Preface xxiii
Audience xxiv
How to Use This Guide xxiv
Related Documentation xxvi
Symbols and Conventions xxix
Obtaining Documentation xxx
   Cisco.com xxx
   Ordering Documentation xxx
Documentation Feedback xxxi
Obtaining Technical Assistance xxxi
   Cisco Technical Support Website xxxii
   Submitting a Service Request xxxii
   Definitions of Service Request Severity xxxiii
Obtaining Additional Publications and Information xxxiii

CHAPTER 1
Content Load-Balancing Overview 1-1
   Service, Owner, and Content Rule Overview 1-1
   Overview of CSS Flow 1-4

CHAPTER 2
Configuring Flow and Port Mapping Parameters 2-1
   Configuring Flow Parameters 2-1
      Configuring Permanent Connections for TCP or UDP Ports 2-2
      Configuring TCP Maximum Segment Size 2-3
      Reclaiming Reserved Telnet and FTP Control Ports 2-4
   Showing Flow Statistics 2-4
Configuring Flow Inactivity Timeouts on Content Rules and Source Groups 2-6
  Timeout Value Precedence 2-6
  Configuring Flow Timeouts 2-7
  Displaying Flow Timeout Statistics 2-8
  Displaying Content Rule and Source Group Information 2-9
Configuring Flow Processing for Fragmented IP Packets 2-9
  What Is IP Packet Fragmentation? 2-10
  Configuration Restrictions 2-11
  Enabling Flow Processing for Fragmented IP Packets 2-11
  Configuring the Maximum Assembled Size 2-12
  Configuring the Minimum Fragment Size 2-13
  Displaying IP Fragment Statistics 2-14
  Resetting IP Fragment Statistics 2-16
Configuring a CSS to Send a TCP Reset if a VIP Is Unavailable 2-17
Configuring the Flow-State Table 2-18
  Configuring the Flow State of a Port 2-21
  Resetting the Flow-State Table Hit Counters 2-23
  Displaying the Flow-State Table 2-23
Configuring CSS Port Mapping 2-24
  Overview of Global Port Mapping 2-24
  Configuring Global Port Mapping 2-25
  Displaying Global Port Mapping Statistics 2-27
  Configuring No-Flow Port Mapping 2-29
  Displaying No-Flow Port Mapping Statistics 2-30

CHAPTER 3
Configuring Services 3-1
  Service Configuration Quick Start 3-2
  Creating a Service 3-4
  Assigning an IP Address to the Service 3-4
Specifying a Port
Specifying a Protocol
Specifying a Domain Name
Specifying an HTTP Redirect String
Prepending “http://” to a Redirect String or a Domain
Configuring an Advanced Load-Balancing String
Configuring a Service HTTP Cookie
Configuring Weight and Graceful Shutdown
Specifying a Service Type
How the CSS Accesses Server Types
Configuring Service Access
Bypassing Content Rules on Caches
Configuring Network Address Translation for Transparent Caches
Configuring a Service to Bypass a Cache Farm
Configuring Maximum TCP Connections
Configuring Keepalives for a Service
Activating a Service
Suspending a Service
Showing Service Configurations
Clearing Service Statistics Counters
Where to Go Next

Configuring Service, Global, and Script Keepalives
CSS Keepalive Overview
Configuring Service Keepalives
Configuring Global Keepalives
Creating and Naming a Global Keepalive
Configuring a Global Keepalive IP Address
Configuring a Global Keepalive Description 4-7
Activating the Global Keepalive 4-7
Suspending a Global Keepalive 4-8
Associating a Service with a Global Keepalive 4-8
Configuring Service and Global Keepalive Attributes 4-8
   Configuring a Keepalive Frequency 4-9
   Configuring a Keepalive Retry Period 4-10
   Configuring the Maximum Number of Failures for a Keepalive 4-11
   Configuring a Keepalive Type 4-11
   Configuring a TCP Keepalive with Graceful Socket Close (FIN) 4-14
   Configuring a Keepalive Port Number 4-15
   Configuring the HTTP Keepalive Method 4-16
   Configuring a Keepalive HTTP Response Code 4-17
   Configuring a Keepalive URI 4-18
   Configuring a Keepalive Hash Value 4-19
Showing Keepalive Configurations 4-21
Using Script Keepalives with Services 4-23
   Script Keepalive Considerations 4-24
   Configuring Script Keepalives 4-26
   Viewing a Script Keepalive in a Service 4-27
   Script Keepalive Status Codes 4-28
   Script Keepalives and Upgrading WebNS Software 4-28

CHAPTER 5
Configuring Source Groups for Services 5-1
   Overview of Source Groups and Port Mapping 5-2
   Source Group Configuration Quick Start 5-5
   Creating a Source Group 5-8
   Configuring the Source Group 5-8
      Configuring a VIP Address for a Source Group 5-9
Configuring a Service on a Source Group 5-10
Adding a Destination Service to the Source Group 5-10
Activating and Suspending a Source Group 5-11
Configuring Source Group Port Mapping 5-12
  Configuring the Starting Port Number 5-12
  Configuring the Total Number of Ports in a Port-Map Range 5-12
  Configuring a VIP Address Range for Port Mapping 5-13
  Disabling Port Mapping 5-15
Configuring Source Groups and ACLs 5-16
Configuring a Source Group for FTP Connections 5-18
Configuring Source Groups to Allow Servers to Resolve Domain Names Using the Internet 5-19
Showing Source Groups 5-21
Clearing Source Group Counters 5-24

CHAPTER 6

Configuring Loads for Services 6-1
Configuring Relative Load for Services 6-1
  Relative Load Overview 6-2
  Configuring Relative Load 6-4
    Relative Load Configuration Quick Start 6-5
    Configuring Global Load Reporting 6-6
    Configuring the Relative Load Step 6-6
    Configuring the Global Load Threshold 6-7
    Configuring the Load Teardown Timer 6-8
    Configuring the Load Ageout Timer 6-8
  Showing Global Service Loads 6-9
Configuring the Absolute Load Calculation Method 6-11
  Overview of Calculating Absolute Load 6-12
  Configuration Requirements and Restrictions 6-13
Contents

Absolute Load Configuration Quick Start 6-13
Configuring Load Calculation 6-15
Using the load absolute-sensitivity Command 6-16
  Configuring load absolute-sensitivity 6-16
  Optimizing the Absolute Load Number Scale 6-17
Configuring Load Variance 6-18
Displaying Relative Load Statistics 6-19
Displaying Absolute Load Calculation Ranges 6-20
Using ArrowPoint Content Awareness Based on Server Load and Weight 6-26
  Using ACA Based on Server Load 6-26
  Using ACA Based on Server Weight and Load 6-26
Configuring the Load Command for Use with ACA 6-27

CHAPTER 7
Configuring Dynamic Feedback Protocol for Server Load Balancing 7-1
  DFP Overview 7-2
    Functions of a DFP Agent 7-3
    Types of DFP Messages 7-3
    DFP System Flow 7-4
  Configuring a DFP Agent 7-5
  Maintaining a Consistent Weight Range Among Services 7-8
  Displaying Configured DFP Agents 7-9
  Displaying Services Supported by Configured DFP Agents 7-10
  Displaying DFP Information 7-11
    Using the show service Command 7-11
    Using the show rule services Command 7-12

