



Creating and Managing IP Access Control Lists for Standalone Content Engines

This chapter describes how to create and manage Internet Protocol (IP) access control lists (ACLs) on standalone Content Engines. This chapter contains the following sections:

- [Introducing IP ACLs for Standalone Content Engines, page 19-2](#)
- [Understanding the Basics About Working with IP ACLs, page 19-4](#)
- [Defining and Activating IP ACLs on Standalone Content Engines, page 19-6](#)
- [Creating or Modifying IP ACLs on Standalone Content Engines, page 19-10](#)
- [Activating an IP ACL on an Interface, page 19-16](#)
- [Applying an IP ACL to an Application, page 19-17](#)
- [Deleting an IP ACL, page 19-23](#)
- [Viewing the Configuration of an IP ACL, page 19-23](#)
- [Clearing an IP ACL Counter, page 19-24](#)



Note

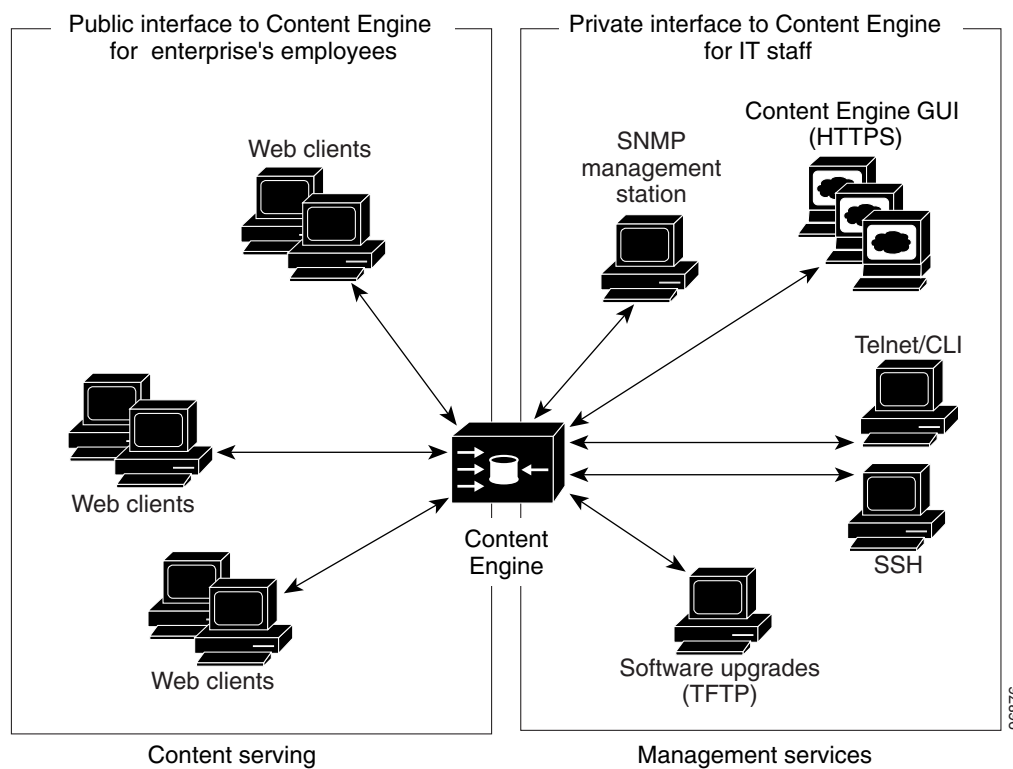
Throughout this chapter, the term *IP ACLs* is used to refer to IP access control lists.

Introducing IP ACLs for Standalone Content Engines

In the ACNS 5.1 software and later releases, IP ACLs are supported. IP ACLs provide IP packet filtering. These IP ACLs provide a means to filter packets by allowing you to permit or deny IP packets from crossing specific interfaces on the Content Engine.

In environments that have standalone Content Engines, you may want to use this feature to control access to content services and management services on the Content Engine. For example, you can use IP ACLs to define a public interface on the Content Engine for content serving and a private interface for management services (for example, Telnet, Secure Shell (SSH), SNMP, HTTPS, and software upgrades). (See [Figure 19-1](#).)

Figure 19-1 Using IP ACLs to Control Access to Specific Interfaces on a Standalone Content Engine



Note

In the ACNS 5.4.1 software and later releases, IP ACLs are also supported for nontransparent (proxy-mode requests) incoming FTP native requests and transparently-redirectioned incoming FTP native requests. For more information, see the [“Using IP ACLs to Control Native FTP Access”](#) section on [page 19-19](#).

The following are some examples of how IP ACLs can be used in environments that have standalone Content Engines:

- A Content Engine resides on the customer premises and is managed by a service provider, and the service provider wants to secure the device for its management only.
- A Content Engine is deployed anywhere within the enterprise. As with routers and switches, the administrator wants to limit Telnet, SSH, and Content Engine GUI access to the IT source subnets.
- An application layer proxy firewall with a hardened outside interface has no ports exposed. (*Hardened* means that the interface carefully restricts which ports are available for access, primarily for security reasons. With an outside interface, many types of security attacks are possible.) The Content Engine's outside address is Internet global, and its inside address is private. The inside interface has an IP ACL to limit Telnet, SSH, and Content Engine GUI access to the Content Engine.
- A Content Engine is deployed as a reverse proxy in an untrusted environment. The Content Engine administrator wants to allow only port 80 inbound traffic on the outside interface and outbound connections on the backend interface.
- A Content Engine using WCCP is positioned between a firewall and an Internet router or a subnet off the Internet router. Both the Content Engine and the router must have IP ACLs.

Implementing IP ACLs for Standalone Content Engines

To implement IP ACLs, follow these steps:

-
- Step 1** Define the IP ACLs on the standalone Content Engine by using the **ip access-list** command.
- Step 2** Apply the defined IP ACL either inbound or outbound to an interface on the standalone Content Engine by using the **ip access-group** command.



Note You can also use IP ACLs to permit or deny Telnet, SSH, and SNMP access to this standalone Content Engine.

