



CHAPTER 1

Configuring Security Access Control Lists

This chapter describes how to configure security access control lists (ACLs) on your Cisco 4700 Series Application Control Engine (ACE) appliance. ACLs provide basic security for your network by filtering traffic and controlling network connections. This chapter contains the following major sections:

- [ACL Overview](#)
- [ACL Configuration Quick Start](#)
- [Configuring ACLs](#)
- [Simplifying Access Control Lists with Object Groups](#)
- [Applying an ACL to an Interface](#)
- [Applying an ACL Globally to All Interfaces in a Context](#)
- [Filtering Traffic with an ACL](#)
- [ACL Configuration Examples](#)
- [Displaying ACL Configuration Information and Statistics](#)
- [Clearing ACL Statistics](#)

ACL Overview

An ACL consists of a series of statements called ACL entries that define the network traffic profile. Each entry permits or denies network traffic (inbound and outbound) to the parts of your network specified in the entry. Each entry also contains a filter element that is based on criteria such as the source address, the destination address, the protocol, and protocol-specific parameters such as ports and so on.

An implicit deny-all entry exists at the end of each ACL, so you must configure an ACL on each interface that you want to permit connections. Otherwise, the ACE denies all traffic on the interface.

ACLs allow you to control network connection setups rather than processing each packet. Such ACLs are commonly referred to as security ACLs.

You can configure ACLs as parts of other features (for example, security, Network Address Translation (NAT), server load balancing (SLB), and so on). The ACE merges these individual ACLs into one large ACL called a merged ACL. The ACL compiler then parses the merged ACL and generates the ACL lookup mechanisms. A match on this merged ACL can result in multiple actions.

For example, one use of ACLs could be to permit all e-mail traffic on a VLAN, but block Telnet traffic. You can also use ACLs to allow one client to access a part of the network and prevent another client from accessing that same area.

When configuring ACLs, you must apply an ACL to an interface to control traffic on that interface. Applying an ACL on an interface assigns the ACL and its entries to that interface.

You can apply only one extended ACL to each direction (inbound or outbound) of an interface. You can also apply the same ACL on multiple interfaces. You can apply EtherType ACLs only in the inbound direction and only on Layer 2 interfaces.

This section contains the following topics:

- [ACL Types and Uses](#)
- [ACL Guidelines](#)

ACL Types and Uses

You can configure the following two types of ACLs on the ACE:

- Extended—Control network access for IP traffic
- EtherType—Control network access for non-IP traffic

**Note**

The ACE does not explicitly support standard ACLs. To configure a standard ACL, specify the destination address as **any** and do not specify ports in an extended ACL. For details about configuring an extended ACL, see the [“Configuring an Extended ACL”](#) section.

ACL Guidelines

This section describes the guidelines to observe when you configure and use ACLs in your network. This section contains the following topics:

- [ACL Entry Order](#)
- [ACL Implicit Deny](#)
- [Maximum Number of ACLs and ACL Entries](#)

ACL Entry Order

An ACL consists of one or more entries. Depending on the ACL type, you can specify the source and destination addresses, the protocol, the ports (for TCP or UDP), the ICMP type, the ICMP code, or the EtherType as the match criteria. By default, the ACE appends each ACL entry at the end of the ACL. You can also specify the location of each entry within an ACL.

The order of the entries is important. When the ACE decides whether to accept or refuse a connection, the ACE tests the packet against each ACL entry in the order in which the entries are listed. After it finds a match, the ACE does not check any more entries. For example, if you create an entry at the beginning of an ACL that explicitly permits all traffic, the ACE does not check any other statements in the ACL.

ACL Implicit Deny

All ACLs have an implicit deny entry at the end of the ACL, so, unless you explicitly permit it, traffic cannot pass. For example, if you want to allow all users to access a network through the ACE except for those users with particular IP addresses, then you must deny the particular IP addresses in one entry and permit all other IP addresses in another entry.

Maximum Number of ACLs and ACL Entries

The ACE supports a maximum of 8192 unique ACLs and 40,000 ACL entries. Some ACLs use more memory than others, such as an ACL that uses large port number ranges or overlapping networks (for example, one entry specifies 10.0.0.0/8 and another entry specifies 10.1.1.0/24). Depending on the type of ACL, the actual limit that the ACE can support may be less than 40,000 entries.

If you use object groups in ACL entries, you enter fewer actual ACL entries, but the same number of *expanded* ACL entries as you did when you entered entries without object groups. Expanded ACL entries count toward the system limit. To view the number of expanded ACL entries in an ACL, use the **show access-list name** command.

If you exceed the memory limitations of the ACE, the appliance generates a syslog message and increments the Download Failures counter in the output of the **show interface vlan number** command. The configuration remains in the running-config file and the interface stays enabled. The ACL entries stay the same as they were before the failing configuration was attempted.

For example, if you add a new ACL with ten entries, but the addition of the sixth entry fails because the ACE runs out of memory, the ACE removes the five entries that you successfully entered.

ACL Configuration Quick Start

[Table 1-1](#) provides a quick overview of the steps required to configure ACLs. Each step includes the CLI command or a reference to the procedure required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following [Table 1-1](#).

Table 1-1 ACL Configuration Quick Start

Task and Command Example

1. If you are operating in multiple contexts, observe the CLI prompt to verify that you are operating in the desired context. If necessary, change to the correct context.

```
host1/Admin# changeto c1  
host1/C1#
```

The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco 4700 Series Application Control Engine Appliance Virtualization Configuration Guide*.

2. Enter configuration mode.

```
host1/Admin# config  
host1/Admin(config)#
```

3. Create an ACL.

```
host1/Admin(config)# access-list INBOUND extended deny ip  
192.168.12.0 255.255.255.0 any
```

4. As required by your application, add entries to the ACL using the same access list name. For example, enter:

```
host1/Admin(config)# access-list INBOUND extended permit ip any  
any
```

5. Apply the ACL to an individual interface.

```
host1/Admin(config)# interface vlan 10  
host1/Admin(config-if)# access-group input INBOUND
```

6. Alternatively, you can apply an ACL globally to all interfaces in a context.

```
host1/Admin(config)# access-group input INBOUND
```

7. (Optional) Save your configuration changes to flash memory.

```
host1/Admin(config)# exit  
host1/Admin# copy running-config startup-config
```

8. Display and verify the ACL configuration information.

```
host1/Admin# show running-config access-list
```

Configuring ACLs

This section contains the following topics:

- [Configuring an Extended ACL](#)
- [Configuring Comments in an Extended ACL](#)
- [Configuring an EtherType ACL](#)
- [Resequencing Entries](#)

Configuring an Extended ACL

An extended ACL allows you to specify both the source and the destination IP addresses of traffic as well as the following parameters:

- Protocol
- TCP or UDP ports
- ICMP types and codes

You can specify these parameters directly when you use the **access-list** command or you can use object groups for each parameter. For more information about object groups, see the [“Simplifying Access Control Lists with Object Groups”](#) section.

For TCP, UDP, and ICMP connections, you do not need to apply an ACL on the destination interface to allow returning traffic, because the ACE allows all returning traffic for established connections.



Note

The ACE does not explicitly support standard ACLs. To configure a standard ACL, specify the destination address as **any** and do not specify the ports in an extended ACL.



Tip

Enter the ACL name in uppercase letters so that the name is easy to see in the configuration. You may want to name the ACL for the interface (for example, INBOUND) or for the purpose (for example, NO_NAT or VPN).

