



# Configuring Content Rules

---

This chapter describes how to create and configure content rules. Information in this chapter applies to all CSS models except where noted.

This chapter contains the following major sections:

- [Content Rule Overview](#)
- [Naming and Assigning a Content Rule to an Owner](#)
- [Configuring a Virtual IP Address](#)
- [Configuring a Domain Name Content Rule](#)
- [Adding Services to a Content Rule](#)
- [Activating a Content Rule](#)
- [Suspending a Content Rule](#)
- [Removing a Content Rule](#)
- [Removing a Service from a Content Rule](#)
- [Configuring a Protocol](#)
- [Configuring a Port](#)
- [Configuring Load Balancing](#)
- [Configuring a DNS Balance Type](#)
- [Configuring Hot Lists](#)
- [Configuring HTTP Method Parsing](#)
- [Configuring Extension Qualifier Lists](#)
- [Configuring URL Qualifier Lists](#)

- [Specifying a Uniform Resource Locator](#)
- [Specifying the Number of Spanned Packets](#)
- [Specifying a Load Threshold](#)
- [Including Services in a CSS Ping Response Decision](#)
- [Enabling TCP Flow Reset Reject](#)
- [Configuring Persistence, Remapping, and Redirection](#)
- [Defining Failover](#)
- [Specifying an Application Type](#)
- [Enabling Content Requests to Bypass Transparent Caches](#)
- [Showing Content](#)
- [Showing Content Rules](#)
- [Clearing Counters in a Content Rule](#)

For information on how service, owners and content rules work together, see [Chapter 1, Content Load-Balancing Overview](#).

## Content Rule Overview

The CSS uses content rules to determine:

- Where the content physically resides, whether local or remote
- Where to direct the request for content (which service or services)
- Which load-balancing method to use

The type of rule also implies the layer at which the rule functions.

- A Layer 3 content rule implies a destination IP address of the host or network.
- A Layer 4 content rule implies a combination of destination IP address, protocol, and port.
- A Layer 5 content rule implies a combination of destination IP address, protocol, port, and URL that may or may not contain an HTTP cookie or a domain name.

## Content Rule Hierarchy

Content rules are hierarchical. That is, if a request for content matches more than one rule, the characteristics of the most specific rule apply to the flow. The CSS uses this order of precedence to process requests for the content, with 1 being the highest match and 9 being the lowest match. The hierarchy for content rules is as follows:

1. Domain name, IP address, protocol, port, URL
2. Domain name, protocol, port, URL
3. IP address, protocol, port, URL
4. IP address, protocol, port
5. IP address, protocol
6. IP address
7. Protocol, port, URL
8. Protocol, port
  - Protocol

## Matching Precedence for Layer 5 Rules

In a Layer 5 content rule, the CSS matches the URL after the CSS matches the IP address, protocol, and port. An HTTP header field group in a Layer 5 content rule enables a rule to be more specific than if the rule defined just a URL. For more information on configuring HTTP header field groups, refer to the [Chapter 12, Configuring HTTP Header Load Balancing](#).

Because content rules are hierarchical, if a request for content matches more than one rule, the characteristics of the most specific rule apply to the flow. For a Layer 5 content rule, the CSS uses the following order of precedence to process requests for the content, with 1 being the highest match and 10 being the lowest match.

1. Exact URL (for example, /test/index.html) with a header field group configuration.
2. Exact URL (for example, /test/index.html).
3. Wildcard URL length (for example, /test/ind\* or /test/index.h\*) with a header field group configuration.

4. Wildcard URL length (for example, /test/ind\* or /test/index.h\*) with a partial path match before applying the wildcard.
5. Wildcard URL extension (for example, /test/\*.html) with a header field group configuration.
6. Wildcard URL extension (for example, /test/\*.html).
7. Wildcard Extension Qualifier List (for example, “/test/\*” eql *EQL\_LIST*) with a header field group configuration. For more information on Extension Qualifier Lists (EQLs), see the [“Configuring Extension Qualifier Lists”](#) section.
8. Wildcard EQL (for example, “/test/\*” eql *EQL\_LIST*).
9. Wildcard URL (for example, /test/\*) with a header field group configuration.
10. Wildcard URL (for example, /test/\*) where the entire path segment is wildcarded without regard to a partial path match.

In the following example, the content rules ruleWap and ruleNoWap are identical except ruleWap includes a header field group.

- The content rule ruleWap matches any TCP port 80 traffic destined for VIP 192.168.128.151 that has the MSISDN field in the HTTP header, as defined in the header field group configuration.
- The content rule ruleNoWap matches any TCP port 80 traffic destined for VIP 192.168.128.151 that does not have the MSISDN field in the HTTP header.

Because content rule ruleWap includes a header field group, the CSS will try to match on it before trying to match on content rule ruleNoWap.

```
header-field-group wap
  header-field 1 msisdn exist

owner arrowpoint
  content ruleWap
    vip address 192.168.128.151
    protocol tcp
    port 80
    url "/*"
    add service server1
    add service server2
    header-field-rule wap
    active
```

```
content ruleNoWap
  vip address 192.168.128.151
  protocol tcp
  port 80
  url "/"*
  add service server21
  add service server22
  active
```

## Content Rule Configuration Quick Start

[Table 10-1](#) provides a quick overview of the steps required to create and configure a Layer 3 content rule. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the content rule configuration options, see the sections following [Table 10-1](#).

Ensure that you have already created and configured a service and owner for the content rules. The command examples in [Table 10-1](#) create a Layer 3 content rule for owner arrowpoint.

**Table 10-1 Content Rule Configuration Quick Start**

---

### Task and Command Example

---

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

```
# config
(config)#
```

---

2. Enter the owner mode for which you wish to create content rules.

```
(config)# owner arrowpoint
```

---

3. Create the content rule for the owner.

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

The CSS enters the owner-content rule mode.

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

---

**Table 10-1 Content Rule Configuration Quick Start (continued)****Task and Command Example**

4. Configure a VIP address or domain name for the owner content. This example configures a VIP address, which implies a Layer 3 content rule.

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

If you require a Layer 4 content rule, specify a protocol in the content rule and a specific TCP/UDP port number (in addition to the VIP address or domain name).

```
(config-owner-content[arrowpoint-rule1]# protocol tcp  
(config-owner-content[arrowpoint-rule1]# port 80
```

If you require a Layer 5 content rule, specify a URL in the content rule (in addition to the protocol and port number).

```
(config-owner-content[arrowpoint-rule1]# url  
"/www.arrowpoint.com/*"
```

5. Specify a load-balancing type.

```
(config-owner-content[arrowpoint-rule1]# balance aca
```

6. Add previously configured services to the content rule.

```
(config-owner-content[arrowpoint-rule1]# add service serv1  
(config-owner-content[arrowpoint-rule1]# add service serv2
```

7. Activate the content rule.

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

8. Display the content rules (optional).

```
(config-owner-content[arrowpoint-rule1]# show rule
```

The following running-configuration example shows the results of entering the commands in [Table 10-1](#).

```
!***** OWNER *****
owner arrowpoint
  address "200 Beaver Brook Road, Boxborough, MA 01719"

content rule1
  add service server1
  vip address 192.168.3.6
  balance aca
  add service serv2
  protocol tcp
  port 80
  url "http://www.arrowpoint.com/"
```

## Naming and Assigning a Content Rule to an Owner

By assigning content rules to an owner, you can manage access to the content. Assign content rules to an owner by creating the content rule in the mode for that owner. The CSS identifies content rules by the names you assign.

To name a content rule and assign it to an owner, use the **content** command. Enter a content rule name from 1 to 31 characters.



### Note

The CSS supports a maximum of 31 characters for content rule names. In a content rule-based DNS configuration, CSS peers share content rules over APP sessions. When a CSS learns a content rule from a peer, it appends an "at" sign (@) and the VIP address of the CSS peer to the content rule name. Depending on the length of the original content rule name and the VIP address of the peer, the learned content rule name may exceed 31 characters.

To maintain the maximum length of 31 characters, the CSS drops characters from the left side of the learned content rule name. If you have content rule names greater than 15 characters with content-rule based DNS configured, this process could cause a CSS to have two content rules with the same name, which renders both content rules inoperable. To prevent this occurrence, always place the unique characters in a content rule name at the end of the name.

The following example assigns:

- The name `rule1` to the content rule
- Content rule `rule1` to owner `arrowpoint`

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

Once you assign a content rule to an owner, the CLI prompt changes to reflect the specific owner and content rule mode.

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

Within owner and content mode, you can configure how the CSS will handle requests for the content. To remove an existing content rule from an owner, use the **no content** command from owner mode. For example, enter:

```
(config-owner[arrowpoint])# no content rule1
```

## Configuring a Virtual IP Address

A VIP address is an address that an Internet Domain Name System (DNS) provides when asked to resolve a domain name. For example, a DNS server may translate `www.arrowpoint.com` to the VIP address `192.217.4.15`. Internet service providers (ISPs) generally assign VIP addresses. ISPs request VIP addresses from the Internet Assigned Numbers Authority (IANA).

Assigning a VIP address to owner content enables the CSS to translate (using Network Address Translation (NAT)) the VIP address to the IP address of the service where the content resides.



### Note

---

The CSS allows you to configure a domain name instead of a VIP address. See the next section for information on configuring a domain name. You may configure either a VIP address, a domain name, or both in a content rule.

---

To enable the CSS to translate an owner's Internet IP address to the IP address of the service where the content resides, configure a VIP address to the owner content. By translating a VIP address to the service IP address, the CSS enhances network security because it prevents users from accessing your private network IP addresses.

**Caution**

Ensure that all VIP addresses are unique IP addresses. Do not configure a VIP address to the same address as an existing IP address on your network or a static Address Resolution Protocol (ARP) entry.

**Note**

The CSS supports Adaptive Session Redundancy (ASR) on Cisco 11500 series CSS peers in an active-backup VIP redundancy and virtual interface redundancy environment to provide stateful failover of existing flows. For details on ASR, refer to the *Cisco Content Services Switch Global Server Load-Balancing Configuration Guide*.

**Note**

When you configure a rule without a VIP address (wildcard VIP rule), the rule matches any VIP address that matches the other configured rule attributes (for example, port and protocol). When you configure a rule without a VIP address and without a port (double-wildcard caching rule), the rule matches any VIP address or port that matches the other configured rule attributes (for example, protocol). For more information on double-wildcard caching rules, see [Chapter 13, Configuring Caching](#). If you have a configuration that requires either type of rule, be aware that the client request will match this rule when the client request attempts to connect directly to a server IP address.

The variables and options for the **vip address** command include:

- *ip\_address* or *host* - The IP address or name for the content rule. Enter the address in either dotted-decimal IP notation (for example, 192.168.11.1) or mnemonic host-name format (for example, myhost.mydomain.com). The CSS restricts VIP and IP addresses on content rules to class A, B, or C addresses. The CSS does not allow multicast (class D and E) and IP addresses with ranges beyond the end of the address range.
- **range number** - The range option and variable allows you to specify a range of IP addresses starting with the VIP address. Enter a number from 1 to 65535. The default range is 1. The *ip\_or\_host* variable is the first address in the range. For example, if you enter a VIP address of 172.16.3.6 with a range of 10, the VIP addresses will range from 172.16.3.6 to 172.16.3.15.

When configuring a content rule with the same VIP address as a source group, the VIP address range for a content rule must be the same as the range for a source group.

When configuring a port mapper in a source group with the same VIP address as the content rule, you must configure the port mapper and content rule with the same VIP address ranges.

Note that the maximum VIP address range for a port mapper is 255. If you need to create a rule with a VIP address range greater than 255, create multiple rules with smaller ranges instead.

You cannot configure content rules with VIP address ranges that overlap, including rules with different port numbers. However, you can configure content rules with the same VIP address range.

**Note**


---

The CSS does not support VIP address ranges on the SSL module. The **ssl-proxy-list** and **ssl-server vip** commands cannot be configured as part of a content rule VIP configured using the **vip address** command with the **range** option.

---

**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 Services](#)).

---

To configure a VIP address, issue the **vip address** command and specify either an IP address or a host name. For example, enter:

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

**Note**


---

When you ping a VIP address, the CSS responds only if there is at least one live service, live sorry server, or redirect string configured for the VIP address, or if the service is associated with a source group. If the services or sorry servers are down and you have not defined a redirect string for the VIP address, the CSS does not respond to the ping.

---

To configure a VIP address with a range of 10, use the **vip address** command with the **range** option. For example, enter:

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

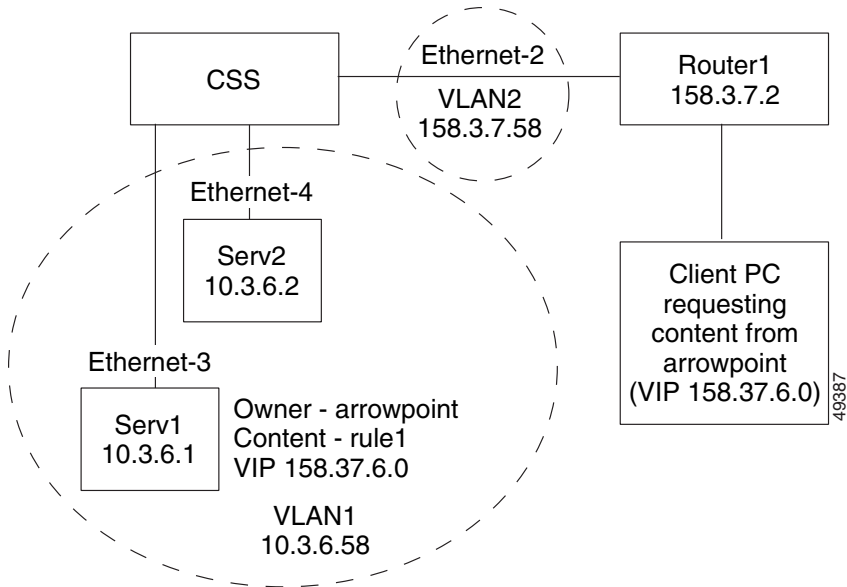
When using the **vip address range** command, use IP addresses that are within the subnet you are using. The CSS does not use the ARP for IP addresses that are not on the circuit subnet. For example, if you configure the circuit for 10.10.10.1/24 and configure the VIP address range as 10.10.10.2 range 400, the CSS will not use the ARP for any IP addresses beyond 10.10.10.254. Using the same example with a VIP address range of 200, the CSS will use the ARP for all IP addresses in the range.

