



Configuring Source Groups, ACLs, EQLs, URQLs, NQLs, and DQLs

This chapter describes how to configure source groups, access control lists (ACLs), extension qualifier lists (EQLs), Uniform Resource Locator qualifier lists (URQLs), network qualifier lists (NQLs), and domain qualifier lists (DQLs). Information in this chapter applies to all CSS models, except where noted.

This chapter contains the following major sections:

- [Configuring Source Groups](#)
- [Controlling CSS Network Traffic Through Access Control Lists](#)
- [Configuring Extension Qualifier Lists](#)
- [Configuring URL Qualifier Lists](#)
- [Configuring Network Qualifier Lists](#)
- [Configuring Domain Qualifier Lists](#)

Configuring Source Groups

Group configuration mode allows you to configure a maximum of 255 source groups on a CSS. A source group is a collection of local servers that initiate flows from within the local web farm. The CSS enables you to treat a source group as a virtual server with its own source IP address.

For example, if you configure several streaming audio transmitters as a group, the CSS will process flows from the group members and give them all the same source IP address.

The following sections describe how to configure a source group:

- [Source Group Configuration Quick Start](#)
- [Creating a Source Group](#)
- [Configuring a Source Group for FTP Connections](#)
- [Configuring Source Groups to Allow Servers to Resolve Domain Names Using the Internet](#)
- [Showing Source Groups](#)

Source Group Configuration Quick Start

Use the procedure in [Table 5-1](#) to configure a source group for TCP/UDP traffic. To configure a source group for FTP traffic, see the next section. Note that each source group requires a content rule that contains the same services and VIP as the source group.

Table 5-1 Source Group Configuration Quick Start

Task and Command Example

1. Create the source group. Source group names can have a maximum of 16 characters. The following example creates a source group `ftpgroup`.

```
(config)# group ftpgroup
```

The CLI transitions into `config-group` mode where you can activate the source group and configure attributes for it.

```
(config-group[ftpgroup])#
```

Table 5-1 Source Group Configuration Quick Start (continued)

Task and Command Example

2. Configure the source group VIP address to which all service IP addresses will be translated. You can assign the same VIP address to multiple source groups, but only one of the source groups can be active at a time. For example, enter:

```
(config-group[ftpgroup])# vip address 172.16.36.58
```

3. Add previously defined services to the source group. For example, enter:

```
(config-group[ftpgroup])# add service server1  
(config-group[ftpgroup])# add service server2
```

4. Activate the source group. Because a VIP address can belong to only one active source group at a time, the CSS will not allow you to activate a second source group that contains the same VIP address as the one in the active source group.

```
(config-group[ftpgroup])# active
```

To remove service server1 from the source group, enter:

```
(config-group[ftpgroup])# remove service server1
```

5. Create a content rule, add the same services and VIP that are configured in the source group, and activate the content rule. The content rule enables the CSS to match requests for the content rule VIP. When either server1 or server2 replies to the request, the CSS NATs the server IP addresses to the source group VIP.

For example, enter:

```
(config-owner[arrowpoint.com])# content ftpsource1
```

```
(config-owner-content[arrowpoint.com-ftpource1])# add service  
server1
```

```
(config-owner-content[arrowpoint.com-ftpource1])# add service  
server2
```

```
(config-owner-content[arrowpoint.com-ftpource1])# vip address  
172.16.36.58
```

```
(config-owner-content[arrowpoint.com-ftpource1])# activate
```

Creating a Source Group

To access group configuration mode, use the **group** command from any mode except ACL and boot configuration modes. The syntax for this command is:

```
group groupname
```

Enter an existing or a new source group name from 1 to 31 characters.

For example, enter:

```
(config)# group ftpgroup  
(config-group[ftpgroup])#
```

To view a list of existing source groups, enter:

```
(config)# group ?
```



Note

You can also use the **group** command from within group mode to access or create another source group.

To remove a source group, enter:

```
(config)# no group ftpgroup
```



Note

To make certain modifications to an active source group, you must first suspend the source group using the **suspend** command. Such modifications include: changing the IP address to 0 or using the **no ip address** command, adding or removing a service or destination service, or using the **portmap** command.

Source Group Commands

Use the following commands in group mode:

- **active** - Activates a source group.
- **add destination service** *service_name* - Adds a destination service to a source group. You can configure a maximum of 64 destination services per source group. You cannot use a service with the same name in other source groups or the source service list within the same source group. You can use services with duplicate addresses among destination services because the actual service is chosen through content rule selection. The destination service must be active and must be added to a content rule in order for it to perform destination source address NATing for the source group (see [Chapter 3, Configuring Content Rules](#)).



Note

Adding a destination service to a source group does not allow the destination service flows to be NATed by the source group, when the service initiates the flows. This is because the destination service applies group membership based on rule and service match. To ensure that service initiated connections are NATed, you must also configure ACL match criteria or additional service names with duplicate addresses, then add those services to a source group. The source group used could be the current source group with the destination service or any other configured source group.

If your topology consists of a CSS 11800 using ECMP to the servers and server port NAT configured on the services, to ensure the correct processing of packets either:

- Enable Service Remapping with the **persistence reset remap** command.
- Create source groups for the services in the content rule with the **add destination service** command.
- **add service** - Adds a service to a source group. You can configure a maximum of 64 services per source group. A service may belong to only one group at a time. When the source group is active and the same service is hit through a content rule, ACL preferred service, or sorry service, the source group is used to NAT (Network Address Translation) the source address. The service must be active in order for it to perform source address NATing for the source group (see [Chapter 1, Configuring Services](#)).

