cl switch resources available per-port to support ACL operation, define the policies you want your ACLs to enforce, and understand how your ACLs will impact your network users.

Switch Resource Usage

ACLs, IGMP, QoS, and Rate Limiting share certain 3400cl/6400cl switch per-port resources and load these resources in ways that require more careful attention to per-port resource usage when planning a configuration using these features. Otherwise, there is an increased possibility of fully consuming some port resources, which means that at some point the switch would not support further ACL, QoS, and/or Rate-Limiting configurations on one or more ports (and/or IGMP on the switch). This section describes resource planning for ACLs on a 3400cl or 6400cl switch. For QoS resource planning, refer to chapter 8, "Quality of Service (QoS): Managing Bandwidth More Effectively". For Rate-Limiting resource planning, refer to the "Rate Limiting" section in the chapter titled "Port Traffic Controls" of the *Management and Configuration Guide* for your switch.

Prioritizing and Monitoring ACL, IGMP, QoS, and Rate Limiting Feature Usage

If you want to configure ACLs and either QoS or Rate-Limiting (or both) on the same 3400cl or 6400cl port(s), plan and implement your per-port configuration in descending order of feature importance. This will help to ensure that the most important features are configured first on any given port. Also, if insufficient resources become a problem, this approach can help you recognize how to distribute the desired feature implementations across multiple switches to achieve your objectives.

Note

ACLs on the Series 3400cl and Series 6400cl switches are applied per-port. Except for the source-port classifier, QoS on 3400cl/6400cl switches is applied across either all physical interfaces on the switch or across all physical interfaces on a specified VLAN. This means that in most cases a QoS configuration applies to multiple ports while an ACL configuration applies only to specifically designated ports.

Adding ACLs consumes per-port ACL mask resources rapidly. If ACLs are more important on particular 3400cl or 6400cl switch ports than IGMP, then you should plan and configure your ACL resource usage first for those ports, then give attention to configuration of IGMP. If insufficient resources remain for IGMP, try applying IGMP on other switches.

ACL Resource Usage and Monitoring

ACL configurations on the 3400cl/6400cl switches use internal rule and mask resources on a per-port basis. Per-Port rule and mask usage is reserved as shown below:

Feature	Maximum Internal Masks Available Per-Port	Maximum Internal Rules Available Per-Port
ACLs and IGMP*	8 ACL Masks*	120 maximum

*Enabling IGMP on one or more VLANs consumes one per-port ACL mask on all ports. If all per-port ACL masks are used up on any port in the switch, IGMP cannot be configured. If all rules are used, but at least one mask remains, IGMP can be configured.

The switch consumes per-port (internal) rule and mask resources required by the ACEs in an ACL when you apply the ACL to one or more port and/or static trunk interfaces.

Standard ACLs:

- Each ACE, including the implicit **deny any** ACE in a standard ACL, uses one port rule.
- Contiguous ACE entries with the same subnet mask use the same port mask. Contiguous ACE entries with different subnet masks use one port mask per entry. To conserve ACL mask resources, group ACEs with identical subnet masks together. For example:

Table 10-2. Minimizing Per-Port Mask Usage

Contiguous ACEs with the Same Subnet Mask		Contiguous ACEs with Different Subnet Masks	
The ACEs in this sequence use **two** port masks because entries with identical subnet masks are contiguous. This method optimizes the capacity of an ACL to accept ACEs requiring different port masks because it minimizes port mask usage.		This sequence uses the same entries as the column to the left, but each consecutive entry has a subnet mask that differs from its predecessor, and requires **four** port masks. This method of ordering ACEs unnecessarily consumes port masks and reduces the capacity of an ACL to accept ACEs requiring different port masks.	
15.28.247.1/24	(15.28.247.1 255.255.255.0)	15.28.247.1/24	(15.28.247.1 255.255.255.0)
15.28.253.1/24	(15.28.253.1 255.255.255.0)	10.0.8.0/32	(10.0.8.0 0.0.0.0)
10.0.8.0/32	(10.0.8.0 0.0.0.0)	15.28.253.1/24	(15.28.253.1 255.255.255.0)
10.0.8.105/32	(10.0.8.0 0.0.0.0)	10.0.8.105/32	(10.0.8.0 0.0.0.0)

- An ACL with no ACEs except a **permit any** or a **deny any** uses only one rule and one mask because the IP address and subnet mask are duplicates of the IP address and subnet mask used for the implicit **deny any** ACE that the switch automatically includes at the end of each ACL.

Table 10-3 on page 10-21 summarizes switch use of resources to support ACES.

Extended ACLs:

- Each ACE, including the implicit **deny ip any any** ACE in an extended ACL uses one port rule.
- Contiguous ACE entries with the same subnet mask and the same IP or TCP/UDP protocol applications use the same port mask. Contiguous ACE entries with different subnet masks or different IP-TCP/UDP applications use one port mask per entry. To conserve ACL mask resources, group ACEs with identical subnet masks and IP or TCP/UDP applications together. (The effect of this grouping is the same as above for the standard ACLs, but with more elements to consider.)
- An extended ACL with no ACEs except a **permit ip any any** or **deny ip any any** uses one rule and one mask. This is because the IP address and subnet mask are duplicates of the IP address and subnet mask used for the implicit **deny ip any any** ACE that the switch automatically includes at the end of every ACL.

Table 10-3. ACL Rule and Mask Resource Usage

ACE Type	Per-Port Rule Usage	Per-Port Masks Usage
Standard ACLs		
Implicit **deny any** (automatically included in any standard ACL, but not displayed by **show access-list** < *acl-#* > command).	1	1
First ACE entered	1	1
Next ACE entered with same ACL mask[1]	1	0
Next ACE entered with a different ACL mask[1]	1	1
Closing ACL with a deny any or permit any ACE having the same ACL mask as the preceding ACE	0	0
Closing ACL with a deny any or permit any ACE having a different ACL mask than the preceding ACE	1	1
Extended ACLs		
Implicit **deny ip an any** (automatically included in any standard ACL, but not displayed by **show access-list** < *acl-#* > command).	1	1
First ACE entered	1	1
Next ACE entered with same SA/DA ACL mask and same IP or TCP/UDP protocols specified[2]	1	0
Next ACE entered with any of the following differences from preceding ACE in the list: – Different SA or DA ACL mask – Different protocol (IP as opposed to TCP/UDP) specified in either the SA or DA[3]	1	1
Closing an ACL with a **deny ip any any** or **permit ip any any** ACE preceded by an IP ACE with the same SA and DA ACL masks	0	0
Closing an ACL with a **deny ip any any** or **permit ip any any** ACE preceded by an IP ACE with different SA and/or DA ACL masks	1	1

[1] In a given standard ACL, consecutive ACEs must have identical ACL masks in their SA entries to avoid using a separate per-port mask for each ACE. In a given standard ACL, If two ACEs having identical SA ACL masks are separated by an ACE with a different SA ACL mask, then three per-port masks are used instead of two; one for each sequential change in SA ACL masks. Thus, you can conserve per-port resources by grouping SA entries with the same ACL mask together.

[2] In a given extended ACL, consecutive ACEs must have the same SA and DA ACL mask and the same protocol application (IP as opposed to TCP/UDP) to avoid using a separate per-port mask for each ACE. If consecutive ACEs have different SA or DA ACL masks, or different protocol applications, then each such ACE consumes a separate per-port mask.

[3] TCP and UDP are the same for the purpose of determining per-port mask use. Also, actual TCP or UDP port numbers can vary between ACEs without affecting per-port mask usage. However, if one ACE specifies a TCP/UDP source port and another does not, another per-port mask will be used.

The following two CLI commands are unique to the 3400cl/6400cl switches and are useful for planning and monitoring rule and mask usage in an ACL configuration.

Syntax: access-list resources help

Provides a quick reference on how ACL, QoS and Rate-Limiting use rule resources and how ACL uses mask resources for each configuration option. Includes most of the information in table 10-3, plus an ACL usage summary.

Syntax: show access-list resources

Shows the number of rules and ACL masks currently available on each port. This command is useful for verifying rule and ACL mask availability as you proceed with configuring ACL, IGMP, QoS, and/or Rate-Limiting features available on the switch.

Managing ACL Resource Consumption

As shown in table 10-3, changes in IP subnet masks or changes in IP or TCP/UDP applications among consecutive ACEs in an assigned ACL can rapidly consume per-port mask resources. Also, in almost all cases, adding a new ACE to an ACL consumes one per-port rule. An extensive ACL configuration can fully subscribe the 120 rule resources available on one or more ports, especially when QoS and Rate-Limiting are also configured on the switch. (Configuring IGMP uses one per-port ACL mask, but does not use any per-port rules.) However, a relatively short ACL can fully subscribe the eight mask resources available on one or more ports. (The switch allows one ACL per-port.)

Oversubscribing Available Resources

If a given ACL requires more mask or rule resources on a port than are available, then the switch cannot apply the ACL to *any* of the interfaces specified for that ACL. In this case, the **access-group** command fails and the CLI displays the following:

- In the CLI:

  ```
  Unable to apply access control list.
  ```

- In the Event Log (and in a Syslog server, if configured on the switch):

  ```
  ACL: unable to apply ACL <acl-#> to port <port-#>, failed
  to add entry < # >
  ```

  (Note that <*port-#*> is the first port in the assignment command that was unable to support the ACL.)

Troubleshooting a Shortage of Per-Port Resources

As noted above, a lack of available per-port rules can be caused by a combination of ACL, IGMP, QoS, and Rate-Limiting applications. A lack of available ACL masks is caused by configuring an ACL to oversubscribe the number of per-port masks available for ACLs. (Also, note that enabling IGMP on a VLAN consumes one ACL mask per-port for all ports on the switch, leaving seven available per-port masks for ACL applications.)

Do the following to determine how to change resource usage to allow the ACL you want to configure:

1. Use the **show access-list resources** command to identify the port(s) on which there are insufficient rule resources. For example, figure 10-6 includes ports that can be the source of problems due to rule consumption by policies configured earlier:

```
ProCurve(config)# show qos resources
QoS/ACL Resource Usage

 Port | Rules Available | ACL Masks Available
------+-----------------+--------------------
   1  |  104            |    8
   2  |   40            |    6
   3  |    2            |    6
   4  |    1            |    6
   5  |    0            |    6
   6  |   86            |    7
   .  |    .            |    .
   .  |    .            |    .
   .  |    .            |    .
```

In this example, suppose that earlier configuration of QoS policies have depleted the rule resources on ports 4 and 5 to the point where there are not enough rules remaining for applying an ACL, and only enough rules on port 3 for a minimal ACL.

At a minimum, the policies previously configured on ports 4 and 5 must be reduced to free up enough rule resources to allow you to apply an ACL to these ports. Depending on the ACL you want to apply to port 3, existing QoS policies on port 3 may have to be reduced.

Port 3 has enough rules available to accept an ACL that uses 1 or 2 rules.

Port 4 can accept only an ACL with one entry that has either the same (standard) ACL mask as **deny any** or the same (extended) ACL that has the same SA/DA ACL mask and same IP protocol.

Figure 10-6. Example of Inspecting Available Rule (and Mask) Resources

2. Use **show** commands to identify the currently configured ACL, QoS, and Rate-Limiting policies, and any per-VLAN IGMP configuration.

3. Determine which of the existing policies you can remove to free up rule resources for the ACL policy you want to implement. Depending on your network topology and configuration, you can free up rule resources by moving some policies to other devices. Another alternative is to inspect

the switch's existing configuration for unnecessary QoS and rate-limiting entries or inefficient applications that could be removed or revised to achieve the desired policies with less resource usage. Tables10-3 on page 10-21 and 8-10 on page 8-17, and the information displayed by the **access-list resources help** command, can help you to determine the resource usage of ACL and QoS policies.

Guidelines for Reconfiguring an ACL to Use Fewer Port Masks. If an ACL requires more mask resources than are reserved on a port for ACL use (maximum: eight per-port; seven if IGMP is configured), then the remedy is to reduce mask consumption by:

a. Ensuring that the ACEs in the list are in a sequence that takes optimum advantage of the switch's ability to re-use a mask on consecutive ACEs in a list. (Refer to table 10-2 on page 10-20.)

b. Removing enough ACEs from the ACL to reduce mask consumption to no more than the available maximum.

If an ACL requires more rule resources on a port than are available (a maximum of 120), then the remedy is to reduce rule consumption by:

a. Examining the ACEs in the list and, where feasible, combining multiple ACEs into a single ACE with a broader application.

b. If QoS or Rate-Limiting are applied to the same port(s) where you want to apply the ACL, prioritize your use of resources and eliminate enough of the lower-priority applications to allow you to apply the ACL. This may include shifting some applications to other switches.

Example of ACL Resource Usage

This example illustrates how to check for current per-port rule and mask availability, and then how to create and assign an ACL, and then to verify its effect on per-port rule and mask resources. (For more detailed information on configuring and applying ACLs, refer to the later sections of this chapter.)

Viewing the Current Per-Port Rule and Mask Usage

The **show access-list resources** command displays the currently available per-port rules and masks.

```
ProCurve(config)# show access-list resources

QoS/ACL Resource Usage

 Port  | Rules Available | ACL Masks Available
 ------+-----------------+--------------------
   1   | 120             |  8
   2   | 120             |  8
   3   | 120             |  8
   4   | 120             |  8
   5   | 120             |  8
   6   | 120             |  8
   7   | 120             |  8
   8   | 120             |  8
   .     .                 .
   .     .                 .
   .     .                 .
```

In the default configuration, there are 120 rules and 8 per-port ACL masks available on each port. These masks are reserved for ACLs and IGMP use.

Figure 10-7. Example of Available Per-Port Rules and ACL Masks

Standard ACL Using a Subset of the Switch's Ports. Suppose that ports 1 - 4 on a 3400cl or 6400cl switch belong to the following VLANs:

- VLAN 1: 10.10.10.1
- VLAN 2: 10.10.11.1
- VLAN 3: 10.10.12.1

(Assume that ports 1-4 are tagged members of VLAN 22, although tagged/untagged ports do not affect ACL operation because ACLs examine all inbound traffic, regardless of VLAN membership.)

The system administrator wants to:

- Permit inbound VLAN 1 traffic on all ports
- Permit inbound VLAN 2 traffic on ports 1 - 4 from hosts 10.10.10.1-30
- Deny inbound VLAN 2 traffic on ports 1 - 4 from hosts 10.10.10.31-255

- Permit inbound VLAN 3 traffic on all ports.

Because all ports in the example have the same inbound traffic requirements for ACL filtering, the system administrator needs to create only one ACL for application to all four ports.

- All inbound 10.10.10.*x* (VLAN 1) traffic is allowed on all ports.
- For the inbound 10.10.11.*x* (VLAN 2) traffic, the fourth octet of the ACL mask includes an overlap of permit and deny use on the "16" bit, which will require two different ACEs in the ACL. That is:
  - To deny hosts in the range of 31-255 in the fourth octet, it is necessary to use an ACE that specifies the leftmost four bits of the octet.
  - To permit hosts in the range of 1-30 in the fourth octet, it is necessary to use and ACE that specifies the rightmost five bits of the octet.