To remove a VIP address from a content rule, enter:

```
(config-owner-content[arrowpoint-rule1])# no vip address
```

**Figure 10-1** shows an example of configuring a VIP address. In this example, a user requests content from arrowpoint. The content physically resides on the server with IP address 10.3.6.1. By configuring VIP address 158.37.6.0 to the content, the CSS translates the VIP address to the server IP address where the content actually resides without exposing internal IP addresses.

Figure 10-1 Example of Configuring a Virtual IP Address



## Configuring a Domain Name Content Rule

The CSS allows you to use a domain name in place of, or in conjunction with, a VIP address in a content rule. Using a domain name in a content rule enables you to:

- Enable service provisioning to be independent of IP-to-domain name mappings
- Provision cache bandwidth as needed based on domain names



### Note

Domain names in content rules are case-insensitive, regardless of the **case** command setting.

To configure a domain name in a content rule, use the **url** command and place two slash characters (*//*) at the front of the quoted *url\_name* or *url\_path*.

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# url "/www.arrowpoint.com/*"
```

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-owner-content[arrowpoint-rule1])# url  
"/www.arrowpoint.com:8080/*"
```

Use domain name rules rather than VIP rules when you have several transparent caches and you want certain domains to use the most powerful cache server. You want all other domains load balanced among the remaining cache servers. For this configuration, set up a domain name rule for the specific domains you want directed to the powerful cache server. Then configure a wildcard VIP rule (specify port 80 and no VIP address) to balance all other HTTP traffic among the remaining caches.

You may use a single VIP address in front of a server that is hosting many domain names. Over time, some of the domain names may receive more traffic and could benefit from having their content on a separate server. To segregate the traffic, configure the domain names you want directed to specific services. You do not need to configure additional VIP addresses for the domain names because the CSS will use the domain names as the matching criteria in the content rules.

## Matching Content Rules to Multiple Domain Names

When you have a requirement for a content rule to match 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. 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 that 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.

DQLs exist independently of any range mapping. You can use them as matching criteria to balance across servers that do not have IP addresses or port ranges. If you want to use range mapping when using a service range, you need to consider the index of any domain name in the DQL.

**Note**

---

The DQL indexes need to map to the service range. If the indexes do not map properly, an error message appears when you activate the rule.

---

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 load-balancing method, the CSS examines a service (S2 for this example) to determine the back-end 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 addresses and 0 (any) as its port, the DQL index of 3 matches the service VIP address range index of 3, which is address 10.0.0.3.

To delete a DQL, use the **no dql** command. For example, enter:

```
(config)# no dql Woodworker
```

**Note**

---

You cannot delete a DQL currently in use by a content rule.

---

For a complete description of DQLs, see the [“Configuring Domain Qualifier Lists”](#) section.

## Configuring a Content Rule Using a Domain Name and a VIP Address

Use a domain name and a VIP address in a content rule when you want the CSS to match content requests going to a specific domain at a specific VIP address. If the CSS is serving more than one VIP address at the domain name, configure two domain name content rules and specify the different VIP addresses.

This configuration is shown in the following sample running-config. Note that because the IP addresses in the example are contiguous, you could use the **vip address range** command to specify a VIP address range of 2.

For example:

```
content domainRule1
  vip address 192.168.1.1
  protocol tcp
  port 80
  url "://domain.com/*"
  add service Serv1
  activate

content domainRule2
  vip address 192.168.1.2
  protocol tcp
  port 80
  url "://domain.com/*"
  add service Serv1
  activate
```

If your network topology does not require that the CSS use an ARP reply for VIP addresses, you do not need to configure separate content rules for the domain name and VIP address. In this situation, a domain name content rule without a VIP address is sufficient because it will match all content requests going to the domain regardless of the VIP address. For example:

```
content domainRule3
  protocol tcp
  port 80
  url "://domain.com/*"
  add service Serv1
  active
```

An example of a topology where an ARP reply is not required is when an upstream router has the CSS statically configured as the next-hop router for the VIP addresses.

## Using Wildcards in Domain Name Content Rules

You can use wildcards in domain names as part of the matching criteria for a content rule. Domain name wildcards work within the content rule hierarchy. That is, if a request for content matches more than one rule (including a wildcard domain name), the characteristics of the most specific rule determine how the CSS sets up the flow.

**Note**

You cannot use wildcards with either a DQL or a Uniform Resource Locator Qualifier List (URQL).

For example, the following content rule criteria have the highest precedence because, as a set, they provide the greatest specificity in matching content:

Domain name, IP address, protocol, port, URL

If you want to create a content rule using all these criteria, such as the configuration shown below, then the content rule matches only the JPEG files that are found in the domain whose name starts with “arr,” as well as the other criteria, including VIP address, protocol, and port number.

```
(config-owner-content[arrowpoint-rule1]) # vip address 192.168.3.6
(config-owner-content[arrowpoint-rule1]) # protocol tcp
(config-owner-content[arrowpoint-rule1]) # port 80
(config-owner-content[arrowpoint-rule1]) # url "//arr*.com/*.jpg"
```

When the CSS encounters a content rule with a wildcard domain name and matches according to the content rule hierarchy, it stops the search at that point. This behavior is consistent with the way that the CSS manages content rules in general.

For example, if the content request matches the rule with VIP address 192.168.3.6 and URL /\*, the CSS does not continue the search to match a second rule with a wildcard VIP address (no address specified) and a URL of /\*.jpg. The specific address match makes the first rule more specific than the second rule.

To further clarify, if the match occurs on a rule with //arrowpoint\*.com/\*, the search stops at that point and does not continue to match a rule with //arr\*.com/\*.gif, because the first rule is a more specific match. Also note that a fully specified domain name rule (arrowpoint.com) is more specific than a wildcard domain name rule (arr\*.com).

For example, to have the content rule match on all instances of the text string “arr” in the domain name portion of the content rule, enter:

```
(config-owner-content[arrowpoint-rule1]) # url "//arr*.com/*"
```

## General Guidelines for Domain Name Wildcards in Content Rules

A domain name is made up of text strings called “words” and word separators called “dots” (.). The CSS parses the domain name from right word to left word. The CSS allows wildcards to be used as part of the domain name in one word or more than one word, but the wildcard cannot start the word.

For example, the CSS supports the following domain names:

- www.arr\*.com
- arr\*.com
- \*.arr\*.com
- arr\*.home.com

Notice that the wildcard character either appears by itself as a domain word or appears to the *right* of any characters that start a domain word. However, a wildcard character cannot start a domain name word.

For example, the CSS does not support the following domain names:

- \*point.com
- \*.\*point.com
- \*point.home.com

**Note**

---

You cannot use wildcards on the *rightmost* portion of the domain name (for example, .com, .org, .gov) . For this reason, the wildcard domain name syntax f\* is not supported. You can use wildcards in any other words that make up the domain name.

---

## 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

Assigning multiple domain names to a DQL enables you to have many domain names match one content rule. 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 "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"
```

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 10-2 describes the fields in the **show dql** command output.

**Table 10-2** 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 3, 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 10-3](#) for the steps required to configure virtual Web hosting.

**Table 10-3 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 10-3 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 10-3 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
```

The following running-configuration example shows the results of entering the commands in [Table 10-3](#).

```
!***** SERVICE *****
service serv1
 ip address 10.3.6.1 range 200
 protocol tcp
 keepalive type http
 keepalive method get
 keepalive uri "/index.html"
 active

!***** DQL *****
dql pet_domains
 domain www.birds.com index 1 "idaho-based"
 domain www.cats.com index 2 "worldwide"
```

```
domain www.horses.com index 3 "florida-based"
!***** OWNER *****
owner arrowpoint

content rule1
  vip address 192.168.3.6 range 10
  add service serv1
  protocol tcp
  port 80
  url "/"* dql pet_domains
  active

!***** GROUP *****
group group1
  vip address 192.168.5.7 range 10
  add service serv1
  active
```

## Adding Services to a Content Rule

Adding a service to a content rule includes it in the resource pool that the CSS uses for load-balancing requests for content. The maximum number of services that you can add to a single content rule is 64. Note that a service may belong to multiple content rules.

To add an existing service to a content rule, use the **add** command. To see a list of services you can add to a content rule, use **add service ?** command.



### Note

---

You can add local services only to a content rule that contains either a Domain Qualifier List (DQL) or a service port range.

---

The **add service** command enables you to add the following types of services to a content rule:

- Service
- Primary sorry server
- Secondary sorry server

For information on configuring service types, see the “[Specifying a Service Type](#)” section in [Chapter 3, Configuring Services](#).

When you configure a Layer 3 or Layer 4 content rule, the rule matches the local services. If:

- The local services are not active or configured, the rule matches the primary sorry server
- The primary sorry server fails, the rule matches the secondary sorry server

Redirect services and redirect content strings cannot be used with Layer 3 or Layer 4 rules because they use the HTTP protocol.

When you configure a Layer 5 content rule, the CSS directs content requests to local services. If:

- The local services are not active or configured, the rule sends the HTTP redirects with the location of the redirect services to the clients
- The local and redirect services are not active or configured, the rule forwards the HTTP requests to the primary sorry server
- All services are down except the secondary sorry server, the rule forwards the HTTP requests to the secondary sorry server

**Note**

---

A Layer 5 content rule supports the RFC-2518 HTTP CONNECT, GET, HEAD, POST, PUSH, and PUT methods. In addition, the CSS recognizes and forwards the following RFC-2518 extension methods directly to the destination server in a transparent caching environment but does not load balance them: OPTIONS, TRACE and PROPFIND, PROPPATCH, MKCOL, MOVE, LOCK, UNLOCK, COPY, DELETE. To configure the CSS to support the RFC-2518 extension methods, see the “[Configuring HTTP Method Parsing](#)” section.

---

**Note**

In some environments, URL, cookie strings, or HTTP header information can span over multiple packets. In these environments, the CSS can parse multiple packets for Layer 5 information before making load-balancing decisions. Through the global configuration mode **spanning-packets** command, the CSS can parse up to 20 packets; the default is 6. The CSS makes the load-balancing decision as soon as it finds a match and does not require parsing of all of the configured number of spanned packets. Because parsing multiple packets does impose a longer delay in connection, performance can be impacted by longer strings that span multiple packets. For information on using the **spanning-packets** command, see the [“Specifying the Number of Spanned Packets”](#) section later in this chapter.

## Adding a Service to a Content Rule

Use the **add service** command to add a service to a content rule. The maximum number of services that you can add to a single content rule is 64.

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# add service serv2
```

## Specifying a Service Weight

The CSS uses the weight for a service when you configure weighted roundrobin load balancing on the content rule. When you assign a higher weight to the service, the CSS redirects more requests to the service.

When you add a service to a content rule, you can assign a weight for the service using the **add service service\_name weight** command or the **change service service\_name weight** command as described as follows:

- **add service service\_name weight** - This command allows you to assign a weight to the service used when you configure weighted roundrobin load balancing on the content rule.
- **change service service\_name weight** - This command allows you to modify the weight of a service without removing the service from the content rule and adding it back again. Removing the service causes all existing sticky sessions created on the service, as a result of matching on the sticky content rule, to terminate. Enter the server name as a case-sensitive unquoted text string with no spaces.

Both commands override the server-specific weight and apply only to the content rule to which you add the service.

To set the weight for a service, enter a number from 0 (graceful shutdown) to 10. The default is the weight configured for a service through the **(config-service) weight** command (see the “[Configuring Weight and Graceful Shutdown](#)” section in [Chapter 3, Configuring Services](#)). By default, all services have a weight of 1.

**Note**


---

When you configure weighted roundrobin load balancing on the content rule, the configured weight takes precedence over the service weight reported by a configured DFP agent for that content rule as well as the weight configured in service mode.

---

If you want to perform a graceful shutdown of an overloaded service or take a service offline gracefully for maintenance, when you specify a weight of 0, no new connections, except the connections for existing sticky sessions, will be directed to the service. Over time, as existing sticky sessions complete, the load on the service begins to diminish. Changing the weight from 0 to a value between 1 and 10 causes the service to be brought back into rotation for all load-balancing methods.

For example, to specify a service weight of 3, enter:

```
(config-owner-content[arrowpoint-rule1]) add service serv2
weight 3
```

For example, to specify a weight of 0 to gracefully shut down an active service, enter:

```
(config-owner-content[arrowpoint-rule1]) change service serv2 weight 0
```

To restore the weight to the weight configured in service mode, enter:

```
(config-owner-content[arrowpoint-rule1]) no change service serv2
```

Note the following guidelines for the **add service *service\_name* weight** and the **change service *service\_name* weight** commands when configuring the CSS for a graceful shutdown:

- If you do not have a weighted roundrobin load-balancing method specified for the content rule or do not have DFP specified for server load-balancing, use the **weight** command only in service mode (see the “[Configuring Weight and Graceful Shutdown](#)” section in [Chapter 3, Configuring Services](#)). For these load-balancing methods, using the **add service *service\_name* weight** or the **change service *service\_name* weight** command in content mode has no effect on the service weights and cannot be used to gracefully shut down the service.
- Weight is not configurable on a content rule for primary or secondary sorry servers. Sorry servers can be gracefully shut down only when you set the weight to 0 in service mode.
- We recommend that you use the **sticky-inact-timeout** command to specify an inactivity timeout period if you use advanced load-balancing methods such as **sticky-srcip** or **sticky-srcip-dstport** in conjunction with a graceful shutdown. Once the sticky entries time out as a result of inactivity, the connection count to the shutdown service decreases.

## Adding a Primary Sorry Server to a Content Rule

The CSS directs content requests to the primary sorry server when all other services are unavailable. You can configure this service to contain content, or to provide a drop or redirect message. This service is not used in load balancing.



### Note

---

If you configure the **persistence reset remap** command in the global configuration and **no persistent** command on the content rule, when a local service becomes available again, the CSS remaps any new or in-progress persistent connections to the local server from the sorry server. Otherwise, new connections go to the available local services, but in-progress persistent connections stay on the sorry server. For more information on service remapping and redirection, see the [Configuring HTTP Redirection and Service Remapping](#) section.

---

Use the **primarySorryServer** command to configure the primary sorry service for a content rule. Enter the server name as a case-sensitive unquoted text string with no spaces.

**Note**

You can only add a primary sorry server to a rule if its range for the IP address or port is equal to the range for the IP address or port of each service on the rule. For example, if the rule has two services each with a range of three addresses, the primary sorry server must have a range of three addresses.

For example, enter:

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

To remove a primary sorry service, enter:

```
(config-owner-content[arrowpoint-rule1])# no primarySorryServer
```

## Adding a Secondary Sorry Server to a Content Rule

A secondary sorry service is a backup service the CSS uses when the primary sorry service is unavailable. You can configure this service to contain content, or to provide a drop or redirect message. This service is not used in load balancing.

Use the **secondarySorryServer** command to configure the secondary sorry service for a content rule. Enter the server name as a case-sensitive unquoted text string with no spaces.

**Note**

You can only add a secondary sorry server to a rule if its range for the IP address or port is equal to the range for the IP address or port of each service on the rule. For example, if the rule has two services each with a range of three addresses, the secondary sorry server must have a range of three addresses.

For example, enter:

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

To remove a secondary sorry service, enter:

```
(config-owner-content[arrowpoint-rule1])# no secondarySorryServer
```

## Adding a DNS Name to a Content Rule

To specify a DNS name that maps to a content rule, use the **add dns** command. The options for this command are:

- **add dns *dns\_name*** - The DNS name to be mapped to the content rule. Enter the name as a case-sensitive unquoted text string with no spaces and a length of 1 to 31 characters.
- **add dns *dns\_name ttl\_value*** - The DNS name to be mapped to the content rule with the optional Time to Live (TTL) value, in seconds. This value sets how long the DNS client remembers the IP address response to the query. Enter a value from 0 to 255. The default is 0.

**Note**

---

When using the content **add dns** command, you must add DNS names in lowercase only. If you enter DNS names with a combination of uppercase and lowercase characters, a startup error appears and you must reenter the names in all lowercase characters.

---

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# add dns arrowpoint 120
```

To remove a DNS name mapped to the content rule, enter:

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

**Note**

---

To configure DNS server functionality on the CSS, use the (**config**) **dns-server** command.

---

## Disabling DNS in a Content Rule

If the services related to a content rule are not available for DNS activities, the CSS informs other CSSs through an Application Peering Protocol (APP) session. However, the services remain active for other functions.

To disable DNS in a content rule, use the **dns-disable-local** command.

If you configure the **dns-disable-local** command on a content rule in a GSLB environment, the rule is active, and there is no DNS peer configured for the domain name, the CSS responds with SERVERFAIL to the server that requested the DNS resolution.

For example, to disable DNS for a specific content rule, enter:

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

To enable DNS in the content rule, use the **no dns-disable-local** command. For example, enter:

```
(config-owner-content [arrowpoint-rule1])# no dns-disable-local
```

## Activating a Content Rule

Activating content enables the CSS to provide access to the content. To activate content, use the **active** command in content mode to activate specific content.



### Note

Once a content rule is activated, the following commands cannot be changed for the active content rule: **port**, **protocol**, **balance**, **dnsbalance**, **header-field-rule**, and **url**. In addition, you cannot remove the last remaining service from the content rule. If you need to make modifications to an active content rule, you must first suspend it.

For example, enter:

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

## Suspending a Content Rule

Suspending a content rule deactivates it. Suspending a content rule:

- Prevents the CSS from providing access to the content
- Does not affect existing flows to the content

To suspend a content rule, use the **suspend** command in content mode. For example, enter:

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

## Removing a Content Rule

To remove an existing content rule, use the **no content** command from owner mode. For example, enter:

```
(config-owner[arrowpoint])# no content rule1
```

## Removing a Service from a Content Rule

Removing a service removes it from the resource pool that the CSS uses for balancing the load of requests for content governed by a rule. When you remove a service, the remaining services are rebalanced.

To remove an existing service from a content rule, use the **remove** command from owner-content mode. For example, enter:

```
(config-owner-content[arrowpoint-rule1])# remove service serv1
```

## Configuring a Protocol

Specifying a protocol in a content rule enables the CSS to direct requests for content associated with the content rule to use a specific protocol. You may specify the following protocols for content:

- **any** (default; means the rule will match a TCP or UDP port)
- **tcp**
- **udp**

If you specify Session Initiation Protocol (SIP) as the application type and you have not previously configured a protocol in the content rule, the CSS automatically enters the default SIP protocol of UDP in the running-configuration file. See the [“Specifying an Application Type”](#) section.

To configure the TCP protocol for content, enter:

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

To reset the protocol to the default of **any**, enter:

```
(config-owner-content[arrowpoint-rule1])# no protocol
```

## Configuring a Port

Specifying a port enables the CSS to associate a content rule with a specific TCP/UDP port number. Specify a port number ranging from 0 to 65535. The default is 0, which indicates any port.

If you specify SIP as the application type and you have not previously specified a port in the content rule, the CSS automatically enters the default SIP port number of 5060 in the running-configuration file. See the [“Specifying an Application Type”](#) section.

To configure a port for content, enter:

```
(config-owner-content[arrowpoint-rule1])# port 80
```

To reset the port number to the default of 0 value, enter:

```
(config-owner-content[arrowpoint-rule1])# no port
```

## Configuring Load Balancing

To specify the load-balancing algorithm for a content rule, use the **balance** command available in content configuration mode.



### Note

We strongly recommend that if you configure a **balance** method on a content rule that requires the TCP protocol for Layer 5 (L5) spoofing, you should configure a default URL string, such as **url “/\*”**. The addition of the URL string forces the content rule to become an L5 rule and ensures L5 load balancing or stickiness. If you do not configure a default URL string, unexpected results can occur.

In the following configuration example, if you configure a Layer 3 (L3) content rule with an L5 balance method, the CSS performs L5 load balancing but will reject UDP packets.

```
content testing
vip address 192.168.128.131
add service s1
balance url
active
```

The **balance url** method is an L5 load-balancing method in which the CSS must spoof the connection and examine the HTTP GET content request to perform load balancing. The CSS rejects the UDP packet sent to this rule because a UDP connection cannot be L5. Though the CSS allows this rule configuration, its expected behavior would be more clear if you promote the rule to L5 by configuring the **url “/\*”** command.

---

The options are:

- **balance aca** - ArrowPoint Content Awareness load-balancing algorithm (see the “Using ArrowPoint Content Awareness Based on Server Load and Weight” section in [Chapter 6, Configuring Loads for Services](#)). ACA balances the traffic over the services based on load or on server weight and load.
- **balance destip** - Destination IP address division algorithm. The CSS directs all client requests with the same destination IP address to the same service. This option is typically used in a caching environment.
- **balance domain** - Domain name division algorithm. The CSS divides the alphabet evenly across the number of caches. It parses the host tag for the first four letters following the first dot and then uses these characters of the domain name to determine to which server it should forward the request. This option is typically used in a caching environment.
- **balance domainhash** - Internal CSS hash algorithm based on the domain string. The CSS parses the host tag and does an exclusive XOR hash across the entire host name. It then uses the XOR hash value to determine to which server to forward the request. This method guarantees that all requests with the same host tag will be sent to the same server in order to increase the probability of a cache hit. This option is typically used in a caching environment.

**Note**

When you configure a content rule with **no persistent** command, the global **persistent reset remap** command, and the **domainhash** load-balancing method, the CSS remaps a server when a subsequent HTTP GET on an HTTP 1.1 connection causes a different hash value than the previous GET.

---

- **balance leastconn** - Least connection algorithm. This balance method chooses a running service that has the fewest number of connections.

When configuring this method, you can optionally configure the slow-start feature that slowly increases the number of connections that a new service receives. This feature prevents the service from being flooded with connections. You can configure the slow-start timer to determine how long the service remains in the slow-start process. You can also adjust the rate that the service receives connections during the slow-start process. For more information on configuring the slow-start feature, see the “[Slowly Starting Connections on a Service](#)” section.

We do not recommend that you use UDP content rules with the leastconn load-balancing algorithm. The service connection counters do not increment and remain at 0 because UDP is a connectionless protocol. Because the counters remain at 0, the CSS will give inconsistent results.

- **balance roundrobin** - Roundrobin algorithm (default). The CSS resolves the request by evenly distributing the load to resolve domain names among local and remote content domain sites.
- **balance srcip** - Source IP address division algorithm. The CSS directs all client requests coming from the same source IP address to the same service. This option is generally used in a caching configuration.
- **balance url** - URL division algorithm. The CSS divides the alphabet evenly across the number of caches. It then parses the URL for the first four characters located after the portion of the URL matched by the rule. For example, if the URL in a content rule is configured for `"/news/*"`, the CSS will balance on the first four characters following `"/news/"`. This option is typically used in a caching environment.
- **balance weightedrr** - Weighted roundrobin algorithm. The CSS uses roundrobin but weighs some services more heavily than others depending on the server's configured weight. All servers have a default weight of 1. To set a server weight, use the **add service weight** command in owner-content mode.
- **balance urlhash** - Internal CSS hash algorithm based on the URL string. The CSS parses the URL and performs an XOR hash across the URL. It then uses the XOR hash value to determine to which server to forward the request. This method guarantees that all requests for the same URL will be sent to the same server in order to increase the probability of a cache hit. This option is typically used in a caching environment.

**Note**

When you configure a content rule with **no persistent** command, the global **persistent reset remap** command, and the **urlhash** load-balancing method, the CSS remaps a server when a subsequent HTTP GET on an HTTP 1.1 connection causes a different hash value than the previous GET.

**Note**

A Layer 5 content rule supports the RFC-2518 HTTP CONNECT, GET, HEAD, POST, PUSH, and PUT methods. In addition, the CSS recognizes and forwards the following RFC-2518 extension methods directly to the destination server in a transparent caching environment but does not load balance them: OPTIONS, TRACE and PROPFIND, PROPPATCH, MKCOL, MOVE, LOCK, UNLOCK, COPY, DELETE. To configure the CSS to support the RFC-2518 extension methods, see the [“Configuring HTTP Method Parsing”](#) section.

For example, to specify **weightedrr** load balancing, enter:

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

To revert the balance type to the default of roundrobin, enter:

```
(config-owner-content[arrowpoint-rule1])# no balance
```

## Slowly Starting Connections on a Service

When you configure a content rule with the least connection (leastconn) load-balancing method, a service on the rule with the fewest connections receives the next request. If you activate a service on this rule, the service may become flooded with requests.

To prevent the flooding of connections on a newly activated or reactivated service, you can enable the slow-start feature on a content rule configured with the leastconn load-balancing method. Through this feature, you can configure:

- The rate that an activated service receives connections. The slow-start rate is applied globally to all leastconn content rules on the CSS. By default, the rate is enabled with a default value of 3. If you disable the rate by setting it to 0, the slow-start feature is disabled on all leastconn content rules configured on the CSS.



---

**Note** We recommend that you do not change the slow-start rate default value.

---

- The maximum time that the service remains in the slow-start process. The slow-start timer sets the number of seconds that the service on a leastconn content rule is in the slow-start process. The timer is applied to a leastconn content rule. By default, the timer is disabled. If the timer is disabled, the slow-start feature is disabled on the rule.

When you enable the slow-start feature on the CSS and a leastconn rule, a service on the rule enters the slow-start process when you:

- Add and activate a service to the rule
- Reactivate a service after suspending it on the rule
- Activate the rule

When you activate a rule, it starts to load balance the connections on its services. The service with the least number of connections is selected to enter the slow-start process.

When a rule has only two services, only one service can enter the slow-start process. When the rule has more than two services, a newly activated service can enter the slow-start process when one of the services is currently in the slow-start process and the other services are out of the slow-start process.

A service in the slow-start process slowly continues to receive connections until either the slow-start timer expires or its connections equal the number of connections on the other active services of the rule. Then, the service exits the slow-start process and starts receiving connections as it normally would.



---

**Note** When you enable the slow-start feature on a leastconn rule, avoid adding any of its services to other content rules using different load-balancing methods. The slow-start process on the service may not be observable. For optimal slow-start performance on a service, add the service to only one leastconn content rule.

---



---

**Note** The slow-start feature is not designed to be maintained across failover scenarios. However, the probability of failover during a slow-start process is extremely small.

---

For information on configuring the slow-start timer and rate, see the following sections:

- [Configuring the Slow-Start Timer](#)
- [Configuring the Slow-Start Rate](#)

## Configuring the Slow-Start Timer

By default, the slow-start timer is disabled on the `leastconn` content rule. The slow-start timer applies to content rules configured with the `leastconn` load-balancing method. The slow-start timer ignores all other load-balancing methods.

By setting the timer, you:

- Enable the slow-start feature on the rule
- Set the amount of time in seconds that an activated or reactivated service on the rule remains in the slow-start process.

When the slow-start timer times out for the service, the service exits the slow-start process.



### Note

---

When the connections for a slow-start service equal the number of connections for the other active services on the rule, the service exits the slow-start process even when time remains on the slow-start timer. If the slow-start rate is set to 0, the CSS disables the slow-start feature for all `leastconn` rules on the CSS. For information on the slow-start rate, see the [“Configuring the Slow-Start Rate”](#) section.

---

To configure the slow-start timer, use the **`leastconn-slow-start`** command in content-rule configuration mode. The syntax for this command is:

**`leastconn-slow-start`** *seconds*

The *seconds* variable is the time in seconds that the service remains in the slow-start process. Enter a number from 0 to 65535. By default, this variable is set to 0 which disables the slow-start feature on the content rule.

**Note**

---

If the rule is active, you must suspend the rule before configuring the slow-start timer.

---

For example, to configure the slow-start timer with a value of 25 seconds, enter:

```
(config-owner-content[arrowpoint-rule1])# leastconn-slow-start 25
```

To disable the slow-start timer on the rule, enter:

```
(config-owner-content[arrowpoint-rule1])# no leastconn-slow-start
```

## Configuring the Slow-Start Rate

By default, the slow-start rate has a value of 3. By increasing or decreasing this value, you can change the rate that a service receives connections during the slow-start process. When you decrease the rate value, the service receives connections at a slower rate. When you increase the rate value, the service receives connections at a faster rate.