To create an extended ACL, use the **access-list extended** command in configuration mode. There are two major types of extended ACLs:

- Non-ICMP ACLs
- ICMP ACLs

You can permit or deny network connections based on the IP protocol, source and destination IP addresses, and TCP or UDP ports. To syntax of a non-ICMP extended ACL is as follows:

```
access-list name [line number] extended {deny | permit}
  {protocol {any | host src_ip_address | src_ip_address netmask
  | object-group net_obj_grp_name} [operator port1 [port2]] {any | host
  dest_ip_address | dest_ip_address netmask | object-group
  net_obj_grp_name} [operator port3 [port4]]}
  | {object-group service_obj_grp_name} {any | host src_ip_address
  | src_ip_address netmask | object-group net_obj_grp_name} {any | host
  dest_ip_address | dest_ip_address netmask | object-group
  net_obj_grp_name}
```

You can also permit or deny network connections based on the ICMP type (for example, echo, echo-reply, unreachable, and so on). To syntax of an ICMP extended ACL is as follows:

```
access-list name [line number] extended {deny | permit}
  {icmp {any | host src_ip_address | src_ip_address netmask |
  object-group net_obj_grp_name} {any | host dest_ip_address |
  dest_ip_address netmask | object-group network_grp_name}
  [icmp_type [code operator code1 [code2]]]}
  | {object-group service_obj_grp_name} {any | host src_ip_address
  | src_ip_address netmask | object-group net_obj_grp_name} {any | host
  dest_ip_address | dest_ip_address netmask | object-group
  net_obj_grp_name}
```

The keywords, options, and arguments are as follows:

- *name*—Unique identifier of the ACL. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *line number*—(Optional) Specifies the line number position where you want the entry that you are configuring to appear in the ACL. The position of an entry affects the lookup order of the entries in an ACL. If you do not configure the line number of an entry, the ACE applies a default increment and a line number to the entry and appends it at the end of the ACL.

- **extended**—Specifies an extended ACL. Extended ACLs allow you to specify the destination IP address and subnet mask and other parameters not available with a standard ACL.
- **deny**—Blocks connections on the assigned interface.
- **permit**—Allows connections on the assigned interface.
- *protocol*—Name or number of an IP protocol. Enter a protocol name or an integer from 0 to 255 that represents an IP protocol number from [Table 1-2](#).

Table 1-2 Supported Protocol Keywords and Numbers

Protocol Name	Protocol Number	Description
ah	51	Authentication Header
eigrp	88	Enhanced IGRP
esp	50	Encapsulated Security Payload
gre	47	Generic Routing Encapsulation
icmp	1	Internet Control Message Protocol
igmp	2	Internet Group Management Protocol
ip	any	Internet Protocol
ip-in-ip	4	IP-in-IP Layer 3 Tunneling Protocol
ospf	89	Open Shortest Path First
pim	103	Protocol Independent Multicast
tcp	6	Transmission Control Protocol
tcp-udp	6 and 17	TCP and UDP
udp	17	User Datagram Protocol

- **any**—Specifies network traffic from any source.
- **host *src_ip_address***—Specifies the IP address of the host from which the network traffic originates. Use this keyword and argument to specify the network traffic from a single IP address.
- ***src_ip_address netmask***—Traffic from a source defined by the IP address and the network mask. Use these arguments to specify network traffic from a range of source IP addresses.

- **object-group** *net_obj_grp_name*—Specifies the identifier of an existing network object group. For details, see the “[Simplifying Access Control Lists with Object Groups](#)” section.
- *operator*—(Optional) Operand used to compare source and destination port numbers for TCP, TCP-UDP, and UDP protocols. The operators are as follows:
 - **eq**—Equal to.
 - **gt**—Greater than.
 - **lt**—Less than.
 - **neq**—Not equal to.
 - **range**—An inclusive range of port values. If you enter this operator, enter a second port number value to define the upper limit of the range.
- *port1* [*port2*]—TCP or UDP source port name or number from which you permit or deny services access. Enter an integer from 0 to 65535. To enter an inclusive range of ports, enter two port numbers. *port2* must be greater than or equal to *port1*. See [Table 1-3](#) for a list of well-known TCP port names and numbers and [Table 1-4](#) for a list of well-known UDP port names and numbers.

Table 1-3 Well-Known TCP Port Numbers and Keywords

Keyword	Port Number	Description
aol	5190	America-Online
bgp	179	Border Gateway Protocol
chargen	19	Character Generator
citrix-ica	1494	Citrix Independent Computing Architecture Protocol
cmd	514	Same as exec, with automatic authentication
ctiqbe	2748	Computer Telephony Interface Quick Buffer Encoding
daytime	13	Daytime
discard	9	Discard
domain	53	Domain Name System

Table 1-3 Well-Known TCP Port Numbers and Keywords (continued)

Keyword	Port Number	Description
echo	7	Echo
exec	512	Exec (RSH)
finger	79	Finger
ftp	21	File Transfer Protocol
ftp-data	20	FTP data connections
gopher	70	Gopher
h323	1720	H.323 call signaling
hostname	101	NIC hostname server
http	80	Hypertext Transfer Protocol
https	443	HTTP over TLS/SSL
ident	113	Ident Protocol
imap4	143	Internet Message Access Protocol, version 4
irc	194	Internet Relay Chat
kerberos	88	Kerberos
klogin	543	Kerberos Login
kshell	544	Kerberos Shell
ldap	389	Lightweight Directory Access Protocol
ldaps	636	LDAP over TLS/SSL
login	513	Login (rlogin)
lotusnotes	1352	IBM Lotus Notes
lpd	515	Printer Service
matip-a	350	Mapping of Airline Traffic over Internet Protocol Type A
netbios-ssn	139	NetBIOS Session Service
nntp	119	Network News Transport Protocol
pcanywhere-data	5631	PC Anywhere data

Table 1-3 Well-Known TCP Port Numbers and Keywords (continued)

Keyword	Port Number	Description
pim-auto-rp	496	PIM Auto-RP
pop2	109	Post Office Protocol v2
pop3	110	Post Office Protocol v3
pptp	1723	Point-to-Point Tunneling Protocol, RFC 2637
rtsp	554	Real-Time Streaming Protocol
sip	5060	Session Initiation Protocol
skinny	2000	Cisco Skinny Client Control Protocol (SCCP)
smtp	25	Simple Mail Transfer Protocol
sqlnet	1521	Structured Query Language Network
ssh	22	Secure Shell
sunrpc	111	Sun Remote Procedure Call
tacacs	49	Terminal Access Controller Access Control System
talk	517	Talk
telnet	23	Telnet
time	37	Time
uucp	540	UNIX-to-UNIX Copy Program
whois	43	Nickname
www	80	World Wide Web (HTTP)

Table 1-4 Well-Known UDP Key Words and Port Numbers

Keyword	Port Number	Description
biff	512	Mail notification
bootpc	68	Bootstrap Protocol client

Table 1-4 Well-Known UDP Key Words and Port Numbers (continued)

Keyword	Port Number	Description
bootps	67	Bootstrap Protocol server
discard	9	Discard
dnsix	195	DNSIX Security protocol auditing (dn6-nlm-aud)
domain	53	Domain Name System
echo	7	Echo
isakmp	500	Internet Security Association Key Management Protocol
kerberos	88	Kerberos
mobile-ip	434	Mobile IP registration
nameserver	42	Host Name Server
netbios-dgm	138	NetBIOS datagram service
netbios-ns	137	NetBIOS name service
netbios-ssn	139	NetBIOS Session Service
ntp	123	Network Time Protocol
pcanywhere-status	5632	PC Anywhere status
radius	1812	Remote Authentication Dial-in User Service
radius-acct	1813	RADIUS Accounting
rip	520	Routing Information Protocol
snmp	161	Simple Network Management Protocol
snmptrap	162	SNMP Traps
sunrpc	111	Sun Remote Procedure Call
syslog	514	System Logger
tacacs	49	Terminal Access Controller Access Control System
talk	517	Talk

Table 1-4 Well-Known UDP Key Words and Port Numbers (continued)

Keyword	Port Number	Description
tftp	69	Trivial File Transfer Protocol
time	37	Time
who	513	Who service (rwho)
wsp	9200	Connectionless Wireless Session Protocol
wsp-wtls	9202	Secure Connectionless WSP
wsp-wtp	9201	Connection-based WSP
wsp-wtp-wtls	9203	Secure Connection-based WSP
xdmcp	177	X Display Manager Control Protocol

- *dest_ip_address netmask*—IP address of the network or host to which the packet is being sent and the network mask bits to be applied to the destination IP address. Use these arguments to specify a range of destination IP addresses.
- **any**—Specifies the network traffic that goes to any destination.
- **host dest_address**—Specifies the IP address and subnet mask of the destination of the packets in a flow. Use this keyword and argument to specify the network traffic destined to a single IP address.
- *operator*—(Optional) Operand used to compare source and destination port numbers for TCP and UDP protocols. The operators are as follows:
 - **lt**—Less than.
 - **gt**—Greater than.
 - **eq**—Equal to.
 - **neq**—Not equal to.
 - **range**—Inclusive range of port values. If you enter this operator, enter a second port number value to define the upper limit of the range.
- *port3 [port4]*—TCP or UDP destination port name or number to which you permit or deny services access. To enter an optional inclusive range of ports, enter two port numbers. *port4* must be greater than or equal to *port3*. See [Table 1-3](#) for a list of well-known ports.