CHAPTER 8
Configuring Owners 8-1
  Owner Configuration Quick Start 8-2
Contents

Creating an Owner 8-3
Configuring an Owner DNS Balance Type 8-3
Specifying Owner Address 8-4
Specifying Owner Billing Information 8-4
Specifying Case 8-5
Specifying Owner DNS Type 8-5
Specifying Owner E-Mail Address 8-6
Removing an Owner 8-6
Showing Owner Information 8-7
    Showing Owner Summary 8-10
Where to Go Next 8-11

CHAPTER 9

Configuring Content Rules 9-1
Content Rule Overview 9-2
    Content Rule Hierarchy 9-3
        Matching Precedence for Layer 5 Rules 9-3
        Content Rule Configuration Quick Start 9-5
    Naming and Assigning a Content Rule to an Owner 9-7
    Configuring a Virtual IP Address 9-8
Configuring a Domain Name Content Rule 9-11
    Matching Content Rules to Multiple Domain Names 9-12
    Configuring a Content Rule Using a Domain Name and a VIP Address 9-13
    Using Wildcards in Domain Name Content Rules 9-14
        General Guidelines for Domain Name Wildcards in Content Rules 9-16
Configuring Domain Qualifier Lists 9-16
    Creating a DQL 9-18
    Describing a DQL 9-19
    Adding a Domain to a DQL 9-19
Adding a DQL to a Content Rule 9-20
Removing a DQL from a Content Rule 9-20
Showing DQL Configurations 9-21
Configuring Virtual Web Hosting 9-21
Adding Services to a Content Rule 9-25
   Adding a Service to a Content Rule 9-27
      Specifying a Service Weight 9-27
   Adding a Primary Sorry Server to a Content Rule 9-29
   Adding a Secondary Sorry Server to a Content Rule 9-30
   Adding a DNS Name to a Content Rule 9-31
Disabling DNS in a Content Rule 9-31
Activating a Content Rule 9-32
Suspending a Content Rule 9-32
Removing a Content Rule 9-33
Removing a Service from a Content Rule 9-33
Configuring a Protocol 9-33
Configuring a Port 9-34
Configuring Load Balancing 9-34
Configuring a DNS Balance Type 9-37
Configuring Hot Lists 9-38
   Configuring a Domain Hotlist 9-40
Configuring Extension Qualifier Lists 9-40
   Specifying an EQL in a Uniform Resource Locator 9-42
   Showing EQL Extensions and Descriptions 9-43
Configuring URL Qualifier Lists 9-43
   URQL Quick Start 9-44
   Creating a URQL 9-46
   Configuring a URL in a URQL 9-46
      Specifying the URL Entry 9-47
Clearing Service Statistics Counters in a Content Rule 9-88
Where to Go Next 9-89

CHAPTER 10
Configuring Sticky Parameters for Content Rules 10-1

Sticky Overview 10-2
  Why Use Stickiness? 10-3
  Using Layer 3 Sticky 10-4
  Using Layer 4 Sticky 10-4
  Using Layer 5 Sticky 10-5
Configuring Sticky on the CSS 10-5
Specifying an Advanced Load-Balancing Method for Sticky Content 10-9
Configuring SSL-Layer 4 Fallback 10-13
Configuring Sticky Serverdown Failover 10-15
Configuring Sticky Mask 10-16
Configuring Sticky Inactive Timeout 10-16
Configuring Sticky Content for SSL 10-17
Configuring String Range 10-18
Specifying a String Operation 10-19
  Comparing Hash Method with Match Method 10-20
Enabling or Disabling String ASCII Conversion 10-22
Configuring the Handling of Multiple String Matches 10-22
Specifying End-of-String Characters 10-23
Specifying a String Prefix 10-24
Specifying a String Process Length 10-24
Specifying a String Skip Length 10-25
Configuring Sticky-No-Cookie-Found-Action 10-25
Configuring Sticky Parameters for E-Commerce and Other Internet Applications 10-26
Configuring an advanced-balance arrowpoint-cookie  10-26
Configuring an Arrowpoint Cookie  10-27
  Configuring an Arrowpoint Cookie Name  10-28
  Configuring an Arrowpoint Cookie Path  10-29
  Configuring an Arrowpoint-Cookie Expiration Time  10-29
  Configuring Arrowpoint-Cookie Browser Expire  10-30
  Configuring Arrowpoint-Cookie Expire Services  10-31
  Configuring an Arrowpoint Cookie Domain  10-31
Configuring a Location Cookie  10-31
  Overview  10-32
  Location Cookie Quick Start  10-33
  Configuring the location-cookie Command  10-42
  Configuring a Domain Name for the Location Cookie  10-43
  Configuring Location Services  10-43
  Examples of Location Cookie Flow  10-44
  Displaying Location Cookie Information  10-49
Configuring Wireless Users for E-Commerce Applications  10-49
Configuring Session Initiation Protocol Load Balancing  10-50
Configuration Requirements and Restrictions  10-51
SIP Load Balancing Configuration Quick Start  10-52
Showing Sticky Attributes  10-54
Showing Sticky Table Configurations  10-64
  Showing Layer 3 Sticky Table Information  10-67
  Showing Layer 4 Sticky Table Information  10-68
  Showing SIP Call-ID Sticky Table Information  10-69
  Showing SSL Sticky Table Information  10-70
  Showing WAP Sticky Table Information  10-71
Showing Sticky Connection Statistics  10-72
CHAPTER 11
Configuring HTTP Header Load Balancing 11-1
HTTP Header Load-Balancing Overview 11-2
   Using HTTP Header Load Balancing in a Content Rule 11-2
HTTP Header Load Balancing Configuration Quick Start 11-3
Creating a Header Field Group 11-4
Describing the Header Field Group 11-5
Configuring a Header-Field Entry 11-6
Associating a Header Field Group with a Content Rule 11-10
Showing a Content Rule Header Field Group Configuration 11-11
Showing Header Field Groups 11-11
Header Field Group Configuration Examples 11-12

CHAPTER 12
Configuring Caching 12-1
Caching Overview 12-1
   Content Caching 12-2
      Using Proxy Caching 12-3
      Using Reverse Proxy Caching 12-4
      Using Transparent Caching 12-5
      Using Cache Clustering 12-7
Caching Configuration Quick Start 12-8
Configuring Caching Content Rules 12-10
   Specifying a Service Type 12-10
   Specifying a Failover Type 12-11
   Configuring Load Balancing 12-14
   Configuring a Double-Wildcard Caching Content Rule 12-16
Enabling Content Requests to Bypass Caches 12-17
   Using the param-bypass Command 12-17
   Using the cache-bypass Command 12-18
   Using the bypass-hosttag Command 12-18
Configuring Network Address Translation for Transparent Caches 12-19
Configuring Network Address Translation Peering 12-20
Configuring NAT Peering 12-22