**Note**

---

We recommend that you do not change the slow-start rate default value without adequate testing in the intended network environment.

---

When the connections for the slow-start service equals the connections for the other services on the rule, the service exits the slow-start process even when time remains on the slow-start service.

**Note**

---

Disabling the slow-start rate disables the slow-start feature on all leastconn content rules even when the slow-start timer is set on a content rule. When the slow-start timer times out for the service, the service exits the slow-start process even though its connections does not equal the number of connections for the other services on the rule.

---

To configure the slow-start rate, use the **slowstart rate** command in global configuration mode. The syntax for this command is:

```
slowstart rate value
```

The *value* argument is the rate value to determine the rate that a service receives connections. Enter a number from 0 to 10. The default value is 3. Decreasing the value slows the rate that a service receives connections. Increasing the value quickens the rate that a service receives connections. A value of 0 disables the slow-start feature on all leastconn content rules configured on the CSS.

For example, to change the slow-start rate to 5, enter:

```
(config)# slowstart rate 5
```

To disable the slow-start rate, enter:

```
(config)# slowstart rate 0
```

To reset the default rate value of 3, enter:

```
(config)# no slowstart rate
```

## Displaying the Slow-Start Timer

To view the current slow-start timer setting and the slow-start time remaining for each service in a leastconn content rule, use the **show rule owner rule all** command. The fields that are displayed in the **show** output when the slow-start timer is configured on the content rule. The output displays the following slow-start fields:

- Slow Start Timer - Displays the configured slow-start timer setting for the content rule.
- SlowStart - Displays the time remaining before each service exits the slow-start process. If a service is not in the slow-start process, this field displays Out of SS.

You can also view the slow-start time remaining for each service on a content rule by using the **show rule owner rule services** command.



### Note

If you previously created automated scripts to parse output of the **show rule owner rule all | services** command, you must modify your script to accommodate the enabling of the slow-start feature on a leastconn content rule. Otherwise, the script may fail when it tries to locate an specific entry at a specific column of the output.

For information on all fields displayed through the **show rule** command, see the [“Showing Content Rules”](#) section.

## Configuring a DNS Balance Type

To determine where to resolve a request for a domain name into an IP address, use the **dnsbalance** command. The syntax and options for this content mode command are:

- **dnsbalance preferlocal** - Resolve the request to a local VIP address. If all local systems exceed their load threshold, the CSS chooses the least-loaded remote system VIP address as the resolved address for the domain name.
- **dnsbalance roundrobin** - Resolve the request by evenly distributing the load to resolve domain names among local and remote content domain sites. The CSS does not include sites that exceed their local load threshold.
- **dnsbalance leastloaded** - Resolve the request to the least-loaded of all local or remote domain sites. The CSS first compares load numbers. If the load number between domain sites is within 50, then the CSS compares their response times. The site with the fastest response time is considered the least-loaded site.
- **dnsbalance useownerdnsbalance** - Resolve the request by using the DNS load-balancing method assigned to the owner. This is the default method for the content rule. If you do not configure an owner method, the CSS uses the default owner DNS load-balancing method of roundrobin. To configure a DNS balancing method for an owner, see [Chapter 9, Configuring Owners](#).

For example, enter:

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

To restore the DNS balance type to the default setting of using the owner’s method, enter:

```
(config-owner-content[arrowpoint-rule1])# no dnsbalance
```

# Configuring Hot Lists

The CSS enables you to configure hot-list attributes for content rules. Defining hot-list attributes for a content rule enables you to determine which content is heavily accessed. With this information, you can accurately determine which content should be replicated. Use the **hotlist** command to define a hot list that lists the content most requested (hot content) during a user-defined period of time.

**Note**

---

You must configure and enable a hot list for replication-store and replication-cache to work.

---

You can configure the following attributes for hot lists for specific content from config-owner-content mode:

- **hotlist** - Enable the hot list. To enable a hot list for a specific content rule, use the **hotlist** command from the corresponding owner-content mode. For example, enter:

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

To disable a hot list, enter:

```
(config-owner-content[arrowpoint-rule1])# no hotlist
```

- **hotlist interval** - Set the hot-list refresh interval. Enter the interval time in minutes from 1 to 60. The default is 1. For example, enter:

```
(config-owner-content[arrowpoint-rule1])# hotlist interval 10
```

To restore the hot-list interval to the default of 1, enter:

```
(config-owner-content[arrowpoint-rule1])# no hotlist interval
```

- **hotlist size** - Set the size of the hot list. Enter the total number of entries maintained for this rule from 1 to 100. The default is 10. For example, enter:

```
(config-owner-content[arrowpoint-rule1])# hotlist size 10
```

To restore the hot-list size to the default of 10, enter:

```
(config-owner-content[arrowpoint-rule1])# no hotlist size
```

- **hotlist threshold** - Set the hot-list threshold. Enter an integer from 0 to 65535 to specify the threshold above which a piece of content is considered hot. The default is 0. For example, enter:

```
(config-owner-content[arrowpoint-rule1])# hotlist threshold 9
```

To restore the hot-list threshold default of 0, enter:

```
(config-owner-content[arrowpoint-rule1])# no hotlist threshold
```

- **hotlist hitcount** - Set the hot-list type to hit count, which is the number of times the content was accessed. For example, enter:

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

To restore the hot-list type to the default setting **hitcount**, enter:

```
(config-owner-content[arrowpoint-rule1])# no hotlist type
```

To display hot-list information, use the **show domain hotlist** command.

[Table 10-4](#) describes the fields in the **show domain hotlist** command output.

**Table 10-4 Field Descriptions for the show domain hotlist Command Output**

Field	Description
Hotlist Enabled/Disabled	Enable the domain hot list. The domain hot list is disabled by default.
Size	The configured maximum number of domain entries contained in the hot list. The range is from 1 to 100. The default is 10.
Interval	The configured interval, in minutes, to refresh the domain hot list and start a new list. The interval range is from 1 to 60. The default is 1.
Threshold	The configured number of domain hits per interval, which must be exceeded for a domain to be considered hot and added to the list. The threshold range is from 0 to 65535. The default is 0, which indicates that the threshold is disabled.
# Hot Domains	The total number of hot domains.
Hits	The number of hits for a hot domain.
Domain	The name of the hot domain associated with the Hits field.

## Configuring a Domain Hotlist

A domain hot list lists the most accessed domains on a CSS during a user-defined period of time. Use the **domain** command to enable the domain hot list and configure domain hot-list parameters. The syntax and options are:

- **domain hotlist** - Enable the domain hot list. The domain hotlist is disabled by default.
- **domain hotlist interval** *minutes* - Configure the interval to refresh the domain hot list and start a new list. Enter the interval from 1 to 60 minutes. The default is 1 minute.
- **domain hotlist size** *max\_entries* - Configure the maximum number of domain entries contained in the hot list. Enter the maximum number of entries from 1 to 100. The default is 10 entries.
- **domain hotlist threshold** *number* - Configure the threshold, which is the number of domain accesses per interval that must be exceeded for a domain to be considered hot and added to the list. Enter the threshold from 0 to 65535. The default is 0, which disables the threshold.

To enable a domain hot list, enter:

```
(config)# domain hotlist
```

To disable the domain hot list, enter:

```
(config)# no domain hotlist
```

To display the domain hot list and its configuration, use the **show domain hotlist** command (see [Table 10-4](#)).

# Configuring HTTP Method Parsing

Before the CSS can compare the HTTP request to the content rules, it parses the Request packet into tokens. This parsing evaluates the Request-Line and the different header fields. The Request-line field in the packet helps to define how the packet should be parsed. The method token in the Request-line field determines how to interpret the Request-URI field.

After the CSS parses the information in the HTTP packet into tokens, it attempts to match the tokens against criteria of content rules that define how the CSS should load balance the request. The CSS compares the HTTP request tokens to active Layer 5 content rules. When a match occurs, the CSS forwards the packet according to the configured rule. If a Layer 5 rule match does not occur, then the CSS attempts to match the tokens to a Layer 4 rule, and then a Layer 3 rule.

By default, a Layer 5 content rule supports the HTTP CONNECT, GET, HEAD, POST, and PUT methods. Unless configured, the CSS recognizes and forwards the following HTTP methods directly to the destination server in a transparent caching environment but does not load balance them: OPTIONS, TRACE, PROPFIND, PROPPATCH, MKCOL, MOVE, LOCK, UNLOCK, COPY, and DELETE.

You can configure the CSS to extend its support for parsing and matching of all extension methods defined in RFC-2518, including RFC-2616. You can also configure user-defined methods. For more information, see the following sections:

- [Configuring RFC-2518 Extension Methods Parsing](#)
- [Configuring User-Defined Methods](#)
- [Showing the HTTP Method Parsing Configuration and Status](#)

**Note**

---

The CSS provides scripts for the configuration of RFC-2518 and custom methods required for Outlook Web Access (OWA). The **setup\_owa\_methods** script enables RFC-2518 methods and configures the POLL, SEARCH, SUBSCRIBE, BMOVE, BCOPY, BDELETE, and BPROPPATCH user-defined methods. The **remove\_owa\_methods** script disables the RFC-2518 methods and removes the OWA methods configured with the **setup\_owa\_methods** script.

---

## Configuring RFC-2518 Extension Methods Parsing

By default, the CSS supports the parsing of HTTP CONNECT, GET, HEAD, POST, and PUT methods. You can enable the CSS to support all RFC-2518 extension methods parsing and processing, including OPTIONS, TRACE, PROPFIND, PROPPATCH, MKCOL, MOVE, LOCK, UNLOCK, COPY, and DELETE.

When you enable the CSS to support all RFC-2518 methods, the CSS parses the Request-URI field in an attempt to match a Layer 5 rule. If the contents of the Request-URI field are not in a compliant format of an absolute URI or an absolute path, then the CSS tries to match the field to the next best wildcard (“/\*”) rule. If the match fails, the CSS attempts to match the Layer 4 rule, and then the Layer 3 rule.

To enable support for the extension methods defined RFC-2518, enter:

```
(config)# http-method parse RFC2518-methods
```

To disable support for the extension methods defined RFC-2518, enter:

```
(config)# no http-method parse RFC2518-methods
```

## Configuring User-Defined Methods

If you require parsing of a method for proprietary web applications, the CSS allows you to configure a maximum of 16 user-defined methods. To configure a user-defined method, use the **http-method parse user-defined-method** command. The syntax for this global configuration mode command is:

```
http-method parse user-defined-method METHOD_NAME {uri  
[wildcard|authority|url]}
```

The variables and options are:

- *METHOD\_NAME* - The name for the method that processes the Request-URI field. Enter the name as an unquoted alphanumeric text string with a minimum of 3 characters and a maximum of 15 characters, including the hyphen (-) and underscore (\_) characters. You must capitalize the alphabetic

characters. You cannot use control, tspecial, and space characters. The tspecial characters include “(”, “)”, “<”, “>”, “@”, “;”, “:”, “\”, “|”, “/”, “[”, “]”, “?”, “=”, “{”, “}”, space (SP), and horizontal tab (HT) characters.

The method name cannot conflict with currently supported HTTP methods defined in RFC-2616, extension methods defined in RFC-2518, or any user-defined methods.

- **uri** - (Optional) Allows you to identify the resource in the Request-URI field that is applied to the method. By default, the method processes a resource in an absoluteURI or absolute path format. The absoluteURI format is required when the request is to a proxy. The proxy either forwards the request or services it from a valid cache, and then returns a response. For example:

```
GET http://www.test3.org/pub/www/Test.html HTTP/1.1
```

The absolute path identifies a resource on an origin server or gateway. The absolute path of the URI is transmitted as the Request-URI, and the network location of the URI (authority) is transmitted in a Host header field.

```
GET /index.html HTTP/1.1
Host:www.test3.org
```

- **url** - Indicates that the Request-URI field contains an absoluteURI or absolute path format. This keyword is the default resource.
- **wildcard** - Indicates that the Request-URI field can contain a wildcard (\*) character, an absolute URI, or an absolute path. The wildcard character indicates that the request does not apply to a particular resource, but to the server itself, and is only allowed when the method used does not necessarily apply to a resource. For example:

```
OPTIONS * HTTP/1.1
```

When you configure a user-defined method with a wildcard URI, you must configure a Layer 5 rule (url “/\*”) with a header-field group that contains a request line with the method name for the CSS to match the user-defined method to the rule.

- **authority** - Indicates that the Request-URI field contains an authority format. The CONNECT method is the only method that uses the authority format. For example:

```
CONNECT server.cisco.com:80 HTTP/1.1
```

For example, to configure a user-defined method named TREE with a wildcard in the Request-URI field, enter.

```
(config)# http-method parse user-defined-method TREE uri wildcard
```

**Note**

To change the URI resource for an existing user-defined method, you must remove the method and reconfigure it.

To remove a user-defined method, use the **no http-method parse user-defined-method** command. For example, to remove the user-defined method, enter:

```
(config)# no http-method parse user-defined-method TREE
```

## Showing the HTTP Method Parsing Configuration and Status

To display the HTTP method parsing configuration and processing status on the CSS, use the **show http-methods** command. For example, enter:

```
(config)# show http-methods
```

[Table 10-5](#) describes the fields in the **show http-methods** command output.