- **object-group** *service_obj_grp_name*—(Optional) Specifies the identifier of an existing service object group. For details, see the “[Simplifying Access Control Lists with Object Groups](#)” section.
- *icmp_type*—(Optional) Type of ICMP messaging. Enter either an integer that corresponds to the ICMP code number or an ICMP type as described in [Table 1-5](#).

Table 1-5 ICMP Types

ICMP Code Number	ICMP Type
0	echo-reply
3	unreachable
4	source-quench
5	redirect
6	alternate-address
8	echo
9	router-advertisement
10	router-solicitation
11	time-exceeded
12	parameter-problem
13	timestamp-request
14	timestamp-reply
15	information-request
16	information-reply
17	mask-request
18	mask-reply
30	traceroute
31	conversion-error
32	mobile-redirect

- **code**—(Optional) Specifies that a numeric operator and ICMP code follows.

- *operator*—Operator that the ACE applies to the ICMP code that follows. Enter one of the following operators:
 - **lt**—Less than.
 - **gt**—Greater than.
 - **eq**—Equal to.
 - **neq**—Not equal to.
 - **range**—Inclusive range of ICMP code values. When you use this operator, specify two code numbers to define the range.
- *code1, code2*—ICMP code number that corresponds to an ICMP type. See [Table 1-5](#). If you entered the **range** operator, enter a second ICMP code value to define the upper limit of the range.

**Note**

For security reasons, the ACE does not allow pings from an interface on a VLAN on one side of the ACE through the appliance to an interface on a different VLAN on the other side of the appliance. For example, a host can ping the ACE address that is on the IP subnet using the same VLAN as the host but cannot ping IP addresses configured on other VLANs on the ACE.

For example, to configure a TCP extended ACL, enter:

```
host1/Admin(config)# access-list INBOUND line 10 extended permit tcp
192.168.12.0 255.255.255.0 gt 1024 172.27.16.0 255.255.255.0 lt 4000
```

For example, to remove an entry from an extended ACL, enter:

```
host1/Admin(config)# no access-list INBOUND line 10
```

To control a ping, specify **echo (8)** (host to ACE).

For example, to allow an external host with IP address 192.168.12.5 to ping a host behind the ACE with an IP address of 10.0.0.5, enter:

```
host1/Admin(config)# access-list INBOUND extended permit icmp host
192.168.12.5 host 10.0.0.5 echo code eq 0
```

For example, to remove an entry from an ICMP ACL, enter:

```
host1/Admin(config)# no access-list INBOUND extended permit icmp host
192.168.12.5 echo
```

Configuring Comments in an Extended ACL

You can add comments about an extended ACL to clarify the function of the ACL. To add a comment to an ACL, use the **access-list name remark** command in configuration mode. You can enter only one comment per ACL and the comment always appears at the beginning of the ACL. The syntax of this command is as follows:

```
access-list name remark text
```

The keywords and arguments are as follows:

- **name**—Unique identifier of the ACL. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- **remark text**—Specifies any comments that you want to include about the ACL. Comments appear at the top of the ACL. Enter an unquoted text string with a maximum of 100 alphanumeric characters. You can enter leading spaces at the beginning of the text. Trailing spaces are ignored.

For example, enter:

```
host1/Admin(config)# access-list INBOUND remark This is a remark
```

For example, to remove entry comments from an ACL, enter:

```
host1/Admin(config)# no access-list INBOUND line 200 remark
```

If you delete an ACL using the **no access-list name** command, then all the remarks are also removed.

Configuring an EtherType ACL

You can configure an ACL that controls traffic based on its EtherType. An EtherType is a subprotocol identifier. EtherType ACLs support Ethernet V2 frames. EtherType ACLs do not support 802.3-formatted frames because they use a length field instead of a type field. The only exception is a bridge protocol data unit (BPDU), which is SNAP encapsulated. The ACE can specifically handle BPDUs.

You can permit or deny BPDUs. By default, all BPDUs are denied. The ACE receives trunk port (Cisco proprietary) BPDUs because ACE ports are trunk ports. Trunk BPDUs have VLAN information inside the payload, so the ACE modifies

the payload with the outgoing VLAN if you allow BPDUs. If you configure redundancy, you must allow BPDUs on both interfaces with an EtherType ACL to avoid bridging loops. For details about configuring redundancy, see the *Cisco 4700 Series Application Control Engine Appliance Administration Guide*.

If you allow Multiprotocol Label Switching (MPLS), ensure that Label Distribution Protocol (LDP) and Tag Distribution Protocol (TDP) TCP connections are established through the ACE by configuring both MPLS routers connected to the ACE to use the IP address on the ACE interface as the router ID for LDP or TDP sessions. LDP and TDP allow MPLS routers to negotiate the labels (addresses) used to forward packets.

**Note**

You can configure an EtherType ACL on a Layer 2 interface in the inbound direction only.

On Cisco IOS routers, enter the appropriate command for your protocol: LDP or TDP. The *interface* is the interface connected to the ACE:

```
host1/Admin(config)# mpls ldp router-id interface force
```

or

```
host1/Admin(config)# tag-switching tdp router-id interface force
```

**Tip**

Enter the ACL name in uppercase letters so that the name is easy to see in the configuration. You may want to name the ACL for the interface (for example, INBOUND), or for the purpose (for example, MPLS).

To configure an EtherType ACL, use the **access-list ethertype** command in configuration mode. The syntax of this command is as follows:

```
access-list name ethertype {deny | permit} {any | bpdu | ipv6 | mpls}
```

The keywords and arguments are as follows:

- *name*—Unique identifier of the ACL. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *ethertype*—Name that specifies a subprotocol. Valid values are as follows:
 - **deny**—Blocks connections on the assigned interface
 - **permit**—Allows connections on the assigned interface

- **any**—Specifies any EtherType
- **bpdud**—Specifies a bridge protocol data unit



Note The ACE does not forward multiple spanning tree (MST) BPDUs.

- **ipv6**—Specifies Internet Protocol version 6
- **mpls**—Specifies Multiprotocol Label Switching



Note When you specify the **mpls** keyword in an EtherType ACL, the ACE denies or permits both MPLS-unicast and MPLS-multicast traffic.

For example, to configure an EtherType ACL for MPLS, enter:

```
host1/Admin(config)# access-list INBOUND ethertype permit mpls
```

To remove an entry from an EtherType ACL, enter:

```
host1/Admin(config)# no access-list INBOUND ethertype permit mpls
```

Resequencing Entries

You can resequence the entries in an ACL with a specific starting number and interval by using the **access-list name resequence** command in configuration mode. The ability to resequence entries in an ACL is supported only for extended ACLs.

The syntax of this command is as follows:

```
access-list name resequence [number1] [number2]
```

The keywords, options, and arguments are as follows:

- **name**—Unique identifier of the ACL. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- **resequence**—Specifies the renumbering of the entries in an ACL.