Example of Defining and Activating IP ACLs

The following example shows how to define and activate an IP ACL on a standalone Content Engine. As the example shows, the first step is to use the **ip access-list** global configuration command to create an IP ACL for a standalone Content Engine. In this case, the IP ACL is named **example** and permits all web traffic but limits SSH access to a specific host:

```
ContentEngine(config)# ip access-list extended example
ContentEngine(config-ext-nacl)# permit tcp any any eq www
ContentEngine(config-ext-nacl)# permit tcp host 64.101.215.21 any eq ssh
ContentEngine(config-ext-nacl)# exit
```

After you create the IP ACL, use the **interface** global configuration command and the **ip access-group** configuration interface command to apply and activate the IP ACL for a specific interface on the Content Engine:

```
ContentEngine(config)# interface gigabitethernet 1/0
ContentEngine(config-if)# ip access-group example in
ContentEngine(config-if)# exit
```

After defining and activating the IP ACLs, view the running configuration on the Content Engine:

```
ContentEngine# show running-config
.
.
.
!
interface GigabitEthernet 1/0
 ip address 10.1.1.50 255.255.0.0
 ip access-group example in
 exit
.
.
.
ip access-list extended example
 permit tcp any any eq www
 permit tcp host 10.101.215.21 any eq ssh
 exit
.
.
.
```

**Note**

IP ACLs are defined for individual ACNS software devices only. IP ACLs cannot be managed globally across the ACNS network or through device groups. For information about using the Content Distribution Manager to create and manage IP ACLs on ACNS network devices (for example, a Content Engine that is registered with a Content Distribution Manager), see the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.5*.

For more background information about IP ACLs, see the next section, “[Understanding the Basics About Working with IP ACLs](#).” For information about how to configure IP ACLs, see the “[Defining and Activating IP ACLs on Standalone Content Engines](#)” section on page 19-6.

Understanding the Basics About Working with IP ACLs

An IP ACL consists of one or more condition entries that specify the kind of packets that the Content Engine will drop or accept for further processing. The Content Engine applies each condition in the order in which it occurs in the IP ACL, which by default, is the order in which you configured the condition.

In the ACNS 5.1 software and later releases, there are two different types of IP ACLs:

- Standard ACLs
- Extended ACLs

**Note**

The ACNS software CLI must be used to create and manage IP ACLs on a standalone Content Engine. The Content Engine GUI does not currently support the configuration of IP ACLs on a standalone Content Engine.

Working with Standard IP ACLs

Typically, standard ACLs are used for the following reasons:

- To allow connections from a host with a specific IP address
- To allow connections from hosts on a specific network

Accessing Standard IP ACL Configuration Mode

To work with standard IP ACLs, you must enter standard IP ACL configuration mode on a Content Engine. To access standard IP ACL configuration mode, enter the **ip access-list standard** global configuration command:

```
ContentEngine(config)# ip access-list standard {acl-name | acl-num}
```

- *acl-name* is the name of the standard IP ACL that you want to create or modify.
- *acl-num* is the number of the standard IP ACL that you want to create or modify.

After you enter standard IP ACL mode, the ContentEngine(config)# prompt changes to ContentEngine(config-std-nacl)#, where nacl stands for the specified standard access list.

For example, the following example shows how to enter standard IP ACL configuration mode in order to modify the standard IP ACL that has the ACL number of 2. The CLI enters the standard IP ACL configuration mode, in which all subsequent commands apply to the currently specified standard IP ACL (for example, the standard IP ACL nac2).

```
ContentEngine(config)# ip access-list standard 2
ContentEngine(config-std-nacl)#
```

Working with Extended IP ACLs

Extended IP ACLs generally use the following elements to control connections:

- Destination IP address
- IP protocol type
- UDP or TCP source or destination port
- ICMP message type or code
- TCP flag bits (established)

To create more restrictive conditions, these conditions can be combined with information about the source IP address. [Table 19-3](#) lists the keywords that you can use to match specific Internet Control Message Protocol (ICMP) message types and codes.

Accessing Extended IP ACL Configuration Mode

To work with extended IP ACLs, you must enter extended IP ACL configuration mode on the Content Engine. To access extended IP ACL configuration mode, enter the **ip access-list extended** global configuration command:

```
ContentEngine(config)# ip access-list extended {acl-name | acl-num}
```

- *acl-name* is the name of the extended IP ACL that you want to create or modify.
- *acl-num* is the number of the extended IP ACL that you want to create or modify.

After you enter extended IP ACL mode, the ContentEngine(config)# prompt changes to a ContentEngine(config-ext-nacl)# prompt, where nacl represents the specified extended access list.

The following example shows how to enter extended IP ACL configuration mode in order to modify the extended IP ACL that has the ACL number of 101. The CLI enters extended IP ACL configuration mode where all subsequent commands apply to the currently specified extended IP ACL (for example, the extended IP ACL 101).

```
ContentEngine(config)# ip access-list extended 101
ContentEngine(config-ext-nacl)#
```

**Note**

For information about how to create or modify an extended IP ACL, see the [“Creating or Modifying IP ACLs on Standalone Content Engines” section on page 19-10](#).

Defining and Activating IP ACLs on Standalone Content Engines

In some service provider deployments, a Content Engine can have one interface in the customer’s IP address space that serves content, and another interface in a private IP address space that the administrator uses for management purposes. The ACNS 5.1 software and later releases provide controls that allow various services to be associated with a particular interface (such as management services to the private IP address space) so that the enterprise customer can only access the Content Engine for serving content and not access it for management purposes.

To use IP ACLs in environments that have standalone Content Engines that are running the ACNS 5.1 software and later releases, the system administrator must complete the following tasks through the CLI:

1. Define the IP ACLs by using the **ip access-list** command.
2. Apply an IP ACL to a specific interface on the Content Engine, by using the **interface** and **ip access-group** commands.

**Tip**

Use the **ip access-group** command to apply an IP ACL to either inbound or outbound IP traffic on an interface.

Usage Guidelines

When creating or modifying IP ACLs on a standalone Content Engine, remember the following important points:

- To create an entry in a standard or extended IP ACL, use the **deny** or **permit** keyword and specify the type of packets that you want the Content Engine to drop or to accept for further processing. By default, an access list denies everything because the list is terminated by an implicit **deny any** entry. Therefore, you must include at least one **permit** entry to create a valid access list.

**Note**

To allow connections from a specific network, use the **permit source-ip wildcard** command. Replace *source-ip* with a network ID or the IP address of any host on the network that you want to specify. Replace *wildcard* with the dotted decimal notation for a mask that is the reverse of a subnet mask, where a 0 indicates a position that must be matched and a 1 indicates a position that does not matter. For instance, the wildcard 0.0.0.255 causes the last 8 bits in the source IP address to be ignored. Therefore, the entry **permit 192.168.1.0 0.0.0.255** allows access from any host on the 192.168.1.0 network.