  The overlap[1] can be illustrated as shown here:

Bit Values in the Fourth Octet	128	64	32	16	8	4	2	1
Bits Needed To Deny Hosts 31 - 255 (4th Octet Mask: 0.0.0.224)								
Bits Needed To Permit Hosts 1 - 30 (4th Octet Mask: 0.0.0.31)								

[1] For more on this topic, refer to "Rules for Defining a Match Between a Packet and an Access Control Entry (ACE)" on page 10-33, and "Using CIDR Notation To Enter the ACL Mask" on page 10-45.

  The overlap on the "16" bit means that it is necessary for the ACL to deny the host at 10.10.11.31 before permitting the hosts in the range of 10.10.10.1 - 30. The complete sequence is:

1. Permit all inbound traffic from 10.10.10.*x*.
2. Permit all inbound traffic from 10.10.12.*x*.
3. Deny the host at 10.10.11.31.
4. Permit the hosts in the range of 10.10.11.1 - 30.
5. Allow the implicit deny (automatically present in all ACLs) to deny all other traffic, which will automatically include the hosts in the range 10.10.10.32 - 255.

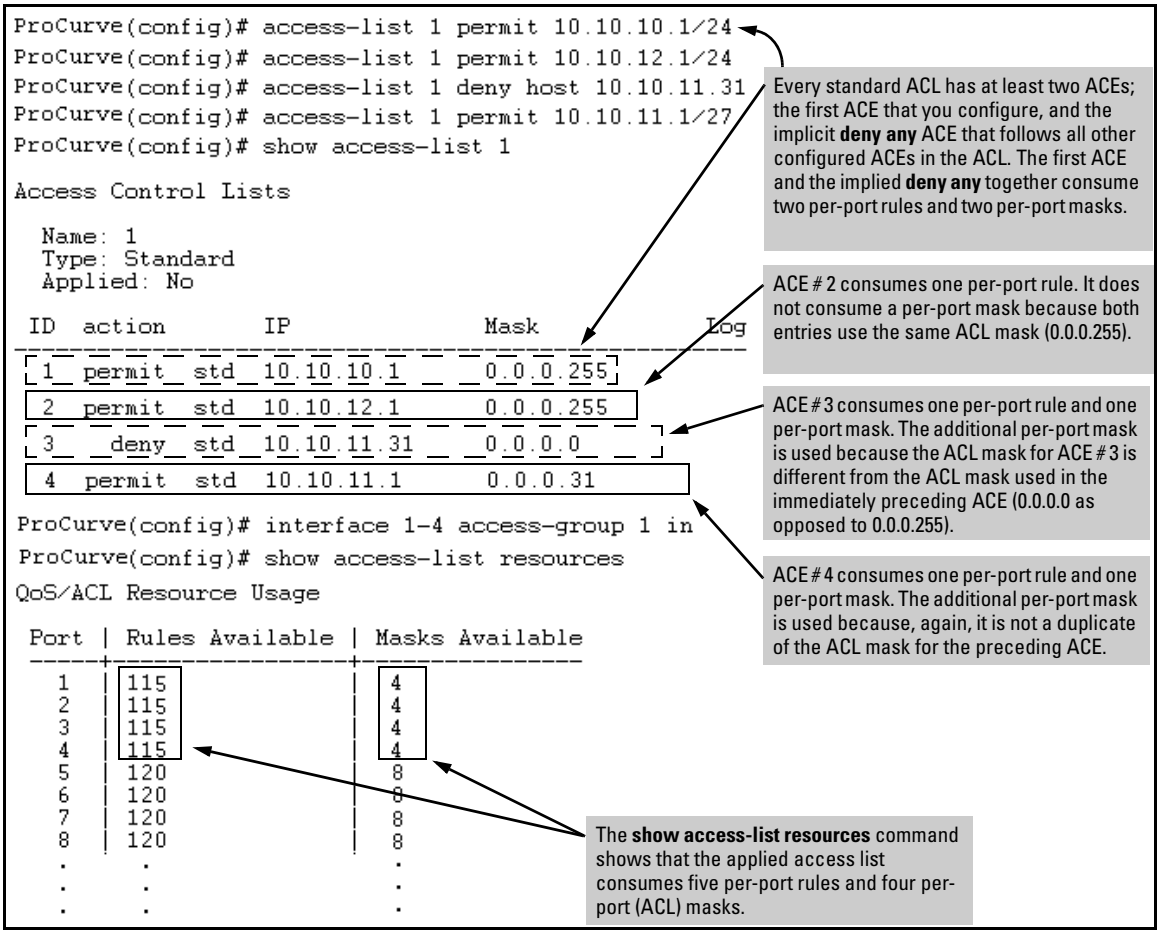


Figure 10-8. Example of Show Access List Command

Traffic Management and Improved Network Performance

You can use ACLs to block unnecessary traffic caused by individual hosts, workgroups, or subnets, and to block user access to subnets, devices, and services. Answering the following questions can help you to design and properly position ACLs for optimum network usage.

- What are the logical points for minimizing unwanted traffic? In many cases it makes sense to block unwanted traffic from the core of your network by configuring ACLs to drop such traffic at or close to the edge of the network. (The earlier in the network path you block unwanted traffic, the greater the benefit for network performance.)
- What traffic should you explicitly block? Depending on your network size and the access requirements of individual hosts, this can involve creating a large number of ACEs in a given ACL (or a large number of ACLs), which increases the complexity of your solution and rapidly consumes the per-port rule and mask resources.
- What traffic can you implicitly block by taking advantage of the implicit **deny any** to deny traffic that you have not explicitly permitted? This can reduce the number of entries needed in an ACL and make more economical use of switch resources.
- What traffic should you permit? In some cases you will need to explicitly identify permitted traffic. In other cases, depending on your policies, you can insert a **permit any** (standard ACL) or **permit ip any any** (extended ACL) entry at the end of an ACL. This means that all IP traffic not specifically matched by earlier entries in the list will be permitted.

Security

ACLs can enhance security by blocking inbound IP traffic carrying an unauthorized source IP address (SA). This can include:

- Blocking access to or from subnets in your network
- Blocking access to or from the internet
- Blocking access to sensitive data storage or restricted equipment
- Preventing the use of specific TCP or UDP functions (such as Telnet, SSH, web browser) for unauthorized access

You can also enhance switch management security by using ACLs to block inbound IP traffic that has the switch itself as the destination address (DA).

Caution

ACLs can enhance network security by blocking selected IP traffic, and can serve as one aspect of maintaining network security. *However, because ACLs do not provide user or device authentication, or protection from malicious manipulation of data carried in IP packet transmissions, they should not be relied upon for a complete security solution.*

Note

ACLs in the 3400cl/6400cl switches do not screen non-IP traffic such as AppleTalk and IPX.

Guidelines for Planning the Structure of an ACL

The first step in planning a specific ACL is to determine where you will apply it. (Refer to "ACL Inbound Application Points" on page 10-10.) You must then determine the order in which you want the individual ACEs in the ACL to filter traffic. Some applications require high usage of the per-port resources the switch uses to support ACLs (as well as the rules used by QoS and Rate-Limiting applications). In these cases it is important to order the individual ACEs in a list to avoid unnecessarily using resources. For more on this topic, refer to "Planning an ACL Application on a Series 3400cl or Series 6400cl Switch" on page 10-17.

- The first match dictates the action on a packet. possible, subsequent matches are ignored.
- On any ACL, the switch implicitly denies packets that are not explicitly permitted or denied by the ACEs configured in the ACL. If you want the switch to forward a packet for which there is not a match in an ACL, add **permit any** as the last ACE in an ACL. This ensures that no packets reach the implicit **deny any** case.
- Generally, you should list ACEs from the most specific (individual hosts) to the most general (subnets or groups of subnets) unless doing so permits traffic that you want dropped. For example, an ACE allowing a small group of workstations to use a specialized printer should occur earlier in an ACL than an entry used to block widespread access to the same printer.

ACL Configuration and Operating Rules

- **Per-Interface ACL Limits.** At a minimum an ACL must have one, explicit "permit" or "deny" Access Control Entry. You can assign one ACL per interface, as follows:
  - Standard ACLs—Numeric range: 1 - 99
  - Extended ACLs—Numeric range: 100 - 199
  - Named (Extended or Standard) ACLs: Up to the maximum number of ports on the switch (minus any numeric ACL assignments)

- **Implicit "deny any":** In any ACL, the switch automatically applies an implicit "deny IP any" that does not appear in **show** listings. This means that the ACL denies any packet it encounters that does not have a match with an entry in the ACL. Thus, if you want an ACL to permit any packets that you have not expressly denied, you must enter a **permit any** or **permit ip any any** as the last visible ACE in an ACL. Because, for a given packet the switch sequentially applies the ACEs in an ACL until it finds a match, any packet that reaches the **permit any** or **permit ip any any** entry will be permitted, and will not encounter the "deny ip any" ACE the switch automatically includes at the end of the ACL. For an example, refer to figure 10-5 on page 10-16.

- **Explicitly Permitting Any IP Traffic:** Entering a **permit any** or a **permit ip any any** ACE in an ACL permits all IP traffic not previously permitted or denied by that ACL. Any ACEs listed after that point do not have any effect and unnecessarily use rule and mask resources.

- **Explicitly Denying Any IP Traffic:** Entering a **deny any** or a **deny ip any any** ACE in an ACL denies all IP traffic not previously permitted or denied by that ACL. Any ACEs listed after that point have no effect.

- **An ACL Assignment Is Exclusive:** The switch allows one ACL assignment on an interface. If a port or static trunk already has an ACL assigned, you cannot assign another ACL to the interface without first removing the currently assigned ACL.

- **Replacing One ACL with Another:** Where an ACL is already assigned to an interface, you must remove the current ACL assignment before assigning another ACL to that interface. If an assignment command fails because one or more interfaces specified in the command already have an ACL assignment, the switch generates this message in the CLI and in the Event Log:

  `<` *acl-list-#* `>: Unable to apply access control list.`

- **ACLs Operate On Ports and Static Trunk Interfaces:** You can assign an ACL to any port and/or any statically configured trunk on the switch. ACLs do not operate with dynamic (LACP) trunks.
- **ACLs Screen Only the Traffic Entering the Switch on a Port or Static Trunk Interface:** On a given interface, ACLs can screen inbound traffic at the point where it enters the switch. In the 3400cl/6400cl switches, ACLs do not screen traffic routed between VLANs within the switch, between subnets in a multinetted VLAN, or at the interface where the traffic exits from the switch. (See figure 10-2 on page 10-11.)
- **Before Modifying an Applied ACL, You Must First Remove It from All Assigned Interfaces:** An ACL cannot be changed while it is assigned to an interface.
- **Before Deleting an Applied ACL, You Must First Remove It from All Interfaces to Which It Is Assigned:** An assigned ACL cannot be deleted.
- **Port and Static Trunk Interfaces:**
  - Removing a port from an ACL-assigned trunk returns the port to its default settings.
  - To add a port to a trunk when an ACL is already assigned to the port, you must first remove the ACL assignment from the port.
  - Adding a new port to an ACL-assigned trunk automatically applies the ACL to the new port.

How an ACE Uses a Mask To Screen Packets for Matches

When the switch applies an ACL to inbound traffic on an interface, each ACE in the ACL uses an IP address and *ACL mask* to enforce a selection policy on the packets being screened. That is, the mask determines the range of IP addresses (SA only or SA/DA) that constitute a match between the policy and a packet being screened.

What Is the Difference Between Network (or Subnet) Masks and the Masks Used with ACLs?

In common IP addressing, a network (or subnet) mask defines which part of the IP address to use for the network number and which part to use for the hosts on the network. For example:

IP Address	Mask	Network Address	Host Address
18.38.252.195	255.255.255.0	first three octets	The fourth octet.
18.38.252.195	255.255.248.0	first two octets and the left-most five bits of the third octet	The right most three bits of the third octet and all bits in the fourth octet.

Thus, the bits set to 1 in a network mask define the part of an IP address to use for the network number, and the bits set to 0 in the mask define the part of the address to use for the host number.

In an ACL, IP addresses and masks provide the criteria for determining whether to deny or permit a packet, or to pass it to the next ACE in the list. If there is a match, the deny or permit action occurs. If there is not a match, the packet is compared with the next ACE in the ACL. Thus, where a standard network mask defines how to identify the network and host numbers in an IP address, the mask used with ACEs defines which bits in a packet's IP address must match the corresponding bits in the IP address listed in an ACE, and which bits can be *wildcards*.

Rules for Defining a Match Between a Packet and an Access Control Entry (ACE)

- For a given ACE, when the switch compares an IP address and corresponding mask in the ACE to an IP address carried in a packet:
  - **A mask-bit setting of 0 ("off")** requires that the corresponding bit in the packet's IP address and in the ACE's IP address must be the same. That is, if a bit in the ACE's IP address is set to 1 ("on"), the same bit in the packet's IP address must also be 1.
  - **A mask-bit setting of 1 ("on")** means the corresponding bit in the packet's IP address and in the ACE's IP address do not have to be the same. That is, if a bit in the ACE's IP address is set to 1, the same bit in the packet's IP address can be either 1 or 0 ("on" or "off").

  For an example, refer to "Example of How the Mask Bit Settings Define a Match" on page 10-35.

- In any ACE, a mask of all ones means *any* IP address is a match. Conversely, a mask of all zeros means the *only* match is an IP address identical to the host IP address specified in the ACL.

- Depending on your network, a single ACE that allows a match with more than one source or destination IP address may allow a match with multiple subnets For example, in a network with a prefix of 31.30.240 and a subnet mask of 255.255.240.0 (the leftmost 20 bits), applying an ACL mask of 0.0.31.255 causes the subnet mask and the ACL mask to overlap one bit, which allows matches with hosts in two subnets: 31.30.224.0 and 31.30.240.0.

Bit Position in the Third Octet of Subnet Mask 255.255.240.0								
Bit Values	128	64	32	16	8	4	2	1
Subnet Mask Bits	1	1	1	1	n/a	n/a	n/a	n/a
Mask Bit Settings Affecting Subnet Addresses	0	0	0	**1 or 0**	n/a	n/a	n/a	n/a

This ACL supernetting technique can help to reduce the number of ACLs you need. You can apply it to a multinetted VLAN and to multiple VLANs. However, ensure that you exclude subnets that do not belong in the policy. If this creates a problem for your network, you can eliminate the unwanted match by making the ACEs in your ACL as specific as possible, and using multiple ACEs carefully ordered to eliminate unwanted matches.

- Every IP address and mask pair (source or destination) used in an ACE creates one of the following policies:
  - **Any IP address fits the matching criteria.** In this case, the switch automatically enters the IP address and mask in the ACE. For example:

    access-list 1 deny any

    produces this policy in an ACL listing:

IP Address	Mask
0.0.0.0	255.255.255.255

    This policy states that every bit in every octet of a packet's SA is a wildcard, which covers any IP address.
  - **One IP address fits the matching criteria.** In this case, you provide the IP address and the switch provides the mask. For example:

    access-list 1 permit host 18.28.100.15

    produces this policy in an ACL listing:

IP Address	Mask
18.28.100.15	0.0.0.0

    This policy states that every bit in every octet of a packet's SA must be the same as the corresponding bit in the SA defined in the ACE.
  - **A group of IP addresses fits the matching criteria.** In this case you provide both the IP address and the mask. For example:

    access-list 1 permit 18.28.32.1 0.0.0.31

IP Address	Mask
18.28.32.1	0.0.0.31

    This policy states that:
    - In the first three octets of a packet's SA, every bit must be set the same as the corresponding bit in the SA defined in the ACE.
    - In the last octet of a packet's SA, the first three bits must be the same as in the ACE, but the last five bits are wildcards and can be any value.