- *number1*—(Optional) Number assigned to the first entry in the ACL. Enter any integer. The default is 10.
- *number2*—(Optional) Number added to each entry in the ACL after the first entry. Enter any integer. The default is 10.

For example, enter:

```
host1/Admin(config)# access-list INBOUND resequence 5 15
```

Simplifying Access Control Lists with Object Groups

This section describes how to use object groups to simplify ACL creation and maintenance. It contains the following topics:

- [Overview of Object Groups](#)
- [Configuring Network Object Groups](#)
- [Configuring Service Object Groups](#)
- [Using Object Groups in an ACL](#)
- [Applying an ACL to an Interface](#)
- [Applying an ACL Globally to All Interfaces in a Context](#)
- [Filtering Traffic with an ACL](#)

Overview of Object Groups

Object groups allow you to streamline the configuration of multiple ACL entries in an ACL. By grouping like objects together, you can use an object group in an ACL entry instead of having to enter an ACL entry for each object separately. You can create the following types of object groups:

- Network object groups
- Service object groups

For example, consider the following three object groups:

- **MyServices**—Includes the TCP and UDP port numbers of the service requests that are allowed access to the internal network
- **TrustedHosts**—Includes the host and network addresses that are allowed access to the greatest range of services and servers
- **PublicServers**—Includes the host addresses of servers to which the greatest access is provided

After you create these groups, you can use a single ACL entry to allow trusted hosts to make specific service requests to a group of public servers.

**Note**

You can configure a maximum of 4 K object groups in an ACE. Each object group can have up to 40,000 elements. The maximum number of ACL entries in an ACE is 40,000.

The system-wide ACL entry limit of 40,000 entries applies to expanded ACL entries. An expanded ACL entry is the individually entered entry equivalent of an object-group element. If you use object groups in an ACL, you enter fewer actual ACL entries. When the ACE expands an ACL that references an object group, internally, multiple ACL entries will exist based on the number of elements present in the object group. To view the number of expanded ACL entries in an ACL, enter the **show access-list *name*** command. For details, see the “[Displaying ACL Configuration Information and Statistics](#)” section.

Configuring Network Object Groups

This section describes how to configure object groups to streamline the creation of ACL entries in an ACL. It includes the following topics:

- [Creating a Network Object Group](#)
- [Adding a Description to a Network Object Group](#)
- [Configuring a Network IP Address for a Network Object Group](#)
- [Configuring a Host IP Address](#)

Creating a Network Object Group

To create an object group, use the **object-group** command in configuration mode. The syntax of this command is as follows:

```
object-group network name
```

The keywords and arguments are as follows:

- **network**—Specifies a group of hosts or subnet IP addresses.
- *name*—Unique identifier of the object group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a network object group, enter:

```
host1/Admin(config)# object-group network NET_OBJ_GROUP1  
host1/Admin(config-objgrp-netw)#
```

To remove the network object group from the configuration, enter:

```
host1/Admin(config)# no object-group network NET_OBJ_GROUP1
```



Note

If you add new elements to an existing object group that is already in use by an entry in a large ACL, recommitting the ACL can take a long time, depending on the size of the ACL and the number of elements in the object group. In extreme cases, recommitting this ACL may cause the ACE to respond to commands slowly or even to become temporarily unresponsive. We recommend that you first remove the ACL entry that refers to the object group, make your modifications to the relevant object group, and then add the ACL entry back into the ACL.

Adding a Description to a Network Object Group

To add an optional description to a network object group, use the **description** command in object group network configuration mode. The syntax of this command is as follows:

```
description text
```

The *text* argument is an unquoted text string with a maximum of 240 alphanumeric characters.

For example, to add a description to a network object group, enter:

```
host1/Admin(config-objgrp-netw)# description intranet network object group
```

To remove a description from a network object group, enter:

```
host1/Admin(config-objgrp-netw)# no description intranet network object group
```

Configuring a Network IP Address for a Network Object Group

To associate a network IP address with a network object group, use the *ip_address* argument in object-group network configuration mode. The syntax of this command is as follows:

```
ip_address netmask
```

The arguments are as follows:

- *ip_address* —IP address assigned to the network object group. Enter an IP address in dotted-decimal notation (for example, 192.168.12.15).
- *netmask*—Network mask applied to the IP address. Enter a network mask in dotted decimal notation (for example, 255.255.255.0).

For example, to add the IP address 192.168.12.15 and network mask 255.255.255.0 to a network object group, enter:

```
host1/Admin(config-objgrp-netw)# 192.168.12.15 255.255.255.0
```

Enter additional object-group IP addresses as required.

To remove an IP address from the network object group, enter:

```
host1/Admin(config-objgrp-netw)# no 192.168.12.15 255.255.255.0
```

Configuring a Host IP Address

To associate a host IP address with a network object group, use the **host** command in object-group network configuration mode. The syntax of this command is as follows:

```
host ip_address
```

The *ip_address* specifies the IP address of the host. Use this argument to specify a single IP address. Enter an IP address in dotted-decimal notation (for example, 192.168.12.15).

For example, to create a network object group that includes three host addresses, enter:

```
host1/Admin(config)# object-group network NET_OBJ_GROUP1
host1/Admin(config-objgrp-netw) # description Administrator Addresses
host1/Admin(config-objgrp-netw) # host 192.168.12.15
host1/Admin(config-objgrp-netw) # host 192.168.12.21
host1/Admin(config-objgrp-netw) # host 192.168.12.27
```

Configuring Service Object Groups

This section describes how to configure service object groups to streamline the creation of ACL entries that include protocol names and port names in an ACL. It includes the following sections:

- [Creating a Service Object Group](#)
- [Adding a Description to a Service Object Group](#)
- [Defining Protocol Parameters for a Service Object Group](#)

Creating a Service Object Group

To create a service object group, use the **object-group** command in configuration mode. The syntax of this command is as follows:

```
object-group service name
```

The keywords and arguments are as follows:

- **service**—Specifies a group of IP protocol and port specifications.
- *name*—Unique identifier of the object group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a service object group, enter:

```
host1/Admin(config)# object-group service SERV_OBJ_GROUP1
host1/Admin(config-objgrp-serv) #
```

To remove the servicobject group from the configuration, enter:

```
host1/Admin(config)# no object-group service SERV_OBJ_GROUP1
```



Note

If you add new elements to an existing object group that is already in use by an entry in a large ACL, recommitting the ACL can take a long time, depending on the size of the ACL and the number of elements in the object group. In extreme cases, recommitting this ACL may cause the ACE to respond to commands slowly or even to become temporarily unresponsive. We recommend that you first remove the ACL entry that refers to the object group, make your modifications to the relevant object group, and then add the ACL entry back into the ACL.

Adding a Description to a Service Object Group

To add an optional description to a service object group, use the **description** command in object group service configuration mode. The syntax of this command is as follows:

```
description text
```

The *text* argument is an unquoted text string with a maximum of 240 alphanumeric characters.

For example, to add a description to a service object group, enter:

```
host1/Admin(config)# object-group service SERV_OBJ_GROUP1
host1/Admin(config-objgrp-serv)# description intranet network object
group
```

To remove a description from a service object group, enter:

```
host1/Admin(config)# object-group service SERV_OBJ_GROUP1
host1/Admin(config-objgrp-serv)# no description intranet network
object group
```

Defining Protocol Parameters for a Service Object Group

To define protocol parameters for a service object group, use the *protocol* argument in object group service configuration mode. For TCP or UDP, the syntax of this command is as follows:

```
protocol [source {{operator} port1 | port1 port2}] [{operator} port3 | port3 port4}]
```

For ICMP, the syntax of this command is:

```
icmp [icmp-type] [code {{operator} icmp-code1 | range icmp-code1 icmp-code2}]
```

The keywords, arguments, and options are as follows:

- *protocol*—Name or number of an IP protocol. Enter a protocol name or an integer from 1 to 255 that represents an IP protocol number. See [Table 1-2](#).
- **source**—(Optional) Specifies a source port for TCP, TCP-UDP, or UDP.