- You can also apply an extended IP ACL to a specific application using the appropriate command. A reference to an IP ACL that does not exist is the equivalent of a **permit any** condition statement.
- In the ACNS 5.4.1 software and later releases, you can use IP ACLs to grant or deny access to the native FTP proxy service that is running on a standalone Content Engine. To support this feature, the following two CLI commands were added in the ACNS 5.4.1 software release:

```
ftp-native access-list in {std-acl-num | std-acl-name}
ftp-native access-list out {ext-acl-num | ext-acl-name}
```

For more information, see the [“Using IP ACLs to Control Native FTP Access”](#) section on page 19-19.

- In the ACNS 5.1 software and later releases, the SNMP and TFTP applications have a specific CLI command to configure the use of an IP ACL. The commands are as follows:

```
snmp-server access-list {std-acl-num | std-acl-name}
tftp-server access-list {std-acl-num | std-acl-name}
```



Note The **snmp-server access-list** and **tftp-server access-list** global configuration commands can only accept the name or number for a standard IP ACL and not an extended IP ACL.

Other application traffic (for example, Telnet and SSH) can be controlled by applying an IP ACL to an interface (typically, to inbound traffic) on the standalone Content Engine.

- In the ACNS 5.2.1 software and later releases, use the **wccp access-list** global configuration command to specify an IP ACL that the Content Engine applies to WCCP GRE inbound traffic.

```
wccp access-list {acl-num | acl-name}
```

The WCCP access control list feature supports both standard and extended access control lists, and is not restricted to only standard access control lists, as is the case with SNMP and TFTP server access lists. For more information about configuring WCCP access control lists, see the [“Configuring WCCP Access Lists for Standalone Content Engines”](#) section on page 19-20.

- For standard IP ACLs, the **wildcard** parameter of the **ip access-list** command is always optional. If the **host** keyword is specified for a standard IP ACL, then the **wildcard** parameter is not allowed, as shown in the following example:

```
ContentEngine(config)# ip access-list standard 1
ContentEngine(config-std-nacl)# permit ?
  A.B.C.D Source address
  any      Any source host
  host     A single host address
ContentEngine(config-std-nacl)# permit 10.1.1.1 ?
  A.B.C.D Source wildcard bits <=== *** Wildcard parameter is optional here ***
<cr>
ContentEngine(config-std-nacl)# permit host 10.1.1.1 ? <=== *** Wildcard parameter is
not allowed here because the host keyword is used***
<cr>
ContentEngine(config-std-nacl)# permit 10.1.1.1
ContentEngine(config-std-nacl)# exit
```

- For extended IP ACLs, the **wildcard** parameter is always required unless the **host** keyword is specified. If the **host** keyword is specified for an extended IP ACL, then the **wildcard** parameter is not allowed, as shown in the following example:

```
ContentEngine(config)# ip access-list extended 100
ContentEngine(config-ext-nacl)# permit ?
<1-255> An IP Protocol Number
gre     Cisco's GRE Tunneling
icmp    Internet Control Message Protocol
ip      Any IP Protocol
tcp     Transport Control Protocol
udp     User Datagram Protocol
ContentEngine(config-ext-nacl)# permit ip ?
A.B.C.D Source address
any     Any source host
host    A single host address
ContentEngine(config-ext-nacl)# permit ip 10.1.1.1 ?
A.B.C.D Source wildcard bits
<=== *** Wildcard parameter is required here because the host keyword is not
specified***
ContentEngine(config-ext-nacl)# permit ip host ?
A.B.C.D Source address
ContentEngine(config-ext-nacl)# permit ip host 10.1.1.1 ? <=== *** Wildcard parameter
is not allowed here because the host keyword is used***
A.B.C.D Destination address
any     Any destination host
host    A single host address
```

- When you are in standard or extended IP ACL configuration mode, you can use the editing commands (**list**, **delete**, and **move**) to display entries, to delete a specific entry (a condition), or to change the order in which the entries will be evaluated.

```
ContentEngine(config)# ip access-list standard 1
ContentEngine(config-std-nacl)#?
delete Delete a condition
deny    Specify packets to reject
exit    Exit from this submode
insert  Insert a condition
list    List conditions
move    Move a condition
no      Negate a command or set its defaults
permit  Specify packets to accept
ContentEngine(config-std-nacl)#
```

- To identify the line numbers that conditions map to, use the **list** command. This command lists the specified entries (or all entries when none are specified). Without this command, you would have to return to EXEC mode and then enter the **show ip access-list EXEC** command to obtain this mapping.

The following example shows how to use the **list** command.

```
Content Engine(config-ext-nacl)# list
1 permit tcp host 10.1.1.1 any
2 permit tcp host 10.1.1.2 any
3 permit tcp host 10.1.1.3 any
Content Engine(config-ext-nacl)#
```

- For information about how to delete an entire IP ACL from the Content Engine's database, see the [“Deleting an IP ACL”](#) section on page 19-23.

Usage Guidelines for IP ACL Configuration Modes

When working with IP ACLs, remember the following important points about the IP ACL configuration modes:

- You must enter the standard IP ACL configuration mode to work with standard IP ACLs.

```
ContentEngine(config)# ip access-list standard ?
<1-99> Standard IP access-list number
WORD Access-list name (max 30 characters)
```

- You must enter the extended IP ACL configuration mode to work with extended IP ACLs.

```
ContentEngine(config)# ip access-list extended ?
<100-199> Standard IP access-list number
WORD Access-list name (max 30 characters)
```

Usage Guidelines for IP ACL Names

When creating IP ACL names, use the following naming guidelines:

- IP ACL names must be unique within the Content Engine
- When an IP ACL name is numeric (for example, **ip access-list standard** *acl-num* or **ip access-list extended** *acl-num*):
 - It can contain only numeric characters (for example, 101).
 - Numbers 1–99 represent standard IP ACLs.
 - Numbers 100–199 represent extended IP ACLs.
- When an IP ACL name is a word (for example, **ip access-list standard** *acl-name* or **ip access-list extended** *acl-name*):
 - It must begin with a nonnumeric character for example, snmpaccesslist).
 - It is limited to 30 characters.
 - It can contain the digits 0–9 within the string of characters (for example, snmpaccesslist7).
 - It can contain most of the printable special characters but no white space. The list of acceptable special characters includes the following: ~!@#\$%^&*()_+={ }[]\|:;<>.,/. The list of unacceptable special characters includes the following: `|"?.