**Table 10-5** Field Descriptions for the show http-methods Command Output

Field	Description
HTTP Method Processing Summary	
RFC-2518 Methods Full Processing Support:	<p>Processing state of RFC-2518 methods. The states are:</p> <ul style="list-style-type: none"> <li>enabled - The CSS processes all RFC-2518 methods and attempts to match them to a content rule.</li> <li>disabled - The CSS processes the RFC-2518 HTTP CONNECT, GET, HEAD, POST, and PUT methods. However, the CSS recognizes, but only forwards, the OPTIONS, TRACE, PROPFIND, PROPPATCH, MKCOL, MOVE, LOCK, UNLOCK, COPY, and DELETE methods to transparent caching applications. This behavior is the default setting.</li> </ul>
User-defined methods:	The number of user-defined methods.
Method Processing Detail:	
Method Name	The method name in the request line. The method name can be an RFC-2518, RFC-2616, or user-defined method.
URI Format:	<p>How the CSS processes the URI. The values are:</p> <ul style="list-style-type: none"> <li>wildcard - The wildcard character (*), absoluteURI format, or absolute path format is valid.</li> <li>url - The absoluteURI or absolute path format is valid.</li> <li>authority - The authority format is valid.</li> </ul>

**Table 10-5** Field Descriptions for the *show http-methods* Command Output

Field	Description
Defined-In	The document that defines the method and URI format. The values are: <ul style="list-style-type: none"><li data-bbox="686 370 1210 423">• RFC-2616 - RFC-2616 defined method and URI behavior</li><li data-bbox="686 444 1210 498">• RFC-2518 - RFC-2518 defined method and URI behavior</li><li data-bbox="686 519 1210 573">• Custom - User-defined method name and URI behavior</li></ul>
Hit Counter	The number of times that the CSS matched the method. To clear all of the hit counters, use the global configuration mode <b>http-method statistics clear</b> command.

# 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. Each path defined within the 252 URL character string cannot exceed a maximum of 32 characters. A URL path includes all characters between the two slashes (*//*).



### 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 10, Configuring Content Rules](#).

## Showing EQL Extensions and Descriptions

To display a list of existing EQLs names, use **eq1 ?** command.

For example, enter:

```
(config)# eq1 ?
```

To display the extensions configured for a specific EQL including any descriptions, use the **show eq1** command and the EQL name. For example, enter:

```
(config)# show eq1 graphics
```

[Table 10-6](#) describes the fields in the **show eq1** command output.

**Table 10-6** Field Descriptions for the show eq1 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 specify a URQL with subscriber services.

See the following sections to configure a URQL:

- [Creating a URQL](#)
- [Configuring a URL in a URQL](#)
- [Designating the Domain Name of URLs in a URQL](#)
- [Adding a URQL to a Content Rule](#)
- [Describing the URQL](#)
- [Activating a URQL](#)
- [Suspending a URQL](#)
- [URQL Configuration in a Startup-Config File](#)
- [Showing URQLs](#)

## URQL Quick Start

Use the quick-start procedure in [Table 10-7](#) to configure a URQL. Each step includes the CLI command required to complete the task. For a complete description of each feature, see the sections following this procedure.

**Table 10-7 URQL Configuration Quick Start**

---

### Task and Command Example

---

1. Create a URQL.

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

---

2. Optionally, describe the URQL.

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

---

**Table 10-7 URQL Configuration Quick Start (continued)****Task and Command Example**

3. Configure the URLs you want to group in the URQL:

- a. Specify the URL entry.

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

- b. Define the URL.

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

- c. Optionally, describe the URL

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

4. Designate the domain name of the URLs in a URQL. For example:

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

5. Add the URQL to a content rule using the owner-content **url** command.

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

The following running-configuration example shows the results of entering the commands in [Table 10-7](#).

```
!***** URQL *****
urql videos
  description "cooking streaming video"
  url 10
  url 10 url "/cooking/cookies.avi"
  url 10 description "making cookies"
  domain "www.arrowpoint.com"

!***** OWNER *****
owner chefsbest
  address "200 Beaver Brook Road, Boxborough, MA 01719"

content recipes
  vip address 192.1.1.100
  protocol tcp
  port 80
  url "urql videos"
  add service server1
  active
```

## 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, configure the URLs you want to group in the URQL. The following section describes how to complete this task.

## 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. Each path defined within the 252 URL character string cannot exceed a maximum of 32 characters. A URL path includes all characters between the two slashes (*//*). In addition, an extension after the "." character cannot exceed 7 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 specify both **url urql** and subscriber services within the same content rule.

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 10, 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 the [“Designating the Domain Name of URLs in a URQL”](#) section 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 excellencel  
    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 10-8 describes the fields in the **show urql** command output.

**Table 10-8 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 10-9 describes the additional fields when you display a specified URQL.

**Table 10-9 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
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

# Specifying a Uniform Resource Locator

To specify the Uniform Resource Locator (URL) for content and enable the CSS to access a remote service when a request for content matches the rule, use the **url** command. Enter the URL as a quoted text string with a maximum length of 252 characters. Each path defined within the 252 URL character string cannot exceed a maximum of 32 characters. A URL path includes all characters between the double slash (//) at the beginning of the host name and the single slash (/) at the end of the host name. In addition, an extension after the period character (.) cannot exceed 7 characters.

**Note**

If the CSS receives a URL request that has a path between the slashes(/) greater than 252 characters, the CSS ignores the request.

**Note**

Do not include the ? or # parameter character in the URL string. The CSS terminates the URL at these parameter characters.

Before you can change the URL for a content rule, you must remove the current URL first using the **no url** command.

**Note**

We strongly recommend that if you configure a **balance** or **advanced-balance** method on a content rule that requires the TCP protocol for Layer 5 (L5) spoofing, you should configure a default URL string, such as **url “/\*”**. The addition of the URL string forces the content rule to become an L5 rule and ensures L5 load balancing or stickiness. If you do not configure a default URL string, unexpected results can occur.

In the following configuration example, if you configure a Layer 3 (L3) content rule with an L5 balance method, the CSS performs L5 load balancing but will reject UDP packets.

```
content testing
vip address 192.168.128.131
add service s1
balance url
active
```

The **balance url** method is an L5 load-balancing method in which the CSS must spoof the connection and examine the HTTP GET content request to perform load balancing. The CSS rejects the UDP packet sent to this rule because a UDP connection cannot be L5. Though the CSS allows this rule configuration, its expected behavior would be more clear if you promote the rule to L5 by configuring the **url “/\*”** command.

In the next example, if you configure an L3 content rule with an L5 advanced-balance method, L5 stickiness will not work as expected.

```
content testing
vip address 192.168.128.131
add service s1
advanced-balance arrowpoint-cookie
active
```

The **advanced-balance arrowpoint-cookie** method causes the CSS to spoof the connection, however, the CSS still marks it as an L3 rule. Thus, the CSS does not insert the generated cookie and the rule defaults to L3 stickiness (sticky-srcip). You must configure a URL like **url “/\*”** to promote this rule to L5, ensuring that L5 stickiness works as expected.

The syntax and options for the **url** content mode command are:

- **url “/url\_name”** - Specify the URL for the content as a quoted text string with a maximum length of 252 characters. The *url\_name* is the URL for the content. Enter a quoted text string with a maximum length of 252 characters. You must place a slash (/) at the beginning of the URL (for example, “/announcements/prize.html”).

To specify a domain name, place two slashes (//) at the beginning of the URL. For example, “//www.arrowpoint.com/\*” allows the rule to match on HTTP traffic that contains the www.arrowpoint.com domain name in the HTTP host tag.

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-owner-content[arrowpoint-rule1])# url
"/www.arrowpoint.com:8080/*"
```

You can specify certain wildcard operations for wildcard matching. Use an asterisk (\*) to specify a wildcard match. You can specify a maximum of eight directories. Each directory name can be a maximum of 32 characters with a total maximum of 252 characters in the URL. You can specify only one wildcard per URL.

Examples of supported wildcards are:

- **/\*.html** - Matches all requests with the .html extension
- **/announcements/\*** - Matches all requests for files in the *announcements* directory
- **/announcements/\*.html** - Matches requests for files in the announcements directory that have .html extensions
- **/announcements/new/\*.jpg** - Matches requests for all files in the announcements/new directory that contain the .jpg extension
- **url “/url\_path/\*” eql eql\_name** - Specify the URL for any content file that has its file extension defined in the specified Extension Qualifier List (EQL). The *url\_path* is the path to any content file that has its file extension defined in the EQL. Enter a quoted text string. You must place:
  - A slash (/) at the beginning of the quoted path. For caching environments, you can configure a domain content rule by placing two slashes (//) at the front of the *url\_path*.
  - A slash and asterisk (/\*) at the end of the quoted path.

For example, “/announcements/new/\*”.

The *eql\_name* is the name of the EQL. To see a list of EQLs, use the **eql ?** command.

- **url “/url\_path/\*” dql dql\_name {eql\_name}** - Specify the URL for any content file that has its domain name defined in the specified Domain Qualifier List (DQL). You cannot use a DQL in conjunction with a domain name in a URL. You may include an EQL name after the DQL name to specify specific file extensions as part of the DQL matching criteria.

The *url\_path* variable is the path to any content file that has its domain defined in a DQL. Enter a quoted text string. You must place:

- A slash (/) at the beginning of the quoted path. For caching environments, you can configure a domain content rule by placing two slashes (//) at the front of the *url\_path*.
- Two slashes (//) at the beginning of the quoted path

The *dql\_name* variable is the name of the DQL. To see a list of DQLs, use the **dql ?** command.

- **url urql *urql\_name*** - Specify a URL qualifier list (URQL) consisting of a group of URLs to this content rule. Note that you cannot specify both **url urql** and **application ssl, application sip**, or subscriber services for the same content rule.

The *urql\_name* variable is the name of the URQL. You can assign only one URQL per rule. To see a list of URQLs, enter the **urql ?** command.



#### Note

For caching environments, you can configure a domain content rule by placing two slashes (//) at the front of the *url\_name* or *url\_path*. The rule matches HTTP traffic that contains the domain name in the HTTP host tag.

For example, to specify a URL that matches all requests for content in the announcements directory with .html extensions, enter:

```
(config-owner-content[arrowpoint-products.html])# url
"/announcements/*.html"
```

To remove a URL, enter:

```
(config-owner-content[arrowpoint-products.html])# no url
```

To remove a URQL from a URL, enter:

```
(config-owner-content[arrowpoint-products.html])# no url urql
```

To display a URL for a content rule, use the **show rule** command for the content rule.

## Specifying an Extension Qualifier List in a URL

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. For information on creating an EQL, see the [“Configuring Extension Qualifier Lists”](#) section.

Specify a URL as a quoted text string with a maximum of 252 characters followed by **eql** and the EQL name. Each path defined within the 252 URL character string cannot exceed a maximum of 32 characters. A URL path includes all characters between the two slashes (*//*).



### Note

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

For example, enter:

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

To display a content rule EQL, use the **show rule** command.

## Specifying the Number of Spanned Packets

In some environments, URLs, cookie strings, or HTTP header information can span multiple packets. In these environments, the CSS can parse up to 20 packets for Layer 5 information before making a load-balancing decision. By default, the CSS parses six packets.

The CSS makes the load-balancing decision as soon as it finds a match, and it does not require parsing of all the spanned packets. Because parsing multiple packets does impose a longer delay in connection, performance can be impacted by longer strings that span multiple packets.

Use the **spanning-packets** command to configure the number of packets spanned for the search of the HTTP header termination string. To change the number of packets, enter a number from 1 to 20. The default value is 6. For example, to configure the number of packets spanned to 10, enter:

```
(config)# spanning-packets 10
```

To reset the number of packets spanned to the default value of 6, enter:

```
(config)# no spanning-packets
```

## Specifying a Load Threshold

When the service load metric exceeds this threshold, the local service becomes unavailable and is redirected to remote services. To define a remote service, use the service mode **type redirect** command (see the “[Specifying a Service Type](#)” section in [Chapter 3, Configuring Services](#)).

Use the **load-threshold** command to set the normalized load threshold for the availability of each local service on a content rule. Enter the load threshold as an integer from 2 to 254. The default is 254, which is the maximum threshold a service can reach before becoming unavailable. To view the load on services, use **show service**. For example, enter:

```
(config-owner-content[arrowpoint-rule1])# load-threshold 100
```

To reset the load threshold to its default value of 254, enter:

```
(config-owner-content[arrowpoint-rule1])# no load-threshold
```

# Including Services in a CSS Ping Response Decision

By default, a CSS responds to a ping request to a Virtual IP (VIP) address configured on a content rule if any of the local services on the content rule are alive. To include remote services, for example services of type redirect, in the decision to respond to a ping request to the VIP address, use the **vip-ping-response local-remote** command. For example, enter:

```
(config-owner-content[arrowpoint-rule1])# vip-ping-response  
local-remote
```

To reset the CSS to its default behavior of including only local services in the ping response decision, enter:

```
(config-owner-content[arrowpoint-rule1])# vip-ping-response local
```

# Enabling TCP Flow Reset Reject

By default, the CSS disables the sending of the TCP RST frame to the client when a flow for requested content is mapped to a destination IP address that is no longer reachable. Use the **flow-reset-reject** command to enable the CSS flow manager subsystem to send a TCP RST (reset) frame. The **flow-reset-reject** command prevents a CSS client from hanging up and retransmitting when the request can never be serviced. The **flow-reset-reject** command is applied on a per-content rule basis.

To enable the CSS to send a TCP RST frame, enter:

```
(config-owner-content[rule1])# flow-reset-reject
```

To reset the CSS back to the default state of not sending a TCP RST frame, enter:

```
(config-owner-content[rule1])# no flow-reset-reject
```

# Configuring Persistence, Remapping, and Redirection

During the life of a persistent connection, a CSS must determine when it needs to move a client connection to a new service based on content rules, load balancing, and service availability. In some situations, moving the client connection is not necessary; in other situations, it is mandatory. This section describes how to configure the CSS to make these decisions using:

- Content rule persistence
- Bypass persistence
- HTTP redirection
- Service remapping

## Configuring Content Rule Persistence

When a CSS receives a request for content from a client, the software checks if the request matches a content rule to determine the best service to handle the request. If the request matches a content rule, the CSS establishes a client connection to the best service specified by the content rule. By default, the CSS keeps the client on the same connection for an entire flow session as long as a new content request:

- Matches the same content rule that specified the current service
- Matches a new content rule that contains the current service, even if a different best service is specified by the content rule

This CSS behavior is known as *content rule persistence*. If you are using transparent caches (which prefetch content) or mirrored-content servers, this scheme works well because the same content is available on each service.

Use the **persistent** command in content configuration mode to maintain a persistent connection with a server as long as the above criteria are met. By default, persistence is enabled. Disabling persistence allows the CSS to move a connection to a better service on the same rule or to use cache bypass functionality (EQLs or failover bypass).

For example, enter:

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

Use the **no persistent** command on a content rule with:

- A balance method of domain or domain hash when using proxy caches
- A balance method of url or urlhash when using transparent caches
- A failover method of bypass when using transparent caches
- An EQL bypass with a transparent cache
- Adding a sorry server to a content rule

**Note**

---

If you configure an ArrowPoint cookie on a content rule using the **advanced-balance arrowpoint-cookie** command and the CSS receives a subsequent GET with no ArrowPoint cookie on a persistent HTTP connection, the CSS ignores all persistence settings in the running-config, remaps the back-end connection to a new server, and inserts a new ArrowPoint cookie.

---

To disable persistence:

```
(config-owner-content [arrowpoint-rule1]) # no persistent
```

**Note**

---

If a request for content on a persistent connection matches a new content rule that does not contain the current service, or if persistence is disabled and there is a better service configured in the content rule, the CSS redirects or remaps the current connection to a new best service based on the setting of the **persistence reset** command, if configured. If you do not configure **persistence reset**, the CSS performs an HTTP redirect by default. For details on HTTP redirection, see the [“Configuring HTTP Redirection and Service Remapping”](#) section later in this chapter.

---

## Configuring Bypass Persistence

If a CSS bypasses a service (for example, a transparent cache is down and **failover bypass** is configured) and the next content request on the same TCP connection matches a content rule that contains the transparent cache that was down, the CSS will continue to bypass the cache, by default, even after the bypassed cache is back online. In this case, the CSS typically sends the content request to the origin server. This behavior is called *bypass persistence*.

You can configure the CSS to redirect or remap a bypassed connection using the **bypass persistence** global config command in conjunction with the **persistence reset** command.

Use the **bypass persistence** command to determine when the CSS performs either a remapping or redirection operation to reset a bypassed service when a content request matches on a content rule, but a previous request caused the bypass. This global command affects all flows. By default, bypass persistence is enabled.

For example, enter:

```
(config)# bypass persistence disable
```

The CSS uses remapping or redirection to reset the connection according to the setting of the **persistence reset** method.