Be aware that you cannot use a service with:

- The same name in other source groups or the destination service list within the same source group
- The same address as a source service on another source group
- **vip address** - Specifies the source Virtual IP address (VIP) for the group. The CSS substitutes this IP address for the source address in flows originating from one of the group's sources. You can assign the same VIP address to multiple source groups, but only one of the source groups can be active at a time.
- **remove service** - Removes a previously configured service from a source group.
- **portmap** - Defines the source port translation of flows from the services configured in a source group. By default, portmapping is enabled for source groups on source ports greater than 1023. The CSS translates such source ports to a range starting at 8192. Use the following portmap options to change the default portmapping behavior of the CSS. The syntax and options for this group mode command are:
 - **portmap base-port** *base_number* - Defines the base port (starting port number) for the CSS. Enter a base number from 2016 to 63456. The default is 2016.

To reset the starting port number to its default value of 2016, use the **no portmap base-port** command.
 - **portmap number-of-ports** *number* - Defines the number of ports in the portmap range for each Switch Fabric Processor (SFP) in a CSS.

Enter a number from 1 to 57216. The default for a CSS 11800 is 14304; otherwise, the default is 57216. If the value entered is not a multiple of 32, it will be rounded up to the next possible multiple of 32.

To reset the number of ports to the default value, use the **no portmap number-of-ports** command.
 - **portmap disable** - Instructs the CSS to perform Network Address Translation (NAT) only on the source IP addresses and not on the source ports of UDP traffic hitting a particular source group. Use this option for Wireless Application Protocol (WAP) or other applications where you need to preserve the registered UDP port number for return traffic.



Note This command does not affect TCP flows.

The CSS maintains but ignores any **base-port** or **number-of ports** (see the previous options) values configured in the source group. If you later reenables port mapping for that source group, any configured **base-port** or **number-of ports** values will take effect. The default behavior for a configured source group is to NAT both the source IP address and the source port for port numbers greater than 1023.

To restore the default CSS behavior of NATing source IP addresses *and* source ports for a configured source group, use the **portmap enable** command.

- **suspend** - Suspends a source group. The group and its attributes remain the same but no longer have an effect on flow creation.

Configuring a Source Group for FTP Connections

To use source groups to support FTP sessions to a VIP that is load balanced across multiple services, configure a content rule for the VIP and then a source group.



Note

When you use an FTP content rule with a configured VIP address range, be sure to configure the corresponding source group with the same VIP address range (see [Chapter 3, Configuring Content Rules](#)).

To configure FTP sessions to a VIP:

1. Configure a content rule as required using the VIP that will be load balanced across multiple servers. The following example shows the portion of a running-config for content rule `ftp_rule`. Ensure that you use the **application ftp-control** command to define the application type.

```
content ftp_rule
  vip address 192.168.3.6
  protocol tcp
  port 21
  application ftp-control
  add service serv1
  add service serv2
  add service serv3
  active
```

2. Configure a source group defining the same VIP and services as configured in the content rule.



Note If you are load-balancing passive FTP servers, you must configure services directly in the associated source groups as shown in the following example.

The following running-config example shows source group ftp_group.

```
group ftp_group
  vip address 192.168.3.6
  add service serv1
  add service serv2
  add service serv3
  active
```

Configuring Source Groups to Allow Servers to Resolve Domain Names Using the Internet

The CSS provides support to enable servers to resolve domain names using the Internet. If you are using private IP addresses for your servers and wish to have the servers resolve domain names using domain name servers that are located on the Internet, you must configure a content rule and source group. The content rule and source group are required to specify a public Internet-routable IP address (VIP address) for the servers to allow them to resolve domain names.

To configure a server to resolve domain names:

1. If you have not already done so, configure the server.

The following example creates Server1 and configures it with a private IP address 10.0.3.251 and activates it.

```
(config)# service Server1
(config-service[Server1])# ip address 10.0.3.251
(config-service[Server1])# active
```

2. Create a content rule to process DNS replies. The content rule to process DNS replies is in addition to the content rules you created to process Web traffic. The content rule example below enables the CSS to NAT inbound DNS replies from the public VIP address (192.200.200.200) to the server's private IP address (10.0.3.251).

The following example creates content rule `dns1` with a public VIP `192.200.200.200` and adds server `Server1`.

```
(config-owner[arrowpoint.com])# content dns1
(config-owner-content[arrowpoint.com-dns1])# vip address
192.200.200.200
(config-owner-content[arrowpoint.com-dns1])# add service Server1
(config-owner-content[arrowpoint.com-dns1])# active
```

3. Create a source group to process DNS requests. The source group enables the CSS to NAT outbound traffic source IP addresses from the server's private IP address (`10.0.3.251`) to the public VIP address (`192.200.200.200`).

To prevent server source port collisions, the CSS NATs the server's source IP address and port by translating the:

- Source IP address to the IP address defined in the source group.
- Port to the port selected by the source group. The source group assigns each server a unique port for a DNS query so that the CSS can match the DNS reply with the assigned port. This port mapping enables the CSS to direct the DNS reply to the correct server.