CHAPTER 13
Configuring Content Replication 13-1
Configuring Demand-Based Content Replication 13-2
  Demand-Based Content Replication Quick Start 13-3
  Configuring Hot Lists 13-5
  Specifying Service Type for Replication 13-6
  Configuring Max Age 13-7
  Configuring Max Content 13-8
  Configuring Max Usage 13-8
  Configuring FTP Access for Content Replication 13-8
  Creating an FTP Record 13-9
Configuring Content Staging and Replication 13-10
  Content Staging and Replication Quick Start 13-12
  Configuring FTP Access for Publishing and Subscribing 13-14
  Configuring a Publishing Service 13-15
    Displaying Publisher Configurations 13-16
  Configuring a Subscriber Service 13-18
    Displaying Subscriber Configurations 13-19
  Configuring a Content Rule for Content Staging and Replication 13-20
  Configuring Publisher Content Replication 13-21
  Configuring File-Error Handling for Content Replication 13-22
  Displaying Content 13-23
Figures

Figure 12-8 Removing a Cache Service Configured for Failover Linear 12-14
Figure 12-9 NAT Peering Configuration Example 12-21
Table 2-1  Field Descriptions for the flow statistics Command  2-4
Table 2-2  Field Descriptions for show flow-timeout default Command  2-8
Table 2-3  Field Descriptions for the show flow-timeout configured Command  2-9
Table 2-4  Field Descriptions for the show ip-fragment-stats Command  2-14
Table 2-5  Default CSS Flow-Disabled Ports  2-18
Table 2-6  Flow-State Table Default Values  2-19
Table 2-7  Field Descriptions for the show flow-state-table Command  2-23
Table 2-8  Field Descriptions for show global-portmap Command  2-28
Table 2-9  Field Descriptions for show noflow-portmap Command  2-31
Table 3-1  Service Configuration Quick Start  3-2
Table 3-2  Field Descriptions for the show service Command Output  3-19
Table 4-1  Keepalive Class, Types, and Limitations  4-2
Table 4-2  Global Keepalive Configuration Quick Start  4-5
Table 4-3  Field Descriptions for the show keepalive Command Output  4-22
Table 5-1  Adding Modules (SPs) to a CSS 11506 Decreases the Number of Eligible Source Ports While Adding VIP Addresses for Port Mapping Increases the Number of Eligible Source Ports  5-4
Table 5-2  Adding Destination Ports or Configuring a VIP Address Range for Port Mapping Increases the Number of Eligible Source Ports  5-5
Table 5-3  Source Group Configuration Quick Start  5-6
Table 5-4  Field Descriptions for the show group Command Output  5-21
Table 6-1  Relative Load Configuration Quick Start  6-5
Table 6-2  Field Descriptions for the show load Command Output  6-10
Table 6-3  Absolute Load Configuration Quick Start  6-13
Table 6-4  Service-Specific Field Descriptions for the show load Command Output  6-19
Table 6-5  Output for the show load absolute Command (load absolute-sensitivity = 21)  6-21
Table 6-6  Field Descriptions for the show load absolute Command Output  6-22
Table 6-7  Output for the show load absolute Command (load absolute-sensitivity = 22)  6-23
Table 6-8  Output for the show load absolute Command (load absolute-sensitivity = 1)  6-24
Table 6-9  Output for the show load absolute Command (load absolute-sensitivity = 2)  6-25
Table 7-1  Field Descriptions for the show dfp Command Output  7-9
Table 7-2  Field Descriptions for the show dfp-reports Command Output  7-11
Table 8-1  Owner Configuration Quick Start  8-2
Table 8-2  Field Descriptions for the show owner name Command Output  8-7
Table 8-3  Field Descriptions for the show owner name statistics Command Output  8-9
Table 8-4  Field Descriptions for the show summary Command Output  8-11
Table 9-1  Content Rule Configuration Quick Start  9-5
Table 9-2  Field Descriptions for the show dq1 Command Output  9-21
Table 9-3  Virtual Web Hosting Configuration Quick Start  9-22
Table 9-4  Field Descriptions for the show domain hotlist Command Output  9-39
Table 9-5  Field Descriptions for the show eql Command Output  9-43
Table 9-6  URQL Configuration Quick Start  9-44
Table 9-7  Field Descriptions for the show urql Command Output  9-51
Table 9-8  Field Descriptions for a Specified URQL  9-51
Table 9-9  Field Descriptions for the show remap Command Output  9-63
Table 9-10  Field Descriptions for the show content Command Output  9-71
Table 9-11  Field Descriptions for the show rule Command Output  9-73
<table>
<thead>
<tr>
<th>Table 10-1</th>
<th>Applying Sticky Rules to Content Rules</th>
<th>10-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 10-2</td>
<td>Location Cookie Configuration Quick Start for Site1</td>
<td>10-33</td>
</tr>
<tr>
<td>Table 10-3</td>
<td>Location Cookie Configuration Quick Start for Site2</td>
<td>10-36</td>
</tr>
<tr>
<td>Table 10-4</td>
<td>Location Cookie Configuration Quick Start for Site3</td>
<td>10-39</td>
</tr>
<tr>
<td>Table 10-5</td>
<td>SIP Configuration Quick Start</td>
<td>10-52</td>
</tr>
<tr>
<td>Table 10-6</td>
<td>Field Descriptions for the show rule sticky Command Output</td>
<td>10-55</td>
</tr>
<tr>
<td>Table 10-7</td>
<td>Field Descriptions for the show sticky-table all-sticky Command Output</td>
<td>10-64</td>
</tr>
<tr>
<td>Table 10-8</td>
<td>Field Descriptions for the show sticky-stats Command Output</td>
<td>10-72</td>
</tr>
<tr>
<td>Table 11-1</td>
<td>HTTP Load Balancing Configuration Quick Start</td>
<td>11-3</td>
</tr>
<tr>
<td>Table 11-2</td>
<td>Field Descriptions for the show header-field-group Command Output</td>
<td>11-12</td>
</tr>
<tr>
<td>Table 12-1</td>
<td>Caching Configuration Quick Start</td>
<td>12-8</td>
</tr>
<tr>
<td>Table 12-2</td>
<td>NAT Configuration Quick Start</td>
<td>12-23</td>
</tr>
<tr>
<td>Table 13-1</td>
<td>Demand-Based Content Replication Configuration Quick Start</td>
<td>13-3</td>
</tr>
<tr>
<td>Table 13-2</td>
<td>Content Staging and Replication Configuration Quick Start</td>
<td>13-12</td>
</tr>
<tr>
<td>Table 13-3</td>
<td>Field Descriptions for the show publisher Command</td>
<td>13-17</td>
</tr>
<tr>
<td>Table 13-4</td>
<td>Field Descriptions for the show subscriber Command</td>
<td>13-19</td>
</tr>
<tr>
<td>Table 13-5</td>
<td>Field Descriptions for the show content Command</td>
<td>13-23</td>
</tr>
</tbody>
</table>
Preface

This guide provides instructions for the content load-balancing configuration of the Cisco 11500 Series Content Services Switch (CSS). Information in this guide applies to all CSS models except where noted.

The CSS software is available in a Standard or optional Enhanced feature set. The Enhanced feature set contains all of the Standard feature set and also includes Network Address Translation (NAT) Peering, Domain Name Service (DNS), Demand-Based Content Replication (Dynamic Hot Content Overflow), Content Staging and Replication, and Network Proximity DNS. Proximity Database and Secure Management, which includes Secure Shell Host and SSL strong encryption for the Device Management software, are optional features.