Note

For more information about how to create or modify an IP ACL on a standalone Content Engine, see the next section, “[Creating or Modifying IP ACLs on Standalone Content Engines](#).”

Creating or Modifying IP ACLs on Standalone Content Engines

To configure IP ACLs on a standalone Content Engine, follow these steps:

- Step 1** Access the Content Engine CLI in global configuration mode.

```
ContentEngine(config)#
```

- Step 2** From global configuration mode, access the appropriate IP ACL configuration mode, and specify the name or number of the IP ACL that you want to create, modify, or view.

- To create or modify a standard IP ACL, use the **ip access-list standard** global configuration command to enter into standard IP ACL configuration mode.

```
ip access-list standard {acl-name | acl-num}
```

The following example shows how to create or modify a standard IP ACL that has the ACL number of 59:

```
ContentEngine(config)# ip access-list standard 59
```

The CLI enters standard IP ACL configuration mode, in which all subsequent commands apply to the current standard IP ACL, and the following prompt appears:

```
ContentEngine(config-std-nacl)#
```

- To create or modify an extended IP ACL, use the **ip access-list extended** command to enter into extended IP ACL configuration mode.

```
ip access-list extended {acl-name | acl-num}
```

The following example shows how to create or modify an extended IP ACL named test2 by specifying its name:

```
ContentEngine(config)# ip access-list extended test2
```

The CLI enters extended IP ACL configuration mode, in which all subsequent commands apply to the current extended IP ACL, and the following prompt appears:

```
ContentEngine(config-ext-nacl)#
```

- Step 3** To add, delete, or modify conditions in a standard ACL, enter the following commands from the standard IP ACL configuration mode:

- a.** To add a line to the standard IP ACL, use the following syntax.

For example, choose a purpose (permit or deny) that specifies whether a packet is to be passed or dropped, enter the source IP address, and enter the source IP wildcard address.

```
[insert line-num] {deny | permit} {source-ip [wildcard] | host source-ip | any}
```

- b.** To delete a line from the standard IP ACL, use the following syntax:

```
delete line-num
```

- c.** To move a line to a new position within the standard IP ACL, use the following syntax:

```
move old-line-num new-line-num
```



Note For a list of extended IP ACL conditions, see [Table 19-4](#).

Step 4 To add, delete, or modify conditions in an extended ACL, enter the following commands from extended IP ACL configuration mode:

- a. To delete a line from the extended IP ACL, use the following syntax:

```
delete line-num
```

- b. To move a line to a new position within the extended IP ACL, use the following syntax:

```
move old-line-num new-line-num
```

- c. To add a condition to the extended IP ACL, enter the options according to the protocol you choose:

- For IP, use the following syntax to add a condition:

```
[insert line-num] {deny | permit} {gre | ip | proto-num}
{source-ip wildcard | host source-ip | any} {dest-ip wildcard |
host dest-ip | any}
```

```
[no] {deny | permit} {gre | ip | proto-num} {source-ip wildcard |
host source-ip | any} {dest-ip wildcard | host dest-ip | any}
```

- For TCP, use the following syntax to add a condition:

```
[insert line-num] {deny | permit} tcp {source-ip wildcard |
host source-ip | any} [operator port [port]] {dest-ip wildcard |
host dest-ip | any} [operator port [port]] [established]
```

```
no {deny | permit} tcp {source-ip wildcard | host source-ip | any}
[operator port [port]] {dest-ip wildcard | host dest-ip | any}
[operator port [port]] [established]
```

- For UDP, use the following syntax to add a condition:

```
[insert line-num] {deny | permit} udp {source-ip wildcard |
host source-ip | any} [operator port [port]] {dest-ip wildcard |
host dest-ip | any} [operator port [port]]
```

```
no {deny | permit} udp {source-ip wildcard | host source-ip | any}
[operator port [port]] {dest-ip wildcard | host dest-ip | any} |
[operator port [port]]
```

- For ICMP, use the following syntax to add a condition:

```
[insert line-num] {deny | permit} icmp {source-ip wildcard |
host source-ip | any} {dest-ip wildcard | host dest-ip | any}
[icmp-type [code] | icmp-msg]
```

```
no {deny | permit} icmp {source-ip wildcard | host source-ip | any}
{dest-ip wildcard | host dest-ip | any} [icmp-type [code] | icmp-msg]
```



Note For extended IP ACLs, the **wildcard** parameter is required if the **host** keyword is not specified. For a list of the keywords that you can use to match specific ICMP message types and codes, see [Table 19-3](#). For a list of supported UDP and TCP keywords, see [Table 19-1](#) and [Table 19-2](#). For a list of extended IP ACL conditions, see [Table 19-5](#).

Step 5 To add another condition to a standard IP ACL, repeat [Step 3](#). To add another condition (entry) to an extended IP ACL, repeat [Step 4](#).

Step 6 Activate and apply this IP ACL to a specific interface on this Content Engine by using the **interface** and **ip access-group** commands.

For more information about activating and applying an IP ACL to a specific interface, see the “[Activating an IP ACL on an Interface](#)” section on page 19-16 and the “[Applying an IP ACL to an Application](#)” section on page 19-17.

List of Keywords for Extended IP ACLs

Table 19-1 lists the UDP keywords that you can use with extended IP ACLs.

Table 19-1 UDP Keywords and Port Numbers

CLI Keyword	Description	UDP Port Number
bootpc	Bootstrap Protocol (BOOTP) client	68
bootps	Bootstrap Protocol (BOOTP) server	67
domain	Domain Name Service (DNS)	53
mms	Microsoft Media Server Protocol	1755
netbios-dgm	NetBIOS datagram service	138
netbios-ns	NetBIOS name service	137
netbios-ss	NetBIOS session service	139
nfs	Network File Server service	2049
ntp	Network Time Protocol	123
snmp	Simple Network Management Protocol	161
snmptrap	SNMP traps	162
tacacs	Terminal Access Controller (TAC) Access Control System	49
tftp	Trivial File Transfer Protocol	69
wccp	Web Cache Communication Protocol	2048

Table 19-2 lists the TCP keywords that you can use with extended IP ACLs.