The following example creates source group `dns1` with public VIP address `192.200.200.200` and adds the service `Server1`.

```
(config)# group dns1
(config-group[dns1])# vip address 192.200.200.200
(config-group[dns1])# add service Server1
(config-group[dns1])# active
```

Showing Source Groups

To display source group configuration information, use the **show group** commands in SuperUser, User, Global Configuration, and Group modes. The options are:

- **show group** - Display all source group configurations
- **show group group_name** - Display the source group configuration specified by *group_name*
- **show group group_name portmap** - Display the starting port number and number of ports configured on each SFP in a CSS

For example, enter:

```
(config)# show group
```

Table 5-2 describes the fields in the **show group** command output.

Table 5-2 *Field Descriptions for the show group Command Output*

| Field | Description |
|-----------------------------|---|
| Group | The name of the group, whether the group is activated (Active) or suspended (Suspend), and the source IP address for the group. |
| Associated ACLs | Any ACLs associated with the group. |
| Source/Destination Services | The source or destination services of the source group. |
| Name | The name of the service. |
| Hits | The number of content accessed (hit) on the service. This field is incremented for traffic from a group server going out from the source group. Traffic coming into the group does not increment the counter. |
| State | The state of the service. The possible states are Alive, Dying, or Dead. |
| DNS Load | The DNS load for the service. A load of 255 indicates that the service is down. An eligible load range is from 2 to 254. |

Table 5-2 *Field Descriptions for the show group Command Output (continued)*

| Field | Description |
|---------------------------|---|
| Trans | The number of times that the state of the service has transitioned. |
| Keepalive | The keepalive type of the service. The possible types are FTP, HTTP, ICMP, NAMED, SCRIPT, or TCP. |
| Conn | The number of connections currently on the service. |
| Flow Timeout Multiplier | Number of seconds that a flow remains idle before the CSS reclaims the flow resources, as configured with the flow-timeout-multiplier command. For details on the flow-timeout-multiplier command, refer to the <i>Cisco Content Services Switch Administration Guide</i> . |
| Group Cumulative Counters | The counters for the group. |
| Hits/Frames/Bytes | The number of group hits, frames, and bytes. This field is incremented for traffic from a group server going out from the source group. Traffic coming into the group does not increment the counter. |
| Connection Total/Current | The total number of connections and the current number of connections for the group. |
| FTP Control Total/Current | The total number of FTP control channels that were mapped and monitored by the CSS, and the current number of those connections that are mapped. |
| SP (or SFP) Port Map Info | The port map information for each SFP in a CSS. Includes the status of the portmap command (Enabled or Disabled). |
| SP (or SFP) | The slot and port number of the SFP in a CSS. |
| Base Port | The starting SFP port number in a CSS. |

Table 5-2 *Field Descriptions for the show group Command Output (continued)*

| Field | Description |
|------------------------------|--|
| Configured Base Port | The configured starting port number. |
| Configured Ports SP (or SFP) | The configured number of ports allowed on each SFP in a CSS. |
| Current Mapped Ports | The current number of mapped ports. |
| Last Mapped Port | The most recently mapped port number for each SFP in a CSS. |
| High Water Mark | The highest number of ports that this source group has had concurrently mapped since the last group was activated. |
| No Portmap Errors | The number of times no port could be allocated by the portmapper. |

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](#)
- [ACL Configuration Quick Start](#)
- [Creating an ACL](#)
- [Deleting an ACL](#)
- [Configuring Clauses](#)
- [Adding a Clause When ACLs are Globally Enabled](#)
- [Deleting a Clause](#)
- [Applying an ACL to a Circuit or DNS Queries](#)
- [Removing an ACL from Circuits or DNS Queries](#)
- [Enabling ACLs on the CSS](#)
- [Disabling ACLs on the CSS](#)
- [Showing ACLs](#)
- [Setting the Show ACL Counters to Zero](#)
- [Logging ACL Activity](#)
- [ACL Example](#)

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.

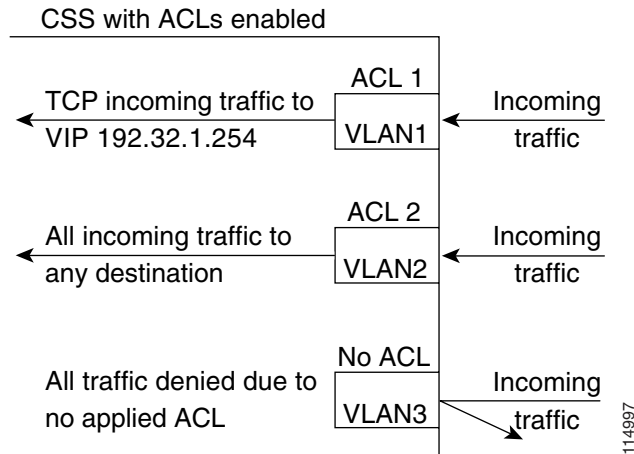
An ACL consists of clauses that you define. The CSS uses these clauses to determine how to handle each packet it processes on a VLAN circuit. When the CSS examines each packet, it either forwards or blocks the packet based on whether or not the packet matches a clause in the ACL. You must configure a 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 5-1](#) shows three VLAN circuits on the CSS.

Figure 5-1 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 will 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 will 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 *will be 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.

