Controlling CSS Access

This chapter describes how to configure access to the CSS including network traffic. Information in this chapter applies to all models of the CSS, except where noted.

This chapter contains the following major sections:

- Changing the Administrative Username and Password
- Creating Usernames and Passwords
- Controlling Remote User Access to the CSS
- Controlling Administrative Access to the CSS
- Controlling CSS Network Traffic Through Access Control Lists
- Configuring Network Qualifier Lists for ACLs

Changing the Administrative Username and Password

During the initial log in to the CSS you enter the default user name **admin** and the default password **system** in lowercase text. For security reasons, you should change the administrative username and password. Security on your CSS can be compromised because the administrative username and password are configured to be the same for every CSS shipped from Cisco Systems.

The administrative username and password are stored in nonvolatile random access memory (NVRAM). Each time you reboot the CSS, it reads the username and password from NVRAM and reinserts them in to the user database. SuperUser status is assigned to the administrative username by default.

You can change the administrative username and password, but because the information is stored in NVRAM, you cannot permanently delete them. If you delete the administrative username using the **no username** command, the CSS deletes the username from the running-config file, but restores the username from NVRAM when you reboot the CSS.

Use the **username-offdm** *name* **password** *text* command to change the administrative username or password.

Note You can also use the Security Options menu from the Offline DM menu (accessed during the boot process) to change the administrative username and password. Refer to the *Cisco Content Services Switch Administration Guide* for information on the Offline DM menu.

For example, to change the default administrative username and password to a different username and password, enter.

```
(config)# username-offdm bobo password secret
```

Creating Usernames and Passwords

Logging into the CSS requires a username and password. The CSS supports a maximum of 32 usernames, including the administrator and technician usernames. You can assign each user with SuperUser or User status.

- **User** - Allows access to a limited set of commands that enable you to monitor and display CSS parameters, but not change them. A User prompt ends with the > symbol.
- **SuperUser** - Allows access to the full set of CLI commands, including those in User mode, that enable you to configure the CSS. A SuperUser prompt ends with the # symbol.

  From SuperUser mode, you can enter global configuration mode and its subordinate configuration modes. If you do not specify **superuser** when configuring a new user, the new user has only user-level status by default.

Caution Creating or modifying a username and password is restricted to CSS users who are identified as either administrators or technicians, and it is contingent on whether the **restrict user-database** command has been entered.

Use the **username** command to create usernames and passwords to log in to the CSS. The syntax for this global configuration mode command is:

username *name* [**des-password**|**password**] *password* {**superuser**} {**dir-access** *access*}

The following example creates a SuperUser named *picard* with a password of *captain*.

```
(config)# username picard password "captain" superuser
```

The options and variables are as follows:

- *name* - Sets the username you want to assign or change. Enter an unquoted text string with no spaces and a maximum of 16 characters. To see a list of existing usernames, enter **username ?**.
- **des-password** - Specifies the password is Data Encryption Standard (DES) encrypted. Use this option only when you are creating a file for use as a script or a startup configuration file. Enter the DES password as a case-sensitive unquoted text string 6 to 64 characters in length.

- **password** - Specifies the password is not encrypted. Use this option when you use the CLI to dynamically create users.
- *password* - The password. Enter an unquoted text string with no spaces and a length of 6 to 16 characters. The CSS allows all special characters in a password except for the percent sign (%).

Note If you specify the **des-password** option, you must know the encrypted form of this password to successfully log in to the CSS. You can find the CSS encrypted password in the running configuration. To display the CSS running configuration, use the **show running-config** command (see the "Creating Usernames and Passwords" section).

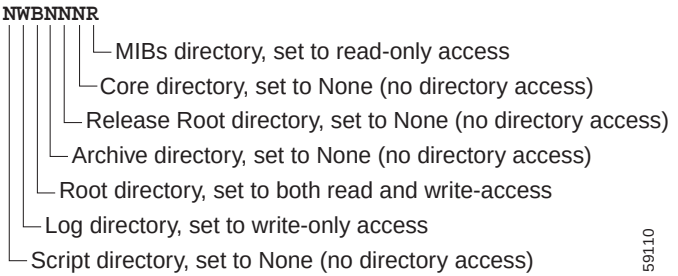
- **superuser** - Specifies SuperUser privileges to allow a user to access SuperUser mode. If you do not enter this option, the user can only access User mode.
- **dir-access** - (Optional) Defines the CSS directory access privileges for the username. There are access privileges assigned to the seven CSS directories, in the following order: Script, Log, Root (installed CSS software), Archive, Release Root (configuration files), Core, and MIBs. By default, users have both read- and write-access privileges (B) to all seven directories. Administrators or technicians can use the **dir-access** option to selectively implement a set of directory access privileges for each user. Changing the access level also affects the use of the CLI commands associated with directories.

  To use the **dir-access** option, you must first specify the **restrict user-database** command to implement security restrictions for the CSS user database.

- *access* - Specifies directory access privileges for the username. By default, users have both read- and write-access privileges (B) to all seven directories. Enter, in order, one of the following access privilege codes for each of the seven CSS directories:
  - **R** - Read-only access to the CSS directory
  - **W** - Write-only access to the CSS directory
  - **B** - Both read- and write-access privileges to the CSS directory
  - **N** - No access privileges to the CSS directory

Figure 1-1 illustrates the directory access privileges for a username.

Figure 1-1 CSS Directory Access Privileges



For example, to define directory access for username *picard*, enter:

```
(config)# username picard password "captain" superuser NWBNNNR
```

To display a list of existing usernames, enter:

```
(config)# username ?
```

To remove an existing username, enter:

```
(config)# no username picard
```

To change a user password, reenter the **username** command and specify the new password. Remember to include SuperUser privileges if required. For example:

```
(config)# username picard password "flute" superuser
```

Caution The **no username** command removes a user permanently. Make sure you want to perform this action because you cannot undo this command.

Controlling Remote User Access to the CSS

To control access to the CSS, you can configure the CSS to authenticate remote (virtual) or console users. The CSS can authenticate users by using the local user database, RADIUS server, or TACACS+ server. You can also allow user access without authenticating or disallowing all remote user access to the CSS.