- Unlike subnet masks, the wildcard bits in an ACL mask need not be contiguous. For example, 0.0.7.31 is a valid ACL mask. However, a subnet mask of 255.255.248.224 is not a valid subnet mask.

Example of How the Mask Bit Settings Define a Match . Assume an ACE where the second octet of the mask for an SA is 7 (the rightmost three bits are "on", or "1") and the second octet of the corresponding SA in the ACE is 31 (the rightmost five bits). In this case, a match occurs when the second octet of the SA in a packet being filtered has a value in the range of 24 to 31. Refer to table 10-4, below.

Table 10-4. Example of How the Mask Defines a Match

Location of Octet	Bit Position in the Octet							
	128	64	32	16	8	4	2	1
SA in ACE	0	0	0	1	1	1	1	1
Mask for SA	0	0	0	0	0	1	1	1
Corresponding Octet of a Packet's SA	0	0	0	1	1	0/1	0/1	0/1

The shaded area indicates bits in the packet that must exactly match the bits in the source IP in the ACE. Wherever the mask bits are ones (wildcards), the IP bits in the packet can be any value, and where the mask bits are zeros, the IP bits in the packet must be the same as the IP bits in the ACE. **Note:** This example covers only one octet of an IP address. An actual ACE applies this method to all four octets of an IP address.

Example of Allowing Only One IP Address ("Host" Option). Suppose, for example, that you have configured the ACL in figure 10-9 to filter inbound packets on port 20. Because the mask is all zeros, the ACE policy dictates that a match occurs only when the source IP address on such packets is identical to the IP address configured in the ACE.

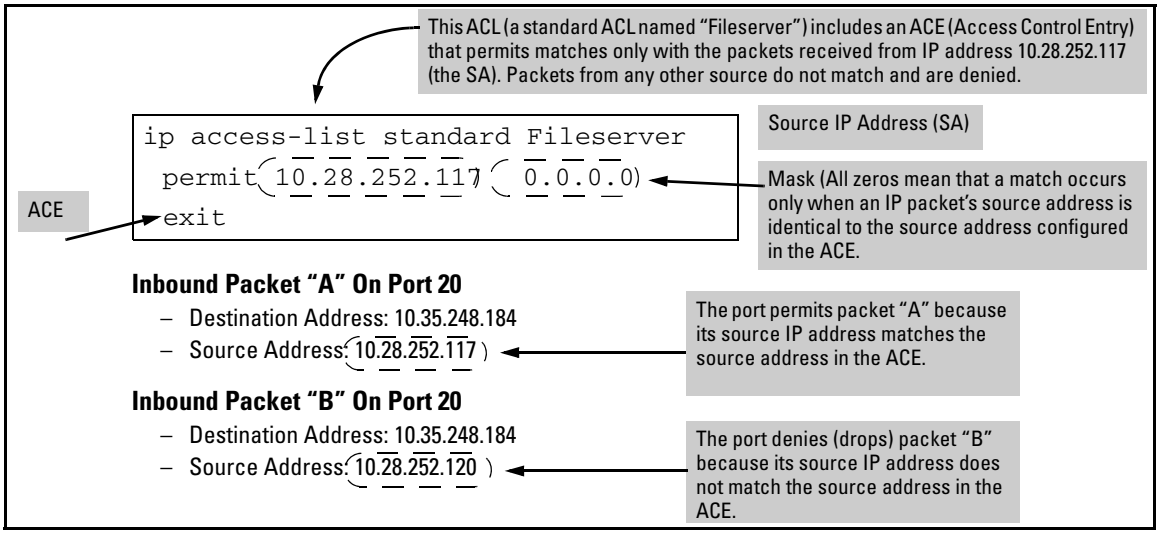


Figure 10-9. Example of an ACL with an Access Control Entry (ACE) that Allows Only One Source IP Address

Examples Allowing Multiple IP Addresses. Table 10-5 provides examples of how to apply masks to meet various filtering requirements.

Table 10-5. Example of Using an IP Address and Mask in an Access Control Entry

IP Address in the ACE	Mask	Policy for a Match Between a Packet and the ACE	Allowed IP Addresses
A: 10.38.252.195	0.0.0.255	Exact match in first three octets only.	10.38.252.< 0-255 > (See row A in table 10-6, below.)
B: 10.38.252.195	0.0.7.255	Exact match in the first two octets and the leftmost five bits (248) of the third octet.	10.38.< 248-255 >.< 0-255 > (In the third octet, only the rightmost three bits are wildcard bits. The leftmost five bits must be a match, and in the ACE, these bits are all set to 1. See row B in table 10-6, below.)
C: 10.38.252.195	0.0.0.0	Exact match in all octets.	10.38.252.195 (There are no wildcard bits in any of the octets. See row C in table 10-6, below.)
D: 10.38.252.195	0.15.255.255	Exact match in the first octet and the leftmost four bits of the second octet.	10.< 32-47 >.< 0-255 >.<0-255> (In the second octet, the rightmost four bits are wildcard bits. See row D in table 10-6, below.)

Table 10-6. Mask Effect on Selected Octets of the IP Addresses in Table 10-5

IP Addr	Octet	Mask	Octet Range	128	64	32	16	8	4	2	1
A	3	**0** all bits	**252**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**
B	3	**7** last 3 bits	**248-255**	**1**	**1**	**1**	**1**	**1**	**0 or 1**	**0 or 1**	**0 or 1**
C	4	**0** all bits	**195**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**
D	2	**15** last 4 bits	**32-47**	**0**	**0**	**1**	**0**	**0 or 1**	**0 or 1**	**0 or 1**	**0 or 1**

Shaded areas indicate bit settings that must be an exact match.

If there is a match between the policy in the ACE and the IP address in a packet, then the packet is either permitted or denied, according to how the ACE is configured. If there is not a match, the next ACE in the ACL is then applied to the packet. The same operation applies to a destination IP address (DA) used in an extended ACE. (Where an ACE includes both source and destination IP addresses, there is one IP-address/ACL-mask pair for the source address, and another IP-address/ACL-mask pair for the destination address. See “Configuring and Assigning an ACL” on page 10-38.)

CIDR Notation. For information on using CIDR notation to specify ACL masks, refer to “Using CIDR Notation To Enter the ACL Mask” on page 10-45.

Configuring and Assigning an ACL

Overview

General Steps for Implementing ACLs

1. Configure at least one ACL. This creates and stores the ACL in the switch configuration.
2. Assign an ACL. This applies the ACL to the inbound traffic on one or more designated interfaces.

Caution Regarding the Use of Source Routing

Source routing is enabled by default on the switch and can be used to override ACLs. For this reason, if you are using ACLs to enhance network security, the recommended action is to disable source routing on the switch. To do so, execute **no ip source-route**.

Types of ACLs

- **Standard ACL:** Uses only a packet's source IP address as a criterion for permitting or denying the packet. For a standard ACL ID, use either a unique numeric string in the range of 1-99 or a unique name string of up to 64 alphanumeric characters.
- **Extended ACL:** Offers the following criteria as options for permitting or denying a packet:
  - Source IP address
  - Destination IP address
  - TCP or UDP criteria

  For an extended ACL ID, use either a unique number in the range of 100-199 or a unique name string of up to 64 alphanumeric characters.

You should carefully plan your ACL application before configuring specific ACLs. For more on this topic, refer to "Planning an ACL Application on a Series 3400cl or Series 6400cl Switch" on page 10-17.

ACL Configuration Structure

After you enter an ACL command, you may want to inspect the resulting configuration. This is especially true where you are entering multiple ACEs into an ACL. Also, it will be helpful to understand the configuration structure when using later sections in this chapter.

The basic ACL structure includes three elements:

1. ACL type and name: This identifies the ACL as **standard** or **extended** and shows the ACL name.
2. One or more deny/permit list entries (ACEs): One entry per line.

Element	Stnd	Ext	Notes
ID Range	1 - 99	100 - 199	You can also use an alphanumeric name of up to 64 characters, including spaces.
Minimum ACEs per ACL	1		
Maximum ACEs Per ACL	120		
Maximum ACEs per Switch	1024		In some cases, rule usage by ACLs, IGMP, QoS, and Rate-Limiting, and mask usage by ACLs may consume available resources to the point where this limit cannot be reached.

3. Implicit **deny any**: Where an ACL is in use, the switch denies any packets that do not have a match with the ACEs explicitly configured in the ACL. The implicit **deny any** does not appear in ACL configuration listings, but always functions when the switch uses an ACL to filter packets. (You cannot delete the implicit "deny any", but you can supersede it with a "permit any" statement.)

Standard ACL Structure

Individual ACEs in a standard ACL include only a permit/deny "type" statement, the source IP addressing, and an optional **log** command (available with "deny" statements).

```
ip access-list < type > "< id-string >"
  permit host < source-ip-address >
  deny < source-ip-address > < acl-mask > [log]
  .
  .
  .
  permit any
  exit
```

Figure 10-10. Example of the General Structure for a Standard ACL

For example, figure 10-11 shows how to interpret the entries in a standard ACL.

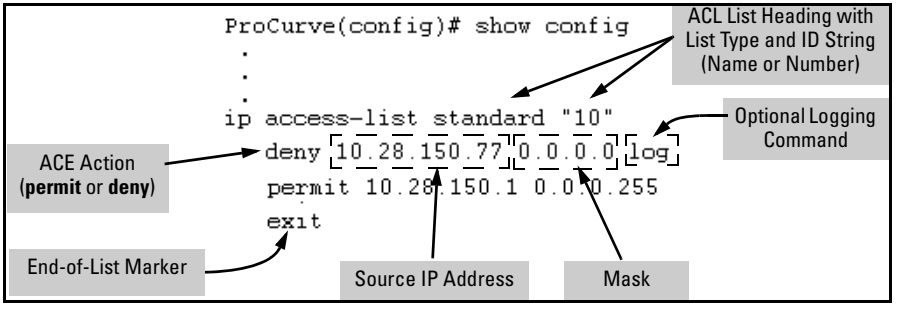


Figure 10-11. Example of a Displayed Standard ACL Configuration with Two ACEs

Extended ACL Configuration Structure

Individual ACEs in an extended ACL include:

- A permit/deny "type" statement
- Source IP addressing
- Optional TCP or UDP port type with optional source port ID and operator and/or optional destination port ID and operator
- Destination IP addressing

- Optional ACL **log** command (available for "Deny" ACLs only)

```
ip access-list < type > "< id-string >" < permit | deny > ip
        < source-ip-address > < source-acl-mask >
        < destination-ip-address > < destination-acl-mask > [log]

        < permit | deny > tcp
            < source-ip-address > < source-acl-mask > [< operator > < port-id >]
            < destination-ip-address > < destination-acl-mask > [< operator > < port-id >] [log]

        < permit | deny > udp
            < source-ip-address > < source-acl-mask > [< operator > < port-id >]
            < destination-ip-address > < destination-acl-mask > [< operator > < port-id >] [log]
        . . .
        exit
```

Note: The optional log function appears only with "deny" aces.

Figure 10-12. General Structure for an Extended ACL

For example, figure 10-13 shows how to interpret the entries in an extended ACL.

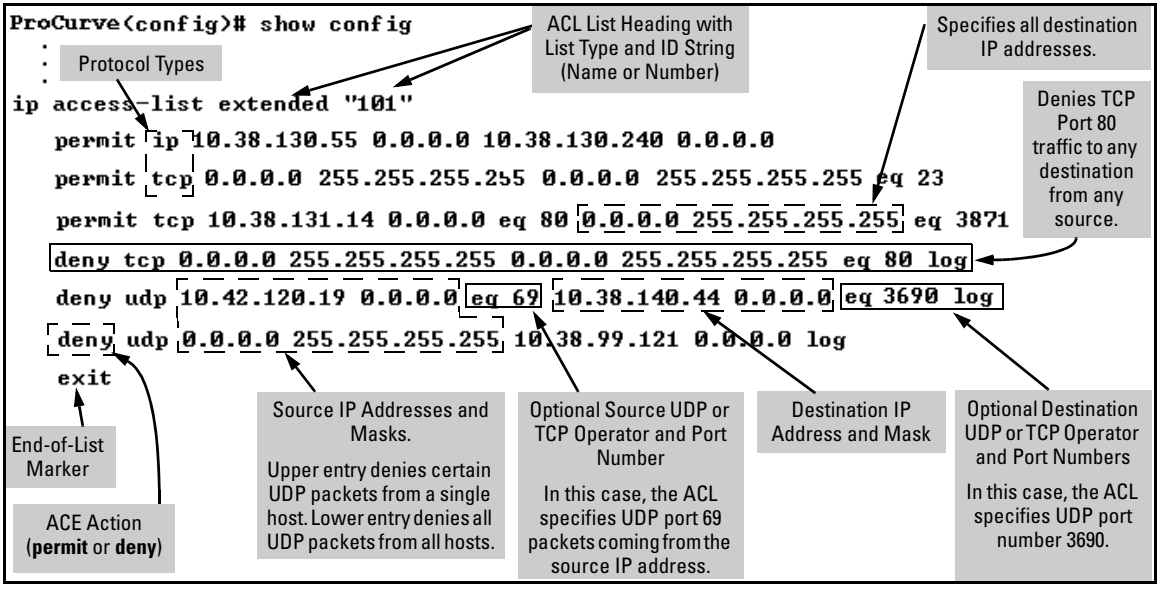


Figure 10-13. Example of a Displayed Extended ACL Configuration

ACL Configuration Factors

ACL Resource Consumption

Consumption of per-port rules and masks can be a significant factor in switches using extensive ACL applications. In this case, resource usage takes precedence over other factors when planning and configuring ACLs. For more information on this topic, refer to "Planning an ACL Application on a Series 3400cl or Series 6400cl Switch" on page 10-17.

The Sequence of Entries in an ACL Is Significant

When the switch uses an ACL to determine whether to permit or deny a packet on a particular interface, it compares the packet to the criteria specified in the individual Access Control Entries (ACEs) in the ACL, beginning with the first ACE in the list and proceeding sequentially until a match is found. When a match is found, the switch applies the indicated action (permit or deny) to the packet. This is significant because, once a match is found for a packet, subsequent ACEs in the same ACL will not be used for that packet, regardless of whether they match the packet.