This preface contains the following major sections:

- Audience
- How to Use This Guide
- Related Documentation
- Symbols and Conventions
- Obtaining Documentation
- Documentation Feedback
- Obtaining Technical Assistance
- Obtaining Additional Publications and Information
Audience

This guide is intended for the following trained and qualified service personnel who are responsible for configuring the CSS:

- Web master
- System administrator
- System operator

How to Use This Guide

This guide is organized as follows:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1, Content Load-Balancing Overview</td>
<td>Assist you in understanding load balancing on the CSS by providing information about the relationship of service, owner, and content rules, and describes how the CSS handles TCP and UDP traffic.</td>
</tr>
<tr>
<td>Chapter 2, Configuring Flow and Port Mapping Parameters</td>
<td>Configure flow and port mapping parameters for the CSS.</td>
</tr>
<tr>
<td>Chapter 3, Configuring Services</td>
<td>Create and configure services. This chapter also provides information on activating and suspending a service and displaying service information.</td>
</tr>
<tr>
<td>Chapter 4, Configuring Service, Global, and Script Keepalives</td>
<td>Configure service, global, and script keepalives. This chapter also provides information on how to display keepalive information.</td>
</tr>
<tr>
<td>Chapter 5, Configuring Source Groups for Services</td>
<td>Configure source groups for services. This chapter also provides information on source group port mapping and displaying source group information.</td>
</tr>
<tr>
<td>Chapter 6, Configuring Loads for Services</td>
<td>Configure the relative and absolute load for services. This chapter also provides information on how to display global load information.</td>
</tr>
</tbody>
</table>
## Chapter Description

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 7, Configuring Dynamic Feedback Protocol for Server Load Balancing</td>
<td>Configure Dynamic Feedback Protocol (DFP) that allows load-balanced servers (both local and remote) to dynamically report changes in their status and their ability to provide services to a CSS.</td>
</tr>
<tr>
<td>Chapter 8, Configuring Owners</td>
<td>Create and configure owners. This chapter also describes how to configure owner attributes such as a DNS balance type, address, billing information, case sensitivity, and DNS type.</td>
</tr>
<tr>
<td>Chapter 9, Configuring Content Rules</td>
<td>Create and configure content rules. This chapter also describes how to assign a content rule to an owner, configure a virtual IP address, add a service to a content rule, and activate, suspend, and remove a content rule.</td>
</tr>
<tr>
<td>Chapter 10, Configuring Sticky Parameters for Content Rules</td>
<td>Configure sticky parameters for content rules such as subnet mask, inactive timeout, string range, string operation, string prefix and string skip length.</td>
</tr>
<tr>
<td>Chapter 11, Configuring HTTP Header Load Balancing</td>
<td>Configure HTTP header load balancing including creating a header field group, configuring a header field entry, associating a header field group to a content rule, and showing header field groups.</td>
</tr>
<tr>
<td>Chapter 12, Configuring Caching</td>
<td>Configure content caching for proxy, reverse proxy, and transparent caching configurations.</td>
</tr>
<tr>
<td>Chapter 13, Configuring Content Replication</td>
<td>Configure demand-based content replication and content synchronization using publisher and subscriber services on a CSS.</td>
</tr>
</tbody>
</table>
In addition to this document, the Content Services Switch documentation set includes the following:

<table>
<thead>
<tr>
<th>Document Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release Note for the Cisco 11500 Series Content Services Switch</td>
<td>This release note provides information on operating considerations, caveats, and command-line interface (CLI) commands for the Cisco 11500 series CSS.</td>
</tr>
<tr>
<td>Cisco 11500 Series Content Services Switch Hardware Installation Guide</td>
<td>This guide provides information for installing, cabling, and powering the Cisco 11500 series CSS. In addition, this guide provides information about CSS specifications, cable pinouts, and hardware troubleshooting.</td>
</tr>
<tr>
<td>Cisco Content Services Switch Getting Started Guide</td>
<td>This guide describes how to perform initial administration and configuration tasks on the CSS, including:</td>
</tr>
<tr>
<td></td>
<td>• Booting the CSS for the first time and a routine basis, and logging in to the CSS</td>
</tr>
<tr>
<td></td>
<td>• Configuring the username and password, Ethernet management port, static IP routes, and the date and time</td>
</tr>
<tr>
<td></td>
<td>• Configuring DNS server for hostname resolution</td>
</tr>
<tr>
<td></td>
<td>• Configuring sticky cookies with a sticky overview and advanced load-balancing method using cookies</td>
</tr>
<tr>
<td></td>
<td>• Finding information in the CSS documentation with a task list</td>
</tr>
<tr>
<td></td>
<td>• Troubleshooting the boot process</td>
</tr>
</tbody>
</table>
### Related Documentation

<table>
<thead>
<tr>
<th>Document Title</th>
<th>Description</th>
</tr>
</thead>
</table>
| *Cisco Content Services Switch Administration Guide* | This guide describes how to perform administrative tasks on the CSS, including upgrading your CSS software and configuring the following:  
  - Logging, including displaying log messages and interpreting sys.log messages  
  - User profile and CSS parameters  
  - SNMP  
  - RMON  
  - XML documents to configure the CSS  
  - CSS scripting language  
  - Offline Diagnostic Monitor (Offline DM) menu |
| *Cisco Content Services Switch Routing and Bridging Configuration Guide* | This guide describes how to perform routing and bridging configuration tasks on the CSS, including:  
  - Management ports, interfaces, and circuits  
  - Spanning-tree bridging  
  - Address Resolution Protocol (ARP)  
  - Routing Information Protocol (RIP)  
  - Internet Protocol (IP)  
  - Open Shortest Path First (OSPF) protocol  
  - Cisco Discovery Protocol (CDP)  
  - Dynamic Host Configuration Protocol (DHCP) relay agent |
<table>
<thead>
<tr>
<th>Document Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Cisco Content Services Switch Global Server Load-Balancing Configuration Guide</em></td>
<td>This guide describes how to perform CSS global load-balancing configuration tasks, including:</td>
</tr>
<tr>
<td></td>
<td>• Domain Name System (DNS)</td>
</tr>
<tr>
<td></td>
<td>• DNS Sticky</td>
</tr>
<tr>
<td></td>
<td>• Content Routing Agent</td>
</tr>
<tr>
<td></td>
<td>• Client-Side Accelerator</td>
</tr>
<tr>
<td></td>
<td>• Network proximity</td>
</tr>
<tr>
<td><em>Cisco Content Services Switch Redundancy Configuration Guide</em></td>
<td>This guide describes how to perform CSS redundancy configuration tasks, including:</td>
</tr>
<tr>
<td></td>
<td>• VIP and virtual interface redundancy</td>
</tr>
<tr>
<td></td>
<td>• Adaptive session redundancy</td>
</tr>
<tr>
<td></td>
<td>• Box-to-box redundancy</td>
</tr>
<tr>
<td><em>Cisco Content Services Switch Security Configuration Guide</em></td>
<td>This guide describes how to perform CSS security configuration tasks, including:</td>
</tr>
<tr>
<td></td>
<td>• Controlling access to the CSS</td>
</tr>
<tr>
<td></td>
<td>• Secure Shell Daemon protocol</td>
</tr>
<tr>
<td></td>
<td>• Radius</td>
</tr>
<tr>
<td></td>
<td>• TACACS+</td>
</tr>
<tr>
<td></td>
<td>• Firewall load balancing</td>
</tr>
<tr>
<td><em>Cisco Content Services Switch SSL Configuration Guide</em></td>
<td>This guide describes how to perform CSS SSL configuration tasks, including:</td>
</tr>
<tr>
<td></td>
<td>• SSL certificate and keys</td>
</tr>
<tr>
<td></td>
<td>• SSL termination</td>
</tr>
<tr>
<td></td>
<td>• Back-end SSL</td>
</tr>
<tr>
<td></td>
<td>• SSL initiation</td>
</tr>
</tbody>
</table>
Symbols and Conventions

This guide uses the following symbols and conventions to identify different types of information.