ACL Configuration Quick Start

Use the quick-start procedure in [Table 5-3](#) 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 5-3 *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])#
```

Table 5-3 ACL Configuration Quick Start (continued)**Task and Command Example**

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 5-4](#) 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.

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.

Table 5-3 ACL Configuration Quick Start (continued)

Task and Command Example

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 5-3](#).

```
!***** 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 contains 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 when ACLs are enabled on the CSS, the ACL is also removed from the circuit, and, therefore, the CSS denies traffic on the circuit. If you do not want to deny 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 will deny 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 effect 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}
```

Table 5-4 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.



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 will be 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 5-4 Clause Command Options

| Variables and Options | Parameters |
|------------------------------|--|
| <i>number</i> | The number you want to assign to the clause. Enter a number from 1 to 254. |
| <i>action</i> | The action to apply to the clause. Enter one of the following: bypass , deny , permit |
| <i>protocol</i> | The protocol for the traffic type. Enter one of the following: any , icmp , igp , igmp , ospf , tcp , udp |

Table 5-4 Clause Command Options (continued)

| Variables and Options | Parameters |
|-----------------------|--|
| <i>source_info</i> | <p>The source of the traffic. Enter one of the following:</p> <ul style="list-style-type: none"> • <i>ip_address</i> (optionally include <i>subnet mask</i> in IP address format only) for the source IP address and optional mask IP address. • <i>hostname</i> 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 <i>nql_name</i> for an existing Network Qualifier List (NQL) consisting of a list of IP addresses. |
| <i>source_port</i> | <p>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:</p> <ul style="list-style-type: none"> • eq <i>port</i> is equal to the port number. • lt <i>port</i> is less than the port number. • gt <i>port</i> is greater than the port number. • neq <i>port</i> is not equal to the port number. • range <i>low high</i> for a range of port numbers, inclusive. Enter numbers from a range of 1 to 65535. Separate the <i>low</i> and <i>high</i> number with a space. |

Table 5-4 Clause Command Options (continued)

| Variables and Options | Parameters |
|-------------------------|--|
| <i>destination_info</i> | <p>The destination information for the traffic. Enter one of the following:</p> <ul style="list-style-type: none"> • destination any for any combination of destination information. • destination content <i>owner_name/rule_name</i> 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 <i>subnet mask</i> as IP address only; no Classless Inter-domain routing (CIDR) address. • destination hostname for the destination host name. To use a <i>hostname</i>, configure the CSS DNS client first to enable the CSS to translate the host name. • nql <i>nql_name</i> for an existing NQL consisting of host IP addresses. Enter the name of the NQL. |

Table 5-4 Clause Command Options (continued)

| Variables and Options | Parameters |
|-------------------------|--|
| <i>destination_port</i> | <p>The destination port. Enter one of the following. You may use a port number or port name with the options.</p> <ul style="list-style-type: none"> • eq <i>port</i> is equal to the port number. • lt <i>port</i> is less than the port number. • gt <i>port</i> is greater than the port number. • neq <i>port</i> is not equal to the port number. • range <i>low high</i> for a range of port numbers, inclusive. Enter numbers from a range of 1 to 65535. Separate the <i>low</i> and <i>high</i> number with a space. • <i>port names</i>: <ul style="list-style-type: none"> – 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 Smt – telnet = Port 23 Telnet, – ftp = Port 21 Ftp – ftp-data = Port 20 Ftp-data – none = None <p>If you do not define a destination port, this clause allows traffic to any port.</p> |

Table 5-4 Clause Command Options (continued)

| Variables and Options | Parameters |
|--------------------------------------|---|
| sourcegroup <i>name</i> | <p>The source group as the destination for the traffic. Enter the group name. To see a list of source groups, enter:</p> <pre>show group ?</pre> <p>Note The clause number bypass command does not affect NATing on a source group.</p> |
| prefer <i>service_name</i> | <p>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.</p> <p>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 be 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:</p> <pre>clause 10 permit any any destination any prefer ap-redirect@192.168.138.118</pre> <p>Note ACLs configured with a preferred service take precedence over stickiness.</p> <p>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.</p> |

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 “clause 200 permit any any destination any” 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 do not want the CSS to deny 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 *will* 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 *will 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, delete the ACL applied to the circuit, or remove an ACL from DNS queries. To remove an ACL from all circuits, an individual circuit, or from 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
```

Cisco recommends 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 reenables the ACLs on the CSS, the CSS will deny traffic on the circuit.

4. Reapply the ACL on the circuit.

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

5. In global configuration mode, reenables 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 will lose network connectivity. When you enable ACLs, all traffic on a circuit that is not configured in an ACL permit clause *will be 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 discards 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, Cisco recommends 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 -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** - Shows the ACL global configuration. This display also shows you which ACLs are applied to which circuits.

For example, enter:

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

Table 5-5 describes the fields in the **show acl** command output.

Table 5-5 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 |
| 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

Cisco does 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, enter:

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

If ACLs are globally enabled on the CSS, to 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, reenables 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, reenables 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 Extension Qualifier Lists

An extension qualifier list (EQL) is a collection of file extensions that enable you to match a content rule based on extensions. You activate an EQL by associating it as part of a URL in a Layer 5 content rule. Use the **eql** command to access EQL configuration mode and configure an extension qualifier list. Enter a name that identifies the extension list you want to create. Enter an unquoted text string with no spaces and a length of 1 to 31 characters.