For example, suppose that you have applied the ACL shown in figure 10-11 to inbound traffic on port 10:

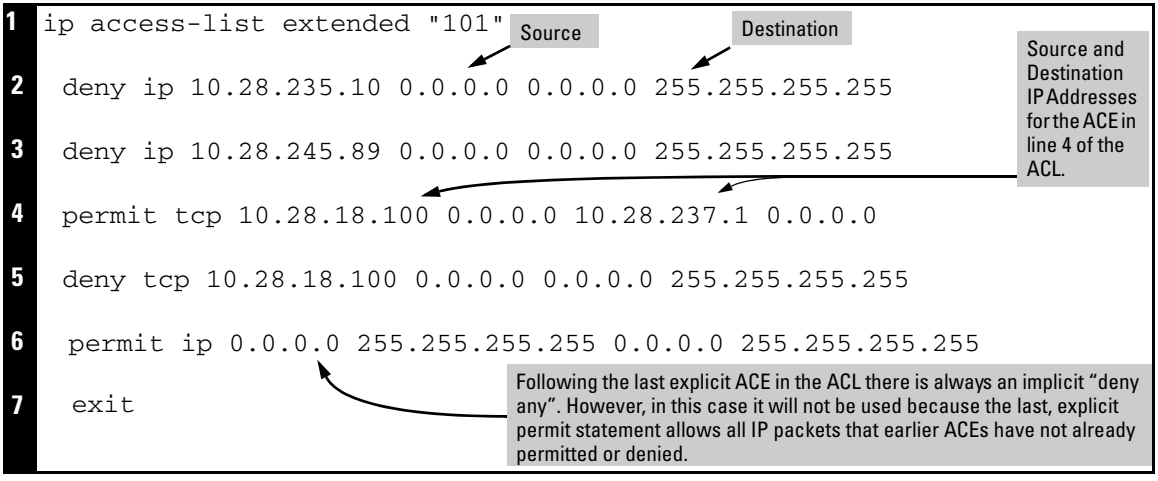


Figure 10-14. Example of a Standard ACL that Permits All Traffic Not Implicitly Denied

Table 10-7. Effect of the ACL in Figure 10-14 on Inbound Traffic on the Assigned Port

Line #	Action
1	Shows list type (extended) and ID (101).
2	A packet from IP source address 10.28.235.10 will be denied (dropped). This line filters out all packets received from 10.28.235.10. As a result, IP traffic from that device will not be routed or switched, and packets from that device will not be compared against any later entries in the list.
3	A packet from IP source 10.28.245.89 will be denied (dropped). This line filters out all packets received from 10.28.245.89. As the result, IP traffic from that device will not be routed or switched and packets from that device will not be compared against any later entries in the list.
4	A packet from TCP source address 10.28.18.100 with a destination address of 10.28.237.1 will be permitted (forwarded). Since no earlier lines in the list have filtered TCP packets from 10.28.18.100 and destined for 10.28.237.1, the switch will use this line to evaluate such packets. Any packets that meet this criteria will be forwarded. (Any packets that do not meet this TCP source-destination criteria are not affected by this line.)
5	A packet from TCP source address 10.28.18.100 to ***any*** destination address will be denied (dropped). Since, in this example, the intent is to block TCP traffic from 10.28.18.100 to any destination ***except*** the destination stated in line 4, this line must follow line 4. (If their relative positions were exchanged, all TCP traffic from 10.28.18.100 would be dropped, including the traffic for the 10.28.18.1 destination.)
6	Any packet from any IP source address to any destination address will be permitted (forwarded). The only traffic to reach this line will be IP packets not specifically permitted or denied in the earlier lines.
n/a	The "implicit deny any any" is a function automatically added as the last action in all ACLs. It denies (drops) any IP traffic from any source to any destination that has not found a match with earlier entries in the list. In this example, line 6 permits (forwards) any IP traffic not already permitted or denied by the earlier entries in the list, so there is no traffic remaining for action by the "implicit deny any any" function.
7	Indicates the end of the ACL.

```
ProCurve(config)# show access-list resources
QoS/ACL Resource Usage

 Port  | Rules Available | ACL Masks Available
-------+-----------------+--------------------
   1   |  120            |   8
   2   |  120            |   8
   3   |  120            |   8
   4   |  120            |   8
   5   |  120            |   8
   6   |  120            |   8
   7   |  120            |   8
   8   |  120            |   8
   9   |  120            |   8
  10   |  115            |   4
  11   |  120            |   8
  12   |  120            |   8
   .       .                .
   .       .                .
   .       .                .
```

The ACL in figure 10-14, when applied to port 10, uses five per-port rules and four per-port masks. Note that the last ACE in the list is a duplicate of the implicit **deny ip any any** that is automatically included at the end of every extended ACL. As a result, the last configured (visible) ACE and the implicit **deny** ACE use the same rule and mask. For more on this topic, refer to table Table 10-3 on page 10-21.

Figure 10-15. Per-Port Rule and Mask Usage for the ACL in Figure 10-14

In Any ACL, There Will Always Be a Match

As indicated in figure 10-14, the switch automatically uses an implicit "deny IP any" (Standard ACL) or "deny IP any any" (Extended ACL) as the last ACE in any ACL. This means that if you configure the switch to use an ACL for filtering inbound traffic, any packets not specifically permitted or denied by the explicit entries you create will be denied by the implicit "deny" action. Note that if you want to preempt the implicit "deny" action, insert an explicit **permit any** or **permit ip any any** as the last line of the ACL.

A Configured ACL Has No Effect Until You Apply It to an Interface

The switch stores ACLs in the configuration file. Thus, until you actually assign an ACL to an interface, it is present in the configuration, but not used.

Using the CLI To Create an ACL

Command	Page
access-list (standard ACLs)	10-47
access-list (extended ACLs)	10-52
ip access-list (named ACLs)	10-58

You can use either the switch CLI or an offline text editor to create an ACL. This section describes the CLI method, which is recommended for creating short ACLs. (To use the offline method, refer to "Editing ACLs and Creating an ACL Offline" on page 10-69.)

General ACE Rules

These rules apply to all ACEs you create or edit using the CLI:

- ACEs are placed in an ACL according to the sequence in which you enter them (last entered, last listed).
- You can use the CLI to delete an ACE from anywhere in a given ACL by using the "no" form of the command to enter that ACE. However, when you use the CLI to add an ACE, the new entry is always placed *at the end of the ACL*.

- Duplicate ACEs are allowed in an ACL. However, multiple instances of an ACE have no effect on filtering because the first instance preempts any subsequent duplicates. Also, duplicate entries unnecessarily consume additional resources on assigned ACLs.

For more information, refer to "Editing ACLs and Creating an ACL Offline" on page 10-69.

Using CIDR Notation To Enter the ACL Mask

You can use CIDR (Classless Inter-Domain Routing) notation to enter ACL masks. The switch interprets the bits specified with CIDR notation as the IP address bits in an ACL and the corresponding IP address bits in a packet. The switch then converts the mask to inverse notation for ACL use.

Table 10-8. Examples of CIDR Notation for Masks

IP Address Used In an ACL with CIDR Notation	Resulting ACL Mask	Meaning
18.38.240.125/15	0.1.255.255	The leftmost 15 bits must match; the remaining bits are wildcards.
18.38.240.125/20	0.0.15.255	The leftmost 20 bits must match; the remaining bits are wildcards.
18.38.240.125/21	0.0.7.255	The leftmost 21 bits must match; the remaining bits are wildcards.
18.38.240.125/24	0.0.0.255	The leftmost 24 bits must match; the remaining bits are wildcards.
18.38.240.125/32	0.0.0.0	All bits must match.

Configuring and Assigning a Numbered, Standard ACL

Configuring Named ACLs	"Configuring a Named ACL" on page 10-58
Configuring Extended, Numbered ACLs	"Configuring and Assigning a Numbered, Extended ACL" on page 10-52

- To configure named ACLs, refer to "Configuring a Named ACL" on page 10-58.
- To configure extended, numbered ACLs, refer to "Configuring and Assigning a Numbered, Extended ACL" on page 10-52.

A standard ACL uses only source IP addresses in its ACEs. This type of ACE is useful when you need to:

- Permit or deny traffic based on source IP address only.
- Quickly control the IP traffic from a specific address, a group of addresses, or a subnet. This allows you to isolate traffic problems generated by a specific device, group of contiguous devices, or a subnet threatening to degrade network performance. This gives you an opportunity to troubleshoot without sacrificing performance for users outside of the problem area.

You can identify each standard ACL with a number in the range of 1 - 99, or an alphanumeric string of up to 64 characters. The CLI command process for using an alphanumeric string to name an ACL differs from the command process for a numeric name. For a description of how to name an ACL with an alphanumeric character string, refer to "Configuring a Named ACL" on page 10-58. To view the command differences, refer to table 10-1, "Comprehensive Command Summary" on page 10-6.

Note

For a summary of ACL commands, refer to table 10-1, "Comprehensive Command Summary", on page 10-6.

Syntax: [no] access-list

Creates an ACE in the specified (1-99) access list and indicates the action (deny or permit) to take on a packet if there is a match between the packet and the criterion in the entry. If the ACL does not already exist, this command creates the specified ACL and its first ACE. To create a named ACL, refer to "Configuring a Named ACL" on page 10-58

< 1-99 >

Specifies the ACL ID number. The switch interprets an ACL with a value in this range as a standard ACL.

Note: *To create an access list with an alphanumeric name (**name-str**) instead of a number, refer to "Configuring a Named ACL" on page 10-58.*

< deny | permit >

Specifies whether to deny (drop) or permit (forward) a packet that matches the ACE criteria.

< any | host < *src-ip-addr* > | *ip-addr / mask-length* >

- **any** — *Performs the specified action on any IP packet. Use this criterion to designate packets from any IP address.*
- **host** < *host ip-address* > — *Performs the specified action on any IP packet having the* < *host ip-address* > *as the source. Use this criterion to designate packets from a single IP address.*
- ***IP-addr / mask-length*** — *Performs the specified action on any IP packet having a source address within the range defined by either*

  < *src-ip-addr / cidr-mask-bits* **>**

  or

  < *src-ip-addr < mask* **>>**

  Use this criterion to filter packets received from either a subnet or a group of contiguous IP addresses. The mask can be in either dotted-decimal format or CIDR format with the number of significant bits. Refer to "Using CIDR Notation To Enter the ACL Mask" on page 10-45.

The mask is applied to the IP address in the ACL to define which bits in a packet's source IP address must exactly match the IP address configured in the ACL and which bits need not match. Note that specifying a group of contiguous IP addresses may require more than one ACE. For more on how masks operate in ACLs, refer to "How an ACE Uses a Mask To Screen Packets for Matches" on page 10-32.

[log]

Optionally generates an ACL log message if:

- *The action is* **deny.**
- *There is a match.*
- *ACL logging is enabled on the switch. (Refer to "Enable ACL "Deny" Logging" on page 10-75.)*

(Use the debug command to direct ACL logging output to the current console session and/or to a Syslog server. Note that you must also use the **logging < i*p-addr*>** *command to specify the IP addresses of Syslog servers to which you want log messages sent. See also "Enable ACL "Deny" Logging" on page 10-75.)*

Syntax: interface < *port-list* | *trunk* > access-group < *ASCII-STR* > in

Assigns an ACL, designated by an ACL ID (< ASCII-STR >), to an interface (list of one or more ports and/or one or more static trunks).

Example of a Standard ACL. Suppose you wanted to configure a standard ACL and assign it to filter inbound traffic on port 10 in a particular switch:

- The ID you selected for this ACL is "50".
- You want the ACL to deny IP traffic from all hosts except these three:
  - 10.128.100.10
  - 10.128.100.27
  - 10.128.100.14

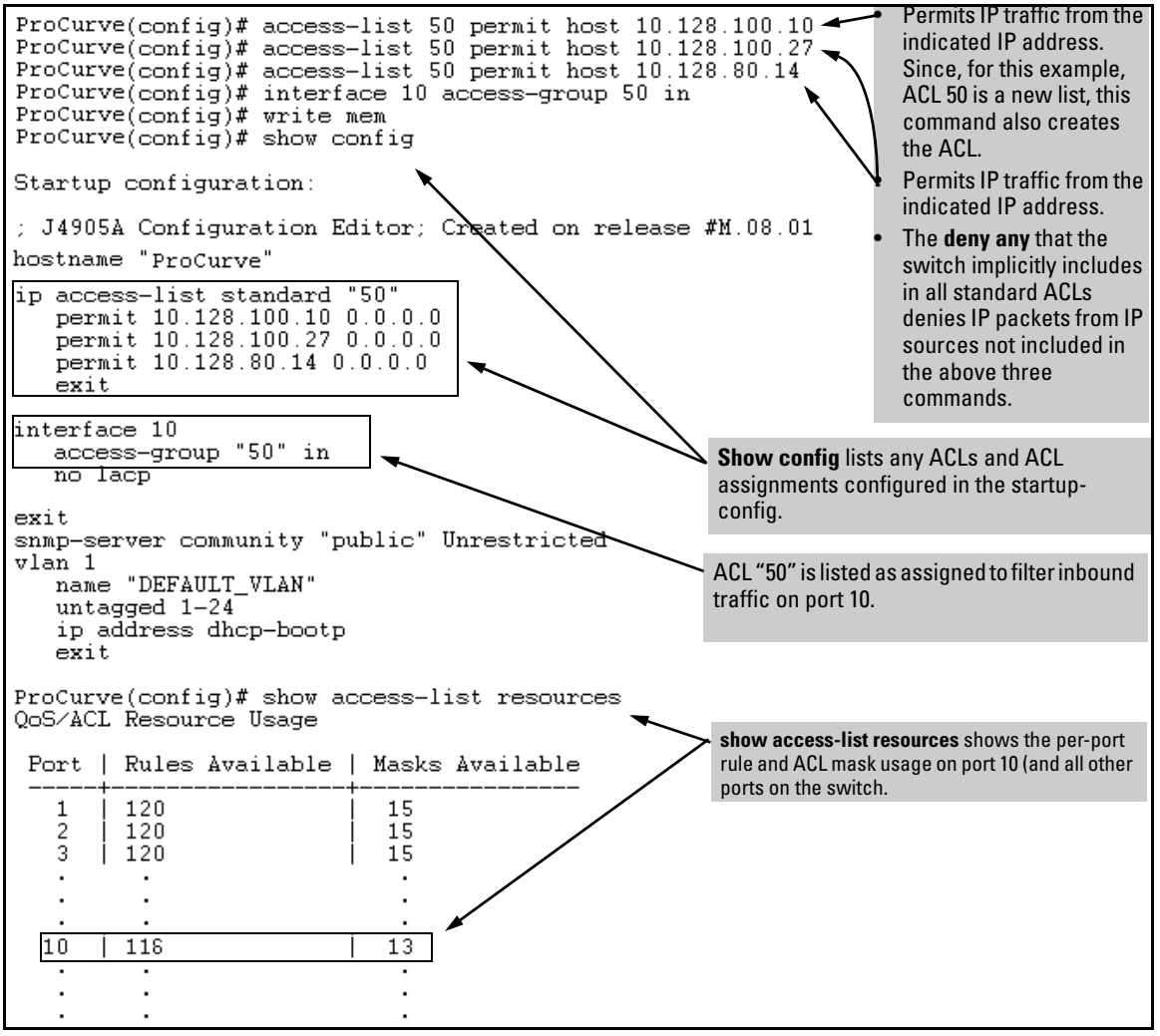


Figure 10-16. Example of Configuring a Standard ACL To Permit Only Traffic from Specific IP Addresses

In a situation opposite to the above, suppose that you wanted to deny inbound IP traffic received on port 20 from 10. 128.93.17 and 10.130.93.25, but permit all other IP traffic on this VLAN. The next ACL achieves this:

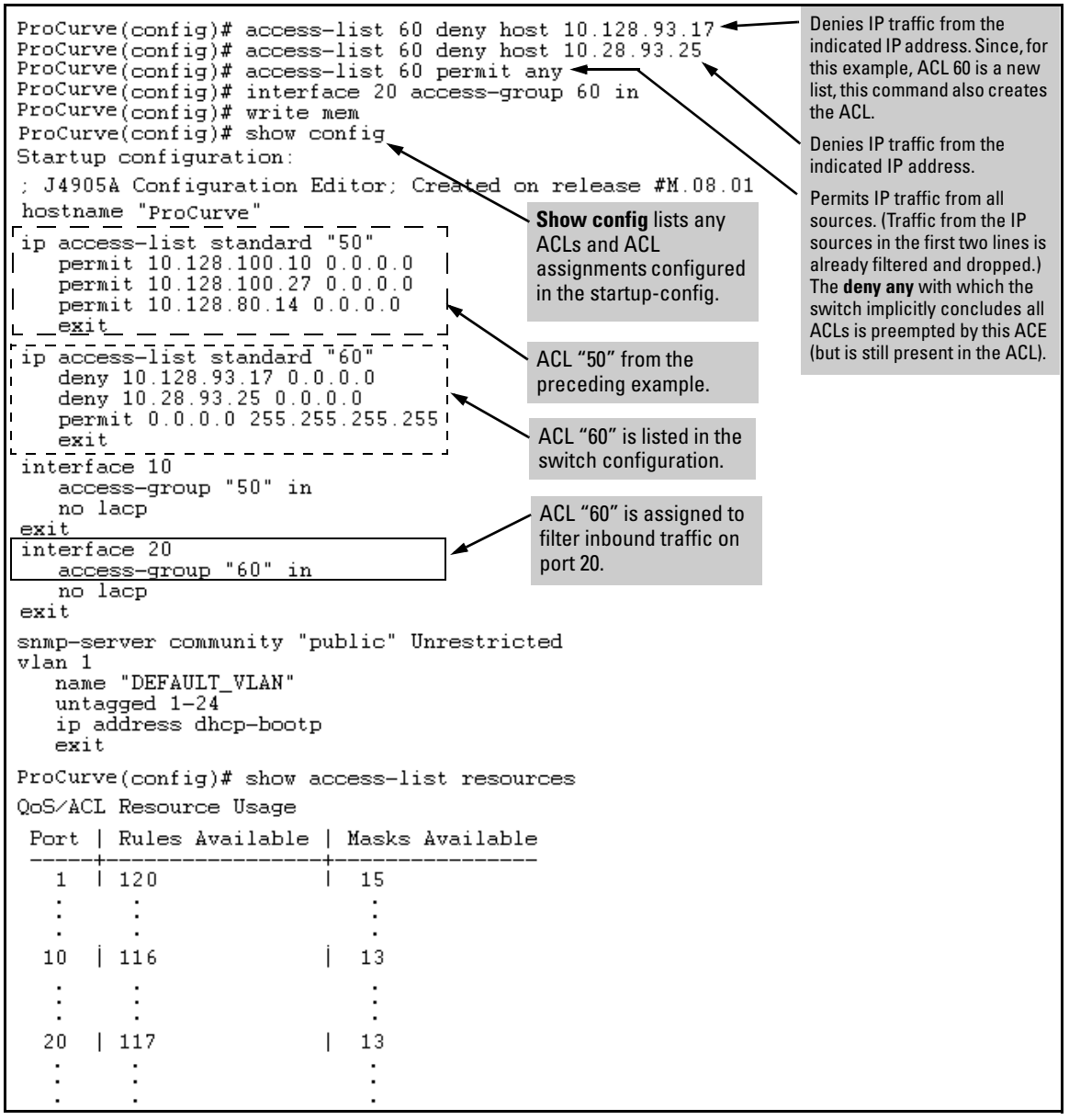


Figure 10-17. Example of Configuring a Standard ACL To Deny Inbound Traffic from Specific IP Addresses

Configuring and Assigning a Numbered, Extended ACL

This section describes how to configure numbered, extended ACLs. To configure other ACL types, refer to the following table.

To Configure:	Refer To:
Standard, numbered ACLs	"Configuring and Assigning a Numbered, Standard ACL" on page 10-47
Named ACLs	"Configuring a Named ACL" on page 10-58

While standard ACLs use only source IP addresses for filtering criteria, extended ACLs allow multiple ACE criteria. This enables you to more closely define your IP packet-filtering criteria. These criteria include:

- Source and destination IP addresses (required), in one of the following options:
  - Specific host IP
  - Subnet or group of IP addresses
  - Any IP address
- IP protocol (IP, TCP, or UDP)
- Source TCP or UDP port (if the IP protocol is TCP or UDP)
- Destination TCP or UDP port (if the IP protocol is TCP or UDP)
- TCP or UDP **eq** operator (if the IP protocol is TCP or UDP)

You can configure extended ACLs with a numeric name in the range of 100 - 199. You can also configure extended ACLs with alphanumeric names. (Refer to "Configuring a Named ACL" on page 10-58.)

Note

For a summary of ACL commands, refer to table 10-1, "Comprehensive Command Summary", on page 10-6.

Syntax: [no] access-list

Creates an ACE in the specified (100-199) access list and:

- *Indicates the action (deny or permit) to take on a packet if there is a match between the packet and the criteria in the complete ACE.*
- *Specifies the packet protocol type (IP, TCP, or UDP).*
- *Specifies the source and destination addressing options described in the remainder of this section.*
- *Allows optional ACL logging where a packet has a match with a* **deny** *ACE.*

If the ACL does not already exist, this command creates the specified ACL and its first ACE. If the ACL already exists, this command adds a new, explicit ACE to the end of the ACL. For a match to occur, the packet must have the source and destination IP addressing criteria specified by this command, as well as any protocol-specific (TCP or UDP port number) criteria specified by the command. To create a named ACL, refer to "Configuring a Named ACL" on page 10-58.

< 100-199 >

Specifies the ACL ID number. The switch interprets an ACL with a value in this range as an extended ACL.

Note: *To create an access list with an alphanumeric name instead of a number, refer to "Configuring a Named ACL" on page 10-58.*

< deny | permit >

Specifies whether to deny (drop) or permit (forward) a packet that matches the ACE criteria.

< ip | tcp | udp >

Specifies the packet protocol type required for a match:

- **ip** *— any IP packet*
- **tcp** *— only tcp packets*
- **udp** *— only udp packets*

< any | host < *src-ip-addr* > | *ip-addr/mask -length* >

In an extended ACL, this parameter defines the source IP address (SA) that a packet must carry in order to have a match with the ACE.

- **any** — *Specifies all inbound IP packets.*
- **host** < ***src-ip-addr***> — *Specifies only inbound packets from a single IP address. Use this option when you want to match only the IP packets from one source IP address (device).*
- ***src-ip-addr/mask-length*** — *Performs the specified action on any IP packet having a source address within the range defined by either*

  < src-ip-addr / cidr-mask-bits >
  or
  < src-ip-addr < mask >>

  Use this criterion to filter packets received from either a subnet or a group of IP addresses. The mask can be in either dotted-decimal format or CIDR format with the number of significant bits. Refer to "Using CIDR Notation To Enter the ACL Mask" on page 10-45.
  The mask is applied to the IP address in the ACL to define which bits in a packet's source IP address must exactly match the IP address configured in the ACL and which bits need not match. Note that specifying a group of contiguous IP addresses may require more than one ACE. For more on how masks operate in ACLs, refer to "How an ACE Uses a Mask To Screen Packets for Matches" on page 10-32.

[*operator* < *src-port tcp/udp-id* >]

In an extended ACL where you have selected either **tcp** *or* **udp** *as the packet protocol type (see above), you can optionally use a TCP or UDP source port number or range of numbers to further define the criteria for a match. To specify a TCP or UDP port number, (1) select the* **eq** *comparison operator and (2) enter the port number or a well-known port name.*

Comparison Operator:

- **eq** < ***tcp/udp-port-nbr*** > — *"Equal To"; to have a match with the ACE entry, the TCP or UDP source port number in a packet must be equal to* < ***tcp/udp-port-nbr*** >.

Port Number or Well-Known Port Name:

Use the TCP or UDP port number required by your application. The switch also accepts these well-known TCP or UDP port names as an alternative to their corresponding port numbers:

- ***TCP***: bgp, dns, ftp, http, imap4, ldap, nntp, pop2, pop3, smtp, ssl, telnet
- ***UDP***: bootpc, bootps, dns, ntp, radius, radius-old, rip, snmp, snmp-trap, tftp

To list the above names, press the **[Shift] [?]** *key combination after entering an operator. For a comprehensive listing of port numbers, visit www.iana.org/assignments/port-numbers.*

< any | host < *dest-ip-addr* > | *ip-addr/mask*-length >

In an extended ACL, this parameter defines the destination IP address (DA) that a packet must carry in order to have a match with the ACE. The options are the same as shown for < ***src-ip-addr*** >.

[< *dest-port tcp/udp-id* >]

In an extended ACL, this parameter defines the TCP or UDP destination port number a packet must carry in order to have a match with the extended ACE. The options are the same as shown above on the preceding page for the source IP address.

[log]

Optional; generates an ACL log message if:

- *The action is* **deny.** *(This option is not configurable for* **Permit.***)*
- *There is a match.*
- *ACL logging is enabled on the switch. (Refer to "Enabling ACL Logging on the Switch" on page 10-77)*

Syntax: interface < *port-list* > access-group < *list-#* | *ascii-str* > in

Assigns an ACL, designated by an ACL list number or ASCII string (alphanumeric list name), to an interface to filter inbound IP traffic on that interface. To configure named ACLs, refer to "Configuring a Named ACL" on page 10-58.

Example of an Extended ACL. Suppose that you want to implement these policies on ports 1, 2, and 3:

A. Permit Telnet traffic from 10.10.10.44 inbound on port 1 to 10.10.20.78, deny all other inbound IP traffic from network 10.10.10.0 (VLAN 10) to 10.10.20.0 (VLAN 20), and permit all other IP traffic from any source to any destination. (See "A" in figure 10-18, below.)

B. Permit FTP traffic from IP address 10.10.20.100 on port 2 to 10.10.30.55. Deny FTP traffic from other hosts on network10.10.20.0 to any destination, but permit all other traffic.

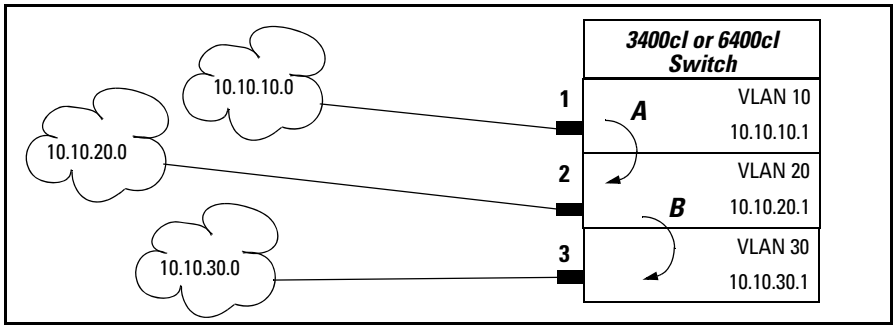


Figure 10-18. Example of an Extended ACL

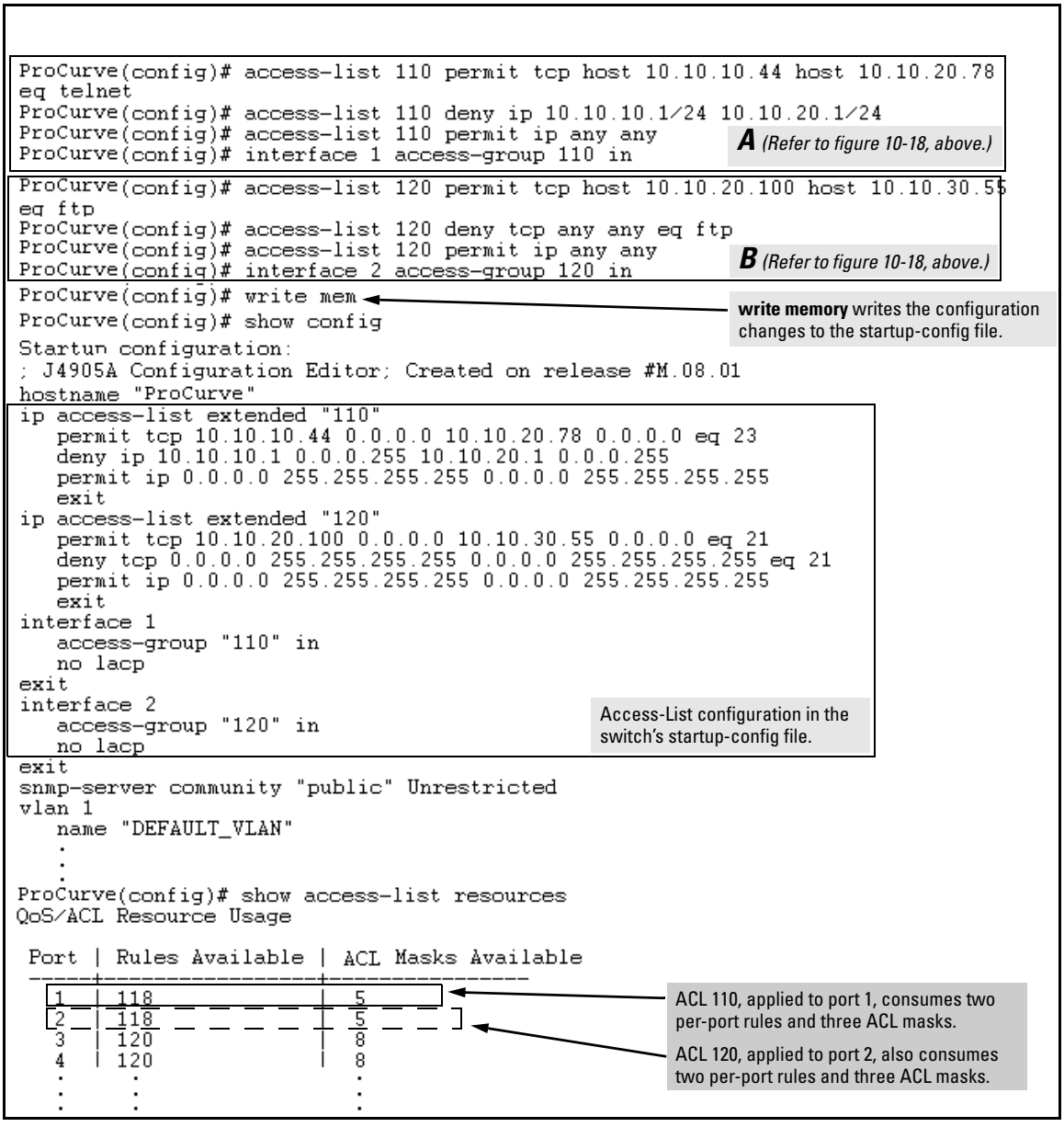


Figure 10-19. Example of Configuration Commands for an Extended ACL

Configuring a Named ACL

You can use the "Named ACL" context to configure a standard or extended ACL with an alphanumeric name instead of a number. Note that the command structure for configuring a named ACL differs from that for a numbered ACL.