You can set a maximum of three authentication methods: a primary, secondary, or tertiary authentication method. The primary method is the first authentication method that the CSS tries. If the primary authentication method fails (for example, the RADIUS server is down or is unreachable), the CSS tries the secondary method. And if the secondary method fails, then the CSS tries the tertiary method. In the event the tertiary method also fails, the CSS displays a message that authentication has failed.

The CSS does not attempt a secondary or tertiary authentication method under the following conditions:

- If the authentication method is **local**, and the local username is not found in the local user database.
- If the authentication method is **local** and the local username is found in the local user database, but the password is invalid.
- If the authentication method is **radius**, and the RADIUS server rejects the primary authentication request from the CSS.
- If the authentication method is **tacacs**, and the TACACS+ server rejects the primary authentication request from the CSS.

Before you can use RADIUS or TACACS+ as either the virtual authentication method or the console authentication method, you must enable communication with the RADIUS or TACACS+ security server. Use either the **radius-server** command (refer to the Chapter 3, Configuring the CSS as a Client of a RADIUS Server) or the **tacacs-server** command (see the Chapter 4, Configuring the CSS as a Client of a TACACS+ Server).

This section includes the following topics:

To display virtual and console authentication settings, use the **show user-database** command.

Configuring Virtual Authentication

Virtual authentication allows remote users to log in to the CSS when they are using FTP, Telnet, SSHD, or the CiscoView Device Manager (CVDM) interface with or without requiring a username and password. The CSS can also deny access to all remote users.

You can configure the CSS to authenticate users by using the local database, RADIUS server, or TACACS+ server. By default, the CSS uses the local database as the primary method to authenticate users and disallows user access for the secondary and tertiary method.

Use the **virtual authentication** command to configure the primary, secondary, or tertiary virtual authentication method. The syntax for this global configuration command is:

virtual authentication [**primary**|**secondary**|**tertiary** [**local**|**radius**|**tacacs**|**disallowed**]]

The options for this command are as follows:

- **primary** - Defines the first authentication method that the CSS uses. The default primary virtual authentication method is the local user database.
- **secondary** - Defines the second authentication method that the CSS uses if the first method fails. The default secondary virtual authentication method is to disallow all user access.

Note If you are configuring a TACACS+ server as the primary authentication method, define a secondary authentication method, such as **local**.

- **tertiary** - Defines the third authentication method that the CSS uses if the second method fails. The default tertiary virtual authentication method is to disallow all user access.
- **local** - The CSS uses the local user database for authentication.
- **radius** - The CSS uses the configured RADIUS server for authentication.
- **tacacs** - The CSS uses the configured TACACS+ server for authentication.
- **disallowed** - The CSS disallows access by all remote users. Entering this option does not terminate existing connections.

To remove users currently logged in to the CSS, use the **disconnect** command.

To define the TACACS+ server as the primary virtual authentication method, enter:

```
#(config) virtual authentication primary tacacs
```

To define local user database as the secondary virtual authentication method, enter:

```
#(config) virtual authentication secondary local
```

Configuring Console Authentication

Console authentication allows users to log in to the CSS through a terminal connected to the console port with or without requiring a username and password. The CSS cannot disallow user access as a primary authentication method; however, it can disallow user access as a secondary or tertiary authentication method.

You can configure the CSS to authenticate users by using the local database, RADIUS server, or TACACS+ server. By default, the CSS uses the local database as the primary method to authenticate users and disallows user access for the secondary and tertiary method.

Use the **console authentication** command to configure the primary, secondary, or tertiary console authentication method. The syntax for this global configuration command is:

console authentication [**primary** [**local**|**radius**|**tacacs**|**none**]
|**secondary**|**tertiary** [**local**|**radius**|**tacacs**|**none**|**disallowed**]]

The options for this command are as follows:

- **primary** - Defines the first authentication method that the CSS uses. The default primary console authentication method is the local user database.
- **local** - The CSS uses the local user database for authentication.
- **radius** - The CSS uses the configured RADIUS server for authentication.
- **tacacs** - The CSS uses the configured TACACS+ server for authentication.
- **none** - The CSS uses no authentication method. All users can access the CSS.

- **secondary** - Defines the second authentication method that the CSS uses if the first method fails. The default secondary console authentication method is to disallow all user access.

Note If you are configuring a TACACS+ server as the primary authentication method, define a secondary authentication method, such as **local**. If you do not configure a secondary method and use the default of **disallowed**, you have the possibility of being locked out of the CSS.

- **tertiary** - Defines the third authentication method that the CSS uses if the second method fails. The default tertiary console authentication method is to disallow all user access.
- **disallowed** - The CSS disallows access by all users (secondary or tertiary authentication method only). Entering this option does not terminate existing connections.

To remove users currently logged in to the CSS, use the **disconnect** command.

To define the TACACS+ server as the primary console authentication method, enter:

```
#(config) console authentication primary tacacs
```

To define local user database as the secondary console authentication method, enter:

```
#(config) console authentication secondary local
```

To disable authentication on the console port allowing users to access the CSS without a username and password, enter:

```
#(config) no console authentication
```

Controlling Administrative Access to the CSS

CSS access through a console, FTP, SSH, SNMP, and Telnet is enabled by default. The CSS supports a maximum of four FTP sessions and a maximum of four Telnet sessions. Use the **restrict** and **no restrict** commands to enable or disable console, FTP, SNMP, SSH, Telnet, user database, secure and unsecure XML, and CVDM data transfer to the CSS.

Specifying the **restrict** command does not prevent the CSS from listening for connection attempts on the restricted port. For TCP connections, the CSS completes the TCP 3-way handshake, then terminates the connection with an error to prevent any data transfer from occurring. For UDP SNMP connections, the CSS simply discards the packets.

To secure restricted ports from unauthorized access, configure ACL clauses to deny packets destined to these ports, while permitting normal traffic to flow through the CSS. You can also use ACLs to secure the CSS itself. See the "Controlling CSS Network Traffic Through Access Control Lists" section for information about configuring ACLs for the CSS.