Table 19-2 TCP Keywords and Port Numbers

CLI Keyword	Description	TCP Port Number
domain	Domain Name Service	53
exec	Exec (RCP)	512
ftp	File Transfer Protocol	21
ftp-data	FTP data connections (used infrequently)	20
https	Secure HTTP	443
nfs	Network File Server service	2049
rtsp	Real-Time Streaming Protocol	554

Table 19-2 TCP Keywords and Port Numbers (continued)

CLI Keyword	Description	TCP Port Number
ssh	Secure Shell login	22
tacacs	Terminal Access Controller (TAC) Access Control System	49
telnet	Telnet	23
www	World Wide Web (HTTP)	80

Table 19-3 lists the keywords that you can use to match specific ICMP message types and codes.

Table 19-3 Keywords for ICMP Message Type and Code

administratively-prohibited	alternate-address
conversion-error	dod-host-prohibited
dod-net-prohibited	echo
echo-reply	general-parameter-problem
host-isolated	host-precedence-unreachable
host-redirect	host-tos-redirect
host-tos-unreachable	host-unknown
host-unreachable	information-reply
information-request	mask-reply
mask-request	mobile-redirect
net-redirect	net-tos-redirect
net-tos-unreachable	net-unreachable
network-unknown	no-room-for-option
option-missing	packet-too-big
parameter-problem	port-unreachable
precedence-unreachable	protocol-unreachable
reassembly-timeout	redirect
router-advertisement	router-solicitation
source-quench	source-route-failed
time-exceeded	timestamp-reply
timestamp-request	traceroute
ttl-exceeded	unreachable

IP ACL Conditions

Table 19-4 describes the standard IP ACL conditions.

Table 19-4 Standard IP ACL Conditions

Parameter	Description
insert	(Optional) Inserts the conditions following the specified line number into the standard IP ACL.
<i>line-num</i>	Identifies the entry at a specific line number in the standard IP ACL.
deny	Causes packets that match the specified conditions to be dropped.
permit	Causes packets that match the specified conditions to be accepted for further processing.
<i>source-ip</i>	Source IP address. The number of the network or host from which the packet is being sent, specified as a 32-bit quantity in 4-part dotted decimal format (for example, 0.0.0.0).
<i>wildcard</i>	Specifies the portions of the preceding IP address to match, expressed using 4-digit, dotted-decimal notation. Bits to match are identified by a digital value of 0; bits to ignore are identified by a 1. For standard IP ACLs, the wildcard parameter of the ip access-list command is always optional. If the host keyword is specified for a standard IP ACL, then the wildcard parameter is not allowed.
host	Matches the following IP address.
any	Matches any IP address.
delete	Deletes the specified entry (condition) from the standard IP ACL.
<i>line-num</i>	Identifies the entry at a specific line number in the standard IP ACL.
list	Lists the specified entries (or all entries when none are specified).
<i>start-line-num</i>	(Optional) Starting line number of the list.
<i>end-line-num</i>	(Optional) Ending line number of the list.
move	Moves the specified entry in the standard IP ACL to a new position in the list.
<i>old-line-num</i>	Specifies the line number of the entry to move.
<i>new-line-num</i>	Specifies the new position of the entry. The existing entry is moved to this new position in the standard IP ACL.

Table 19-5 describes the extended IP ACL conditions.

Table 19-5 Extended IP ACL Conditions

Parameter	Description
insert	(Optional) Inserts the conditions at the specified line number into the extended IP ACL.
<i>line-num</i>	Identifies the entry at a specific line number in the extended IP ACL.
deny	Causes packets that match the specified conditions to be dropped.
permit	Causes packets that match the specified conditions to be accepted for further processing.
<i>source-ip</i>	Source IP address.
<i>wildcard</i>	Specifies the portions of the preceding IP address to match, expressed using 4-digit, dotted-decimal notation. Bits to match are identified by a digital value of 0; bits to ignore are identified by a 1. For extended IP ACLs, the wildcard parameter is always required unless the host keyword is specified. If the host keyword is specified for an extended IP ACL, then the wildcard parameter is not allowed.

Table 19-5 Extended IP ACL Conditions (continued)

Parameter	Description
host	Matches the following IP address.
any	Matches any IP address.
gre	Matches packets using the generic routing encapsulation (GRE) protocol.
ip	Matches all IP packets.
<i>proto-num</i>	Specifies the IP protocol number.
tcp	Matches packets using the TCP protocol.
udp	Matches packets using the UDP protocol.
<i>operator</i>	<p>(Optional) Specifies the operator to use with specified ports, where lt = less than, gt = greater than, eq = equal to, neq = not equal to, and range = an inclusive range. The following shows an example of an extended IP ACL that uses the equal to operator.</p> <pre>ContentEngine(config)# ip access-list extended example ContentEngine(config-ext-nacl)# permit tcp any any eq www ContentEngine(config-ext-nacl)# permit tcp host 10.1.1.5 any eq ssh</pre>
<i>port</i>	<p>(Optional) Specifies the port, using a number (0–65535) or a keyword; 2 port numbers are required with the range operator. Use any of the following keywords with TCP: domain, exec, ftp, ftp-data, https, mms, nfs, rtsp, ssh, tacacs, telnet, and www. Use any of the following keywords with UDP: bootpc, bootps, domain, mms, netbios-dgm, netbios-ns, netbios-ss, nfs, ntp, snmp, snmptrap, tacacs, ftpp, and wccp. For example:</p> <pre>ContentEngine(config)# ip access-list extended example ContentEngine(config-ext-nacl)# permit tcp any any eq www ContentEngine(config-ext-nacl)# permit tcp host 10.1.1.5 any eq ssh</pre>
<i>dest-ip</i>	Destination IP address.
established	(Optional) Matches TCP packets with the ACK or RST bits set.
icmp	Matches ICMP packets.
<i>icmp-type</i>	(Optional) Matches by ICMP message type, expressed as a number (0–255).
<i>code</i>	(Optional) Used with <i>icmp-type</i> to further match by ICMP code type, expressed as a number from 0 to 255.
<i>icmp-msg</i>	(Optional) Matches using a combination of ICMP message type and code types, as expressed by the keywords listed in Table 19-3.
delete	Deletes the specified entry (condition) from the extended IP ACL.
<i>line-num</i>	Identifies the entry at a specific line number in the extended IP ACL.
list	Lists the specified entries (or all entries when none are specified).
<i>start-line-num</i>	(Optional) Starting line number of the list.
<i>end-line-num</i>	(Optional) Ending line number of the list.
move	Moves the specified entry in the list to a new position in the list.
<i>old-line-num</i>	Specifies the line number of the entry to move.
<i>new-line-num</i>	Specifies the new position of the entry. The existing entry is moved to this position in the access list.
exit	Returns to the CLI global configuration mode prompt.