Syntax: ip access-list standard < *name-str* | 1-99 >
< deny | permit >
< any | host < *src-ip-addr* > | *ip-addr / mask-length* >
[log]

ip access-list extended < *name-str* | 100-199 >
< deny | permit > ip
< any | host < *src-ip-addr* > | *ip-addr / mask-length* >
< any | host < *dest-ip-addr* > | *ip-addr / mask-length* >
[log]

ip access-list extended < *name-string* >
< deny | permit > < tcp | udp >
< any | host < *src-ip-addr* > | *ip-addr / mask-length* >
[*oper* < *src-port tcp/udp-id* >]
< any | host < *dest-ip-addr* > | *ip-addr / mask-length* >
[*oper* < *dest-port tcp/udp-id* >]
[log]

These commands create an ACE in the named ACL list and:

- *Indicate the action (deny or permit) to take on a packet if there is a match between a packet and the criteria in the complete ACE.*
- *Specify the packet protocol type (IP, TCP, or UDP) and (if TCP or UDP) the comparison operator.*
- *Specify the source and destination addressing options required for a match.*
- *Allow optional ACL logging where a packet has a match with a* **deny** *ACE. The* **log** *option does not appear when* **permit** *is the action.*

If the ACL does not already exist, these commands create the specified ACL and its first ACE. If the ACL already exists, these commands add a new, explicit ACE to the end of the ACL. For a match to occur, the packet must have the source and destination IP addressing criteria specified by this command, as well as any protocol-specific (TCP or UDP port number) criteria specified by the command.

< *name-str* | 1-99 | 100-199 >

Consists of an alphanumeric string of up to 64 case-sensitive characters. If you include a space in the string, you must also enclose the string with quotes. For example, **"ACL # 1"**. *You can also enter numbers in the ranges associated with standard (1-99) and extended (100-199) ACLs.*

For explanations of the individual parameters in the preceding syntax statements, refer to the syntax descriptions under "Configuring and Assigning a Numbered, Standard ACL" on page 10-47 or "Configuring and Assigning a Numbered, Extended ACL" on page 10-52.

For example, figure 10-20 shows the commands for creating an ACL in the "Named ACL" context with these parameters:

ACL Name:	150
Action:	Deny
Protocol:	TCP
Source IP Address and Mask	10.10.20.100 0.0.0.0
Destination IP Address and Mask	10.10.10.1 0.0.0.255
Protocol Operator and Port Number at Destination	eq telnet

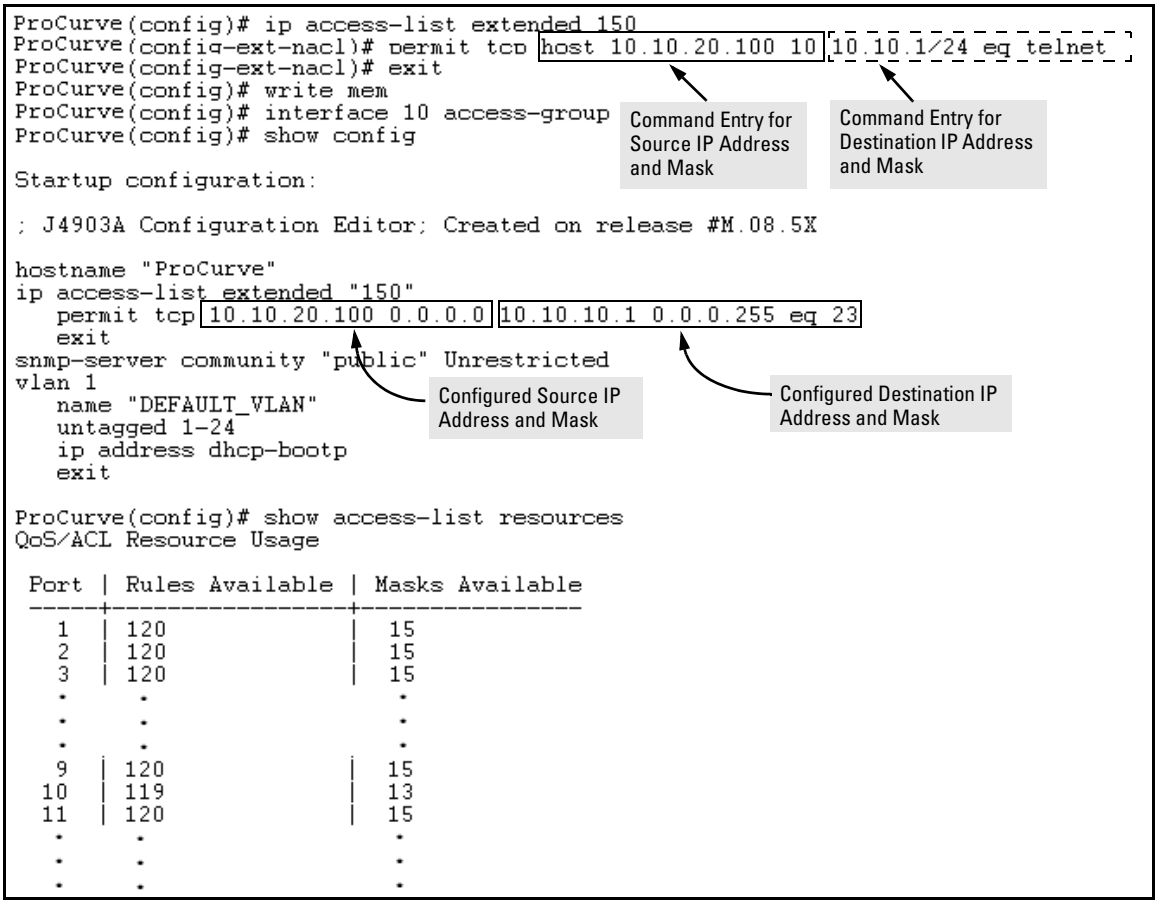


Figure 10-20. Using the "Named ACL" Context To Configure an ACL

Enabling or Disabling ACL Filtering on an Interface

You can configure one ACL to filter inbound traffic on multiple interfaces. For limits and operating rules, refer to "ACL Configuration and Operating Rules" on page 10-30.

Syntax: [no] interface < *port-list* > ip access-group < *ascii-string* > in
where: < *ascii-string* > = either a ACL name or an ACL ID number.

Assigns an ACL to a physical interface, which can be any combination of ports and/or trunks that do not already have an ACL assignment. You can use either the global configuration level or the interface context level to assign an ACL to an interface or remove an ACL from an interface.

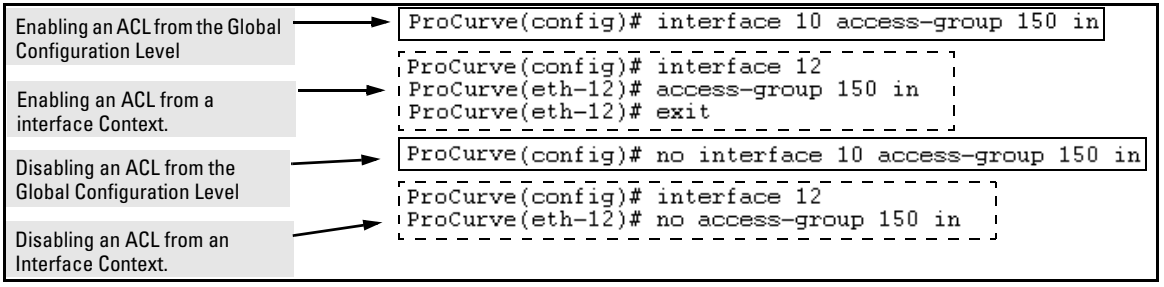


Figure 10-21. Methods for Enabling and Disabling ACLs

Deleting an ACL from the Switch

Syntax: no ip access-list standard < *name-str* | 1-99 >

no ip access-list extended < *name-str* | 100-199 >

Removes the specified ACL from the switch's running-config file.

Note: *You cannot delete an ACL from the switch while the ACL is assigned to any interfaces. Thus, before deleting an ACL from the switch, remove all assignments of the ACL to specific interfaces. If you need to delete an ACL assignment, refer to "Enabling or Disabling ACL Filtering on an Interface" on page 10-61.*

Displaying ACL Data

ACL Commands	Function	Page
show access-list	View a brief listing of all ACLs on the switch.	10-63
show access-list config	Display the ACL lists configured in the switch.	10-63
show access-list ports < all | < *interface* >>	List the name and type of ACLs assigned to all ports on the switch or to a particular port or static trunk configured on the switch.	10-64
show access-list < *acl-name-string* >	Display detailed content information for a specific ACL.	10-65
show access-list resources	Displays the currently available per-port rule and ACL mask resources.	10-67
show config	**show config** includes configured ACLs and assignments existing in the startup-config file.	
show running	**show running** includes configured ACLs and assignments existing in the running-config file.	

Display an ACL Summary

This command lists the configured ACLs, regardless of whether they are assigned to any interfaces.

Syntax: show access-list

List a summary table of the name, type, and application status of all ACLs configured on the switch.

For example:

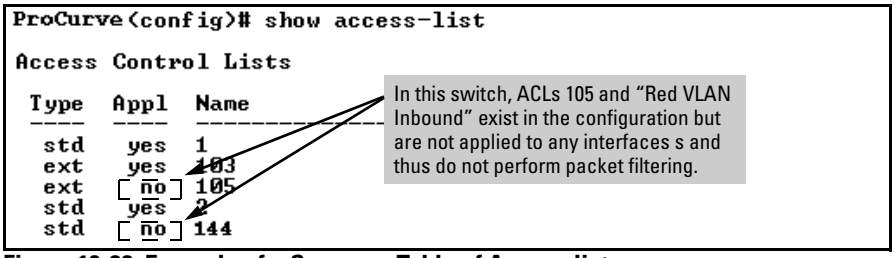


Figure 10-22. Example of a Summary Table of Access lists

Term	Meaning
Type	Shows whether the listed ACL is **std** (Standard; source-address only) or **ext** (Extended; protocol, source, and destination data).
Appl	Shows whether the listed ACL has been applied to an interface (**yes**/**no**).
Name	Shows the name or ID number assigned to each ACL configured in the switch.

Display the Content of All ACLs on the Switch

This command lists the configuration details for every ACL configured in the running-config file, regardless of whether you have assigned any to filter traffic on switch interfaces.

Syntax: show access-list config

List the configured syntax for all ACLs currently configured on the switch.

Note

Notice that you can use the output from this command for input to an offline text file in which you can edit, add, or delete ACL commands. Refer to "Editing ACLs and Creating an ACL Offline" on page 10-69.

This information also appears in the **show running** display. If you executed **write memory** after configuring an ACL, it appears in the **show config** display.

For example, with two ACLs configured in the switch, you will see results similar to the following:

```
ProCurve(config)# show access-list config
ip access-list standard "1"
   deny 18.28.236.77 0.0.0.0
   deny 18.29.140.107 0.0.0.0
   permit 0.0.0.0 255.255.255.255
   exit
ip access-list extended "105"
   permit tcp 18.30.133.27 0.0.0.0 0.0.0.0 255.255.255.255
   permit tcp 18.30.155.101 0.0.0.0 0.0.0.0 255.255.255.255
   deny ip 18.30.133.1 0.0.0.255 0.0.0.0 255.255.255.255
   deny ip 18.30.155.1 0.0.0.255 0.0.0.0 255.255.255.255
   permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
   exit
```

Figure 10-23. Example of an ACL Configured Syntax Listing

Display the ACL Assignments for an Interface

This command briefly lists the identification and type(s) of ACLs currently assigned to a particular interface (one or more ports and/or trunks) in the running-config file. (The switch allows up to one, inbound ACL assignment per interface.)

Syntax: show access-list ports < *interface* >

List the ACLs assigned to interfaces in the running config file.

Note

This information also appears in the **show running** display. If you executed **write memory** after configuring an ACL, it appears in the **show config** display.

For example, if you assigned a standard ACL with an ACL-ID of "1" to filter inbound traffic on port 10, you could quickly verify this assignment as follows:

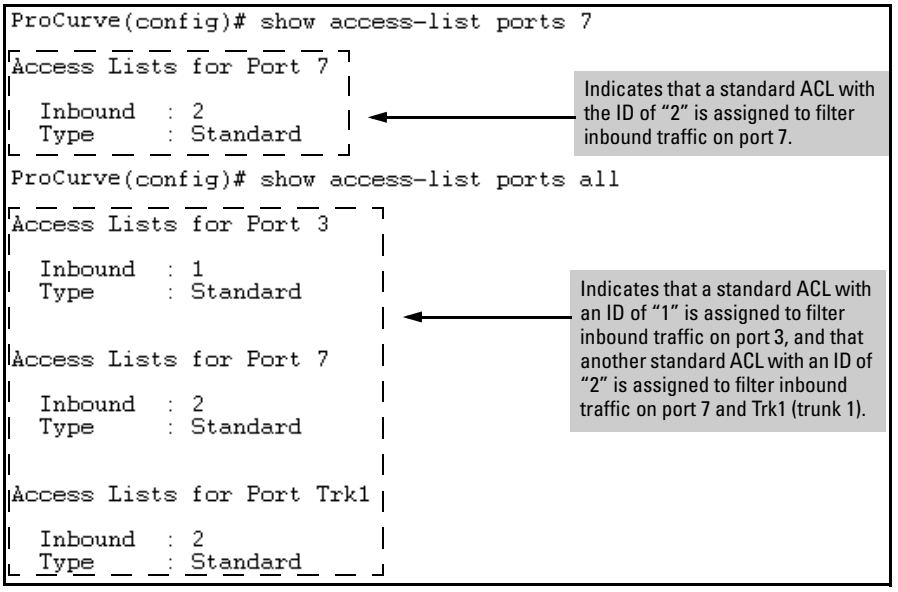


Figure 10-24. Example of Listing the ACL Assignment for an Interface

Displaying the Content of a Specific ACL

This command displays a specific ACL configured in the running config file in an easy-to-read tabular format.

Note

This information also appears in the **show running** display. If you executed **write memory** after configuring an ACL, it appears in the **show config** display.

Syntax: show access-list < *acl-name-string* >

Display detailed information on the content of a specific ACL configured in the running-config file.

For example, suppose you configured the following two ACLs in the switch:

ACL ID	ACL Type	Desired Action
1	Standard	• Deny IP traffic from 18.28.236.77 and 18.29.140.107. • Permit IP traffic from all other sources.
105	Extended	• Permit any TCP traffic from 18.30.133.27 to any destination. • Deny any other IP traffic from 18.30.133.(1-255). • Permit all other IP traffic from any source to any destination.