Note

To specify a destination port for TCP or UDP, use the *operator* argument with no preceding keyword. The destination keyword is implied.

- *operator*—Operand used to compare source and destination port numbers for TCP and UDP protocols or ICMP code numbers for the ICMP protocol. The operators are as follows:
 - **lt**—Less than.
 - **gt**—Greater than.
 - **eq**—Equal to.
 - **neq**—Not equal to.
 - **range**—An inclusive range of port values or ICMP message codes. If you enter this operator, enter a second port number value or a second ICMP message code to define the upper limit of the range.
- *port1 port2*—IP protocol source port name or port number from which you permit or deny access to services. Enter a port name or an integer from 0 to 65535. To enter an inclusive range of ports, enter two port numbers following

the **range** keyword. The *port2* value must be greater than or equal to the *port1* value. See [Table 1-3](#) for a list of well-known TCP keywords and port numbers and [Table 1-4](#) for a list of well-known UDP keywords and port numbers.

- *port3 port4*—IP protocol destination port name or port number to which you permit or deny access to services. To enter an optional inclusive range of ports, enter the **range** keyword followed by two port numbers. The *port4* value must be greater than or equal to the *port3* value. See [Table 1-3](#) for a list of well-known TCP keywords and port numbers and [Table 1-4](#) for a list of well-known UDP keywords and port numbers.
- *icmp-type*—(Optional) If you entered ICMP as the protocol, specifies the type of ICMP messaging. Enter either an integer corresponding to the ICMP code number or one of the ICMP types listed in [Table 1-5](#).
- **code**—(Optional) Specifies that a numeric operator and ICMP code follows.
- *icmp-code1 icmp-code2*—Specifies an ICMP code number that corresponds to an ICMP type. See [Table 1-5](#). To enter an optional inclusive range of ICMP codes, enter the **range** keyword followed by two ICMP code numbers. The *icmp-code1* value must be greater than or equal to the *icmp-code2* value. See [Table 1-5](#) for list of ICMP codes and corresponding ICMP types.

For example, to add only a destination (destination keyword is implied) TCP port to a service object group, enter:

```
host1/Admin(config-objgrp-serv)# tcp eq 41
```

Enter additional object-group protocols as required.

To remove the destination TCP port from a service object group, enter:

```
host1/Admin(config-objgrp-prot)# no tcp
```

For example, to create a service object group for TCP (source port only), UDP (source and destination ports), and ICMP, enter:

```
host1/Admin(config)# object-group service TCP_UDP_ICMP
host1/Admin(config-objgrp-serv)# tcp source eq domain
host1/Admin(config-objgrp-serv)# udp source eq radius eq radius-acct
host1/Admin(config-objgrp-serv)# icmp echo code eq 0
```

To remove the ICMP protocol from the above service object group, enter:

```
host1/Admin(config-objgrp-prot)# no icmp echo code eq 0
```

Using Object Groups in an ACL

To use object groups in an ACL, replace the normal network (*source_address mask*, and so on), service (*protocol operator port*), or ICMP type (*icmp_type*) arguments with the **object-group name** keyword and argument.

For example, to use object groups for all available parameters in the **access-list extended** command, enter the following command:

```
host1/Admin(config)# access-list acl_name extended {deny | permit}
object-group service_grp_name object-group network_grp_name
object-group network_grp_name
```

You do not have to use object groups for all parameters. For example, you can use an object group for the source address, but identify the destination address with an IP address and subnet mask.

The following subsections provide examples of configuring extended ACLs with and without object groups, and how object group entries expand into multiple ACL entries:

- [Example of Configuring an Extended ACL Without Object Groups](#)
- [Example of Configuring the Equivalent Extended ACL Using Object Groups](#)
- [Example of How an Object Group Expands into Multiple ACL Entries](#)

Example of Configuring an Extended ACL Without Object Groups

The following example shows how to configure an extended ACL that does not use object groups to restrict several hosts on the inside network from accessing several web servers. All other traffic is allowed.

```
host1/Admin(config)# access-list ACL_IN remark "object-group acl to
deny specific hosts"
host1/Admin(config)# access-list ACL_IN extended deny tcp host
10.1.1.4 host 209.165.201.29 eq www
host1/Admin(config)# access-list ACL_IN extended deny tcp host
10.1.1.78 host 209.165.201.29 eq www
host1/Admin(config)# access-list ACL_IN extended deny tcp host
10.1.1.89 host 209.165.201.29 eq www
host1/Admin(config)# access-list ACL_IN extended deny tcp host
10.1.1.4 host 209.165.201.16 eq www
host1/Admin(config)# access-list ACL_IN extended deny tcp host
10.1.1.78 host 209.165.201.16 eq www
```

```

host1/Admin(config)# access-list ACL_IN extended deny tcp host
10.1.1.89 host 209.165.201.16 eq www
host1/Admin(config)# access-list ACL_IN extended deny tcp host
10.1.1.4 host 209.165.201.78 eq www
host1/Admin(config)# access-list ACL_IN extended deny tcp host
10.1.1.78 host 209.165.201.78 eq www
host1/Admin(config)# access-list ACL_IN extended deny tcp host
10.1.1.89 host 209.165.201.78 eq www
host1/Admin(config)# access-list ACL_IN extended permit ip any any
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# access-group input ACL_IN

```

Example of Configuring the Equivalent Extended ACL Using Object Groups

The following example shows how to configure the equivalent of the extended ACL in the “[Example of Configuring an Extended ACL Without Object Groups](#)” section using two network object groups, one for the inside hosts, and one for the web servers. Notice how object groups simplify the configuration and allow you to easily modify it to add more hosts as follows:

```

host1/Admin(config)# object-group network DENIED
host1/Admin(config-objgrp-network)# host 10.1.1.4
host1/Admin(config-objgrp-network)# host 10.1.1.78
host1/Admin(config-objgrp-network)# host 10.1.1.89

host1/Admin(config)# object-group network WEB
host1/Admin(config-objgrp-network)# host 209.165.201.29
host1/Admin(config-objgrp-network)# host 209.165.201.16
host1/Admin(config-objgrp-network)# host 209.165.201.78

host1/Admin(config)# access-list ACL_IN remark "object-group acl to
deny specific hosts"
host1/Admin(config)# access-list ACL_IN extended deny tcp object-group
DENIED object-group WEB eq www
host1/Admin(config)# access-list ACL_IN extended permit ip any any
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# access-group input ACL_IN

```

Example of How an Object Group Expands into Multiple ACL Entries

The following examples of **show** command output demonstrate how the ACE expands the single ACL entry that has an object group (see the [“Example of Configuring the Equivalent Extended ACL Using Object Groups”](#) section) into multiple ACL entries. The output of the **show running-config access-list** command displays the unexpanded object-group configuration of the ACL_IN ACL. The output of the **show access-list ACL_IN** command displays the expanded ACL entries.

```
host1/Admin# show running-config access-list
Generating configuration....
```

```
access-list ACL_IN remark "object group acl to deny specific hosts"
access-list ACL_IN line 8 extended deny tcp object-group DENIED
object-group WEB eq www
access-list ACL_IN line 16 extended permit ip any any
```

```
host1/Admin# show access-list ACL_IN
access-list:ACL_IN, elements: 10, status: ACTIVE
  remark : "object group acl to deny specific hosts"
access-list ACL_IN line 8 extended deny tcp object-group DENIED
object-group WEB eq www
  access-list ACL_IN line 8 extended deny tcp host 10.1.1.4 host
209.165.201.29 eq www (hitcount=0)
  access-list ACL_IN line 8 extended deny tcp host 10.1.1.4 host
209.165.201.16 eq www (hitcount=0)
  access-list ACL_IN line 8 extended deny tcp host 10.1.1.4 host
209.165.201.78 eq www (hitcount=0)
  access-list ACL_IN line 8 extended deny tcp host 10.1.1.78 host
209.165.201.29 eq www (hitcount=0)
  access-list ACL_IN line 8 extended deny tcp host 10.1.1.78 host
209.165.201.16 eq www (hitcount=0)
  access-list ACL_IN line 8 extended deny tcp host 10.1.1.78 host
209.165.201.78 eq www (hitcount=0)
  access-list ACL_IN line 8 extended deny tcp host 10.1.1.89 host
209.165.201.29 eq www (hitcount=0)
  access-list ACL_IN line 8 extended deny tcp host 10.1.1.89 host
209.165.201.16 eq www (hitcount=0)
  access-list ACL_IN line 8 extended deny tcp host 10.1.1.89 host
209.165.201.78 eq www (hitcount=0)
access-list ACL_IN line 16 extended permit ip any any (hitcount=0)
```

Applying an ACL to an Interface

Before you can start using a configured ACL, you must apply it to one or more interfaces.