For example, enter:

```
(config)# eql graphics
(config-eql[graphics])#
```

To remove an existing EQL, use the **no eql** command from config mode. For example, enter:

```
(config)# no eql graphics
```

Once you create an EQL, you can configure the following attributes for it:

- **description** - Provides a description for the EQL. Enter a quoted text string with a maximum length of 64 characters. For example, enter:

```
(config-eql[graphics])# description "This EQL specifies graphic
file extensions"
```

- **extension name** - Specifies the extension *name* for content on which you want the CSS to match. Enter a text string from 1 to 7 characters. When configuring EQLs for services, make sure you enter an extension for static content such as .avi, .gif, or .jpg. Do not enter extensions for dynamic content such as .asp and .html. The order in which you enter extensions is irrelevant.

For example, enter:

```
(config-eql[graphics])# extension pcx
```

Optionally, you may provide a *description* of the extension type. Enter a quoted text string with a maximum length of 64 characters. For example, enter:

```
(config-eql[graphics])# extension gif "This is a graphics file"
```

To remove an extension from an EQL, use the **no extension** command. For example, enter:

```
(config-eql[graphics])# no extension gif
```

Specifying an EQL in a Uniform Resource Locator

Server selections are based on the URL specified in the owner content rule. To enable the CSS to access a service when a request for content matches the extensions contained in a previously defined EQL, specify the URL and EQL name for the content.

Specify a URL as a quoted text string with a maximum of 252 characters followed by **eql** and the EQL name.



Note

Do not specify a file extension in the URL when you use an EQL in the URL because doing so will cause the CSS to return an error message. For example, the CSS will “return” an error message for the command **url “/*.txt” eql graphics**. The following command is valid: **url “/*” eql graphics**.

For example, enter:

```
(config-owner-content[arrowpoint.com-products.html])# url “/*” eql
graphics
```

The following example enables the CSS to direct all requests to the correct service for content that matches:

- Pathnames (*/customers/products*)
- Extensions listed in the EQL (*graphics*)

```
(config-owner-content[arrowpoint.com-products.html])# url
“/customers/products/*” eql graphics
```

To display an EQL name and extensions configured for a content rule, use the **show rule** command.

For details on the **show rule** command and its output, see [Chapter 3, Configuring Content Rules](#).

Showing EQL Extensions and Descriptions

To display a list of existing EQLs names, use **eql ?** command.

For example, enter:

```
(config)# eql ?
```

To display the extensions configured for a specific EQL including any descriptions, use the **show eql** command and the EQL name. For example, enter:

```
(config)# show eql graphics
```

Table 5-6 describes the fields in the **show eql** command output.

Table 5-6 Field Descriptions for the show eql Command Output

| Field | Description |
|------------|--|
| EQL | The name of the EQL and its description, if configured |
| Extensions | The extensions of content requests associated with the EQL and their descriptions, if configured |

Configuring URL Qualifier Lists

URQL configuration mode allows you to configure a Uniform Resource Locator qualifier list (URQL). A URQL is a group of URLs for content that you associate with one or more content rules. The CSS uses this list to identify which requests to send to a service. For example, you want all streaming video requests to be handled by your powerful servers. Create a URQL that contains the URLs for the content, and then associate the URQL to a content rule. The CSS will direct all requests for the streaming video URLs to the powerful servers specified in the content rule. Creating a URQL to group the URLs saves you from having to create a separate content rule for each URL.



Note

You cannot specify both **url urql** and **application ssl** within the same content rule. You cannot configure a URQL with subscriber services.

Creating a URQL

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

Enter the URQL name you want to create or enter an existing URQL. Enter the name as an unquoted text string with no spaces and a maximum of 31 characters. When you create a URQL, it remains suspended until you activate it using the **activate** command in URQL mode. To display a list of existing URQL names, enter:

```
(config)# urql ?
```

For example, enter:

```
(config)# urql videos  
(config-urql[videos])#
```

To remove an existing URQL, enter the following command in global configuration mode:

```
(config) no urql videos
```

Once you create a URQL:

1. Configure the URLs you want to group in the URQL by:
 - a. Specifying the URL entry
 - b. Defining the URL
 - c. Optionally, describing the URL
2. Designate the domain name of the URLs in a URQL.
3. Add the URQL to a content rule using the owner-content **url** command.
4. Optionally, describe the URQL.

The following sections describe how to complete these tasks.

Configuring a URL in a URQL

Use the **url** command to include the URL for content requests you want as part of this URQL, and optionally provide a description. The following sections describe how to configure a URL in a URQL:

- [Specifying the URL Entry](#)
- [Defining the URL](#)
- [Describing the URL](#)

**Note**

You must create the URL entry before you can define the URL, describe it, or associate it with a content rule.