Activating an IP ACL on an Interface

In the ACNS 5.1 software and later releases, there are controls that allow various services to be tied to a particular interface. To activate an IP ACL on a particular interface on a standalone Content Engine, use the **ip access-group** interface configuration command. You can use one outbound IP ACL and one inbound IP ACL on each interface.

Before entering the **ip access-group** command, enter interface configuration mode for the interface to which you want to apply the IP ACL.

The following commands apply and activate the IP ACL named `acl-out` pm outbound traffic on the FastEthernet interface slot 0/port 0:

```
ContentEngine(config)# interface FastEthernet 0/0
ContentEngine(config-if)# ip access-group acl-out out
```

The following commands apply and activate the IP ACL named `example` on the inbound traffic on the Gigabit Ethernet interface on port 1/slot 0:

```
ContentEngine(config)# interface gigabitethernet 1/0
ContentEngine(config-if)# ip access-group example in
ContentEngine(config-if)# exit
```

To apply and activate an IP ACL on a specific interface on the standalone Content Engine, follow these steps:

Step 1 Enter interface configuration mode for the interface to which you want to apply the IP ACL.

For example, the following example shows how to enter interface configuration mode for the Fast Ethernet interface on slot 0/port 0 of the Content Engine:

```
ContentEngine(config)# interface FastEthernet 0/0
ContentEngine(config-if)#
```

Step 2 Apply the predefined IP ACL to the specified interface.

For example, the following example shows how apply the predefined IP ACL named `acl-out` to outbound traffic on the FastEthernet interface slot 0/port 0:

```
ContentEngine(config-if)# ip access-group acl-out out
```

[Table 19-6](#) describes the parameters for the **ip access-group** command.

Table 19-6 Parameters for the **ip access-group** CLI Command

Parameter	Description
<i>acl-name</i>	Alphanumeric identifier up to 30 characters, beginning with a letter that identifies the IP ACL to apply to the current interface.
<i>acl-num</i>	Numeric identifier that identifies the IP ACL to apply to the current interface (1–99 for standard IP ACLs; 100–199 for extended IP ACLs).
in	Applies the specified IP ACL to inbound packets on the current interface.
out	Applies the specified IP ACL to outbound packets on the current interface.

Applying an IP ACL to an Application

In the ACNS 5.4.1 software and later releases, FTP has the following specific CLI configuration command to configure the use of an IP ACL for FTP native proxy-mode and transparently redirected (WCCP-redirection) connections:

```
ftp-native access-list in { std-acl-num | std-acl-name }
```

```
ftp-native access-list out { ext-acl-num | ext-acl-name }
```

Use the **ftp-native access-list in** global configuration command to configure a standard ACL for the following types of inbound FTP native connections: proxy mode connections and transparently redirected (WCCP-redirection) connections. For standard ACLs, access control will be based on the source address of the FTP client that sent the native FTP request.

Use the **ftp-native access-list out** global configuration command to configure an extended ACL for the following types of outbound FTP native connections: proxy mode connections and transparently redirected (WCCP-redirection) connections. For extended ACLs, access control will be based on the source and destination addresses.

In the ACNS 5.1 software and later releases, SNMP and TFTP have the following specific CLI configuration commands to configure the use of an IP ACL:

```
snmp-server access-list { std-acl-num | std-acl-name }
```

```
tftp-server access-list { std-acl-num | std-acl-name }
```



Note The **snmp-server access-list** and **tftp-server access-list** global configuration commands can only accept the name or number for a standard IP ACL and not an extended IP ACL.

In the ACNS 5.2.1 software and later releases, specify an IP access list that the Content Engine applies to inbound WCCP GRE encapsulated traffic that it receives by using the **wccp access-list** global configuration command.

```
wccp access-list { acl-num | acl-name }
```

The WCCP access list feature supports both standard and extended access lists, and is not restricted to only standard access lists, as is the case with SNMP and TFTP server access lists.

Other application traffic (for example, Telnet and SSH) can be controlled by applying an IP ACL to an interface (typically to inbound traffic) on the standalone Content Engine.



Note

In the ACNS 5.1 software and later releases, you need to use the **ip access-list** global configuration command to allow or deny access to the user through the TFTP protocol. Unless the IP ACL is configured for the TFTP service, the security of the content can be at risk, and TFTP will not function properly.

In the ACNS 5.0 software, TFTP access was denied to the user by default. To allow access to the user, the administrator had to use the **trusted-host** command. The **trusted-host** command has been deprecated in the ACNS 5.1 software. Consequently, if the **trusted-host** command is used on Content Engines that are running an ACNS software release earlier than Release 5.1, and the device is subsequently upgraded to the ACNS software 5.1 and later releases, the **trusted-host** command shows up in the CLI but does not have any effect on the TFTP protocol. The trusted host configurations can be deleted by using the **no trusted-host** command.

Using IP ACLs to Control SNMP Access

To use standard IP ACLs to control access to the SNMP agent on a standalone Content Engine, follow these steps:

-
- Step 1** Create an IP ACL to control access to the SNMP agent on the Content Engine by using the **ip access-list standard** command.
- Step 2** Associate this IP ACL with the SNMP server (the standalone Content Engine), and activate this standard IP ACL on the Content Engine.

```
ContentEngine(config)# snmp-server access-list {std-acl-num | std-acl-name}
```

- *std-acl-name* is the name of the standard IP ACL that you want to associate with this Content Engine.
- *std-acl-num* is the number of the standard IP ACL that you want to associate with this Content Engine.

The SNMP agent on the Content Engine checks against the specified IP ACL (for example, ACL 1) before accepting or dropping incoming packets.

```
ContentEngine(config)# snmp-server access-list 1
```

Using IP ACLs to Control TFTP Access

To use standard IP ACLs to control access to the TFTP service on a standalone Content Engine, follow these steps:

-
- Step 1** Create an access list to control access to the TFTP service that is running on the standalone Content Engine.