Enabling Administrative Access to the CSS

To enable console, FTP, SNMP, SSH, Telnet, user database, secure and unsecure XML, and CVDM access to the CSS, use the following **no restrict** commands:

- **no restrict console** - Enables console access to the CSS (enabled by default).
- **no restrict ftp** - Enables FTP access to the CSS (enabled by default).
- **no restrict ssh** - Enables SSH access to the CSS (enabled by default).
- **no restrict snmp** - Enables SNMP access to the CSS (enabled by default).
- **no restrict telnet** - Enables Telnet access to the CSS (enabled by default).
- **no restrict user-database** - Enables users to clear the running-config file and create or modify usernames. Only administrator and technician users can perform these tasks (enabled by default).
- **no restrict secure-xml** - Enables the transfer of XML configuration files to the CSS through secure HTTPS SSL connections (disabled by default).
- **no restrict xml** - Enables the transfer of XML configuration files to the CSS through unsecure HTTP connections (disabled by default).

- **no restrict web-mgmt** - Enables CiscoView Device Manager (CVDM) access to the CSS (disabled by default).

Note Disable Telnet access when you want to use the Secure Shell Host (SSH) server. For information about configuring SSH, refer to Chapter 2, Configuring the Secure Shell Daemon Protocol.

For example, to enable CVDM user access, enter:

```
(config)# no restrict web-mgmt
```

Refer to the *Cisco Content Services Switch Administration Guide* for details on configuring the Simple Network Management Protocol (SNMP) features on your CSS. For details on making web-based configuration changes to the CSS using Extensible Markup Language (XML), refer to the *Cisco Content Services Switch Administration Guide*.

Disabling Administrative Access to the CSS

To disable console, FTP, SNMP, SSH, Telnet, user database, secure and unsecure XML, and CVDM access to the CSS, use the following **restrict** commands:

- **restrict console** - Disables console access to the CSS (enabled by default).
- **restrict ftp** - Disables FTP access to the CSS (enabled by default).
- **restrict snmp** - Disables SNMP access to the CSS (enabled by default).
- **restrict ssh** - Disables SSHD access to the CSS (enabled by default).
- **restrict telnet** - Disables Telnet access to the CSS (enabled by default).
- **restrict user-database** - Prevents users from clearing the running-config file and creating or modifying usernames. Only administrator and technician users can perform these tasks (enabled by default).
- **restrict secure-xml** - Disables the transfer of XML configuration files to the CSS through secure HTTPS SSL connections (disabled by default).
- **restrict xml** - Disables the transfer of XML configuration files to the CSS through unsecure HTTP connections (disabled by default).
- **restrict web-mgmt** - Disables CVDM access to the CSS (disabled by default).

For example, to disable Telnet access, enter:

```
(config)# restrict telnet
```

Controlling CSS Network Traffic Through Access Control Lists

The CSS provides traffic filtering capabilities with access control lists (ACLs). ACLs filter inbound network traffic by controlling whether packets are forwarded or blocked at the CSS interfaces. You can configure ACLs for routed network protocols, filtering the protocol packets as the packets pass through the CSS.

The following sections describe how to configure an ACL:

ACL Overview

ACLs configured on the CSS provide a basic level of security for accessing your network. Without ACLs on the CSS, all packets passing through VLAN circuits on the CSS could be allowed onto the entire network. With ACLs, you may want to permit all e-mail traffic on the CSS circuit, 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 the same area.

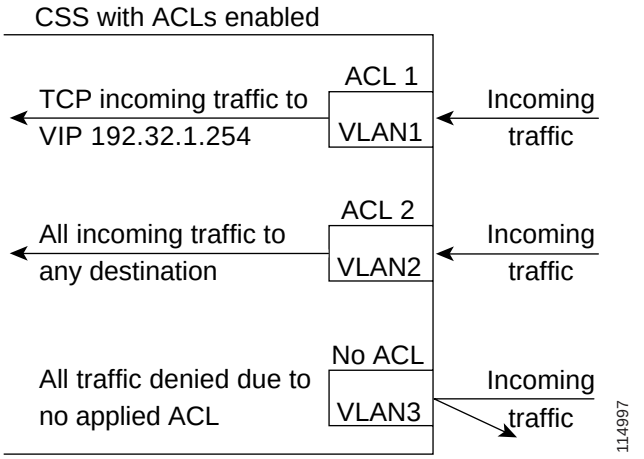
An ACL consists of clauses that you define. The CSS uses these clauses to determine how to handle each packet it processes on a VLAN circuit. When the CSS examines each packet, it either forwards or blocks the packet based on whether or not the packet matches a clause in the ACL. You must configure a permit clause in an ACL to allow traffic through the circuit. An implicit “deny all” clause exists at the end of every ACL.

When configuring ACLs on a CSS, you must apply an ACL to each VLAN circuit on the CSS to control traffic on the VLAN. An applied ACL on a circuit assigns the ACL and its clauses to the circuit.

After you apply an ACL to each CSS circuit, you must enable the ACLs on the CSS. Globally enabling ACLs affect *all* circuits in the CSS. When you enable ACLs, the CSS uses the clauses in all ACLs to permit or deny traffic on all circuits. If a circuit does not have an ACL, the CSS applies an implicit “deny all” clause to this circuit causing the CSS to deny all traffic on it.

For example, Figure 1-2 shows three VLAN circuits on the CSS.

Figure 1-2 ACLs Enabled on the CSS



For VLAN1, if you want to allow any TCP traffic to the destination VIP address 192.32.1.254, create ACL 1 and configure the following clause, *clause 15 permit tcp any destination 192.32.1.254*. Then apply ACL 1 to VLAN1.

For VLAN2, if you want to allow all traffic to any destination, create ACL 2 and configure the following clause, *clause 15 permit any any destination any.* Then apply ACL 2 to VLAN2.

When you enable ACLs on the CSS, VLAN1 and VLAN2 permit traffic as defined by the permit clauses configured for the ACL. Because no ACL is applied to VLAN3, the CSS applies an implicit “deny all” clause to this circuit causing the CSS to deny all traffic on it.

Caution ACLs function as a firewall security feature. It is extremely important that you first configure an ACL for each CSS circuit to permit traffic *before you enable ACLs*. If you do not permit any traffic, you lose network connectivity. Note that the console port is not affected.

Enabling ACLs globally affects all traffic on *all* CSS circuits whether they have ACLs or not. When you enable ACLs, all traffic on a circuit that is not configured in an ACL permit clause *is denied*. If you do not apply an ACL on each circuit, the CSS denies traffic on that circuit.

When the CSS is using ACLs, its hardware implements a maximum of 10 ACLs with simple Layer 3 or Layer 4 clauses. The CSS software implements more complicated ACLs with Layer 5 clauses.