Specifying the URL Entry

To specify a URL entry in a URQL, enter a URL number from 1 to 1000. For example, enter:

```
(config-urql[videos])# url 10
```

To remove a URL entry from a URQL, use the **no url** command. For example, enter:

```
(config-urql[videos])# no url 10
```

To specify additional URL entries in the URQL, reenter the **url** command. For example, enter:

```
(config-urql[videos])# url 20  
(config-urql[videos])# url 30  
(config-urql[videos])# url 40
```

Defining the URL

To define a URL for the entry, use the **url** command. Enter the URL as a quoted text string with a maximum of 252 characters. The URL must match the URL GET request exactly. Wildcards, partial URL paths, and a trailing “/” character in the URL are not allowed in a URQL URL entry. For example, enter:

```
(config-urql[videos])# url 10 url "/cooking/cookies.avi"
```

To remove a URL from an entry, use the **no url number url** command. Use this command to remove a previously assigned URL before you redefine the URL for an entry. For example, enter:

```
(config-urql[videos])# no url 10 url
```

To define additional URL for the entries, reenter the **url entry url** command. For example, enter:

```
(config-urql[videos])# url 20 url "/cooking/fudge.avi"  
(config-urql[videos])# url 30 url "/cooking/pie.avi"  
(config-urql[videos])# url 40 url "/cooking/cake.avi"
```

Describing the URL

You may optionally enter a description for the URL. Enter a quoted text string with a maximum of 64 characters. For example, enter:

```
(config-urql[videos])# url 10 description "making cookies"
```

To remove a description about the URL, enter:

```
(config-urql[videos])# no url 10 description
```

Designating the Domain Name of URLs in a URQL

Use the **domain** command to designate the domain name or IP address of the URLs to a URQL. Enter the domain name in mnemonic host-name format (for example, `www.arrowpoint.com`) from 1 to 63 characters. Enter the IP address as a valid address for the domain name (for example, `192.168.11.1`).



Note

You must assign a domain before you can activate a URQL. To change the domain address of an existing URQL, suspend the URQL and then change the domain.

For example, enter:

```
(config-urql[videos])# domain "www.arrowpoint.com"
```

or

```
(config-urql[videos])# domain "192.168.11.1"
```

Adding a URQL to a Content Rule

Once you create and configure a URQL, use the **url urql** command to add it to a previously configured content rule. You can assign only one URQL per rule. Also, a content rule may contain either a URL or a URQL. To see a list of URQLs, use the **urql ?** command.



Note

You cannot specify both **url urql** and **application ssl** within the same content rule. You cannot configure a URQL with subscriber services.

For example, enter:

```
(config-owner-content[chefsbest-recipes])# url urql videos
```

To remove a URQL from a content rule, enter:

```
(config-owner-content[chefsbest-recipes])# no url urql
```

To display a URL for a content rule, use the **show rule** command for the content rule. For details on the **show rule** command and its output, see [Chapter 3, Configuring Content Rules](#).

Describing the URQL

Use the **description** command to provide a description for a URQL. Enter the description as a quoted text string with a maximum of 64 characters.

For example, enter:

```
(config-urql[videos])# description "cooking streaming video"
```

To clear a description for the URQL, enter:

```
(config-urql[videos])# no description
```

Activating a URQL

Use the **active** command to activate a suspended URQL. When you create a URQL, it is suspended until you use the **active** command to activate it.

**Note**

Before you can activate a URQL, you must assign the domain for the URLs. See [“Designating the Domain Name of URLs in a URQL”](#) in this chapter.

For example, enter:

```
(config-urql[videos])# active
```

Suspending a URQL

Use the **suspend** command to deactivate a URQL on all currently assigned content rules. For example, enter:

```
(config-urql[videos])# suspend
```

To reactivate the URQL, use the **(config-urql) active** command.

URQL Configuration in a Startup-Config File

The following example shows a URQL configuration in a startup-config file.

```
!***** URQL *****
urql excellence1
  url 10
  url 30
  url 30 url "/arrowpoint.gif"
  domain "192.168.128.109"
  url 10 url "/"
urql excellence2
  url 10
  url 10 url "/poweredby.gif"
  domain "192.168.128.109"
```

Showing URQLs

To display a list of URQLs, enter:

```
(config)# urql ?
```

To display all configured URQLs, enter:

```
(config)# show urql
```

To display a specific URQL, enter:

```
(config)# show urql videos
```

[Table 5-7](#) describes the fields in the **show urql** command output.

Table 5-7 Field Descriptions for the show urql Command Output

| Field | Description |
|------------------|---|
| Name | The name of the URQL |
| Description | The configured description for the URQL |
| Domain | The domain name or address of the URLs associated with the URQL |
| Create Type | The create type (static or dynamic) |
| State | The state of the URQL (Active or Suspended) |
| Rules Associated | The number of rules associated with the URQL |

[Table 5-8](#) describes the additional fields when you display a specified URQL.

Table 5-8 Field Descriptions for a Specified URQL

| Field | Description |
|------------------------------|---|
| URQL Table Domain | The domain name or address of the URLs associated with the URQL |
| Number of entries configured | The number of URL entries in the URQL |
| URL | The URL |
| Description | The description associated with the URL |

Table 5-8 Field Descriptions for a Specified URQL (continued)

| Field | Description |
|-------------|---|
| Create Type | The create type (static or dynamic) |
| State | The state of the URL (Active or Suspended) |
| CSD Entries | The number of Content Server Database (CSD) entries |

Configuring Network Qualifier Lists

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

Use the **description** command in NQL mode to provide a description for an NQL. 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

Use the **ip address** command to add a maximum of 512 networks or services to an NQL. 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 5-9](#) describes the fields in the **show nql** command output.

Table 5-9 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. |

Configuring Domain Qualifier Lists

When you have a requirement for a content rule to match on multiple domain names, you can associate a domain qualifier list (DQL) to the rule. A DQL is a list of domain names that you configure and assign to a content rule, instead of creating a content rule for each domain. Assigning multiple domain names to a DQL enables you to have many domain names match one content rule.