For example, the following commands define an access list that permits access to the TFTP service for TFTP clients on the 192.168.1.0 subnetwork:

```
ContentEngine(config)# ip access-list standard 2
ContentEngine(config-std-nacl)# permit 192.168.1.0 0.0.0.255
ContentEngine(config-std-nacl)# exit
ContentEngine(config)#
```

- Step 2** Associate this IP ACL (ACL 2) with the TFTP server (the standalone Content Engine), and activate this standard IP ACL on the Content Engine. The Content Engine checks against the specified access control list before permitting or denying access to the TFTP service that it is running.

The following example configures the Content Engine to check against access control list 2 before permitting or denying TFTP access to a user (a TFTP client).

```
ContentEngine(config)# tftp-server access-list 2
```

Using IP ACLs to Control Native FTP Access

In the ACNS 5.4.1 software and later releases, you can use IP ACLs to permit or deny access to the native FTP proxy service that is running on a standalone Content Engine. This support is available for the following types of incoming native FTP requests:

- Nontransparent FTP native requests
- Transparent FTP native requests (incoming FTP native requests that are transparently intercepted and redirected to a Content Engine by a WCCP-enabled router)

To support this new feature, the following two CLI commands were added in the ACNS 5.4.1 software release:

```
ftp-native access-list in {std-acl-num | std-acl-name}
ftp-native access-list out {ext-acl-num | ext-acl-name}
```

For a standard IP ACL, access control for the incoming FTP native request is based on the source address. For an extended IP ACL, access control for the FTP native requests that are outbound to the origin server is based on the source and destination addresses.

To use IP ACLs to control access to the native FTP proxy service that is running on a standalone Content Engine, follow these steps:

Step 1 Create an access list to control access to the native FTP proxy service that is running on the standalone Content Engine.

For example, the following commands define an access list that permits access to the native FTP proxy service for FTP clients on the 192.168.1.0 subnetwork:

```
ContentEngine(config)# ip access-list standard 3
ContentEngine(config-std-nacl)# permit 192.168.1.0 0.0.0.255
ContentEngine(config-std-nacl)# exit
ContentEngine(config)#
```

Step 2 Associate this IP ACL (ACL 3) with the native FTP proxy service, and activate this standard IP ACL on the Content Engine. The Content Engine checks against the specified access control list before permitting or denying access to the native FTP proxy service.

The following example configures the Content Engine to check against access control list 3 before permitting or denying FTP proxy access to a user (for example, such FTP clients as a Reflection X client, a WS-FTP client, or a UNIX or DOS command line FTP program):

```
ContentEngine(config)# ftp-native access-list in 3
```

Step 3 (Optional). Configure customized response messages that the Content Engine will send to an FTP client in response to an incoming proxy-mode connection.

In the ACNS 5.4.1 software and later releases, you can use the **ftp-native custom-message** global configuration command to configure customized response messages, which the Content Engine sends to an FTP client in response to an incoming proxy mode connection. You can use the **ftp-native custom-message EXEC** command to create, upload, and download a file that contains one of the following custom messages:

- A custom welcome message for welcoming proxy-mode connections from FTP clients
- A custom error message for denying a user (an FTP client) access to the native FTP proxy service based on the configured IP ACLs for the native FTP proxy service

For more information about this topic, see the [“Creating Custom Messages for FTP Proxy Responses for FTP Native Requests”](#) section on page 5-19.

- Step 4** From an FTP client, issue a native FTP request to the Content Engine to verify that the defined access control list works properly.

The following example issues a native FTP request from an FTP client that is on the 192.168.1.0 subnetwork. In this case, the FTP client is a UNIX command line FTP program and the Content Engine has an IP address of 10.1.1.50. Because the source address (the FTP client's IP address) is in the 192.168.1.0 subnetwork, this FTP client is granted access to the FTP service and the Content Engine displays the welcome message to the FTP client:

```
shell# ftp -d 10.1.1.50 8501
Connected to 10.1.1.50
220 Welcome to FTP-proxy. Login to the proxy using username and password.
Name (10.1.1.50:admin):
```

Using IP ACLs to Control WCCP Access

To use standard or extended IP ACLs to control WCCP access on a standalone Content Engine, follow these steps:

- Step 1** Create a standard or extended IP ACL to control WCCP access on the Content Engine.
- To create a standard IP ACL, use the **ip access-list standard** command.
 - To create an extended IP ACL, use the **ip access-list extended** command.
- Step 2** Associate the IP ACL with the Content Engine, and activate this IP ACL on the Content Engine.

```
ContentEngine(config)# wccp access-list {acl-num | acl-name}
```

- acl-name* is the name of the standard or extended IP ACL that you want to associate with this Content Engine.
- acl-num* is the number of the standard or extended IP ACL that you want to associate with this Content Engine.

The Content Engine applies the specified IP ACL (for example, ACL 2) to WCCP GRE inbound traffic.

```
ContentEngine(config)# wccp access-list 2
```

Configuring WCCP Access Lists for Standalone Content Engines

In the ACNS 5.2.1 software and later releases, specify an IP access list that the Content Engine applies to WCCP GRE encapsulated traffic that it receives inbound by using the **wccp access-list** global configuration command.

```
wccp access-list {acl-name | acl-number}
```

acl-name or *acl-number* represent either a standard or an extended IP access list.

The default is that no WCCP access list is configured; therefore, the WCCP access lists are not shown as part of the Content Engine's configuration.

The following example shows sample output from the **show ip access-list EXEC** command if a WCCP access list has been configured on a Content Engine:

```
Content Engine# show ip access-list
Space available:
  48 access lists
  497 access list conditions

Standard IP access list test
  1 permit 10.1.1.1
    (implicit deny any:0 matches)
  total invocations:0
Extended IP access list no_www.linux.org
  1 deny tcp any host 10.1.1.1 (29 matches)
  2 permit ip any any (30 matches)
    (implicit fragment permit:0 matches)
    (implicit deny ip any any:0 matches)
  total invocations:59

Interface access list references:
  GigabitEthernet 2/0 inbound pc_test (Not Defined)

Application access list references:
  snmp-server standard test
    UDP ports:none
  tftp_server standard test4
    UDP ports: 69 (List Not Defined)
  WCCP either no_www.linux.org
    Any IP Protocol
Content Engine#
```

In the ACNS 5.2.1 software and later releases, the output of the **show wccp gre EXEC** command includes two counters that are related to the WCCP access list feature:

```
-----
Packets w/WCCP GRE received too small:    0
Packets dropped due to IP access-list deny:29
-----
```

The first counter represents the number of properly encapsulated WCCP GRE packets that were dropped because they were too short to contain a complete IP packet header.