Note ACLs are not supported on the CSS Ethernet Management port.

ACLs do not block ARP packets.

You cannot use an ACL clause with a source group to perform source address translation of traffic destined to an SSL module. This clause will be accepted by the CSS but will be ignored for flows terminated at the SSL module. You can apply NAT to connections towards servers after SSL processing.

If you are load-balancing passive FTP servers and you want to use an ACL to apply a source group, you must configure services directly in the source group. For details on using source groups to support FTP sessions, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

ACL Configuration Quick Start

Use the quick-start procedure in Table 1-1 to configure an ACL. Each step includes the CLI command required to complete the task. For a complete description of each feature, see the sections following this procedure.

Note You must configure an ACL with at least one permit clause for each CSS circuit. Otherwise, the CSS denies all traffic on the circuit.

Table 1-1 ACL Configuration Quick Start

Task and Command Example
1. Enter global configuration mode. `# config` `(config)#`
2. Create an ACL and access ACL mode. Enter an ACL index number from 1 to 99. `(config)# acl 7` `Create ACL <7>, [y,n]:y` `(config-acl[7])#`
3. Configure clauses in the ACL. The CSS will use the clauses to control traffic on the circuit on which you will apply the ACL (for example, VLAN1). Enter a clause number from 1 to 254 and define the clause parameters. The syntax for defining a clause is: **clause** *number* **permit**|**deny**|**bypass** *protocol* [*source_info* {*source_port*}] **dest** [*dest_info* {*dest_port*}] {**log**} {**prefer** *servicename*} {**sourcegroup** *name*} See Table 1-2 for information on the **clause** command options. For example, to block ports 20 to 23 for all user access coming into the CSS on a circuit from outside the network, enter: `(config-acl[7])# clause 10 deny any any destination range 20 23` To permit all other traffic through the CSS on a circuit, enter: `(config-acl[7])# clause 15 permit any any destination any`
4. Apply the ACL to a specific circuit. In this example, there is only one VLAN, the default VLAN1. For example, to apply acl 7 to circuit VLAN1, enter: `(config-acl[7])# apply circuit-(VLAN1)` You can also apply ACL 7 to all circuits on the CSS by using the **apply all** command.

Table 1-1 ACL Configuration Quick Start (continued)

Task and Command Example
5. You must repeat steps 1 through 4 to create an ACL with at least one permit clause for all other circuits and apply the ACL to them. If a circuit does not have an applied ACL when you enable ACLs on the CSS, the CSS denies traffic on the circuit.
6. Enable all ACLS on the CSS. Enter the global **acl enable** command for all ACLs to take effect on all CSS circuit. **Caution** Because enabling ACLs globally affects all traffic on all CSS circuits, only permit clauses in an ACL allows traffic through the circuit. If you do not apply an ACL to a circuit, the CSS applies an implicit "deny all" clause to this circuit causing the CSS to deny all traffic on it. For example, enter: `(config)# acl enable`

The following running-config example shows the result of entering the commands in Table 1-1.

```
!**************************** ACL ****************************
acl 7
    clause 10 deny any any destination range 20 23
    clause 15 permit any any destination any
  apply circuit-(VLAN1)

!************************** GLOBAL **************************
acl enable
```

Creating an ACL

ACLs contain clauses to control traffic on CSS circuits. Because all circuits are affected when you globally enable ACLs on the CSS, you must create an ACL for each circuit. You can apply an ACL to more than one circuit. You can also apply an ACL to all circuits on the CSS.

Note If a circuit does not have an ACL, the CSS applies an implicit "deny all" clause to this circuit causing the CSS to deny all traffic on it.

To create an ACL and access ACL mode, use the **acl** *index number* command. The index number defines the ACL and can range from 1 to 99. To display a list of existing ACLs, use the **acl ?** command.

```
(config)# acl 7
```

When you access this mode, the prompt changes to the ACL mode of the index number you created. For example:

```
(config-acl[7])#
```

After you create an ACL, you must add clauses to it. For more information, see the "Configuring Clauses" section.

Deleting an ACL

When you no longer need an ACL and its clauses on the CSS, you can delete the ACL. When you delete an ACL, all of its clauses are also deleted. To delete an ACL, use the **no acl** command. For example, to delete ACL 7, enter:

```
(config)# no acl 7
```

If you delete an ACL that is currently applied to a circuit and ACLs are enabled on the CSS, the ACL is removed from the circuit and the CSS denies traffic on the circuit. If you want to permit traffic on the circuit, globally disable the ACLs on the CSS, which permits all traffic on a circuit.

For example:

1. In global configuration mode, disable all ACLs on the CSS.

   ```
   (config)# acl disable
   ```

2. In ACL mode, remove the ACL from the circuit. For example, enter:

   ```
   (config-acl[7])# remove circuit-(VLAN1)
   ```

3. In global configuration mode, delete the ACL. For example, enter:

   ```
   (config)# no acl 7
   ```

4. Apply another ACL on the circuit. If you do not apply an ACL on the circuit, the CSS denies traffic on the circuit when you enable ACLs on the CSS.
5. Reenable all ACLs on the CSS. Enter:

```
(config)# acl enable
```

Configuring Clauses

The clauses you configure on an ACL determine how the CSS controls traffic on a circuit. When you configure a clause, you must assign a number to it. The number assigned to each clause is important. The CSS processes the ACL starting from clause 1 and sequentially progresses through the rest of the clauses. When assigning numbers to clauses, assign the lowest numbers to clauses with the most specific matches. Then, assign higher numbers to clauses with less specific matches.

You do not need to enter the clauses sequentially. The CSS automatically inserts the clause in the appropriate order in the ACL. For example, if you enter clauses 10 and 24, and then clause 15, the CSS inserts the clauses in the correct sequence.

To create a clause to permit, deny, or bypass traffic on a circuit, use the **clause** command. The clause *number* is the number you want to assign to the clause. Enter a number from 1 to 254.

Note Once you add a new clause to an ACL when ACLs are enabled on the CSS, you must reapply the ACL on the circuit. For more information, see the "Adding a Clause When ACLs are Globally Enabled" section.

When you create a clause, you cannot modify it. You must delete the clause and create a new clause. For information on deleting a clause, see the "Deleting a Clause" section.

The CSS applies a hidden default "deny all" clause as clause 255 to all ACLs. You must specify permit clauses that allow traffic including management traffic on the CSS.