Inspect the ACLs as follows:

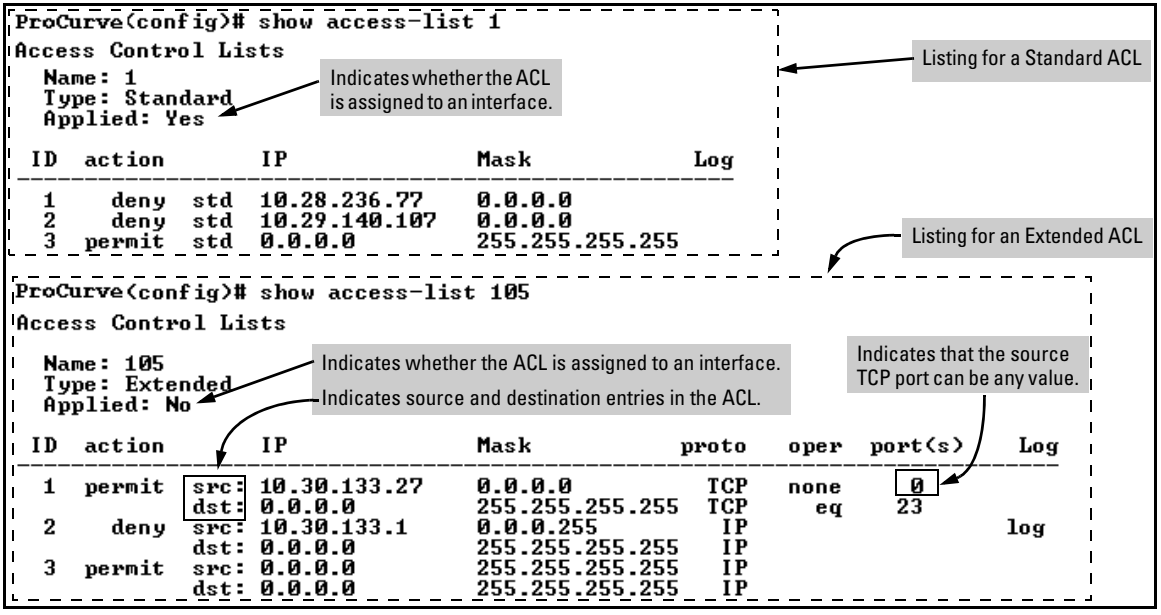


Figure 10-25. Examples of Listings Showing the Content of Standard and Extended ACLs

Table 10-9. Descriptions of Data Types Included in Show Access-List < *interface* > Output

Field	Description
Name	The ACL identifier. Can be a number from 1 to 199, or a name.
Type	Standard or Extended. The former uses only source IP addressing. The latter uses both source and destination IP addressing and also allows TCP or UDP port specifiers.
Applied	"Yes" means the ACL has been applied to an interface. "No" means the ACL exists in the switch configuration, but has not been applied to any interfaces, and is therefore not in use.
ID	The sequential number of the Access Control Entry (ACE) in the specified ACL.
action	Permit (forward) or deny (drop) a packet when it is compared to the criteria in the applicable ACE and found to match.
IP	**In Standard ACLs:** The source IP address to which the configured mask is applied to determine whether there is a match with a packet. **In Extended ACLs:** The source and destination IP addresses to which the corresponding configured masks are applied to determine whether there is a match with a packet.
Mask	The mask configured in an ACE and applied to the corresponding IP address in the ACE to determine whether a packet matches the filtering criteria.
proto	Used only in extended ACLs to specify the packet protocol type to filter. Must be either IP, TCP, or UDP.
oper	Used only in extended ACLs where a TCP or UDP port type and number have been entered. Specifies how to compare the corresponding TCP or UDP port number in a packet to the port number in the ACE.
port(s)	Used only in extended ACLs to show any TCP or UDP port number that has been entered in the ACE.
Log	Shows the status of logging for the entry (ACE). A blank space indicates ACL logging is not enabled for that ACE.

Displaying the Current Per-Port ACL Resources

Assigning an ACL to one or more interfaces reduces the available per-port rule and mask resources for those interfaces. (An unassigned ACL does not affect the rule and mask count.) This command displays the current per-port rule and mask resources available on the switch. For more information on rule and mask usage, refer to "Planning an ACL Application on a Series 3400cl or Series 6400cl Switch" on page 10-17.

Syntax: show access-list resources

Displays the currently available per-port rules and ACL masks on the switch. Note that the available rules can be used by ACL assignments, QoS configurations, Rate-Limiting configurations, and enabling IGMP. For more information, refer to "ACL Resource Usage and Monitoring" on page 10-18.

```
ProCurve(config)# show access-list resources
QoS/ACL Resource Usage

 Port  | Rules Available | ACL Masks Available
 ------+-----------------+-------------------
   1   | 119             |  6
   2   | 120             |  8
   3   | 120             |  8
   4   | 120             |  8
   5   | 120             |  8
   6   | 120             |  8
   7   | 120             |  8
   8   | 120             |  8
   9   | 120             |  8
  10   | 120             |  8
  11   | 120             |  8
  12   | 120             |  8
  13   | 120             |  8
  14   | 120             |  8
  15   | 120             |  8
  16   | 120             |  8
  17   | 120             |  8
  18   | 120             |  8
  19   | 120             |  8
  20   | 120             |  8
  21   | 120             |  8
  22   | 120             |  8
  23   | 120             |  8
  24   | 120             |  8

Maximum Rules per-port : 120
Maximum Masks per-port : 8
```

Indicates that one rule and two masks have been used. All other ports show the default quantity of rules and masks, which means that there are no ACLs or QoS assigned to these other ports on the switch.

Note: Because ACLs and QoS use the same rule resources in the switch, **show access-list resources** and **show qos resources** both list the same resource table. This table indicates the combined resource use of both features (plus Rate-Limiting and IGMP (if configured). Refer to page 10-18.)

Figure 10-26. Example of a Show Access-List Resources Command Output

Display All ACLs and Their Assignments in the Switch Startup-Config File and Running-Config File

The **show config** and **show running** commands include in their listings any configured ACLs and any ACL assignments to interfaces. Refer to figure 10-16 (page 10-50) and figure 10-17 (page 10-51) for examples. Remember that **show config** lists the startup-config file and **show running** lists the running-config file.

Editing ACLs and Creating an ACL Offline

Earlier sections of this chapter describe how to use the CLI to create an ACL. Beginning with "Using the CLI To Edit ACLs", below, describes how to use the CLI to edit existing ACLs. However, you can also create or edit an ACL offline, then use a TFTP server to upload the ACL as a command file. The offline method (page 10-72) provides a useful alternative to using the CLI for creating or editing large ACLs.

Using the CLI To Edit ACLs

The switch applies individual ACEs in the order in which they occur in an ACL. You can use the CLI to delete individual ACEs from anywhere in an ACL and to append new ACEs to the end of an ACL. However, the CLI method does not allow you to insert a new ACE between two existing ACEs.

Note

Before editing an assigned ACL, you must use the **no interface < *interface* > access-group < *acl-#* > in** command to remove the ACL from all interfaces to which it is assigned.

Using the CLI To Edit a Short ACL. To insert a new ACE between existing ACEs in a short ACL, you may want to delete the ACL and then re-configure it by entering your updated list of ACEs in the correct order.

Using the CLI to Edit a Longer ACL. To insert a new ACE between existing ACEs in a longer ACL:

a. Delete the first ACE that is out of sequence and all following ACEs through the end of the ACL.

b. Re-Enter the desired ACEs in the correct sequence.

General Editing Rules

- You can delete any ACE from an ACL by repeating the ACE's entry command, preceded by the "no" statement. When you enter a new ACE, the switch inserts it as the last entry of the specified ACL.
- Deleting the last ACE from a *numeric* ACL, removes the ACL from the configuration. Deleting the last ACE from a *named* ACL leaves the ACL in memory. In this case, the ACL is "empty" and cannot perform any filtering tasks. (In any ACL the implicit "deny any" does not apply unless the ACL includes at least one explicit ACE.)
- When you create a new ACL, the switch inserts it as the last ACL in the startup-config file. (Executing **write memory** saves the running-config file to the startup-config file.)

Deleting Any ACE from an ACL

You can delete an ACE from an ACL by repeating the ACE's entry command, preceded by the "**no**" statement.

Syntax: no access-list < *interface* > < permit | deny > < any | host | *ip-addr/mask-length* >

Deletes an ACE from a standard ACL. All variable parameters in the command must be an exact match with their counterparts in the ACE you want to delete.

no access-list < *interface* > < permit | deny > < ip | tcp | udp >
< src-addr: any | host | *ip-addr/mask-length* > [operator < src-port-num >]
< dest-addr: any | host | ip-addr-mask-length > [operator < dest-port-num >
[log]

Deletes an ACE from a standard ACL. All variable parameters in the command must be an exact match with their counterparts in the ACE you want to delete.

For example, the first of the following two commands creates an ACE in ACL 22 and the second deletes the same ACE:

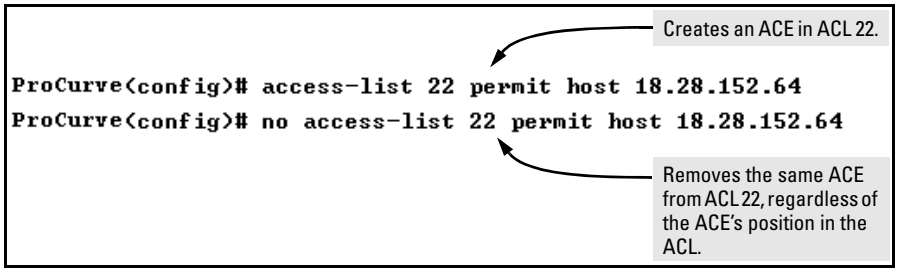


Figure 10-27. Example of Deleting an ACE from a Standard ACL

Figure 10-28 shows an example of deleting an ACE from an extended ACL.

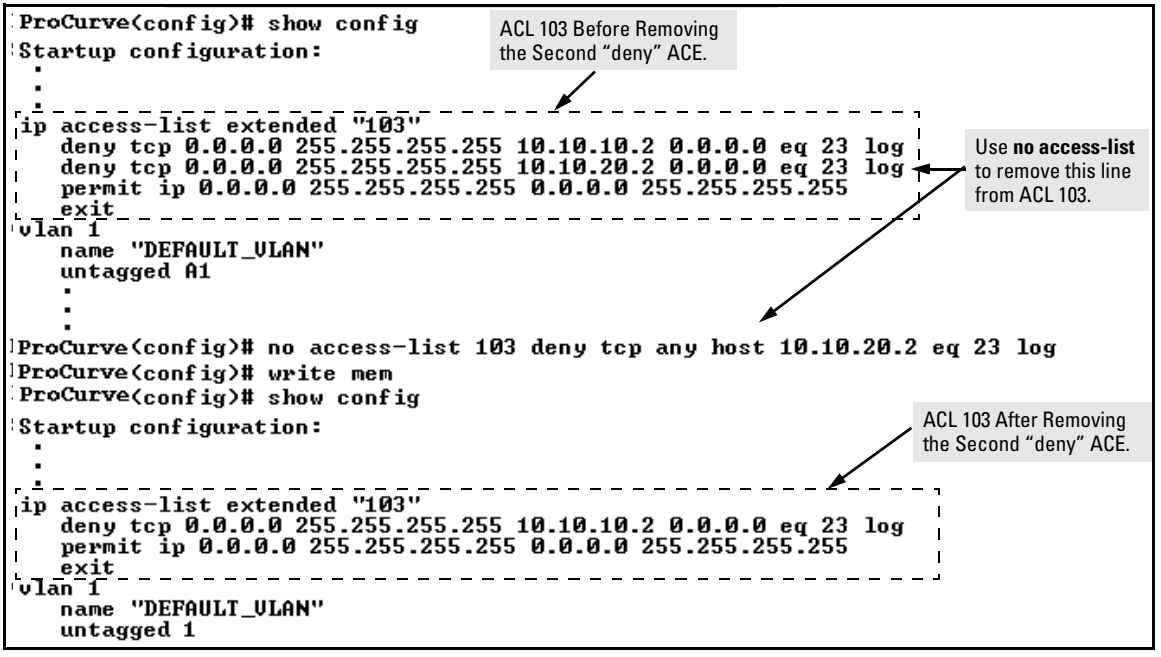


Figure 10-28. Example of Deleting an ACE from an ACL

Working Offline To Create or Edit an ACL

Note

When creating an ACL offline, ensure that the interfaces to which you plan to assign the ACL will have adequate per-port rules and ACL masks available. Note that if you attempt to apply an ACL to multiple interfaces and one of those interfaces does not have sufficient resources to support the ACL, the command will fail for all specified interfaces. For more on per-port ACL resources, refer to "Planning an ACL Application on a Series 3400cl or Series 6400cl Switch" on page 10-17.

For longer ACLs that would be difficult or time-consuming to accurately create or edit in the CLI, you can use the offline method:

1. Begin by doing one of the following:
   - To edit one or more existing ACLs, use **copy command-output tftp** to copy the current version of the ACL configuration to a file in your TFTP server. For example, to copy the ACL configuration to a file named **acl02.txt** in the TFTP directory on a server at 10.28.227.2:

     ```
     ProCurve# copy command-output 'show access-list
     config' tftp 10.28.227.2 acl02.txt pc
     ```
   - To create a new ACL, just open a text file in the appropriate directory on a TFTP server accessible to the switch.
2. Use the text editor to create or edit the ACL(s).
3. Use **copy tftp command-file** to download the file as a list of commands to the switch.

Creating an ACL Offline

Use a text editor that allows you to create an ASCII text file (.txt).

If you are replacing an ACL on the switch with a new ACL that uses the same number or name syntax, begin the command file with a "no" command to remove the earlier version of the ACL from the switch's running-config file. Otherwise, the switch will append the new ACEs in the ACL you download to the existing ACL. For example, if you plan to use the Copy command to *replace* ACL "103", you would place this command at the beginning of the edited file:

```
no ip access-list extended 103
```

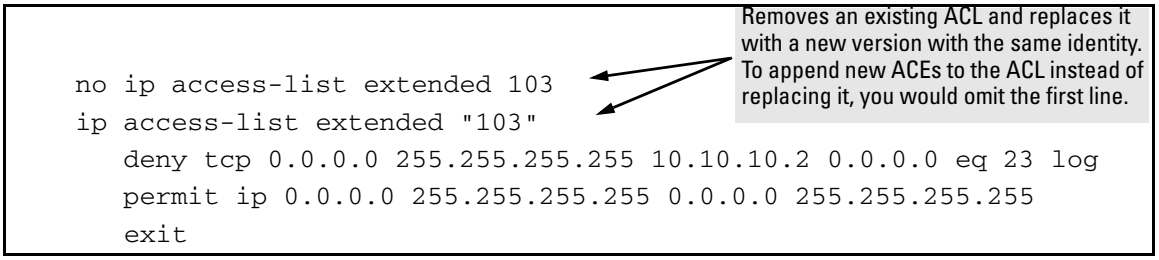


Figure 10-29. Example of an Offline ACL File Designed To Replace An Existing ACL

For example, suppose that you wanted to create an extended ACL to fulfill the following requirements (Assume a subnet mask of 255.255.255.0.):

- ID: 160
- Deny Telnet access to a server at 10.10.10.100 from these three IP addresses on port 2 (with ACL logging):
  - 10.10.20.17
  - 10.10.20.23
  - 10.10.20.40
- Allow any inbound access from all other addresses on port 2:
- Permit internet access to the following two IP addresses through port 24, but deny access to all other addresses through this port (without ACL logging).
  - 10.10.20.98
  - 10.10.20.21
- Deny all traffic from port 3 to the server at 10.10.10.100 (without ACL logging).
- Deny all traffic from port 5 to the server at 10.10.10.100 (without ACL logging), but allow any other traffic from port 5.