**Caution**

A caution means that a specific action you take could cause a loss of data or adversely impact use of the equipment.

**Warning**

A warning describes an action that could cause you physical harm or damage the equipment.

**Note**

A note provides important related information, reminders, and recommendations.

**Bold text** indicates a command in a paragraph.

**Courier text** indicates text that appears on a command line, including the CLI prompt.

**Courier bold text** indicates commands and text you enter in a command line.

**Italic text** indicates the first occurrence of a new term, a book title, emphasized text, or variables for which you supply values.

1. A numbered list indicates that the order of the list items is important.
   
a. An alphabetical list indicates that the order of the secondary list items is important.
Obtaining Documentation

Cisco documentation and additional literature are available on Cisco.com. Cisco also provides several ways to obtain technical assistance and other technical resources. These sections explain how to obtain technical information from Cisco Systems.

Cisco.com

You can access the most current Cisco documentation at this URL:
http://www.cisco.com/univercd/home/home.htm
You can access the Cisco website at this URL:
http://www.cisco.com
You can access international Cisco websites at this URL:

Ordering Documentation

You can find instructions for ordering documentation at this URL:
You can order Cisco documentation in these ways:
• Registered Cisco.com users (Cisco direct customers) can order Cisco product documentation from the Ordering tool:
• Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco Systems Corporate Headquarters (California, USA) at 408 526-7208 or, elsewhere in North America, by calling 800 553-NETS (6387).

Documentation Feedback

You can send comments about technical documentation to bug-doc@cisco.com.

You can submit comments by using the response card (if present) behind the front cover of your document or by writing to the following address:

Cisco Systems
Attn: Customer Document Ordering
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

Obtaining Technical Assistance

For all customers, partners, resellers, and distributors who hold valid Cisco service contracts, Cisco Technical Support provides 24-hour-a-day, award-winning technical assistance. The Cisco Technical Support Website on Cisco.com features extensive online support resources. In addition, Cisco Technical Assistance Center (TAC) engineers provide telephone support. If you do not hold a valid Cisco service contract, contact your reseller.
Cisco Technical Support Website

The Cisco Technical Support Website provides online documents and tools for troubleshooting and resolving technical issues with Cisco products and technologies. The website is available 24 hours a day, 365 days a year at this URL:

http://www.cisco.com/techsupport

Access to all tools on the Cisco Technical Support Website requires a Cisco.com user ID and password. If you have a valid service contract but do not have a user ID or password, you can register at this URL:


Submitting a Service Request

Using the online TAC Service Request Tool is the fastest way to open S3 and S4 service requests. (S3 and S4 service requests are those in which your network is minimally impaired or for which you require product information.) After you describe your situation, the TAC Service Request Tool automatically provides recommended solutions. If your issue is not resolved using the recommended resources, your service request will be assigned to a Cisco TAC engineer. The TAC Service Request Tool is located at this URL:

http://www.cisco.com/techsupport/servicerequest

For S1 or S2 service requests or if you do not have Internet access, contact the Cisco TAC by telephone. (S1 or S2 service requests are those in which your production network is down or severely degraded.) Cisco TAC engineers are assigned immediately to S1 and S2 service requests to help keep your business operations running smoothly.

To open a service request by telephone, use one of the following numbers:

Asia-Pacific: +61 2 8446 7411 (Australia: 1 800 805 227)
EMEA: +32 2 704 55 55
USA: 1 800 553 2447

For a complete list of Cisco TAC contacts, go to this URL:

http://www.cisco.com/techsupport/contacts
Definitions of Service Request Severity

To ensure that all service requests are reported in a standard format, Cisco has established severity definitions.

Severity 1 (S1)—Your network is “down,” or there is a critical impact to your business operations. You and Cisco will commit all necessary resources around the clock to resolve the situation.

Severity 2 (S2)—Operation of an existing network is severely degraded, or significant aspects of your business operation are negatively affected by inadequate performance of Cisco products. You and Cisco will commit full-time resources during normal business hours to resolve the situation.

Severity 3 (S3)—Operational performance of your network is impaired, but most business operations remain functional. You and Cisco will commit resources during normal business hours to restore service to satisfactory levels.

Severity 4 (S4)—You require information or assistance with Cisco product capabilities, installation, or configuration. There is little or no effect on your business operations.

Obtaining Additional Publications and Information

Information about Cisco products, technologies, and network solutions is available from various online and printed sources.

- Cisco Marketplace provides a variety of Cisco books, reference guides, and logo merchandise. Visit Cisco Marketplace, the company store, at this URL:
  http://www.cisco.com/go/marketplace/

- The Cisco Product Catalog describes the networking products offered by Cisco Systems, as well as ordering and customer support services. Access the Cisco Product Catalog at this URL:
  http://cisco.com/univercd/cc/td/doc/pcat/

- Cisco Press publishes a wide range of general networking, training and certification titles. Both new and experienced users will benefit from these publications. For current Cisco Press titles and other information, go to Cisco Press at this URL:
  http://www.ciscopress.com
Packet magazine is the Cisco Systems technical user magazine for maximizing Internet and networking investments. Each quarter, Packet delivers coverage of the latest industry trends, technology breakthroughs, and Cisco products and solutions, as well as network deployment and troubleshooting tips, configuration examples, customer case studies, certification and training information, and links to scores of in-depth online resources. You can access Packet magazine at this URL:

http://www.cisco.com/packet

iQ Magazine is the quarterly publication from Cisco Systems designed to help growing companies learn how they can use technology to increase revenue, streamline their business, and expand services. The publication identifies the challenges facing these companies and the technologies to help solve them, using real-world case studies and business strategies to help readers make sound technology investment decisions. You can access iQ Magazine at this URL:

http://www.cisco.com/go/iqmagazine

Internet Protocol Journal is a quarterly journal published by Cisco Systems for engineering professionals involved in designing, developing, and operating public and private internets and intranets. You can access the Internet Protocol Journal at this URL:

http://www.cisco.com/ipj

World-class networking training is available from Cisco. You can view current offerings at this URL:

Content Load-Balancing Overview

Content load balancing is how the CSS handles requests for content to a specific destination. To assist you in understanding what occurs when load balancing occurs on the CSS, this chapter provides information about the relationship of service, owner, and content rules, and describes how the CSS handles TCP and UDP traffic. This chapter contains the following sections:

- Service, Owner, and Content Rule Overview
- Overview of CSS Flow

Information in this chapter applies to all CSS models except where noted.