The syntax for the **clause** command is:

- **clause** *number* **bypass** - Creates a clause in the ACL to *permit* traffic on a circuit and bypasses (does not process) content rules that apply to the traffic. The syntax for **clause bypass** is:

  clause *number* **bypass** *protocol* [*source_info* {*source_port*}] **dest** [*dest_info* {*dest_port*}] {**sourcegroup** *name*} {**prefer** *servicename*}

Note The **bypass** option bypasses traffic *only* on a content rule, and, therefore, does not cause Network Address Translating (NATing) to occur. Do not use the **bypass** option in an ACL clause with a source group. The **bypass** option does not affect NATing on a source group.

- **clause** *number* **deny** - Creates a clause in the ACL to deny traffic on a circuit. The syntax for **clause deny** is:

  clause *number* **deny** *protocol* [*source_info* {*source_port*}] **dest** [*dest_info* {*dest_port*}] {**sourcegroup** *name*} {**prefer** *servicename*}

- **clause** *number* **permit** - Creates a clause in the ACL to permit traffic on a circuit. When you configure an ACL permit clause, all traffic not specified in a permit clause is denied by default. The syntax for **clause permit** is:

  clause *number* **permit** *protocol* [*source_info* {*source_port*}] **dest** [*dest_info* {*dest_port*}] {**sourcegroup** *name*} {**prefer** *servicename*}

Note When a destination in an ACL clause is a Layer 5 content rule, the CSS does not spoof the connection. Therefore, the ACL clause does not function as would be expected. As a workaround, you may configure an additional clause to permit the TCP/IP addresses and ports. Be aware that content is matched on both clauses. For example,
clause 14 permit any any destination content Layer5/L5 eq 80 (original clause)
clause 15 permit tcp any destination 200.200.200.200 eq 80 (This is an additional clause to handle the SYN, where the destination IP address is the IP address configured in the Layer 5 content rule. Note that this clause number must be greater than the destination content clause number.)

Table 1-2 provides variables and options for the **clause** command. Bolded syntax defines keywords that you enter on the command line. Italics define variables where you enter a value such as an IP address or a host name.

Table 1-2 Clause Command Options

Variables and Options	Parameters
number	The number you want to assign to the clause. Enter a number from 1 to 254.
action	The action to apply to the clause. Enter one of the following: **bypass**, **deny**, **permit**
protocol	The protocol for the traffic type. Enter one of the following: **any**, **icmp**, **igp**, **igmp**, **ospf**, **tcp**, **udp**
source_info	The source of the traffic. Enter one of the following: • *ip_address* (optionally include *subnet mask* in IP address format only) for the source IP address and optional mask IP address. • *hostname* for the source host name. Enter a host name in mnemonic host-name format. Configure the CSS DNS client first to enable the CSS to translate the host name. • **any** for any combination of source IP address and host name information. • **nql** *nql_name* for an existing Network Qualifier List (NQL) consisting of a list of IP addresses.

Table 1-2 Clause Command Options (continued)

Variables and Options	Parameters
source_port	The source port for the traffic. If you do not designate a source port, this clause allows traffic from any port number. Enter one of the following: • **eq** *port* is equal to the port number. • **lt** *port* is less than the port number. • **gt** *port is* greater than the port number. • **neq** *port* is not equal to the port number. • **range** *low high* for a range of port numbers, inclusive. Enter numbers from a range of 1 to 65535. Separate the *low* and *high* number with a space.
destination_info	The destination information for the traffic. Enter one of the following: • **destination any** for any combination of destination information. • **destination content** *owner_name/rule_name* for an owner content rule. Separate the owner and rule name with a / character. • **destination** *ip_address* (for the destination IP address and optional subnet mask IP address. Include *subnet mask* as IP address only; no Classless Inter-domain routing (CIDR) address. • **destination** *hostname* for the destination host name. To use a *hostname*, configure the CSS DNS client first to enable the CSS to translate the host name. • **nql** *nql_name* for an existing NQL consisting of host IP addresses. Enter the name of the NQL.

Table 1-2 Clause Command Options (continued)

Variables and Options	Parameters
destination_port	The destination port. Enter one of the following. You may use a port number or port name with the options. • **eq** *port* is equal to the port number. • **lt** *port* is less than the port number. • **gt** *port* is greater than the port number. • **neq** *port* is not equal to the port number. • **range** *low high* for a range of port numbers, inclusive. Enter numbers from a range of 1 to 65535. Separate the *low* and *high* number with a space. • *port names*: – **https** = Port 443 Https – **ldap** = Port 389 Ldap – **bgp** = Port 179 Bgp – **ntp** = Port 123 Ntp – **nntp** = Port 119 Nntp – **pop** = Port 110 Pop – **http** = Port 80 Http, – **gopher** = Port 70 Gopher – **domain** = Port 53 Domain – **smtp** = Port 25 Smtp – **telnet** = Port 23 Telnet, – **ftp** = Port 21 Ftp – **ftp-data** = Port 20 Ftp-data – **none** = None If you do not define a destination port, this clause allows traffic to any port.

Table 1-2 Clause Command Options (continued)

Variables and Options	Parameters
sourcegroup *name*	The source group as the destination for the traffic. Enter the group name. To see a list of source groups, enter: `show group ?` **Note** The **clause** *number* **bypass** command does not affect NATing on a source group. You cannot use an ACL clause with a source group to perform source address translation of traffic destined to an SSL module. This clause will be accepted by the CSS but will be ignored for flows terminated at the SSL module. You can apply NAT to connections towards servers after SSL processing.

Table 1-2 Clause Command Options (continued)

Variables and Options	Parameters
prefer *service_name*	Prefer the specified service as the traffic destination over other services. To define more than one preferred service, separate each service with a comma (,). You can define a maximum of two services. You cannot configure services learned through an Application Peering Protocol (APP) session as preferred services. A remote service learned through APP is of the form ap-redirect@192.168.138.118 and can been seen on the **show service summary** screen. When configuring an ACL clause, you cannot use this service as a preferred service. If you save this clause in the startup-config and reboot the CSS, a startup error occurs because this service has not been learned through APP at this point. For example: `clause 10 permit any any destination any prefer ap-redirect@192.168.138.118` **Note** ACLs configured with a preferred service take precedence over stickiness. If you specify both a source group and a preferred service in a clause, you must specify the source group before you specify the preferred service within the clause.