```
(config)# bypass persistence enable
```

The CSS does not use remapping or redirection to reset the connection and continues to bypass a service.

## Configuring HTTP Redirection and Service Remapping

If you need to place different content on different servers (for example, to conserve server disk space, for load-balancing considerations, or when using proxy caches), content rule persistence is not useful. In this case, you can disable persistence by using the **no persistent** command, described in the [“Configuring Content Rule Persistence”](#) section earlier in this chapter.

When the CSS receives a request for content that is not available on the current service, it must reset the current connection to the service and establish a new connection to another service (for example, a different proxy cache or the origin server) that contains the requested content. You can accomplish this in either of the following ways:

- **Redirection** - An HTTP technique that resets both the client-to-CSS (front-end) connection and the CSS-to-service (back-end) connection, and then establishes a new flow to the best service that contains the requested content.
- **Service Remapping** - A technique that resets only the back-end connection to the current service and then creates a new back-end connection to the best service that contains the requested content. This technique is faster and more efficient than redirection because the CSS does not need to reset and then reestablish the front-end connection. With service remapping, the CSS strictly manages portmapping to prevent the occurrence of duplicate port numbers.

**Note**

Service remapping is incompatible with stateless redundancy failover (the **redundancy-l4-stateless** command). Service remapping enables CSS portmapping, which source-port NATs all flows. Stateless redundancy failover requires that the CSS not NAT source ports. For more information on stateless redundancy failover, refer to the *Cisco Content Services Switch Redundancy Configuration Guide*.

Use the **persistence reset** global configuration mode command with the **no persistent** content rule command to cause an HTTP redirection or perform a back-end remapping operation when resetting a connection to a new back-end service. The global **persistence reset** command affects all flow setups that require redirection or remapping.

**Note**

When you configure a content rule with **no persistent** command, the global **persistent reset remap** command, and the **urlhash** or **domainhash** load-balancing methods, the CSS remaps a server when a subsequent HTTP GET on an HTTP 1.1 connection causes a different hash value than the previous GET.

For example, to enable redirection:

```
(config)# persistence reset redirect
```

For example, to enable service remapping:

```
(config)# persistence reset remap
```



#### Note

The CSS does not use remapping when selecting redirect type services. See the “[Specifying a Service Type](#)” section in [Chapter 3, Configuring Services](#).

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.



#### Note

If you configure an ArrowPoint cookie on a content rule using the **advanced-balance arrowpoint-cookie** command and the CSS receives a subsequent GET with no ArrowPoint cookie on a persistent HTTP connection, the CSS ignores all persistence settings in the running-config, remaps the back-end connection to a new server, and inserts a new ArrowPoint cookie.

## Redirecting Requests for Content

Use the **redirect** command to set HTTP status code 302 (object moved) for a content rule and specify the alternate location of the content governed by a rule. Use this command to:

- Make the content unavailable to subsequent requests at its current address.
- Provide a URL to send back to the requestor. You must add a URL to the content rule for **redirect** to force the HTTP request. For example, url “/\*”. Enter the URL as a quoted text string with no spaces and a maximum of 252 characters.



#### Note

If you also set status code 404 (drop message) for content, code 302 takes priority.

Do not configure a service for a redirect-only content rule.

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# redirect  
"//www.arrowpoint.com/newlocation.html"
```

To delete the redirect URL, enter:

```
(config-owner-content[arrowpoint-rule1])# no redirect
```

## Setting TCP FIN or RST Flags for HTTP 302 Redirect Messages

By default, when the CSS sends an HTTP 302 redirect message, it sends a FIN flag on an initial connection and RST flags on subsequent requests in a persistent connection. When the CSS sends packets to a client that contain redirect messages to a Microsoft IE browser, the default flag settings may not be suitable for the browser. Use the global configuration **http-redirect-option** command to determine the TCP FIN or RST flags sent with the redirect messages. The syntax for this command is:

```
http-redirect-option [fin-rst|fin-fin|rst-rst]
```

The keywords are:

- **fin-rst** - The default behavior, the CSS sends a FIN flag on an initial connection and an RST flag on subsequent request in a persistent connection
- **fin-fin** - The CSS always sends FIN flags
- **rst-rst** - The CSS always sends RST flags

For example, to configure the CSS to always send FIN flags with 302 redirect messages, enter:

```
(config)# http-redirect-option fin-fin
```

To reset the default behavior, enter:

```
(config)# http-redirect-option fin-rst
```

To display the flag settings for the HTTP redirect messages, use the **show http-redirect-option** command. For example, enter:

```
(config)# show http-redirect-option  
HTTP Redirect Option: fin-rst
```

## Displaying the Persistence Settings

Use the **show remap** command to display the configured **persistence reset** and **bypass persistence** settings. This command is available in all modes except RMON, URQL, and VLAN configuration modes.

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

**Table 10-10 Field Descriptions for the show remap Command Output**

Field	Description
Group SFP Port Map Info	This field is currently not used.
Persistence Reset Method	<p>The configured persistence reset method when resetting a connection to a new back-end service. The possible methods are:</p> <ul style="list-style-type: none"> <li>• <b>redirect</b> - Causes an HTTP redirection when resetting a connection to a new back-end service. An HTTP redirection resets both sides of the connection.</li> <li>• <b>remap</b> - Uses a back-end remapping operation when resetting a connection to a new back-end service.</li> </ul>
Bypass Persistence	<p>The configured bypass persistence setting. The possible settings are:</p> <ul style="list-style-type: none"> <li>• <b>disable</b> - The CSS performs either a service remapping or HTTP redirection operation to reset a bypassed service when a content request matches a content rule, but a previous request caused the bypass.</li> <li>• <b>enable</b> - The CSS does not perform remapping or redirection to reset the connection and continues to bypass a service. By default, bypass persistence is enabled.</li> </ul>

# Defining Failover

**Note**

---

The CSS supports Adaptive Session Redundancy (ASR) on Cisco 11500 series CSS peers in an active-backup VIP redundancy and virtual interface redundancy environment to provide stateful failover of existing flows. For details on ASR, refer to the *Cisco Content Services Switch Global Server Load-Balancing Configuration Guide*.

---

To define how the CSS handles content requests when a service fails or is suspended, use the **failover** command. For the CSS to use this setting, ensure that you configure a keepalive for each service; that is, do not set the keepalive type to **none** (the keepalive default is ICMP). The CSS uses the keepalive settings to monitor the services to determine server health and availability.

The failover command applies to the following caching load-balancing types:

- **balance domain**
- **balance url**
- **balance srcip**
- **balance destip**
- **balance domainhash**
- **balance urlhash**

**Note**

---

If you remove a service (using the **remove service** command), the CSS rebalances the remaining services. The CSS does not use the failover setting.

---

This command supports the following options:

- **failover bypass** - Bypass all failed services and send the content request directly to the origin server. This option is used in a proxy or transparent cache environment when you want to bypass the failed cache and send the content request directly to the server that contains the content.
- **failover linear** (default) - Distribute the content request evenly between the remaining services.

- **failover next** - Send the content requests to the cache service next to the failed service. The CSS selects the service to redirect content requests to by referring to the order in which you configured the services.

For example, enter:

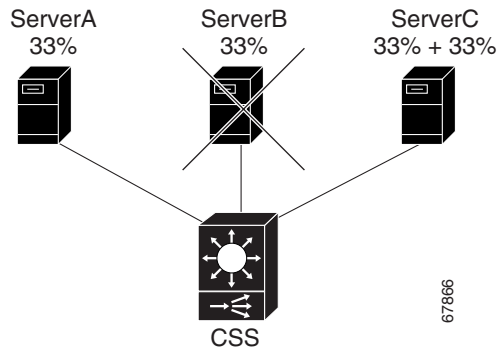
```
(config-owner-content[arrowpoint-rule1])# failover bypass
```

To restore the default setting of **failover linear**, enter:

```
(config-owner-content[arrowpoint-rule1])# no failover
```

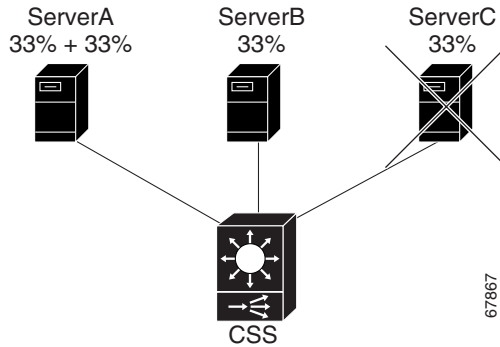
Figure 10-2 shows three cache services configured for failover **next**. If ServerB fails, the CSS sends ServerB content requests to ServerC, which was configured after ServerB in the content rule.

**Figure 10-2 ServerB Configured for Failover Next**



As shown in [Figure 10-3](#), if ServerC fails, the CSS sends ServerC content requests to ServerA because no other services were configured after ServerC.

**Figure 10-3 ServerC Configured for Failover Next**



[Figure 10-4](#) shows three cache services configured for failover **linear**. If you suspend ServerB or if it fails, the CSS does not rebalance the services. It evenly distribute ServerB cache workload between servers A and C.

Note that [Figure 10-4](#) and [Figure 10-5](#) use the alphabet to illustrate division balance.

**Figure 10-4 Suspended or Failed Service Configured for Failover Linear**

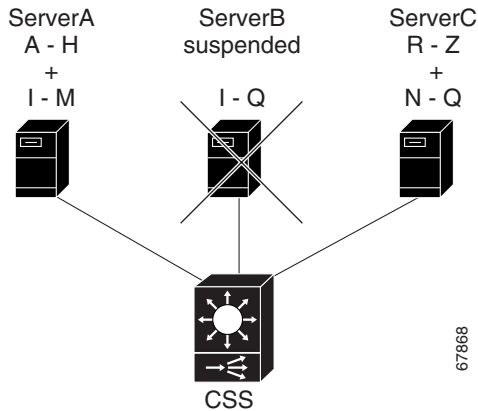
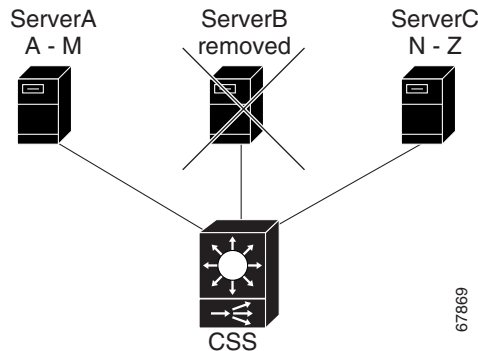


Figure 10-5 also shows three cache services configured for failover **linear**, but in this example, you *remove* ServerB using the **remove service** command from owner-content mode. Because the CSS does not apply the failover setting when you remove a service, it rebalances the remaining services.

**Figure 10-5 Removing a Service Configured for Failover Linear**



## Specifying an Application Type

The application type enables the CSS to correctly interpret the data stream to match and parse the content rule. If you do not specify an application type, the CSS rejects the data stream packets. Always define an application type for nonstandard ports. To specify the application type associated with a content rule, use the **application** command.

When configuring Layer 5 content rules for an application other than HTTP, use the appropriate **application** type to enable the Layer 5 rule to function.

A Layer 5 content rule supports the RFC-2518 HTTP CONNECT, GET, HEAD, POST, PUSH, and PUT methods. In addition, the CSS recognizes and forwards the following RFC-2518 extension methods directly to the destination server in a transparent caching environment but does not load balance them: OPTIONS, TRACE and PROPFIND, PROPPATCH, MKCOL, MOVE, LOCK, UNLOCK, COPY, DELETE. To configure the CSS to support the RFC-2518 extension methods, see the [“Configuring HTTP Method Parsing”](#) section.

The **application** command enables you to specify the following application types:

- **bypass** - Bypass the matching of a content rule and send the request directly to the origin server.
- **ftp-control** - Process FTP data streams. For information on configuring a content rule for FTP connections and allowing the use of a non-standard FTP control port, see the “[Configuring a Content Rule for FTP Connections](#)” and “[Configuring the CSS to Handle Non-Standard FTP Ports](#)” sections.
- **http** (default) - Process HTTP data streams.
- **ssl** - Process Secure Sockets Layer (SSL) protocol data streams.
- **sip** - Process Session Initiation Protocol (SIP) UDP control packets. When you type **application sip** at the CLI, the CSS automatically enters the protocol as UDP and the port number as 5060 in the running-configuration file if you have not previously configured a protocol and a port.