The second counter represents the number of packets that were dropped because they were denied by the specified WCCP access list.

The following example shows sample output from the **show wccp gre EXEC** command when WCCP access lists are defined on the Content Engine:

```
Content Engine# show wccp gre
Transparent GRE packets received:          366
Transparent non-GRE packets received:      0
Transparent non-GRE packets passed through:0
Total packets accepted:                    337
Invalid packets received:                  0
Packets received with invalid service:      0
Packets received on a disabled service:      0
Packets received too small:                 0
Packets dropped due to zero TTL:             0
Packets dropped due to bad buckets:          0
Packets dropped due to no redirect address:0
Connections bypassed due to load:           0
Packets sent back to router:                0
Packets sent to another CE:                 0
GRE fragments redirected:                   0
```

```

Packets failed GRE encapsulation:          0
Packets dropped due to invalid fwd method: 0
Packets dropped due to insufficient memory:0
Packets bypassed, no conn at all:         0
Packets bypassed, no pending connection:  0
Packets due to clean wccp shutdown:       0
Packets bypassed due to bypass-list lookup:0
Packets received with client IP addresses: 0
Conditionally Accepted connections:       0
Conditionally Bypassed connections:       0
Packets w/WCCP GRE received too small:    0
Packets dropped due to IP access-list deny:29
Content Engine#

```

**Note**

You can also display the above output by entering the `show statistics wccp gre EXEC` command.

Configuration Examples

The following example shows how to use the `ip access-list extended` global configuration command to create an extended IP ACL named `example`. This extended IP ACL is created to allow all web traffic but to only allow a specific host (host 10.1.1.5) administrative access using SSH.

```

ContentEngine(config)# ip access-list extended example
ContentEngine(config-ext-nacl)# permit tcp any any eq www
ContentEngine(config-ext-nacl)# permit tcp host 10.1.1.5 any eq ssh
ContentEngine(config-ext-nacl)# exit

```

The following example shows a standalone Content Engine that has been configured to use interface access lists and application access lists:

```

ContentEngine# show ip access-list
Space available:
  47 access lists
  492 access list conditions

Standard IP access list 1
  1 permit 10.1.1.2
  2 deny 10.1.2.1
    (implicit deny any: 2 matches)
  total invocations: 2
Extended IP access list 100
  1 permit tcp host 10.1.1.1 any
  2 permit tcp host 10.1.1.2 any
  3 permit tcp host 10.1.1.3 any
    (implicit fragment permit: 0 matches)
    (implicit deny ip any any: 0 matches)
  total invocations: 0
Standard IP access list test
  1 permit 1.1.1.1 (10 matches)
  2 permit 1.1.1.3
  3 permit 1.1.1.2
    (implicit deny: 2 matches)
  total invocations: 12
Interface access list references:
FastEthernet 0/0 inbound 100
Application access list references:
tftp_server standard 1
UDP ports: 69

```

Deleting an IP ACL

To delete an IP ACL (including all conditions and references in network interfaces and applications) from the Content Engine database, follow these steps:

Step 1 Access the Content Engine CLI in global configuration mode.

```
ContentEngine(config)#
```

Step 2 Specify the name or number of the IP ACL that you want to delete.

- To delete a standard IP ACL, specify the standard IP ACL that you want to delete.

```
ContentEngine(config)# no ip access-list standard {acl-name | acl-num}
```

The following example shows how to delete the standard IP ACL named test2:

```
ContentEngine(config)# no ip access-list standard test2
```

- To delete an extended IP ACL, specify the extended IP ACL that you want to delete.

```
ContentEngine(config)# no ip access-list extended {acl-name | acl-num}
```

The following example shows how to delete the extended IP ACL named example:

```
ContentEngine(config)# no ip access-list extended example
```

Viewing the Configuration of an IP ACL

To view the configuration of the IP ACLs currently defined on a Content Engine, use the **show ip access-list EXEC** command:

```
show ip access-list [acl-name | acl-num]
```

The **show ip access-list EXEC** command allows you to display configuration information about the IP ACLs that have been defined on the current system (in this case, the standalone Content Engine). Unless you identify a specific IP ACL by name or number, the system displays information about all the defined IP ACLs, including the following sections:

- Available space for new lists and conditions
- Defined IP ACLs
- References by interface and application

The following example shows sample output from the **show ip access-list EXEC** command when a specific IP ACL is not specified:

```
ContentEngine# show ip access-list
Space available:
  47 access lists
  492 access list conditions

Standard IP access list 1
  1 permit 10.1.1.2
  2 deny 10.1.2.1
    (implicit deny any: 2 matches)
  total invocations: 2
Extended IP access list 100
```

```

1 permit tcp host 10.1.1.1 any
2 permit tcp host 10.1.1.2 any
3 permit tcp host 10.1.1.3 any
  (implicit fragment permit: 0 matches)
  (implicit deny ip any any: 0 matches)
total invocations: 0
Standard IP access list test
1 permit 1.1.1.1 (10 matches)
2 permit 1.1.1.3
3 permit 1.1.1.2
  (implicit deny: 2 matches)
total invocations: 12

Interface access list references:
FastEthernet 0/0 inbound 100
Application access list references:
tftp_server standard 1
UDP ports: 69

```

The following example shows sample output from the **show ip access-list EXEC** command for the IP ACL named test:

```

ContentEngine# show ip access-list test
Standard IP access list test
1 permit 1.1.1.1 (10 matches)
2 permit 1.1.1.3
3 permit 1.1.1.2
  (implicit deny: 2 matches)
total invocations: 12

```

**Note**

The system displays the number of packets that have matched a condition statement only if the number is greater than zero.

Clearing an IP ACL Counter

To clear the IP ACL counter and to reset the statistical information for an IP ACL on a Content Engine, use the **clear ip access-list counter EXEC** command:

```
ContentEngine# clear ip access-list counters {acl-name | acl-num}
```

Use this EXEC command to clear the IP ACL counters associated with the condition statements of all the existing IP ACLs. If you specify an IP ACL name or number, then only the specified list's counters will be cleared.