After you create clauses for an ACL, you can apply the ACL to a circuit. For more information, see the "Applying an ACL to a Circuit or DNS Queries" section.

Adding a Clause When ACLs are Globally Enabled

If you are adding a new clause to an applied ACL when ACLs are globally enabled on the CSS, you must reapply the ACL to the circuit using the **apply circuit** command for the clause to take effect.

For example, you apply ACL 7 to VLAN1 and then globally enable ACLs on the CSS. At a later time, to add a new clause to ACL 7 and to have the clause take effect on the CSS, enter:

```
(config-acl[7])# clause 200 permit any any destination any
(config-acl[7])# apply circuit-(VLAN1)
```

Deleting a Clause

If you modify an existing clause, you must delete it from the ACL and then readd it. To delete a clause, use the **no clause** command. For example, to delete clause 6, enter:

```
(config-acl[7]) no clause 6
```

When ACLs are applied to a circuit and enabled on a CSS, the CSS considers them in use. You cannot delete a clause from an ACL in use. To delete the clause, remove its applied ACL from the circuit, delete a clause, and then reapply the ACL to the circuit.

For example, to delete clause 6 from ACL 7 on circuit VLAN1:

1. In ACL mode, remove ACL 7 from the circuit VLAN1. Enter:

   ```
   (config-acl[7]) remove circuit-(VLAN1)
   ```

2. Delete clause 6. Enter:

   ```
   (config-acl[7]) no clause 6
   ```

3. Reapply ACL 7 to circuit VLAN1. Enter:

   ```
   (config-acl[7]) apply circuit-(VLAN1)
   ```

Note When you remove an applied ACL from the circuit, the CSS applies an implicit "deny all" clause to this circuit causing the CSS to deny all traffic on it. If you want the CSS to permit traffic on the circuit when removing the applied ACL from the circuit, globally disable ACLs on the CSS with the global configuration mode **acl disable** command. By disabling all ACLs on the CSS, the CSS permits all traffic on all circuits.

Applying an ACL to a Circuit or DNS Queries

After you configure the clauses on an ACL, use the **apply** command to assign an ACL to all circuits, an individual circuit, or to DNS queries.

Note When you add a new clause to an applied ACL, use the **apply circuit** command to reapply the ACL on the circuit for the clause to take effect.

You cannot apply an empty ACL to a circuit. If you attempt to do so, this error message appears: `Cannot apply ACL for it has no clauses`.

The syntax and options for this ACL mode command are:

- **apply all** - Applies the ACL to all existing circuits. For example:

  ```
  (config-acl[7])# apply all
  ```

- **apply circuit - (***circuit_name***)** - Applies the ACL to an individual circuit. For example, to apply acl 7 to circuit VLAN1:

  ```
  (config-acl[7])# apply circuit-(VLAN1)
  ```

  To display a list of circuits, use the **apply ?** command.

- **apply dns** - Adds the ACL to DNS queries.

  ```
  (config-acl[7])# apply dns
  ```

  If you configure a domain name on a content rule on a CSS using the **add dns** *domain_ name* command, a DNS query for that domain name *does* match an ACL that is configured with the **apply dns** command.

However, if you configure a CSS with the **dns-server** command, and the CSS receives a DNS query for a domain name that you configured on the CSS using the **host** command, the DNS query *does not* match an ACL that is configured with the **apply dns** command.

After you apply an ACL and ACLs are disabled on the CSS, you must enter the global configuration **acl enable** command to enable the ACLs on the CSS. For information on the **acl enable** command, see the "Enabling ACLs on the CSS" section later in this chapter.

Removing an ACL from Circuits or DNS Queries

Remove an ACL from the circuit when you need to delete a clause from an ACL, the ACL applied to the circuit, or an ACL from DNS queries. To remove an ACL from all circuits, an individual circuit, or DNS queries, use the **remove** command. The syntax and options for this ACL mode command are:

- **remove all** - Removes the ACL from all circuits.

  ```
  (config-acl[7])# remove all
  ```

- **remove circuit (***circuit_name***)** - Removes the ACL from a specific circuit. For example, enter:

  ```
  (config-acl[7])# remove circuit-(VLAN1)
  ```

  To display a list of circuits that you can remove, use the **remove ?** command.

- **remove dns** - Removes the ACL from DNS queries. For example, enter:

  ```
  (config-acl[7])# remove dns
  ```

We recommend that you globally disable ACLs on the CSS before removing an ACL from a circuit. If you remove an ACL from a circuit when ACLs are enabled on the CSS, the CSS applies an implicit "deny all" clause to this circuit causing the CSS to deny all traffic on it. If you do not want to deny traffic on the circuit, you must disable all ACLs on the CSS and then remove ACL from the circuit. By disabling all ACLs on the CSS, the CSS permits all traffic on all circuits.

For example:

1. In global configuration mode, disable all ACLs on the CSS.

   ```
   (config)# acl disable
   ```

2. In ACL mode, remove the ACL from the circuit.

```
(config-acl[7])# remove circuit-(VLAN1)
```

3. Make any changes to the ACL.

   If you delete an ACL from the circuit, configure another ACL with a permit clause for the circuit, and then apply it to the circuit. Otherwise, when you reenable the ACLs on the CSS, the CSS denies traffic on the circuit.

4. Reapply the ACL on the circuit.

```
(config-acl[7])# apply circuit-(VLAN1)
```

5. In global configuration mode, reenable all ACLs on the CSS.

```
(config)# acl enable
```

Enabling ACLs on the CSS

After you configure ACLs and their clauses, and apply an ACL to each CSS circuit, you can globally enable all ACLs for use on the CSS. When you globally enable all ACLs, the CSS affects all traffic on all circuits and only allows traffic on circuits with ACLs containing a permit clause.

Caution It is extremely important that you first configure an ACL for each CSS circuit to permit traffic *before you enable ACLs*. Enabling ACLS affects all circuits. If you do not permit traffic, you lose network connectivity. When you enable ACLs, all traffic on a circuit that is not configured in an ACL permit clause *is denied*. The CSS applies an implicit "deny all" clause to any circuit that does not have an ACL applied to it.