**Note**

---

You cannot configure both **url urql** and **application ssl** or **application sip** for the same content rule.

---

Always configure the **ssl** application type with the **ssl** advanced load-balancing method. It is important that you configure both the **application** command and **advanced-balance** command together to ensure that the CSS properly interprets the SSL session ID and sticks the client to a server based on the ID. For details, see the “[Specifying an Advanced Load-Balancing Method for Sticky Content](#)” section in [Chapter 11, Configuring Sticky Parameters for Content Rules](#).

To remove an application type, enter:

```
(config-owner-content[arrowpoint-rule1])# no application
```

## Configuring a Content Rule for FTP Connections

If clients are connecting through Port (active) mode FTP, you need to configure the content rule with application type **ftp-control**. This application type instructs the CSS to process only FTP requests coming into the specified port.

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

The following example shows the portion of a running-config for content rule `ftp_rule`. In this content rule, the CSS process FTP requests on port 21.

```
!***** OWNER *****
owner arrowpoint
  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
```

You must also configure a source group because the control channel is a new flow initiated by the server. Configure the source group with the same VIP address as the content rule.



### Note

When you configure the CSS to support passive FTP on a non-standard FTP control port, the CSS inspects the PASV 227 server response payload in order to NAT the embedded server IP address and server TCP port number. If you configure the CSS to perform this NAT through an ACL clause with a preferred source group, then you must configure the ACL clause to match on the FTP control port. The CSS does not perform ACL clause matching based on the embedded FTP PASV payload IP address or TCP port number.

For more information on configuring a source group for FTP connections, see the [“Configuring a Source Group for FTP Connections”](#) section in [Chapter 5, Configuring Source Groups for Services](#).

See the following sections for:

- [Configuring the Teardown Timeout Period for the FTP Control Channel](#)
- [Configuring the Wait Time to Initiate the FTP Data Channel](#)

## Configuring the Teardown Timeout Period for the FTP Control Channel

The CSS tears down the FTP control channel after 10 minutes of idle time. This teardown may occur during a file transfer if the transfer exceeds 10 minutes. The idle timeout applies only to active FTP; it does not apply to PASV FTP.

To configure the timeout to a value that can accommodate the expected duration of FTP file transfers, use the owner-content **flow-timeout-multiplier** command on the associated content rule. This command specifies a value that the CSS uses to derive the number of seconds for which an idle flow can exist before the CSS tears it down. The CSS multiplies the value you specify by 16 to calculate the flow timeout in seconds. Enter an integer for the number variable from 0 to 65533.

For example, to configure a flow timeout period of 16 minutes (960 seconds), enter:

```
(config-owner-content[cisco-rule1])# flow-timeout-multiplier 60
```

## Configuring the Wait Time to Initiate the FTP Data Channel

By default, the CSS waits 5 seconds to initiate the FTP data channel on an active or passive FTP connection for CSS FTP content rules and source groups. You can globally configure the time to wait to initiate the FTP data channel on an active or passive FTP connection. To configure this wait time, use the **ftp data-channel-timeout** command in global configuration mode. The syntax for this command is:

```
ftp data-channel-timeout seconds
```

The *seconds* variable is the wait time in seconds. Enter a number from 5 to 120. For example, to configure a wait time of 10 seconds, enter:

```
(config)# ftp data-channel-timeout 10
```

To reset the default wait time to 5 seconds, use the **no ftp data-channel-timeout** command. For example, enter:

```
(config)# no ftp data-channel-timeout
```

## Configuring the CSS to Handle Non-Standard FTP Ports

By default, the CSS requires the FTP connection to use the standard FTP control port of 21 and data port of 20. The CSS preserves and does not NAT the FTP data port when the FTP data connection is passed through the CSS.

Through the **ftp non-standards-ports** command, you can configure the CSS to allow the FTP control connection to use a non-standard port, not port 21. The CSS does not preserve the FTP data port when the FTP data connection is passed through the CSS.

For example, enter:

```
(config)# ftp non-standard-ports
```



---

**Note**

When you use the **ftp non-standards-ports** command to allow the use of non-standard FTP ports and a content rule is using FTP, you must configure the **application ftp-control** command on the content rule.

---



---

**Note**

When you configure the CSS to support passive FTP on a non-standard FTP control port, the CSS inspects the PASV 227 server response payload in order to NAT the embedded server IP address and server TCP port number. If you configure the CSS to perform this NAT through an ACL clause with a preferred source group, then you must configure the ACL clause to match on the FTP control port. The CSS does not perform ACL clause matching based on the embedded FTP PASV payload IP address or TCP port number.

---

To reset the default behavior of requiring the FTP connection to use the standard control port of 21, use the **no** form of this command. For example, enter:

```
(config)# no ftp non-standard-ports
```

# Enabling Content Requests to Bypass Transparent Caches

The "#" and "?" terminators indicate that the content is dependent on the arguments that follow the terminators. Because the content returned by the server is dependent on the content request itself, the returned content is deemed not cacheable, and the content request is directed to the origin server.

Use the **param-bypass** command to enable content requests to bypass transparent caches when the CSS detects special terminators in the requests. This command contains the following options:

- **param-bypass disable** (default) - Content requests with special terminators do not bypass transparent caches.
- **param-bypass enable** - Content requests with special terminators bypass transparent caches and are forwarded to the origin server.

For example, to enable the **param-bypass** command, enter:

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

## Showing Content

The **show content** command enables you to display content entries in the Content Service Database (CSD) of the CSS. This command is available in all modes.

To display content from a specific module, and content entry location, in either the CSS 11503 or CSS 11506, specify the **show content** command as follows:

```
show content slot_number {start-index index_number}
```

The variables and options are:

- *slot\_number* - Display content from the module located in a specific slot in the CSS 11503 or CSS 11506 chassis. For the CSS 11503, the available choices are 1 through 3. For the CSS 11506, the available choices are 1 through 6. If you do not specify a slot number, the CSS displays the content entries from the SCM in slot 1 of the CSS.

- **start-index** *index\_number* - Display content entries starting at the specified *index\_number* parameter. This variable defines where you want to start browsing CSS content. Starting from the specified index number, you receive up to a maximum of 64K of information. To see additional information, issue the **show content** command again, starting from the last index number displayed. To specify an index number, enter a number from 0 to 4095. If you do not specify a start-index the CSS displays the content entries starting from 0.

Use the **show content** command with no options or variables to show all content entries in the Content Service Database for a CSS 11501, CSS 11503, or CSS 11506.

For example, to look at the content from the module in CSS 11503 chassis slot 2, starting at index 150, enter:

```
(config)# show content slot 2 start-index 150
```

Table 10-11 describes the fields in the **show content** command output.


**Note**

URQL entries are flagged with an asterisk (\*) in the **show content** command output.

**Table 10-11 Field Descriptions for the show content Command Output**

Field	Description
Pieces of Content for Slot	The chassis slot number in which the module resides.
Subslot	The module slot number in which the Session Processor resides.
Total Content	The total number of content entries.
Index	Unique index for known content in the CSD.
<address>	The IP address of the content.
Protocol	The IP Protocol of the content.
Port	Protocol port of the content.

**Table 10-11 Field Descriptions for the show content Command Output**

Field	Description
Best Effort	The QoS class of the content. This field is not used by the CSS at this time.
Streamed	Identifies whether the piece of content is streaming media (video or audio). This field is not used by the CSS at this time.
URL	The Universal Resource Locator of the content.
Domain	The domain name of the content.

## Showing Content Rules

The **show rule** command displays content rule information for specific content rules or all content rules currently configured in the CSS. When using the **show rule** command in content configuration mode, the CSS displays only information for the current rule. You cannot enter the owner and content rule name for another content rule.

Use the following **show rule** commands from any User, SuperUser, global configuration, owner, and content mode:

**Note**

The owner and content rule variables shown in the following commands are not available in content configuration mode.

- **show rule** - Display all owners and content rules currently configured in the CSS
- **show rule-summary** - Display a summary of owner content information
- **show rule** *owner\_name* - Display information identical to the show rule command, but only for the specified owner's content
- **show rule** *owner\_name content\_rule\_name* - Display information identical to the show rule command, but only for a specific owner and content
- **show rule** *owner\_name content\_rule\_name acl* - Display the ACL attributes for the specified content rule

- **show rule** *owner\_name content\_rule\_name* **all** - Display all attributes for the specified content rule
- **show rule** *owner\_name content\_rule\_name* **dns** - Display the DNS attributes for the specified content rule
- **show rule** *owner\_name content\_rule\_name* **header-field** - Display the header-field attributes for the specified content rule
- **show rule** *owner\_name content\_rule\_name* **hot-list** - Display the hot-list attributes for the specified content rule
- **show rule** *owner\_name content\_rule\_name* **services** - Display the services for the specified content rule
- **show rule** *owner\_name content\_rule\_name* **statistics** - Display the statistics for the specified content rule
- **show rule** *owner\_name content\_rule\_name* **sticky** - Display the sticky attributes for the specified content rule

**Note**


---

If you previously created automated scripts to parse output of the **show rule owner rule all | services** command, you must modify your script to accommodate the enabling of the slow-start feature on a leastconn content rule. Otherwise, the script may fail when it tries to locate an specific entry at a specific column of the output.

---

To display all content rule information, enter:

```
# show rule
```

To display the summary for all content rules, enter:

```
# show rule-summary
```

To display all rule attributes for an owner, enter:

```
# show rule owner content_rule all
```

**Note**


---

The CntRuleName and OwnerName fields display the first 16 characters of the configured data. The URL field displays the first 10 characters of configured data.

---

Table 10-12 describes the fields in the **show rule** command output.

**Table 10-12 Field Descriptions for the show rule Command Output**

Field	Description
Name	The name of the content rule.
Owner	The owner of the rule.
Author	The author (Local CSS or remote CSS peer) of the rule.
Index	A CSS assigned unique index for the rule. The number is based in the order that the rule was created.
State	The state of the rule (active or suspend).
Type	The application type associated with the rule. The possible values are: <ul style="list-style-type: none"><li>• <b>bypass</b> - Bypasses the matching of the content rule and sends the request directly to the origin server</li><li>• <b>http</b> - Processes HTTP data streams (default)</li><li>• <b>ftp-control</b> - Processes FTP data streams</li><li>• <b>ssl</b> - Processes Secure Sockets Layer (SSL) protocol data streams</li></ul>


**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
Balance	<p>The load-balancing algorithm for the content rule. The possible values are:</p> <ul style="list-style-type: none"> <li>• <b>ACA</b> - ArrowPoint Content Awareness algorithm. The CSS correlates content request frequency with the server's cache sizes to improve cache hit rates for that server.</li> <li>• <b>destip</b> - Destination IP address division. The CSS directs all client requests with the same destination IP address to the same service.</li> <li>• <b>domain</b> - Domain name division. The CSS uses the domain name in the request URI to direct the client request to the appropriate service.</li> <li>• <b>domainhash</b> - Internal CSS hash algorithm based on the domain string. The CSS uses the algorithm to hash the entire domain string. Then, the CSS uses the hash result to choose the server.</li> <li>• <b>leastconn</b> - Least connections. The CSS chooses a running service that has the least number of connections.</li> <li>• <b>roundrobin</b> - Roundrobin algorithm (default).</li> <li>• <b>srcip</b> - Source IP address division. The CSS directs all client requests with the same source IP address to the same service.</li> <li>• <b>url</b> - URL division. The CSS uses the URL (omitting the leading slash) in the redirect URL to direct the client requests to the appropriate service.</li> </ul>

**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
Balance (continued)	<ul style="list-style-type: none"> <li>• <b>urlhash</b> - Internal CSS hash algorithm based on the URL string. The CSS uses the algorithm to hash the entire URL string. Then, the CSS uses the hash result to choose the server.</li> <li>• <b>weightedrr</b> - Weighted roundrobin algorithm. The CSS uses the roundrobin algorithm but weighs some services more heavily than others. You can configure the weight of a service when you add it to the rule.</li> </ul>
Failover	The type of failover configured for the rule.
Slow Start Timer	The configured value of the slow-start timer for the content rule. This field appears only when the slow-start timer is configured on a rule with the leastconn load-balancing method.
Persistence	Indicates whether or not a persistent connection with a server is maintained. By default, persistence is enabled.
Param-Bypass	Indicates whether or not content requests bypass transparent caches when the CSS detects special terminators in the requests. These “#” and “?” terminators indicate that the content is dependent on the arguments that follow the terminators. Bypass is disabled by default.
IP Redundancy	The state of IP redundancy if configured on the rule. Possible values are Master, Backup, or Down. If IP redundancy is not configured, the state is Not Redundant.
L3	Destination IP address.
L4	Destination protocol and port.
URL	The URL for the content.
URQL	The name of the associated URL Qualifier list.
EQL	The name of the associated EQL.