Service, Owner, and Content Rule Overview

The CSS enables you to configure services, owners, and content rules in order to direct requests for content to a specific destination service (for example, a server or a port on a server). By configuring services, owners, and content rules, you optimize and control how the CSS handles each request for specific content. Services, owners, and content rules are described below:

- A **service** is a destination location where a piece of content resides physically (a local or remote server and port). You add services to content rules. Adding a service to a content rule includes it in the resource pool that the CSS uses for load-balancing requests for content. A service may belong to multiple content rules.
• An owner is generally the person or company who contracts the Web hosting service to host their Web content and allocate bandwidth as required. Owners can have multiple content rules.

• A content rule is a hierarchical rule set containing individual rules that describe which content (for example, an .html file) is accessible by visitors to the Web site, how the content is mirrored, on which server the content resides, and how the CSS should process requests for the content. Each rule set must have an owner.

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

When a request for content is made, the CSS:

1. Uses the owner content rule to translate the owner Virtual IP address (VIP) or domain name using Network Address Translation (NAT) to the corresponding service IP address and port.

2. Checks for available services that match the content request.

3. Uses content rules to choose which service can best process the request for content.

4. Applies all content rules to service the request for content (for example, load-balancing method, redirects, failover, stickiness).
Figure 1-1 illustrates the CSS service, owner, and content rule concepts.

**Figure 1-1  Services, Owners, and Content Rules**

Clients request content from www.arrowpoint.com

**CSS NATs**

www.arrowpoint.com to VIP 192.1.1.43

Owner: arrowpoint
content rule: arrowrule1

- VIP 192.1.1.43
- service Serv1
- protocol tcp
- port 80
- round-robin
- activate rule

Serv1 contains content for arrowpoint.com

Client request content from www.dogsRus.com

**CSS NATs**

www.dogsRus.com to VIP 172.1.1.89

Owner: frednmandi
content rule: fredrules

- VIP 172.1.1.89
- service Serv2
- protocol tcp
- port 8080
- activate rule

Serv2 contains content for dogsRus.com
The following chapters provide information on configuring services, owners, and content rules:

- Chapter 1, Configuring Services
- Chapter 8, Configuring Owners
- Chapter 9, Configuring Content Rules

For information on how TCP and UDP traffic flows through the CSS, see the following "Overview of CSS Flow" section.

### Overview of CSS Flow

A flow is the transfer of a sequence of related packets over a TCP or UDP connection between a source (client) and a destination (server) through the CSS. All packets in an ingress flow (traffic entering the CSS) share a common 5-tuple consisting of:

- Source address
- Destination address
- Protocol
- Source port
- Destination port

TCP flows are bidirectional (Figure 1-2). Packets move from the client to the server and from the server to the client through the CSS. Strictly speaking, a TCP connection consists of two flows, one in each direction. A TCP flow begins with a SYN and ends with an ACK to a FIN/ACK, or an RST.

![Figure 1-2 Example of a TCP Flow](image)
UDP flows (Figure 1-3) are typically unidirectional (for example, streaming audio transmitted by a server to a client). A UDP flow has no definitive beginning or end and is considered completed only after a period of time has elapsed during which the destination device receives no packets that share the same addresses, protocol, and ports that defined the original flow.

**Figure 1-3 Example of a UDP Flow**

A CSS uses data structures called flow control blocks (FCBs) to set up and keep track of ingress flows. FCBs contain all the information the CSS needs to process and manage flows. The creation of an FCB from flow information is called flow mapping. The flow manager in each module session processor is responsible for FCB creation and flow mapping.

Each unidirectional flow uses one FCB. Therefore, a TCP flow uses two FCBs and a UDP flow typically uses one FCB. Front-end SSL, which runs over TCP, requires four FCBs and back-end SSL adds two more FCBs for a total of six FCBs per full-duplex SSL connection. For more information about SSL, refer to the *Cisco Content Services Switch SSL Configuration Guide*. 
Each client-CSS-server connection consists of two parts (Figure 1-4):

- Front-end - Connection between a client and the CSS
- Back-end - Connection between the CSS and a server

A Layer 5 flow begins with a client request for content. After the D-proxy resolves the DNS request (for example, a client types a URL in a Web browser) and points the client to the CSS virtual IP address (VIP), the CSS establishes the front-end TCP connection with the client using the TCP 3-way handshake (Figure 1-5).

When it establishes a Layer 5 flow, a CSS “spoofs” the back-end TCP connection by acting as a proxy for the destination device (server) for the client SYN. In other words, the CSS responds to the client SYN with a SYN/ACK before the CSS sets up the back-end TCP connection with the server.
This process is referred to as delayed binding. Delayed binding causes the client to respond with an ACK and an HTTP GET request. This process allows the CSS to gather the information it needs to select the best service (a server port where content resides or an application running on a server such as FTP) for the content request.

The CSS examines the HTTP header and URL in the HTTP request method (for example, GET, HEAD, or POST). Based on the information in the HTTP header, the URL, and the content rules configured on the CSS, the CSS selects the best site and the best service to satisfy the request. A CSS bases service selection (server load balancing) on factors such as:

- Content rule match
- Service availability
- Service load
- Cookies
- Source IP address

For more information about CSS server load balancing (SLB), see the “Service, Owner, and Content Rule Overview” section.

After the CSS selects the best service to provide the requested content to the client, the CSS establishes the back-end connection with the service using the TCP 3-way handshake and splices the front-end and back-end connections together. The CSS forwards the content request from the client to the service (Figure 1-6). The service responds to the client through the CSS. For the remaining life of the flow, the CSS switches the packets between the client and the service, and performs network address translation (NAT) and other packet transformations as required.

Figure 1-6  Setting Up the Back-End TCP Connection - Delayed Binding
For subsequent content requests from the same client over the same TCP connection (HTTP 1.1 and higher), the CSS attempts to maintain the back-end connection with the service that provided the content for the first HTTP request by default. This condition is called persistence.

During the life of a persistent connection, a CSS must determine if 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.

You can configure the CSS to perform one of the following functions when it becomes necessary to move a client to a new service:

- HTTP redirection - Using the persistence reset redirect command, a CSS closes the back-end connection by sending a RST to the service (Figure 1-7). The CSS sends a 302 redirect to the client’s browser to tell the browser to reconnect using the same DNS name, but this time the HTTP request matches on a different content rule. The CSS then establishes a new flow between the client and the best service.

Figure 1-7 Example of HTTP Redirection

![Figure 1-7 Example of HTTP Redirection](image)

Client

302 Redirect

CSS

RST

Server
Service remapping - Using the **persistence reset remap** command, a CSS closes only the back-end connection by sending a RST to the service (server 1 in Figure 1-8), then establishes a new back-end connection with service server 2 and splices the back-end and front-end connections together. The CSS forwards the content request from the client to server 2. Packets now flow between the client and server 2.

For more information about persistence, HTTP redirection, and service remapping, see Chapter 9, Configuring Content Rules.