For example, you configure three circuits on the CSS (VLAN1, VLAN2, and VLAN3). Then you configure an ACL for VLAN1 only. When you globally enable ACLs, VLAN1 passes traffic based on the ACL. However, VLAN2 and VLAN3 discard all packets because of the implicit "deny all" clause that the CSS applies to the circuits because they do not have an ACL.

Before you globally enable ACLs on the CSS, make sure that you have console access. The console port is not affected if you lose network connectivity because of an ACL configuration problem.

Use the global configuration **acl enable** command to enable all ACLs on the CSS. To globally enable all ACLs, enter:

```
(config)# acl enable
```

Disabling ACLs on the CSS

If you need to add, change, or delete an ACL or delete an ACL clause, we recommend that you disable all ACLs on the CSS before removing the ACL from the circuit. If you remove an ACL before globally disabling ACLs, the CSS applies an implicit "deny all" clause to the circuit from which the ACL is removed and denies traffic on the circuit.

Note Globally disabling ACLs on the CSS disables all ACLs on the CSS and permits all traffic on all CSS circuits.

To globally disable all ACLs on the CSS, enter:

```
(config)# acl disable
```

Showing ACLs

Use the **show acl** commands to display access control lists and clauses. The **show acl** commands are available in all modes.

When you show an ACL clause that is applied to a circuit, the display includes:

- **Content Hits** - A flow can be defined as a stream of UDP and TCP packets between a client and a server. The CSS must receive a number of packets from the client and the server before it can completely set up a flow. All of these packets, received before the flow is completely set up, are subject to ACL checks and can cause increments to the ACL Content Hits counter.
- **Router Hits** - All non-UDP and non-TCP packets subjected to ACL checks cause increments to the ACL Router Hits counter. All UDP and TCP traffic terminating on the CSS (for example, a Telnet or FTP session) cause increments to the ACL Router Hits counter.

- **DNS Hits** - Packets that match an ACL clause for DNS flows when an ACL clause is applied to DNS queries. The display includes a DNS hit counter, which counts DNS lookups.

The total number of ACL hits for each packet received by the CSS can vary depending on the type of flow and whether an ACL match occurred. The CSS performs an ACL check for every packet received until the ACL flow is completely set up. Once the ACL flow is set up, remaining packets received by the CSS that are associated with the flow are not subject to an ACL match and the ACL hit counters do not increment.

The syntax is:

- **show acl** - Displays all ACLs and their clauses.
- **show acl** *index* - Displays the clauses for the specified ACL index number (valid numbers are 1 to 99).
- **show acl config** - Displays the ACL global configuration. This command also shows you which ACLs are applied to which circuits.

For example, enter:

```
(config)# show acl 2
```

Table 1-3 describes the fields in the **show acl** command output.

Table 1-3 Field Descriptions for the show acl Command Output

Field	Description
Acl	The number assigned to the ACL (a number from 1 to 99)
Clause	The number assigned to the clause (a number from 1 to 254)
Action	The method with which incoming traffic is controlled by the clause (permit, deny, or bypass) and the protocol for the type of traffic
Source	The configured source of the traffic
Destination	The configured destination for the traffic
Log	Indicates whether ACL logging is enabled or disabled on the specified clause
Content Hits	Increments for a packet received by the CSS before flow setup

Table 1-3 Field Descriptions for the show acl Command Output (continued)

Field	Description
Router Hits	Increments for a packet directly forwarded to the CSS through a Telnet or FTP session or from a non-TCP or UDP packet
DNS Hits	Increments for a packet that matches an ACL clause for DNS flows

Setting the Show ACL Counters to Zero

Use the **zero counts** command to reset the content and DNS hit counters in the **show acl** command screen to zero for a specific ACL. You must be in an ACL to use this command. The CSS clears counters only for that ACL.

The syntax and options for this command are:

```
(config-acl[7])# zero counts
```

Logging ACL Activity

When you configure the CSS to log ACL activity, it logs the event of the packet matching the clause and ACL. The CSS sends log information to the location you specified in the **logging** command. For information on the **logging** command, refer to the *Cisco Content Services Switch Administration Guide*.

Note We do not recommend logging of an ACL or its clauses. If you enable ACL or clause logging, it may degrade the performance of the CSS.

Before you configure logging for a specific ACL clause, ensure that global ACL logging is enabled. To globally enable ACL logging, use the global configuration mode **logging subsystem acl level debug-7** command.

Because the CSS does not save the **clause log enable** command in the running-config, you must reenable logging if the CSS reboots.

To enable logging on an existing ACL clause, use the **log enable** option for the **clause** command and enter:

```
(config-acl[7])# clause 1 log enable
```

If ACLs are globally enabled on the CSS, configure logging on an existing ACL clause:

1. In global configuration mode, disable all ACLs on the CSS.

   ```
   (config)# acl disable
   ```

2. Enter the ACL mode for which you want to enable logging.

   ```
   (config)# acl 7
   (config-acl[7])#
   ```

3. Remove the ACL from the circuit.

   ```
   (config-acl[7]) remove circuit-(VLAN1)
   ```

4. Enable logging for the existing clause.

   ```
   (config-acl[7])# clause 1 log enable
   ```

5. Reapply the ACL to the circuit.

   ```
   (config-acl[7])# apply circuit-(VLAN1)
   ```

6. In global configuration mode, reenable all ACLs on the CSS.

   ```
   (config)# acl enable
   ```

To disable ACL logging for a specific clause, enter:

1. In global configuration mode, disable all ACLs on the CSS.

   ```
   (config)# acl disable
   ```

2. Enter the ACL mode for which you want to disable logging.

   ```
   (config)# acl 7
   (config-acl[7])#
   ```

3. Remove the ACL from the circuit.

   ```
   (config-acl[7]) remove circuit-(VLAN1)
   ```

4. Disable logging for the existing clause.

   ```
   (config-acl[7])# clause 1 log disable
   ```

5. Reapply the ACL to the circuit.

```
(config-acl[7])# apply circuit-(VLAN1)
```

6. In global configuration mode, reenable all ACLs on the CSS.

```
(config)# acl enable
```

To globally disable logging for all ACL clauses, enter:

```
(config)# no logging subsystem acl
```

ACL Example

The following ACL provides security for a CSS, Server1, and Server2 on one VLAN (VLAN1). The ACL:

- Permits clients from subnet 172.16.107.x to access servers 1 and 2 on VLAN1 using various applications (for example, Telnet, FTP, TFTP)
- Permits clients from subnet 172.16.107.x to launch a browser with the URL 172.16.107.35 (the VIP address)
- Prevents clients on any subnet other than 172.16.107.x from accessing VLAN1 and servers 1 and 2

The individual clauses provide the following security.