You can use a DQL on a rule to specify that content requests for each domain in the list will match the rule. You can determine the order in which the domain names are listed in the DQL. You can arrange the names in a DQL by assigning an index number as you add the name to the list.



Note

The CSS supports a maximum of 512 DQLs, with a maximum of 2,500 DQL domain name entries. This means that a single DQL can have up to 2500 entries, or five DQLs can have up to 500 entries for each DQL.

DQLs exist independently of any range mapping. You can use them as a matching criteria to balance across servers that do not have VIP or port ranges. If you want to use range mapping when using range services, you need to consider the index of any domain name in the DQL. If you are not using service ranges with DQLs, you do not need to configure any index; the default index is 1.

For example, you could configure a DQL named Woodworker.

```
(config)# dql Woodworker
```

The domain names you could add as part of the DQL include www.wood.com, www.woodworker.com, www.maple.com, www.oak.com. You could configure www.wood.com and www.woodworker.com to have the same mapping index. You can enter indexes from 1 to 1000 and provide an optional quoted description for each index.

For example, enter:

```
(config-dql[Woodworker])# domain www.wood.com index 1 "This is the
same as the woodworker domain"
(config-dql[Woodworker])# domain www.woodworker.com index 1
(config-dql[Woodworker])# domain www.maple.com index 2
(config-dql[Woodworker])# domain www.oak.com index 3
```

If you specify a DQL as a matching criteria for content rule WoodSites, and there are two services, S1 and S2, associated with the rule, the CSS checks the services at mapping time for ranges. To add a DQL to a content rule, use the **url** command as shown:

```
(config-owner-content[WoodSites])# url "/" dql Woodworker
```

For example, if the CSS receives a request for www.oak.com along with other criteria, a match on the WoodSites rule occurs on DQL index 3. If the rule has the roundrobin balance method configured, the CSS examines a service (S2 for this example) to determine the backend connection mapping parameters. If you configured S2 with a VIP address of 10.0.0.1 with a range of 5, the addresses include 10.0.0.1 through 10.0.0.5. Because this service has a range of address and **any** as its port, the DQL index of 3 matches the service VIP range index of 3, which is address 10.0.0.3.

To access DQL configuration mode, use the **dql** command from any configuration mode except boot, group, RMON alarm, RMON event, and RMON history configuration modes. The prompt changes to (config-dql [name]). You can also use this command from DQL mode to access an existing DQL.

See the following sections to configure a DQL:

- [Creating a DQL](#)
- [Describing a DQL](#)
- [Adding a Domain to a DQL](#)
- [Adding a DQL to a Content Rule](#)
- [Removing a DQL from a Content Rule](#)
- [Showing DQL Configurations](#)

Creating a DQL

To create a new DQL, enter the name of the DQL you want to create as an unquoted text string with no spaces and a maximum of 31 characters. To access an existing DQL, enter the DQL name. To display a list of existing DQL names, use the **dql ?** command.

For example, to configure a DQL:

```
(config)# dql pet_domains  
(config-dql[pet_domains])#
```

Describing a DQL

Use the **description** command to provide a description for DQL. Enter the description as a quoted text string with a maximum of 63 characters, including spaces.

For example, enter:

```
(config-dql [pet_domains]) # description "pet supplies"
```

Adding a Domain to a DQL

Use the **domain** command to add a domain to the list of domains supported by a DQL. The syntax is:

```
domain name index number {"description"}
```

The variables and option are:

- *name* - The name of the domain. Enter an unquoted text string with a maximum of 63 characters (for example, www.arrowpoint.com). The CSS matches the domain name exactly.
- *number* - The index number for the domain. Enter a number from 1 to 10000. If a domain has more than one domain name, you can assign the same index number to its different names.
- "*description*" - A description of the domain name. Enter a quoted text string with a maximum of 63 characters including spaces.



Note

The CSS supports a maximum of 512 DQLs, with a maximum of 2500 DQL domain name entries. This means that a single DQL can have up to 2500 entries, or five DQLs can have up to 500 entries for each DQL.

For example, enter:

```
(config-dql[pet_domains])# domain www.birds.com index 1
"idahobased"
(config-dql[pet_domains])# domain www.cats.com index 2 "worldwide"
(config-dql[pet_domains])# domain www.horses.com index 3
"floridabased"
```

Normally, port 80 traffic does not use a port number in the domain name. To specify a port other than port 80, enter the domain name with the port number exactly. Separate the domain name and the port number with a colon. For example, enter:

```
(config-dql[pet_domains])# domain www.dogs.com:8080 index 4
```

To add or delete a domain name from a DQL that is assigned to a content rule, you must first suspend the content rule using the **suspend** command. You cannot make changes to a DQL currently in use by a content rule.

For example, to remove a domain from the example DQL, enter:

```
(config-dql[pet_domains])# no domain www.birds.com
```

Adding a DQL to a Content Rule

Once you have configured a DQL, use the **url** command to add it to a content rule. You cannot use wildcards in DQL entries.

For example, enter:

```
(config-owner-content[pets.com-rule1])# url "/" dql pet_domains
```

Removing a DQL from a Content Rule

To remove a DQL that is assigned to a content rule, you must first suspend the content rule using the **suspend** command. You cannot remove a DQL currently in use by a content rule. Once the content rule is suspended, use the **no dql** command to remove the DQL from the content rule.