To apply an ACL to the inbound or outbound direction of an interface and make the ACL active, use the **access-group** command in interface configuration mode. You can apply one ACL of each type (extended and EtherType) to both directions of the interface. See the [“Inbound and Outbound ACLs”](#) section for more information about ACL directions.

**Note**

If you have already applied a global ACL to all interfaces in a context, you cannot apply another ACL to an individual interface in that context. For details about applying an ACL globally, see the [“Applying an ACL Globally to All Interfaces in a Context”](#) section.

For connectionless protocols, you must apply the ACL to the source and destination interfaces if you want traffic to pass in both directions. For example, you can allow BGP in an ACL in transparent mode, and you must apply the ACL to both interfaces.

The syntax of this command is as follows:

```
access-group {input | output} acl_name
```

The keywords and arguments are as follows:

- **input | output**—Specifies the direction (inbound or outbound) of the interface to which you want to apply the ACL.
- *acl_name*—Identifier of an existing ACL that you want to apply to an interface. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin(config)# interface vlan 100  
host1/Admin(config-if)# access-group input INBOUND
```

To remove an ACL from an interface, enter:

```
host1/Admin(config-if)# no access-group input INBOUND
```

Applying an ACL Globally to All Interfaces in a Context

You can apply an ACL to all interfaces in a context at once, subject to the following conditions:

- No interface in the context has an ACL applied to it.
- You can globally apply one Layer 2 and one Layer 3 ACL in the inbound direction only.
- On Layer 2 bridged-group virtual interfaces (BVIS), you can apply both Layer 3 and Layer 2 ACLs.
- On Layer 3 virtual LAN (VLAN) interfaces, you can apply only Layer 3 ACLs.
- In a redundancy configuration, the ACE does not apply a global ACL to the FT VLAN. For details about redundancy, see the *Cisco 4700 Series Application Control Engine Appliance Administration Guide*.

To apply an ACL globally to all interfaces in a context in the inbound direction, use the **access-group input** command in configuration mode. The syntax of this command is as follows:

```
access-group input acl_name
```

For the *acl_name* argument, enter the identifier of an existing ACL as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

You can use this command to allow all traffic on all interfaces in a context by applying an ACL similar to the following example:

```
host1/Admin(config)# access-list ALL_ACCESS permit ip any any
```

Then, apply the ACL globally by entering:

```
host1/Admin(config)# access-group input ALL_ACCESS
```

To remove the ACL from all interfaces in the context, enter:

```
host1/Admin(config)# no access-group input ALL_ACCESS
```

Filtering Traffic with an ACL

You can use an ACL to filter interesting traffic and instruct the ACE to either permit or deny the traffic based on the action in the ACL. To filter traffic using an ACL, use the **match access-list** command in a Layer 3 and Layer 4 class map.

When a packet matches an entry in an ACL, and if it is a **permit** entry, the ACE allows the matching result. If it is a **deny** entry, the ACE blocks the matching result. For details about configuring a Layer 3 and Layer 4 class map and policy map, see [Chapter 4, Configuring Security Access Control Lists](#).

ACL Configuration Examples

This section provides the following examples of the different types of ACLs available in the ACE:

- [Examples of Extended ACLs](#)
- [Examples of EtherType ACLs](#)

Examples of Extended ACLs

This section provides examples of extended ACLs. Use extended ACLs when you want to specify both the source IP address and the destination IP address (IP), ports (TCP or UDP), and ICMP types. For details about configuring extended ACLs, see the “Configuring an Extended ACL” section.

The following ACL allows all hosts (on the interface to which you apply the ACL) to go through the ACE:

```
host1/Admin(config)# access-list ACL_IN extended permit ip any any
```

The following ACL prevents hosts on 192.168.1.0/24 from accessing the 209.165.201.0/27 network. All other addresses are permitted.

```
host1/Admin(config)# access-list ACL_IN extended deny tcp 192.168.1.0
255.255.255.0 209.165.201.0 255.255.255.224
host1/Admin(config)# access-list ACL_IN extended permit ip any any
```

If you want to restrict access to only some hosts, then enter a limited permit entry. By default, all other traffic is denied unless explicitly permitted.

```
host1/Admin(config)# access-list ACL_IN extended permit ip 192.168.1.0
255.255.255.0 209.165.201.0 255.255.255.224
```

For a list of permitted keywords and well-known port assignments, see [Table 1-3](#). DNS, Discard, Echo, Ident, NTP, RPC, SUNRPC, and Talk each require one definition for TCP and one for UDP. TACACS+ requires one definition for port 49 on TCP.

The following ACL example restricts all hosts (on the interface to which you apply the ACL) from accessing a website at address 209.165.201.29. All other traffic is allowed.

```
host1/Admin(config)# access-list ACL_IN extended deny tcp any host
209.165.201.29 eq www
host1/Admin(config)# access-list ACL_IN extended permit ip any any
```

The following ACLs allow all inside hosts to communicate with the outside network but only specific outside hosts to access the inside network:

```
host1/Admin(config)# access-list OUT extended permit ip any any
host1/Admin(config)# access-list IN extended permit ip host
209.168.200.3 any
host1/Admin(config)# access-list IN extended permit ip host
209.168.200.4 any
```

The following examples show how to configure ICMP ACLs. For details about configuring ICMP ACLs, see the “[Configuring an Extended ACL](#)” section.

```
host1/Admin(config)# access-list INBOUND extended permit icmp any any
echo
host1/Admin(config)# access-list INBOUND extended permit icmp host
10.0.0.1 host 20.0.0.1 unreachable code range 0 3
```

This section contains the following topics:

- [Inbound and Outbound ACLs](#)
- [IP Addresses for ACLs with NAT](#)

Inbound and Outbound ACLs

Traffic that flows across an interface in the ACE can be controlled in two ways:

- You can control traffic that enters the ACE by attaching an inbound ACL to the source interface.
- You can control traffic that exits the ACE by attaching an outbound ACL to the destination interface.

To allow any traffic to enter the ACE, you must attach an inbound permit ACL to an interface; otherwise, the ACE automatically refuses all traffic that enters that interface. By default, traffic can exit the ACE on any interface unless you restrict it by using an outbound ACL, which adds restrictions to those ACLs already configured in the inbound ACL.

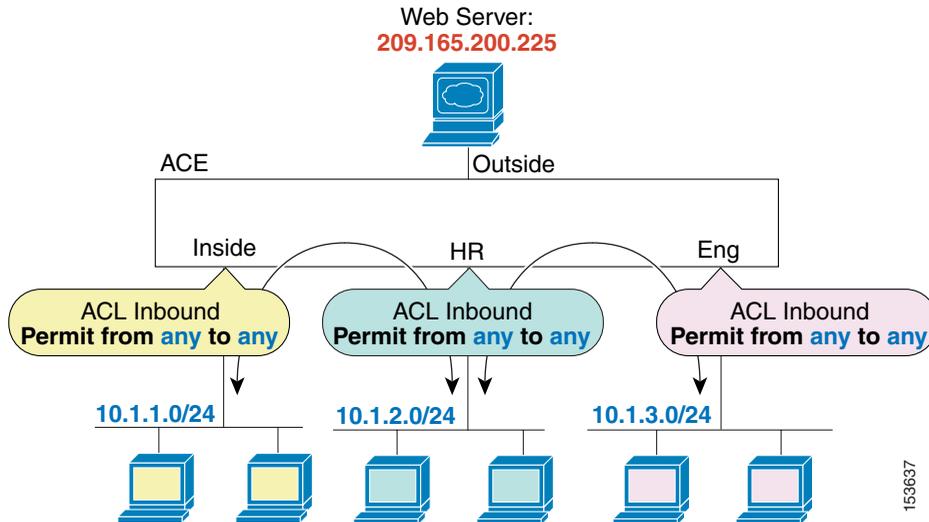


Note

Inbound and outbound refer to the application of an ACL on an interface, either to traffic entering the ACE on an interface or traffic exiting the ACE on an interface.

You may choose to use an outbound ACL to simplify your ACL configuration. For example, if you want to allow three inside networks on three different interfaces to access each other, you can create a simple inbound ACL on each interface that allows all traffic on each inside interface (see [Figure 1-1](#)).

Figure 1-1 Inbound ACLs



The following commands create three inbound ACLs that allow all traffic on each inside interface:

```
host1/Admin(config)# access-list INSIDE extended permit ip any any
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# access-group input INSIDE
```

```
host1/Admin(config)# access-list HR extended permit ip any any
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# access-group input HR
```

```
host1/Admin(config)# access-list ENG extended permit ip any any
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# access-group input ENG
```

If you choose to allow only certain hosts on the inside networks to access a web server on the outside network, you can create a more restrictive ACL that allows only the specified hosts and apply it to the outbound direction of the outside interface (see [Figure 1-2](#)). For information about NAT and IP addresses, see the [“IP Addresses for ACLs with NAT”](#) section. The outbound ACL prevents any other hosts from reaching the outside network.

The following commands create an ACL that allows only specified hosts and apply it to the outbound direction of the outside interface:

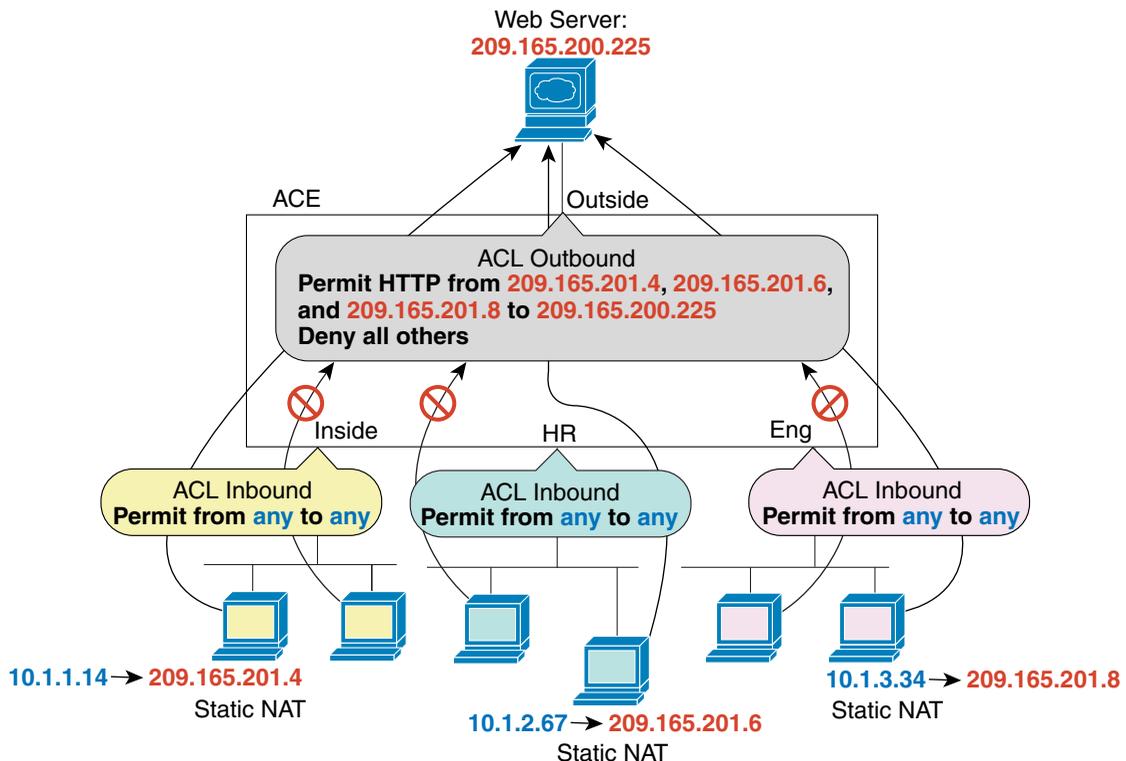
```
host1/Admin(config)# access-list OUTSIDE extended permit tcp host  
209.165.201.4 host 209.165.200.225 eq www
```

```
host1/Admin(config)# access-list OUTSIDE extended permit tcp host  
209.165.201.6 host 209.165.200.225 eq www
```

```
host1/Admin(config)# access-list OUTSIDE extended permit tcp host  
209.165.201.8 host 209.165.200.225 eq www
```

```
host1/Admin(config)# interface vlan 100  
host1/Admin(config-if)# access-group output OUTSIDE
```

Figure 1-2 Outbound ACL



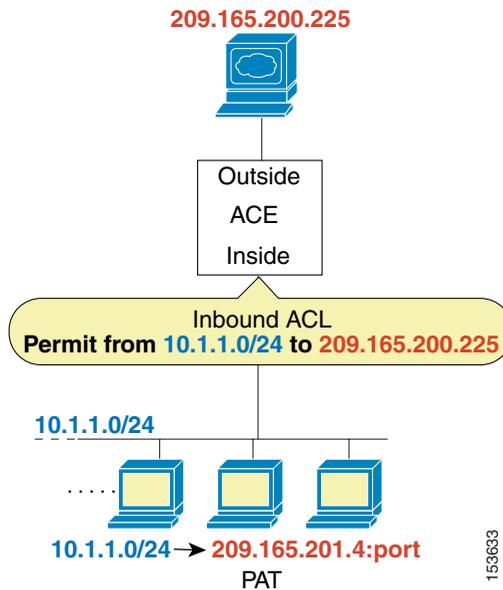
IP Addresses for ACLs with NAT

When you use NAT, the IP addresses that you specify for an ACL depend on the interface to which the ACL is attached. You must use addresses that are valid on a network that is connected to the interface. This guideline applies for both inbound and outbound ACLs. The ACL direction does not determine the address used, only the interface to which the ACL is attached determines the address that is used.

For example, suppose that you want to apply an ACL to the inbound direction of the interface. You configure the ACE to perform NAT on the inside source addresses when they access outside addresses. Because the ACL is applied to the

inside interface, the source addresses are the original untranslated addresses. Because the outside addresses are not translated, the destination address used in the ACL is the real address (see Figure 1-3).

Figure 1-3 IP Addresses in ACLs: NAT Used for Source Addresses

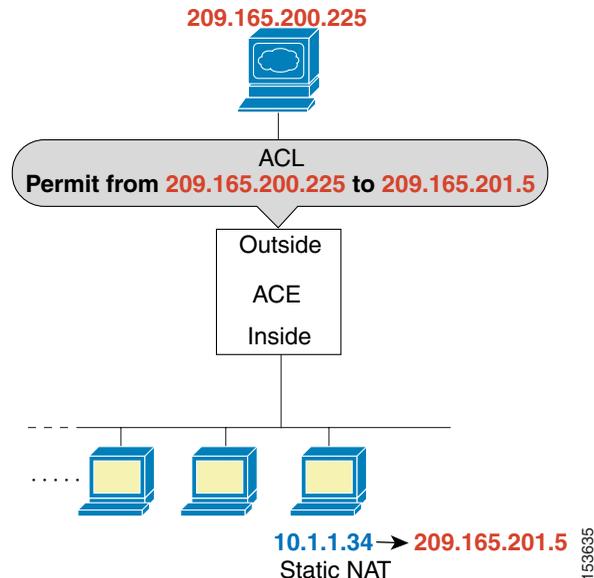


The following commands create an ACL that allows inside source network 10.1.1.0/24 to access the outside destination host 209.165.200.225 and apply the ACL to VLAN interface 100:

```
host1/Admin(config)# access-list INSIDE extended permit ip 10.1.1.0
255.255.255.0 host 209.165.200.225
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# access-group input INSIDE
```

If you want to allow an outside host to access an inside host, you can apply an inbound ACL to the outside interface. You must specify the translated address of the inside host in the ACL because that address is the address that can be used on the outside network (see Figure 1-4).

Figure 1-4 IP Addresses in ACLs: NAT used for Destination Addresses

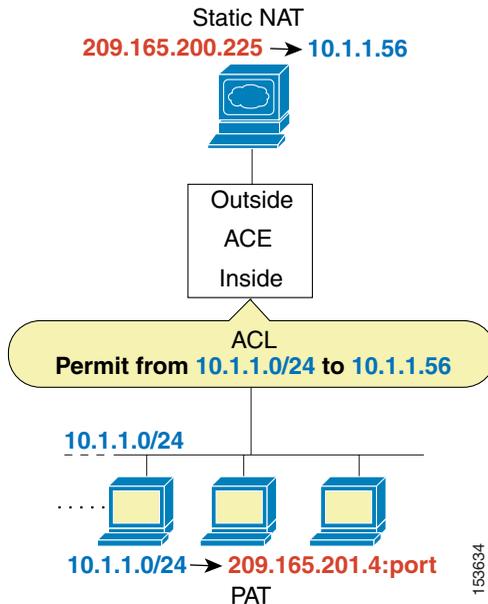


The following commands create an ACL that allows outside host 209.165.200.225 to access inside host 209.165.201.5 (the translated address of the host 10.1.1.34). The last command applies the ACL to VLAN interface 100.

```
host1/Admin(config)# access-list OUTSIDE extended permit ip host
209.165.200.225 host 209.165.201.5
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# access-group input OUTSIDE
```

If you perform NAT on both interfaces, you must verify the addresses that are visible on each interface when you create and apply ACLs. In [Figure 1-5](#), an outside server uses static NAT so that a translated address appears on the inside network.

Figure 1-5 IP Addresses in ACLs: NAT used for Source and Destination Addresses



The following commands create an ACL that allows inside source network 10.1.1.0/24 to access the outside destination host 10.1.1.56 (the translated address of the host 209.165.200.225). The last command applies the ACL to VLAN interface 100.

```
host1/Admin(config)# access-list INSIDE extended permit ip 10.1.1.0  
255.255.255.0 host 10.1.1.56  
host1/Admin(config)# interface vlan 100  
host1/Admin(config-if)# access-group input INSIDE
```

For an example of IP addresses used in outbound ACLs, see [Figure 1-2](#).

Examples of EtherType ACLs

This section provides examples of EtherType ACLs. For details about configuring an EtherType ACL, see the “[Configuring an EtherType ACL](#)” section.

The following example shows an ACL that allows common EtherTypes to originate on the inside interface:

```
host1/Admin(config)# access-list ETHER ethertype permit ipv6
host1/Admin(config)# access-list ETHER ethertype permit bpdu
host1/Admin(config)# access-list ETHER ethertype permit mpls
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# access-group output ethertype ETHER
```

The following example shows an ACL that allows some EtherTypes through the ACE but denies IPv6:

```
host1/Admin(config)# access-list ETHER ethertype deny ipv6
host1/Admin(config)# access-list ETHER ethertype permit bpdu
host1/Admin(config)# access-list ETHER ethertype permit mpls
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# access-group input ethertype ETHER
```

The following example shows an ACL that denies traffic with an EtherType BPDU but allows all others on both interfaces:

```
host1/Admin(config)# access-list nonIP ethertype deny bpdu
host1/Admin(config)# access-list nonIP ethertype permit any
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# access-group input ethertype nonIP
```

Displaying ACL Configuration Information and Statistics

This section describes the **show** commands that you can use to display ACL configurations and statistics. It contains the following topics:

- [Displaying ACL Configuration Information](#)
- [Displaying ACL Statistics](#)

Displaying ACL Configuration Information

You can display all ACL configuration information, including the interfaces on which you applied the ACLs by using the **show running-config** command. The syntax of this command is as follows:

```
show running-config
```

To display only the ACLs and their entries, use the **show running-config access-list** command in Exec mode. The syntax of this command is as follows:

```
show running-config access-list
```

Displaying ACL Statistics

You can display ACL statistics for a particular ACL by using the **show access-list** command. The syntax of this command is as follows:

```
show access-list name [detail]
```

The argument and optional keyword are as follows:

- *name*—Identifier of an existing ACL. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- **detail**—(Optional) Displays detailed ACL information, including a 4-byte MD5-hash value that the ACE uses to identify the ACL entry that caused a deny syslog (106023). See the description of the `0xnnnnnnnnnn` output field in [Table 1-6](#).

[Table 1-6](#) describes the fields in the **show access-list detail** command output.

Table 1-6 *Field Descriptions for the show access-list detail Command Output*

Field	Description
Access-list	Name of the security ACL.
Elements	Number of entries in the ACL.

Table 1-6 *Field Descriptions for the show access-list detail Command Output (continued)*

Field	Description
Status	Current status of the ACL: ACTIVE when the ACL is associated with at least one interface or NOT-ACTIVE when the ACL is not associated with at least one interface.
Remark	Configured comments that describe the ACL.
Entries	Full text of all entries in the ACL
Hitcounts	Hit counts for each ACL entry.
hash 1 (0xn timer)	32-bit hexadecimal MD5-hash value that the ACE computes from the access-list command immediately when you configure an ACL. The ACE includes this hash value in deny syslog messages to help you identify the ACL entry that caused the syslog in the output of this command. This hash value is line-number independent. To prevent possible discrepancies between the hash values in the deny syslog message and the output of this command after a reboot, be sure to use Tab completion or type entire keywords in the CLI when configuring individual entries in an ACL.
hash 2 (0xn timer)	16-bit hexadecimal (0xn timer) MD5-hash value that the ACE computes from the expanded access-list entries resulting from the object groups that you configure in an ACL. The ACE computes the <i>hash 2</i> value when you activate the ACL on an interface. For ACLs that do not have object groups, the <i>hash 2</i> value is always 0x0. The ACE also includes the <i>hash 2</i> value in deny syslog messages to help you identify the expanded ACL entry that caused the syslog. This hash value is also line-number independent. To uniquely identify the expanded ACL entry that caused the syslog, you need to search for an entry in this command output that matches both the <i>hash 1</i> and the <i>hash 2</i> hexadecimal values.

Displaying the ACL Merge Tree Node Usage

You can display the ACL merge tree node usage by entering the following command:

```
show np 1 | 2 access-list resource
```

The output of this command shows the used, guaranteed, and maximum values of the various ACL merge tree nodes. For more details about this command and troubleshooting `acl-memory Denied` counts in the output of the `show resource usage` command, see Step 5 in the [Troubleshooting Access Control Lists](#) section of the ACE Troubleshooting Wiki.

Clearing ACL Statistics

You can clear ACL statistics (hit counts for ACL entries) by using the `clear access-list` command in Exec mode. The syntax of this command is as follows:

```
clear access-list name
```

The *name* argument is an existing ACL. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin# clear access-list acl1
```



Note

If you configured redundancy, then you must explicitly clear ACL statistics (hit counts) on both the active and the standby ACEs. Clearing statistics on the active appliance only will leave the standby appliance's statistics at the old value.