**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
DQL	The name of the associated DQL.
Header Field Group	The name of the associated header-field group.
Total Bytes	Total bytes to the content rule.
Total Frames	Total frames to the content rule.
Total Redirects	Total redirects by the content rule (when the <b>redirect</b> command is configured for a content rule). This field increments whenever a request for content is redirected to an alternate location.
Total Rejects	Total rejects by the content rule. This field increments when all services for a content rule are unavailable.
Overload Rejects	Total rejects on the content rule due to overload on the rule's available services.
Balance	The load-balancing algorithm for the content rule. The entry in this field is the same as previous Balance field.
Advanced Balance	<p>The advanced load-balancing method for the content rule, including stickiness. The possible values are:</p> <ul style="list-style-type: none"> <li>• <b>none</b> - Disables the advanced-balancing method for the rule. This is the default setting.</li> <li>• <b>arrowpoint-cookie</b> - Enables the content rule to stick the client to the server based on the unique service identifier information of the selected server in the ArrowPoint-generated cookie.</li> </ul> <p> <b>Note</b> When you configure the <b>arrowpoint-cookie expiration</b> command and the <b>advanced-balance arrowpoint-cookie</b> command, the CSS CPU may spike and the CSS may experience a degradation in its performance.</p>

**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
Advanced Balance (continued)	<ul style="list-style-type: none"> <li data-bbox="663 293 1231 444">• <b>cookies</b> - Enables the content rule to stick the client to the server based on the configured string found in the HTTP cookie header. You must specify a port in the content rule to use this option. The CSS then spoofs the connection.</li> <li data-bbox="663 467 1231 678">• <b>cookieurl</b> - This is the same as advanced-balance cookies, but if the CSS cannot find the cookie header in the HTTP packet, this type of failover looks up the URL extensions (that is, the portion after the “?” in the URL) based on the same string criteria. Use this option with any Layer 5 HTTP content rule.</li> <li data-bbox="663 701 1231 821">• <b>sticky-srcip</b> - Enables the content rule to stick a client to a server based on the client IP address, also known as Layer 3 stickiness. You can use this option with Layer 3, 4, or 5 content rules.</li> <li data-bbox="663 844 1231 1023">• <b>sticky-srcip-dstport</b> - Enables the content rule to stick a client to a server based on both the client IP address and the server destination port number, also known as Layer 4 stickiness. You can use this option with Layer 4 or 5 content rules.</li> <li data-bbox="663 1045 1231 1256">• <b>ssl</b> - Enables the content rule to stick the client to the server based on the Secure Sockets Layer (SSL) version 3 session ID assigned by the server. The application type must be SSL for the content rule. You must specify a port in the content rule to use this option. The CSS will then spoof the connection.</li> <li data-bbox="663 1279 1231 1430">• <b>url</b> - Enables the content rule to stick a client to a server based on a configured string found in the URL of the HTTP request. You must specify a port in the content rule to use this option. The CSS will then spoof the connection.</li> </ul>

**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
Sticky Mask	The subnet mask used for stickiness. The default is 255.255.255.255.
Sticky Inactivity Timeout	The inactivity timeout period on a sticky connection for a content rule before the CSS removes the sticky entry from the sticky table. The range is from 0 to 65535 minutes. The default value is 0, which means this feature is disabled.
Sticky No Cookie Found Action	<p>The action the CSS should take for a sticky cookie content rule when it cannot locate the cookie header or the specified cookie string in the client request. The possible values are:</p> <ul style="list-style-type: none"> <li>• <b>loadbalance</b> - The CSS uses the configured balanced method when no cookie is found in the client request. This is the default setting.</li> <li>• <b>redirect "URL"</b> - The CSS redirects the client request to a specified URL string when no cookie found in the client request. When using this option, you must also specify a redirect URL. Enter the redirect URL as a quoted text string from 0 to 64 characters.</li> <li>• <b>reject</b> - The CSS rejects the client request when no cookie is found in the request.</li> <li>• <b>service name</b> - The CSS sends the no cookie client request to the specified service when no cookie is found in the request.</li> </ul>

**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
Sticky Server Down Failover	<p>The action that the CSS should take when a sticky string is found but the associated service has failed or is suspended. The possible values are:</p> <ul style="list-style-type: none"> <li>• <b>Balance</b> - The failover method uses a service based on the configured load-balancing method (default).</li> <li>• <b>Redirect</b> - The failover method uses a service based on the currently configured redirect string. If a redirect string is not configured, the load-balancing method is used.</li> <li>• <b>Reject</b> - The failover method rejects the content request.</li> <li>• <b>Sticky-srcip</b> - The failover method uses a service based on the client IP address. This is dependent on the sticky configuration.</li> <li>• <b>Sticky-srcip-dstport</b> - The failover method uses a service based on the client IP address and the server destination port. This is dependent on the sticky configuration.</li> </ul>
ArrowPoint Cookie Path	<p>The pathname where you want to send the ArrowPoint cookie. The default path of the cookie is “/”.</p>
ArrowPoint Cookie Expiration	<p>The expiration time that the CSS compares with the time associated with the ArrowPoint cookie. If you do not set an expiration time, the cookie expires when the client exits the browser.</p>
ArrowPoint Cookie CSS/Browser Expired	<p>Indicates whether the <b>arrowpoint-cookie browser-expire</b> command is enabled to allow the browser to expire the ArrowPoint cookie based on the expiration time. If the command is enabled, the field displays “Browser” in place of “CSS.” The default is “CSS.”</p>

**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
ArrowPoint Cookie Service	Specifies whether the <b>arrowpoint-cookie expire-services</b> command is issued to expire service information when the cookie expires before sending a new cookie. By default, when the cookie expires, the CSS sends a new cookie with the server information from the expired cookie.
ArrowPoint Cookie Advanced	Specifies whether the <b>advanced-balance arrowpoint-cookie</b> command is issued to enable the content rule to stick the client to the server based on the unique service identifier of the selected server in the ArrowPoint-generated cookie.
ArrowPoint Cookie Format	Specifies the format of the ArrowpointCookie expiration time, whether the RFC 2822-compliant format is enabled or disabled. The <b>arrowpoint-cookie rfc2822-compliant</b> command configures the ArrowpointCookie expiration time syntax to be RFC 2822-compliant. This command causes the arrowpoint-cookie expiration time syntax to be only three-character days of the week (for example, “Tue” rather than “Tues”) and to capitalize only the first character of the month (for example, “Jan” rather than “JAN”).

**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
String Match Criteria	The string criteria to derive string results and the method to choose a destination server for the result. The string result is a sticky string in the cookie header, URL, or URL extension based on a sticky type being configured. See the following fields.
String Range	<p>The starting and ending byte positions within a cookie, URL, or URL extension from a client. By specifying the range of bytes, the CSS processes the information located only within the range.</p> <ul style="list-style-type: none"> <li>• The range is from 1 to 1999. The default starting byte position is 1.</li> <li>• The range is from 2 to 2000. The default ending byte position is 100.</li> </ul>
String Prefix	The string prefix located in the sticky range. If you do not configure the string prefix, the string functions start from the beginning of the cookie, URL, or URL extension, depending on the sticky type. If the string prefix is configured but is not found in the specified sticky range, load balancing defaults to the roundrobin method. The default has no prefix (“”).
String Eos-Char	The ASCII characters that are the delimiters for the sticky string.
String Ascii-Conversion	Indicates whether to enable or disable the ASCII conversion of escaped special characters within the specified sticky range before applying any processing to the string. By default, ACSII conversion is enabled.
String Skip-Len	The number of bytes to skip after the end of the prefix to find the string result. The default is 0. The range is from 0 to 64.

**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
String Process-Len	The number of bytes, after the end of the prefix designated by the <b>string prefix</b> command and skipping the bytes designated by the <b>string skip-length</b> command, that the string operation will use. The range is from 0 to 64. The default is 0.
String Operation	The method to choose a destination server for a string result; derived from the settings of the string criteria commands. The possible values are: <ul style="list-style-type: none"> <li>• <b>match-service-cookie</b> - Choose a server by matching a service cookie in the sticky string. This is the default setting. When a match is not found, the server is chosen by using the configured balance method (for example, roundrobin). This is the default method.</li> <li>• <b>hash-a</b> - Apply a basic hash algorithm on the hash string to generate the hash key.</li> <li>• <b>hash-crc32</b> - Apply the CRC32 algorithm on the hash string to generate a hash key.</li> <li>• <b>hash-xor</b> -Perform an Exclusive OR (XOR) on each byte of the hash string to derive the final hash key.</li> </ul>
Location-Cookie	The format (NAME=VALUE) of the location cookie string.
Location-Cookie Expiration	The expiration date and time of the location cookie. This value tells the client browser when the cookie will expire.
Cookie-Domain	A domain name for the location cookie. The cookie domain name allows your browser to send the cookie back to any site that ends with the domain name that you specify.
Redirect	Text used to build an HTTP 302 redirect message that is sent to the client when the rule is matched.

**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
Session Redundancy	Indicates whether ASR is enabled or disabled on the rule. For details on ASR, refer to the <i>Cisco Content Services Switch Redundancy Configuration Guide</i> .
Redund Glb Index	The unique global index value for Adaptive Session Redundancy assigned to the content rule using the <b>redundant-index</b> command in owner-content configuration mode.
IP Redundancy	The state of IP redundancy if configured on the rule. Possible values are Master, Backup, or Down. If IP redundancy is not configured, the state is Not Redundant.
Flow Timeout Multiplier	Number of seconds that a flow remains idle before the CSS reclaims the flow resources, as configured with the <b>flow-timeout-multiplier</b> command. For details on the <b>flow-timeout-multiplier</b> command, see <a href="#">Chapter 2, Configuring Flow and Port Mapping Parameters</a> .
Rule Services	Content rule services to configuration and statistic information, as follows:
Local Load Threshold	The normalized load threshold for the availability of each local service on the content rule. When the service load metric exceeds this threshold, the local service becomes unavailable and is redirected to the remote services. The range is from 2 through 254. The default is 254, which is the maximum load. A load of 255 indicates that the service is down.
PrimarySorryServer	The primary service to be used when all other services for the content rule are unavailable.
SecondSorryServer	The secondary service to be used when all other services for the content rule are unavailable.
Name	The names of the services.
Hits	The number of content accesses on the service.

**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
Wgt	<p>The weight for the service used when you configure ACA, weighted roundrobin, and SASP or DFP load-balancing on the content rule. With a higher weight, the CSS redirects more requests to the service. The letters preceding the weight numbers have the following meanings:</p> <ul style="list-style-type: none"> <li>• A = Weight reported by SASP</li> <li>• D = Weight reported by DFP</li> <li>• R = Weight configured for a service using the <b>add service weight</b> command in owner-content mode</li> <li>• S = Weight configured for a service using the <b>weight</b> command in service mode</li> </ul>
State	The state of the service.
Ld	The service load. The range is from 2 to 255; 255 indicates that the service is unavailable.
KAlive	The service keepalive type.
Conn	The number of connections currently mapped to the service.
DNS	The number of times that the CSS DNS resolver chose the service as the answer to a DNS client query.
SlowStart	The time remaining in seconds for the slow-start process on the service. If the service is not in the slow-start process, the field displays Out of SS. This field appears only when the slow-start timer is configured on the content rule.

**Table 10-12 Field Descriptions for the show rule Command Output (continued)**

Field	Description
DNS Balance	<p>Where the CSS resolves a request for a domain name into an IP address. The possible values are:</p> <ul style="list-style-type: none"> <li>• <b>leastloaded</b> - Resolves the request to the least-loaded local or remote domain site. The CSS first compares load numbers. If the load number between domain sites is within 50, then the CSS compares their response times. The site with the fastest response time is considered the least-loaded site.</li> <li>• <b>Preferlocal</b> - Resolves the request to a local VIP address. If all local systems exceed their load threshold, the CSS chooses the least-loaded remote system VIP address as the resolved address for the domain name.</li> <li>• <b>roundrobin</b> - Resolves the request by evenly distributing the load to resolve domain names among content domain sites, both local and remote. The CSS does not include sites that exceed their local load threshold.</li> <li>• <b>useownerdnsbalance</b> - Resolves the request by using the DNS load-balancing method assigned to the owner. This is the default method for the content rule. If you do not implicitly set an owner method, the CSS uses the default owner DNS load-balancing method of roundrobin.</li> </ul>
DNS Names	Domain Name System names.
DNS TTL	The Time to Live value, in seconds, which determines how long the DNS client remembers the IP address response to the query.

*Table 10-12 Field Descriptions for the show rule Command Output (continued)*

Field	Description
Hotlist	Indicates whether or not hot list is enabled.
Size	The total number of hot-list entries that is maintained for the rule. The range is from 1 to 100. The default is 10.
Type	The hot-list type. Currently, the CSS supports only the hit count hot-list type, which is the default setting. Hit count is the number of times that the content is accessed.
Threshold	The hit count per interval threshold below which content is not considered hot. The range is from 0 to 65535. The default is 0.
Interval	The interval, in minutes, for refreshing the hot list. The range is from 1 to 60. The default is 1.
Associated ACLs	The ACLs associated with a content rule.
TCP RST Client If Service Unreachable	Whether or not the <b>flow-reset-reject</b> command is enabled to allow the CSS's flow manager subsystem to send a TCP RST (reset) frame when a flow is mapped to a service that is no longer reachable. By default, the <b>flow-reset-reject</b> command is disabled.

## Clearing Counters in a Content Rule

The CSS allows you to clear counters:

- Associated with all content rules or only the current content rule
- Associated with a single service or for all services in a content rule

Use the **zero** command and its options to clear the counters for content rules or services associated with content rules, and set the counters to zero.

This section contains:

- Clearing Counters for Content Rules
- Clearing Service Statistics Counters in a Content Rule

## Clearing Counters for Content Rules

To reset the counters for all content rules to zero, use the **zero all** command. The reset counter statistics appear as zero in the **show summary** display.

**Note**

If you issue the **zero** command without an option, only the counters for the current content rule are set to zero.

For example, enter:

```
(config-owner-content[rule1])# zero all
```

## Clearing Service Statistics Counters in a Content Rule

To clear a service statistics counter for all CSS services associated with a content rule, use the **zero** command. To clear a service statistics counter for a specific service in the content rule, use the **zero** command and identify the name of the service. In this case, only the counter for the specified service is set to zero.

The reset statistics appear as 0 in the **show service** display.

You can issue the following **zero** commands from content mode:

- **zero total-connections** - Set the Total Connections counter to zero for all services associated with the specified content rule
- **zero total-reused-connections** - Set the Total Reused Conns. counter to zero for all services associated with the specified content rule
- **zero state-transitions** - Set the State Transitions counter to zero for all services associated with the specified content rule

You can issue the following **zero** commands from content mode:

- **zero total-connections service *service\_name*** - Set the Total Connections counter to zero for only the specified service associated with the content rule
- **zero total-reused-connections service *service\_name*** - Set the Total Reused Conns. counter to zero for only the specified service associated with the content rule
- **zero state-transitions service *service\_name*** - Set the State Transitions counter to zero for only the specified service associated with the content rule

For example, to clear a counter for all services associated with the specified content rule, enter:

```
(config-owner-content[rule1])# zero total-connections
```

For example, to clear a counter for a specific service in a content rule, enter:

```
(config-owner-content[rule1])# zero total-connections service serv1
```

## Where to Go Next

Once you create content rules you can configure sticky parameters for the content rules. For information on configuring sticky parameters, see [Chapter 11, Configuring Sticky Parameters for Content Rules](#).