- Clause 20 permits any protocol from source subnet 172.16.107.0 to Server1 (IP address 172.16.107.15).
- Clause 30 permits any protocol from source subnet 172.16.107.0 to Server2 (IP address 172.16.107.16).
- Clause 40 permits any protocol from source subnet 172.16.107.0 to VIP address 172.16.107.35 port 80 (HTTP).
- Clause 50 permits bidirectional communication to the VLAN for any Internet Control Message Protocol (ICMP) traffic, including keepalives. If you are using service keepalives, you must configure a clause to permit keepalive traffic.
- Clause 60 permits UDP to port 520 on the VLAN for Routing Information Protocol (RIP) updates. This clause is required if your router is on a subnet other than 172.16.107.x.
- Clause 70 denies everything that has not been permitted in the ACL.

```
!************************** ACL ***************************
acl 1
clause 20 permit any 172.16.107.0 255.255.255.0 destination
172.16.107.15
clause 30 permit any 172.16.107.0 255.255.255.0 destination
172.16.107.16
clause 40 permit any 172.16.107.0 255.255.255.0 destination
172.16.107.35 eq 80
clause 50 permit ICMP any destination any
clause 60 permit udp any destination any eq 520
clause 70 deny any any destination any
apply circuit-(VLAN1)
```

Configuring Network Qualifier Lists for ACLs

NQL configuration mode allows you to configure a network qualifier list (NQL). An NQL is a list of networks or specific services, identified by IP address and subnet mask, that you assign to an ACL clause as a source or destination. By grouping networks into an NQL and assigning the NQL to an ACL clause, you have to create only one clause instead of a separate clause for each network.

The CSS enables you to configure a maximum of 512:

- Networks or services per NQL
- NQLs per CSS

This functionality is useful, for example, in a caching environment in which you have a network you want to bypass and send content requests directly to the origin servers (servers containing the content). You can also use an NQL for users who prefer a service based on a specific network.

To access NQL configuration mode, use the **nql** command. The prompt changes to (config-nql [name]). You can also use this command from NQL mode to access another NQL.

See the following sections to configure an NQL:

- Creating an NQL
- Describing an NQL
- Adding Networks to an NQL
- Adding an NQL to an ACL Clause
- Showing NQL Configurations

Creating an NQL

Enter the name of the new NQL you want to create or an existing NQL. Enter the name as an unquoted text string with no spaces and a maximum of 31 characters. You can create a maximum of 512 NQLs per CSS.

For example, enter:

```
(config)# nql bypass_nql
(config-nql[bypass_nql])#
```

To display a list of existing NQLs, use the **nql ?** command. If no NQLs currently exist, the CSS prompts you to enter a new name.

To remove an existing NQL, use the **no nql** command. For example, enter:

```
(config)# no nql bypass_nql
```

Describing an NQL

To provide a description for an NQL, use the **description** command in NQL mode. Enter the NQL description as a quoted text string with a maximum length of
63 characters.

For example, enter:

```
(config-nql[bypass_nql])# description "Bypass services"
```

Adding Networks to an NQL

To add a maximum of 512 networks or services to an NQL, use the **ip address** command. Enter an IP address with either a subnet prefix or a subnet mask. You may also add an optional description for the IP address and turn on logging.

The syntax and options are:

ip address *ip_address*[*/subnet_prefix*| *subnet_mask*] {**"***description***"**}{**log**}

The variables and options are:

- *ip_address* - The destination network address. Enter the IP address in dotted-decimal notation (for example, 192.168.0.0).
- *subnet_prefix*|*subnet_mask* - The IP subnet mask prefix length in CIDR bitcount notation (for example, /16). The valid prefix length range is 8 to 32. Do not enter a space to separate the IP address from the prefix length.
- *subnet_address* - The IP subnet mask in dotted-decimal notation (for example, 255.255.0.0).
- **"***description***"** - A description of the IP address. Enter a quoted text string with a maximum of 63 characters.
- **log** - Logs an event involving an NQL. If you do not enter this option, events are not logged. To log an NQL event, you must enable global NQL logging. To enable global NQL logging, use the **(config) logging subsystem nql level debug-7** command. For logging information, refer to the *Cisco Content Services Switch Administration Guide*.

For example, to add two networks to the NQL bypass_nql, enter:

```
(config-nql[bypass_nql])# ip address 192.168.0.0/16 "Network of
dynamic mail content" log
(config-nql[bypass_nql])# ip address 123.123.123.0/24
```

To log events occurring on a network, you must also enable global NQL logging. For example, enter:

```
(config)# logging subsystem nql level debug-7
```

Note If you do not include a description or turn on logging when you create the entry and later wish to add a description or turn on logging, you must first remove the entry and then add it again with the desired options.

To remove an IP address from an NQL, use the **no ip address** command. For example, enter:

```
(config-nql[bypass_nql])# no ip address 192.168.0.0/16
```

Adding an NQL to an ACL Clause

To add an NQL to an ACL clause:

1. Create the ACL. For example, enter:

```
(config)# acl 10
```

2. Define the clause, including the NQL as either a source or destination.

   This clause example bypasses content rules for any traffic from any source going to the destination networks defined in NQL bypass_nql on port 80.

```
(config-acl[10])# clause 1 bypass any any destination nql
bypass_nql eq 80
```

Showing NQL Configurations

Use the **show nql** command to display NQL configuration information. The syntax for this command is:

- **show nql** - Displays information for all NQLs. If you enter this command in NQL mode, the CSS displays the addresses only for the current NQL.
- **show nql** *nql_name* - Displays information for the specified NQL. Enter the NQL name as a case-sensitive unquoted text string with no spaces. To see a list of existing NQL names, use the **show nql ?** command.

For example, enter:

```
(config-nql[bypass_nql])# show nql
```

Table 1-4 describes the fields in the **show nql** command output.

Table 1-4 Field Descriptions for the show nql Command Output

Field	Description
Name	The name of the NQL.
Description	The description associated with the NQL.
IP Addresses	The IP addresses and subnet mask supported by the NQL. If configured, a description appears after the address.