**Figure 1-8  Example of Remapping the Back-end Connection**

Periodically, the CSS flow manager tears down old, idle flows and reclaims the system resources (FCBs). This process is called **flow resource reclamation**. It is also referred to as **flow cleanup** or **garbage collection**. Flow resource reclamation involves removing FCBs from the TCP and UDP lists. For optimal performance, the CSS reuses FCBs that are no longer needed for flows.

Normally, flow cleanup occurs at a rate that is directly related to the total number of flows that are currently active on a CSS. A CSS always cleans up UDP flows. For TCP flows, a CSS reclaims resources when the number of used FCBs reaches a certain percentage of the total FCBs. A CSS also cleans up long-lived TCP flows that have received a FIN or a RST, or whose timeout values have been met. You can configure various commands to change the default flow-cleanup behavior of the CSS.
In some instances it may not be desirable for the CSS to clean up idle TCP flows. For example, during a connection to a database server that must permanently remain active even when no data passes through the connection. If you observe the CSS dropping long-lived idle connections that need to be maintained you can configure the following TCP flow commands:

- **flow permanent** command - Creates permanent TCP or UDP ports that are not reclaimed
- **flow-timeout-multiplier** command - Configures flow inactivity timeout values for TCP and UDP flows on a per content rule and per source group basis

Refer to Chapter 2, Configuring Flow and Port Mapping Parameters for information on the commands you can use to control how the CSS handles and cleans up TCP and UDP flows.
Configuring Flow and Port Mapping Parameters

This chapter describes how to configure flow and port mapping parameters for the CSS. Information in this chapter applies to all CSS models, except where noted.

This chapter contains the following major sections:

- Configuring Flow Parameters
- Configuring Flow Inactivity Timeouts on Content Rules and Source Groups
- Configuring Flow Processing for Fragmented IP Packets
- Configuring a CSS to Send a TCP Reset if a VIP Is Unavailable
- Configuring the Flow-State Table
- Configuring CSS Port Mapping

For information on how the CSS handles flows, see Chapter 1, Content Load-Balancing Overview.

Configuring Flow Parameters

To configure flow parameters for the CSS, use the `flow` command. The options for this global configuration mode command are as follows:

- `flow permanent` - Creates permanent TCP or UDP ports that are not reclaimed
- `flow reserve-clean` - Reclaims interval flows with port numbers less than or equal to 23
- **flow tcp-mss** - Configures the TCP maximum segment size that the CSS expects to receive from the transmitting device
- **flow statistics** - Displays statistics on currently allocated flows

---

**Note**

Flow parameter setup by the CSS is restricted on the following TCP or UDP ports: 67 (BOOTP server), 68 (BOOTP client), 137 (NETBIOS name service), 138 (NETBIOS datagram service), 161 (SNMP), 162 (SNMP traps), 520 (RIP), and 8089 (restricted UDP only).

This section includes the following topics:

- Configuring Permanent Connections for TCP or UDP Ports
- Configuring TCP Maximum Segment Size
- Reclaiming Reserved Telnet and FTP Control Ports
- Showing Flow Statistics

### Configuring Permanent Connections for TCP or UDP Ports

The CSS allows you to configure a maximum of 20 TCP or UDP ports that have permanent connections and will not be reclaimed by the CSS when the flows are inactive. Use the `flow permanent port1 portnumber` (through `flow permanent port 20 portnumber`) commands to configure a TCP or UDP port as a permanent connection. Enter a port number from 0 to 65535. The default is 0.

A CSS may reclaim flows that have not received an ACK or content request after approximately 15 seconds. To prevent the CSS from reclaiming flows to a specific source or destination port, specify one of the `flow permanent port` commands and identify the TCP or UDP port number you do not want reclaimed.

For example, to configure port 80 as a permanent connection, enter:

```
(config) flow permanent port1 80
```

To reset the port number for port1 to 0, enter:

```
(config) no flow permanent port1
```
We recommend that when you configure a `flow permanent port` command you also enable the `cmd-sched` command to periodically remove the permanent port and allow for cleanup. For details on using the `cmd-sched` command to configure the scheduled execution of any CLI command, refer to the *Cisco Content Services Switch Administration Guide*.

## Configuring TCP Maximum Segment Size

The maximum segment size (MSS) is the largest amount of TCP data that can be transmitted in one segment. The need for a smaller MSS between devices may be necessary in rare instances due to network restrictions between devices. Use the `flow tcp-mss` command to configure the TCP MSS that the CSS expects to receive from the transmitting device. The `flow tcp-mss` command changes the MSS value in the TCP header OPTIONS field of a SYN segment, to reduce the MSS from the default value of 1460 bytes.

The `flow tcp-mss` command applies only when the client is accessing a Layer 5 content rule. The CSS does not negotiate TCP maximum segment size for Layer 3 or Layer 4 content rules.

Enter a maximum segment size (in bytes) from 1 to 1460. The default is 1460 bytes. Use the `no` form of the command to reset the TCP maximum segment size back to the default value of 1460 bytes.

**Caution**

Do not define a smaller than necessary TCP maximum segment size with the `flow tcp-mss` command. Smaller payloads may be less efficient due to increased overhead.

To configure a TCP maximum segment size of 1400 bytes, enter:

```
(config)# flow tcp-mss 1400
```

To reset the TCP maximum segment size to the default value of 1460 bytes, enter:

```
(config)# no flow tcp-mss
```
Reclaiming Reserved Telnet and FTP Control Ports

Control ports have port numbers less than or equal to 23. When the CSS determines that one of these ports has a flow with asymmetrical routing, it reclaims the port.

Use the `flow reserve-clean` command to define how often the CSS scans flows from reserved Telnet and FTP control ports to reclaim them. Enter the `flow reserve-clean` time, in seconds, as the interval the CSS uses to scan flows. Enter an integer from 0 to 100. The default is 10. To disable the port reclaiming process, enter a flow reserve-clean value of 0.

For example, to specify an interval of 36 seconds:

```
(config) # flow reserve-clean 36
```

To disable flow cleanup on Telnet and FTP control ports, enter:

```
(config) # no flow reserve-clean
```

Showing Flow Statistics

Use the `flow statistics` command to display statistics on active flows or Flow Control Blocks (FCBs) on the CSS interfaces.

**Note**
To display summary information about redundant dormant flows, use the `flow statistics dormant` command. Refer to the *Cisco Content Services Switch Redundancy Configuration Guide* for details.