For example, enter:

```
(config) no dql pet_domains
```

Showing DQL Configurations

Use the **show dql** command to display all DQL configurations. To display a specific DQL, include the DQL name in the command line.

For example, enter:

```
(config-dql[pet_domains])# show dql pet_domains
```

Table 5-10 describes the fields in the **show dql** command output.

Table 5-10 Field Descriptions for the show dql Command Output

| Field | Description |
|-------------|---|
| Name | The name of the DQL |
| Index | The CSS unique index which identifies the DQL |
| Description | The description for the DQL |
| Index | The DQL unique index number for this domain |
| Domain | The name of the domain associated with the index number |
| Description | The description for the domain |

Configuring Virtual Web Hosting

Virtual Web hosting enables you to host a large number of Web sites on a small number of servers (typically 2 to 10 servers) that have mirrored content. Each server can virtually host multiple IP addresses, ports, or domain names, and may contain hundreds or thousands of Web sites. The servers determine which Web site is being requested based on IP address, port, or domain name.

Configure virtual Web hosting when using File Transfer Protocol (FTP) or UDP applications.

To use virtual Web hosting, configure:

- Services with either a range of IP addresses or a range of ports.
- Content rules with either a range of VIPs or a DQL (but not both). This configuration allows a CSS to map the range of VIPs or the domain names in the DQL to the servers.

- Content rules with either a range of VIPs or a DQL (but not both) that would map to a server without a range. This configuration allows the CSS to map the range of VIPs or many domain names to one server.
- Source groups with a range of VIPs for NATing source IP addresses and ports when using FTP or UDP applications only. This configuration allows a CSS to map a range of service IP addresses or ports to a range of source group VIPs.

You can configure the CSS to load balance the Web sites by configuring port ranges, VIP ranges, or DQLs. For more information on the service and content rule commands required, see [Chapter 1, Configuring Services](#) and this chapter.

For example, if the destination IP address of an inbound content request matches the second VIP in the range configured on a content rule, the CSS maps the flow to the second IP address or port in the range configured on the corresponding service. If an outbound flow originates from the third IP address or port in the range configured on a service, the CSS maps the flow to the third VIP in the range configured on a matching source group.

See [Table 5-11](#) for the steps required to configure virtual Web hosting.

Table 5-11 Virtual Web Hosting Configuration Quick Start

Task and Command Example

1. Enter config mode by typing **config**.

```
(config)#
```

2. Create a service.

```
(config)# service serv1
(config-service[serv1])#
```

3. Assign an IP address to the service and define the IP address range. Enter a number from 1 to 65535.

When using the **ip address range** command, use IP addresses that are within the subnet you are using. The CSS does not use ARP for IP addresses that are not on the circuit subnet.

```
(config-service[serv1])# ip address 10.3.6.1 range 200
```

Table 5-11 Virtual Web Hosting Configuration Quick Start (continued)**Task and Command Example**

4. Configure other service rules as needed (for example, protocol, keepalive parameters).

```
(config-service[serv1])# protocol tcp
(config-service[serv1])# keepalive type http
(config-service[serv1])# keepalive method get
(config-service[serv1])# keepalive uri "/index.html"
```



Note The CSS uses one keepalive for a service configured with an IP address range or port range and always sends the keepalive to the first IP address or port in that range.

5. Activate the service.

```
(config-service[serv1])# active
```

6. Create the content rule.

```
(config-owner[arrowpoint])# content rule1
(config-owner-content[arrowpoint-rule1])#
```

7. Configure a VIP. You can define a VIP range only if you do not plan to configure a DQL.

```
(config-owner-content[arrowpoint-rule1])# vip address 192.168.3.6
range 10
```

When using the **vip address range** command, use IP addresses that are within the subnet you are using. The CSS does not use ARP for IP addresses that are not on the circuit subnet.

8. Configure other content rule commands as needed (for example, port, protocol, and add a service).

```
(config-owner-content[arrowpoint-rule1])# port 80
(config-owner-content[arrowpoint-rule1])# protocol tcp
(config-owner-content[arrowpoint-rule1])# add service serv1
```

9. Activate the content rule.

```
(config-owner-content[arrowpoint-rule1])# active
```

10. Create a source group.

```
(config)# group group1
(config-group[group1])#
```

Table 5-11 Virtual Web Hosting Configuration Quick Start (continued)**Task and Command Example**

11. Configure a VIP address range on the source group.

```
(config-group[group1])# vip address 192.168.5.7 range 10
```

12. Add the services that you want to be part of the group.

```
(config-group[group1])# add service serv1
```

13. Activate the source group.

```
(config-group[group1])# active
```

14. If you have not configured a VIP range on a content rule, you can create a DQL.

```
(config)# dql pet_domains
(config-dql[pet_domains])#
```

15. Add domains to the DQL you created.

```
(config-dql[pet_domains])# domain www.birds.com index 1
"idaho-based"
(config-dql[pet_domains])# domain www.cats.com index 2
"worldwide"
(config-dql[pet_domains])# domain www.horses.com index 3
"florida-based"
```

16. Add the DQL to the content rule using the **url** command.

```
(config-owner-content[arrowpoint-rule1])# url "/*" dql
pet_domains
```

Where to Go Next

You can configure HTTP header load balancing by creating an HTTP header field group and configuring HTTP header fields. For information, see [Chapter 6, Configuring HTTP Header Load Balancing](#).