1. To create an ACL offline for the above requirements, you would create a **.txt** file with the content shown in figure 10-30.

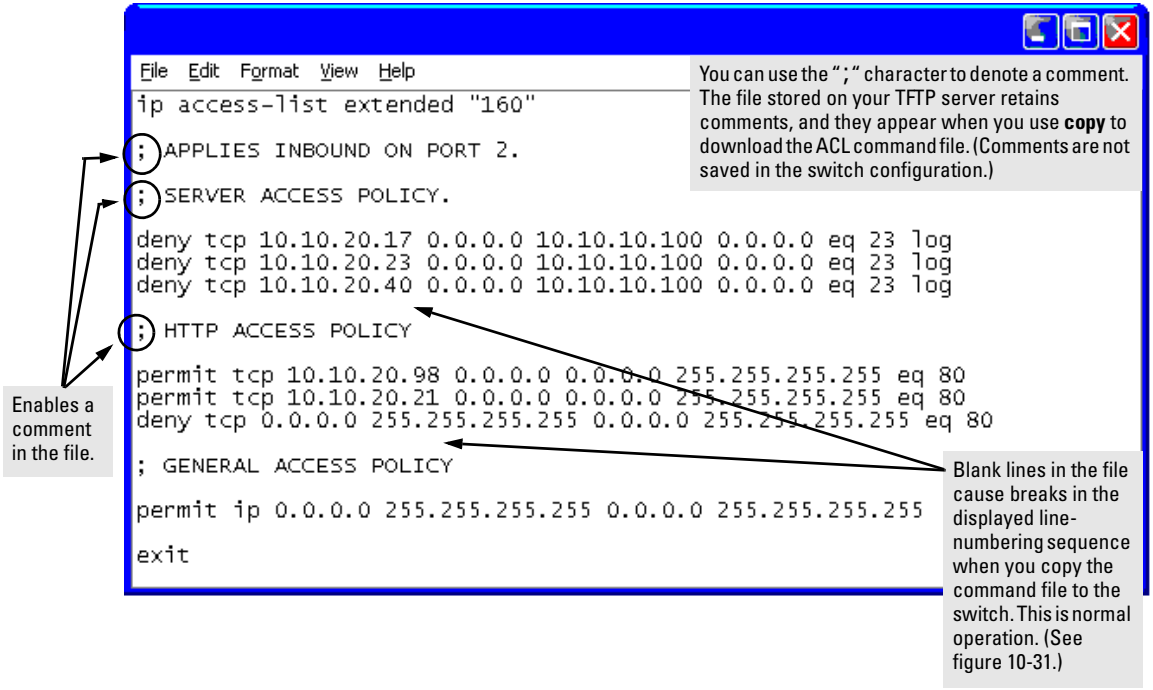


Figure 10-30. Example of a.txt File Designed for Creating an ACL

2. After you copy the above .txt file to a TFTP server the switch can access, you would then execute the following command to download the file to the switch's startup-config file:

```
ProCurve(config)# copy tftp command-file 13.28.234.180 list-160.txt
Running configuration may change, do you want to continue [y/n]?  y
  1. ip access-list extended "160"
  3. ; APPLIES INBOUND ON PORT 2.
  5. ; SERVER ACCESS POLICY.
  7. deny tcp 10.10.20.17 0.0.0.0 10.10.10.100 0.0.0.0 eq 23 log
  8. deny tcp 10.10.20.23 0.0.0.0 10.10.10.100 0.0.0.0 eq 23 log
  9. deny tcp 10.10.20.40 0.0.0.0 10.10.10.100 0.0.0.0 eq 23 log
 11. ; HTTP ACCESS POLICY
 13. permit tcp 10.10.20.98 0.0.0.0 0.0.0.0 255.255.255.255 eq 80
 14. permit tcp 10.10.20.21 0.0.0.0 0.0.0.0 255.255.255.255 eq 80
 15. deny tcp 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255 eq 80
 17. ; GENERAL ACCESS POLICY
 19. permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
 21. exit
```

Figure 10-31. Example of Using "copy tftp command-file" To Configure an ACL in the Switch

Note

If a transport error occurs, the switch does not execute the command and the ACL is not configured.

3. Next, assign the new ACL to the intended interface which, in this example, is for port 2.

   ```
   ProCurve(config)# interface 2 access-group 160 in
   ```

4. Inspect the effect of the ACL on the switch's per-port resources.

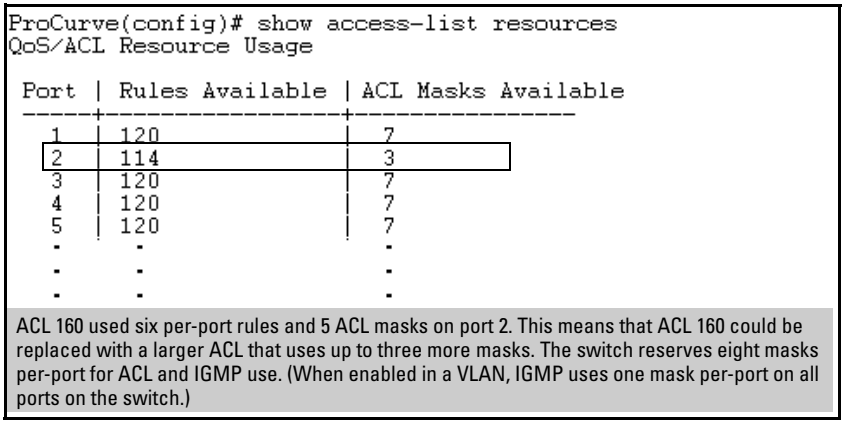


Figure 10-32. Inspection of Per-Port Resource Usage After Assigning an ACL

5. Inspect the new running configuration:

   ```
   ProCurve(config)# show running
   ```

6. If the configuration appears satisfactory, save it to the startup-config file:

   ```
   ProCurve(config)# write memory
   ```

Enable ACL "Deny" Logging

ACL logging enables the switch to generate a message when IP traffic meets the criteria for a match with an ACE that results in an explicit "deny" action. You can use ACL logging to help:

- Test your network to ensure that your ACL configuration is detecting and denying the traffic you do not want forwarded

- Receive notification when the switch detects attempts to transmit traffic you have designed your ACLs to reject

The switch sends ACL messages to Syslog and optionally to the current console, Telnet, or SSH session. You can configure up to six Syslog server destinations.

Requirements for Using ACL Logging

- The switch configuration must include an ACL (1) assigned to an interface and (2) containing an ACE configured with the **deny** action and the **log** option.
- To screen routed packets with destination IP addresses outside of the switch, IP routing must be enabled.
- For ACL logging to a Syslog server, the server must be accessible to the switch and identified (with the **logging < *ip-addr* >** command) in the switch configuration.
- Debug must be enabled for ACLs and one or both of the following:
  - logging (for sending messages to Syslog)
  - Session (for sending messages to the current console interface)

ACL Logging Operation

When the switch detects a packet match with an ACE and the ACE includes both the **deny** action and the optional **log** parameter, an ACL log message is sent to the designated debug destination. The first time a packet matches an ACE with **deny** and **log** configured, the message is sent immediately to the destination and the switch starts a wait-period of approximately five minutes. (The exact duration of the period depends on how the packets are internally routed.) At the end of the collection period, the switch sends a single-line summary of any additional "deny" matches for that ACE (and any other "deny" ACEs for which the switch detected a match). If no further log messages are generated in the wait-period, the switch suspends the timer and resets itself to send a message as soon as a new "deny" match occurs. The data in the message includes the information illustrated in figure 10-33.

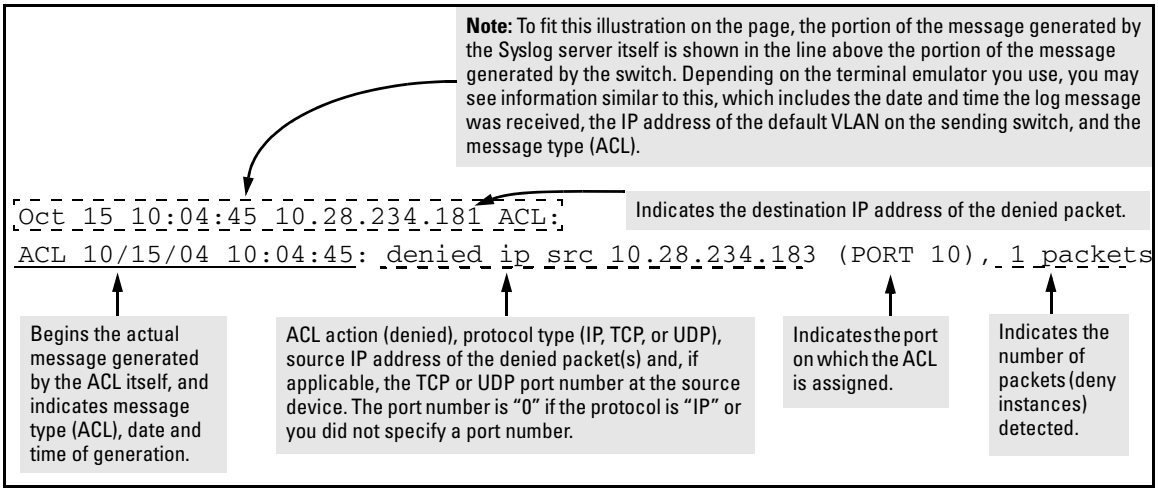


Figure 10-33. Example of the Content of an ACL-Generated Message

Enabling ACL Logging on the Switch

1. Use the debug command to:
   a. Configure one or more log destinations.
   b. If you are using a Syslog server, use the **logging** command to configure the server's IP address. (You can configure up to six Syslog servers.)
   c. Ensure that the switch can access any Syslog servers you specify.
2. Configure one or more ACLs with the deny action and the log option.

For example, suppose that you want to do the following:

- On port 10, configure an extended ACL with an ACL-ID of 143 to deny Telnet traffic from IP address 10.38.100.127.
- Configure the switch to send an ACL log message to the console and to a Syslog server at IP address 10.38.110.54 on port 11 if the switch detects a match denying Telnet access from 10.38.100.127.

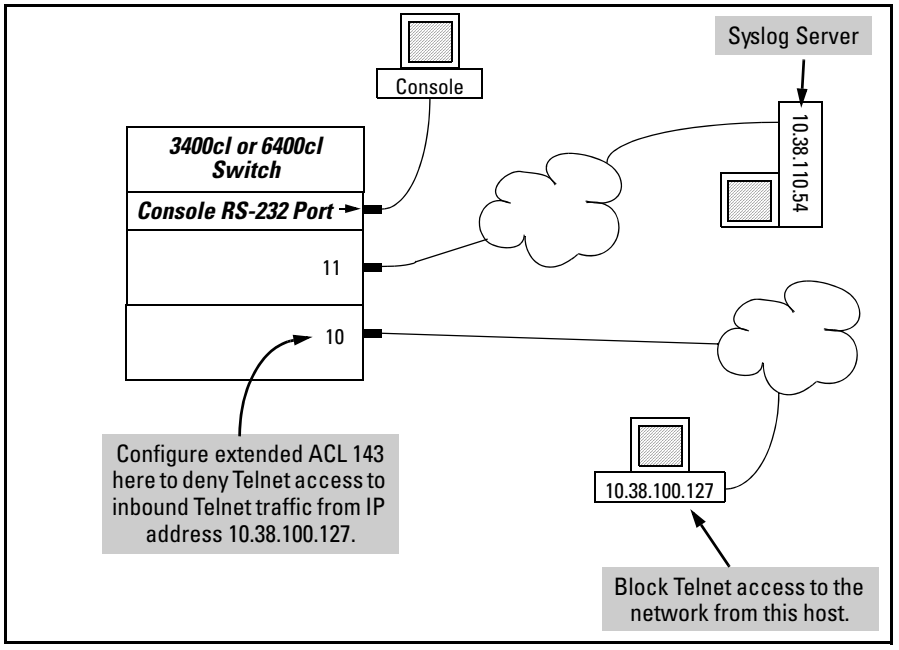


Figure 10-34. Example of an ACL Log Application

```
ProCurve(config)# access-list 143 deny tcp host 10.38.100.127 any eq telnet log
ProCurve(config)# access-list 143 permit ip any any
ProCurve(config)# interface 10 access-group 143 in
ProCurve(config)# logging 10.38.110.54
ProCurve(config)# debug acl
ProCurve(config)# debug destination logging
ProCurve(config)# debug destination session
ProCurve(config)# write memory
ProCurve(config)# show debug
Debug Logging
 Destination:
  Logging
    10.38.110.54
  Session
 Enabled debug types:
  event
  acl log
```

Figure 10-35. Commands for Applying an ACL with Logging to Figure 10-34

Operating Notes for ACL Logging

- The ACL logging feature generates a message only when packets are explicitly denied as the result of a match, and not when explicitly permitted or implicitly denied. To help test ACL logging, configure an ACL with an explicit **deny any** and **log** statements at the end of the list, and apply the ACL to an appropriate interface.
- Logging enables you to selectively test specific devices or groups. However, excessive logging can affect switch performance. For this reason, HP recommends that you remove the logging option from ACEs for which you do not have a present need. Also, avoid configuring logging where it does not serve an immediate purpose. (Note that ACL logging is not designed to function as an accounting method.) See also "Apparent Failure To Log All "Deny" Matches" in the section titled "ACL Problems", found in appendix C, "Troubleshooting" of the Management and Configuration Guide for your switch.
- When configuring logging, you can reduce excessive use by configuring the appropriate ACEs to match with specific hosts instead of entire subnets.

General ACL Operating Notes

ACLs do not provide DNS hostname support.

Protocol Support: ACL criteria includes IP, TCP, and UDP. ACLs do not use these protocols:

- TOS (Type-of-Service)
- Precedence
- MAC information
- QoS

ACLs do not affect switch serial port access.

ACLs filter both Layer 2 and Layer 3 on a port.

There is no performance degradation with ACLs enabled; traffic is at line rate.

When the ACL configuration includes TCP or UDP options, the switch operates in "strict" TCP and UDP mode for increased control. The switch compares all TCP and UDP packets against the ACLs. (In the ProCurve Series 9300 Routing Switches, the Strict TCP and Strict UDP modes are optional and must be specifically invoked.)

Replacing or Adding To an Active ACL Policy. If you assign an ACL to an interface and subsequently want to add or replace ACEs in that ACL, you must first remove the ACL from all assigned interfaces.

Note

When an ACE becomes active, it screens the packets resulting from new traffic connections. It does not screen packets resulting from currently open traffic connections. If you invoke a new ACE to screen packets in a currently open traffic connection, you must force the connection to close before the ACE can begin screening packets from that source.

ACLs Do Not Filter Traffic Generated by the Switch. Because ACLs on the 3400cl/6400cl switches filter only inbound traffic at the inbound physical port, outbound traffic from any source is not filtered by any ACL(s) configured on the switch. Filtering of such traffic must be done at a downstream device.

`<` *acl-list-#* `>: Unable to apply access control list.`

The indicated ACL cannot be applied to an interface because an ACL is already assigned to the interface. The command fails for all included interfaces, including any that do not already have an ACL assigned.

`Duplicate access control entry.`

The switch detects an attempt to create a duplicate ACE in the same ACL.

—This page is intentionally unused—