Table 2-1 describes the fields in the `flow statistics` output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow Manager Statistics - Slot $n$, Subslot $n$</td>
<td>Flow manager statistics for the module in the specified slot and subslot in the CSS chassis. The flow manager is responsible for FCB creation and flow mapping.</td>
</tr>
<tr>
<td>UDP Flows per Second</td>
<td>The number of UDP flows received per second by the CSS (displayed as current, high, and average flows).</td>
</tr>
</tbody>
</table>
Table 2-1  Field Descriptions for the flow statistics Command (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Flows per Second</td>
<td>The number of TCP flows received per second by the CSS (displayed as current, high, and average flows).</td>
</tr>
<tr>
<td>Total Flows Per Second</td>
<td>Total number of TCP and UDP flows received per second by the CSS (displayed as current, high, and average flows).</td>
</tr>
<tr>
<td>Hits Per Second</td>
<td>Cumulative number of content rule hits for all configured rules (displayed as current, high, and average hits). For a Layer 3 and Layer 4 rule, this value increments by one hit for each session. For a Layer 5 rule, this value increments by one hit for each HTTP method.</td>
</tr>
<tr>
<td>Number of Allocated Flows (non-purged)</td>
<td>The allocated FCBs for the flow manager for this module. At CSS boot time, the flow manager designates a number of free flows. If the flow manager runs out of free flows, it can allocate up to a maximum number of FCBs in blocks of 200. The initial free and maximum flows are based on the amount of memory available in the module. The allocated flows equal the cumulative number of active flows for each port on the module.</td>
</tr>
<tr>
<td>Number of Free Flows</td>
<td>The number of FCBs that the flow manager allocates from memory both at initialization time and during system run time. This value is taken from the memory displayed in the <code>show system-resources</code> command.</td>
</tr>
<tr>
<td>Number of Allocated Fast-Path FCBs</td>
<td>The total number of FCBs in use by the fastpath software in the CSS. A TCP flow uses two FCBs and a UDP flow typically uses one FCB.</td>
</tr>
<tr>
<td>Number of Free Fast-Path FCBs</td>
<td>The total number of FCBs available for use by the fastpath software in the CSS.</td>
</tr>
<tr>
<td>Aggregate Flow Statistics Per Port</td>
<td>The summary of TCP and UDP flows for each active port.</td>
</tr>
<tr>
<td>Port</td>
<td>The ingress slot and subslot on the CSS 11501, CSS 11503, or CSS 11506 (for example, 2/1).</td>
</tr>
</tbody>
</table>
Configuring Flow Inactivity Timeouts on Content Rules and Source Groups

Use this feature with a CSS to configure flow inactivity timeout values for TCP and UDP flows on a per content rule and per source group basis. This timeout value is not the frequency with which a CSS reclaims flow resources, but is the time period that must elapse for an idle flow before the CSS marks the flow for cleanup.

Timeout Value Precedence

The CSS uses the following guidelines in the order presented when reclaiming flow resources:

1. If a flow matches on a content rule, the CSS checks for a user-configured timeout value and uses that timeout value if one exists.
2. If the flow matches on a source group, the CSS checks for a user-configured timeout value and uses that timeout value if one exists.
3. If you have configured a permanent port using the `flow permanent port` command (see the “Configuring Permanent Connections for TCP or UDP Ports” section), the CSS sets the flow timeout value to 0, which means that the flow should never time out.
4. If none of the above conditions are met, then the CSS uses the default timeout value for the protocol type. For more information, see the “Displaying Flow Timeout Statistics” section.

Table 2-1 Field Descriptions for the `flow statistics` Command (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>The total number of active TCP and UDP flows.</td>
</tr>
<tr>
<td>Total</td>
<td>The cumulative total of flows.</td>
</tr>
<tr>
<td>TCP</td>
<td>The current number of active TCP flows.</td>
</tr>
<tr>
<td>UDP</td>
<td>The current number of active UDP flows.</td>
</tr>
</tbody>
</table>
Configuring Flow Timeouts

To specify the number of seconds for which an idle flow can exist before the CSS tears it down, use the `flow-timeout-multiplier` command. Specify this command in owner-content or group configuration mode. The syntax for this command is:

```
flow-timeout-multiplier number
```

**Note**

If you configure a source group with destination services for client source NATing, you need to configure the `flow-timeout multiplier` command only on the content rule. The CSS sets the same flow timeout value for flows in both directions. If you configure different timeout values on the content rule and on the source group, the CSS uses the timeout value configured on the content rule for both flows.

Enter an integer for the `number` variable from 0 to 65534. The CSS multiplies the value you specify by 16 to calculate the flow timeout in seconds. The default value depends on the TCP or UDP port number (see the “Displaying Flow Timeout Statistics” section). This default value applies only to flows that you create under a content rule or source group.

A value of zero (no timeout) instructs the CSS to never tear down the flow, resulting in a permanent flow and lost resources. Specifying a value of zero is equivalent to entering the `flow permanent port` command (see the “Configuring Permanent Connections for TCP or UDP Ports” section).

**Note**

We do not recommend that you set the `flow-timeout multiplier` command to 0 for UDP flows on Layer 3 and Layer 4 content rules. If the value is set to 0, the CSS does not clean up the resources for the UDP flows.

**Note**

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. Use the `flow-timeout-multiplier` command on the associated content rule to configure the timeout to a value that can accommodate the expected duration of the FTP file transfers.
These two examples show flow timeout periods of 80 seconds:

```
(config-owner-content[cisco-rule1])# flow-timeout-multiplier 5
(config-group[group1])# flow-timeout-multiplier 5
```

To disable the configured `flow-timeout-multiplier` value and restore the default timeout for the port type, enter:

```
(config-owner-content[cisco-rule1])# no flow-timeout
(config-group[group1])# no flow-timeout
```

### Displaying Flow Timeout Statistics

Use the `show flow-timeout default` command to display the default timeout values for TCP and UDP ports and applications. The default values are not user configurable. Table 2-2 shows the fields in the `show flow-timeout default` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/IP Port</td>
<td>Default TCP or UDP port numbers.</td>
</tr>
<tr>
<td>Application</td>
<td>Names of the default TCP or UDP applications.</td>
</tr>
<tr>
<td>Inactivity</td>
<td>Default flow inactivity timeouts, in seconds, for the TCP or UDP port. If</td>
</tr>
<tr>
<td>Seconds</td>
<td>a flow is idle for the amount of time specified in the timeout value, the CSS</td>
</tr>
<tr>
<td></td>
<td>tears down the flow and reclaims the flow resources.</td>
</tr>
</tbody>
</table>

Use the `show flow-timeout configured` command to display the configured flow timeout values. The command output includes the content rule or source group for which you configured the flow timeout value.

Table 2-3 describes the fields in the `show flow-timeout configured` command output.
Configuring Flow Processing for Fragmented IP Packets

If you configure a flow timeout value in a content rule or a source group, the show rule or show group command output includes an additional field called Flow Timeout Multiplier. This field contains the configured timeout value assigned to flows that match on the rule or group.

Displaying Content Rule and Source Group Information

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>TCP or UDP port number.</td>
</tr>
<tr>
<td>Content Rule</td>
<td>Name of the content rule for which the flow timeout is configured.</td>
</tr>
<tr>
<td>Source Group</td>
<td>Name of the source group for which the flow timeout is configured.</td>
</tr>
<tr>
<td>Timeout</td>
<td>Configured activity timeout in 16-second increments for the TCP or UDP port. When this time period elapses for an idle flow, the CSS tears down the connection and reclaims the FCBs.</td>
</tr>
</tbody>
</table>

Configuring Flow Processing for Fragmented IP Packets

By default, a CSS does not process fragmented TCP and UDP IP packets (IP fragments) in the flow path, but simply routes them according to standard IP routing practices. As a result, IP fragments do not match on configuration items such as content rules and source groups and, therefore, the CSS does not NAT or load balance the fragments.

When you enable flow processing for IP fragments, a CSS processes the IP fragments in the flow path using the IP address and TCP port or UDP port information in the IP header and in the TCP or UDP header. The CSS then forwards and NATs the individual fragments of a packet based on the configured content rules and source groups matched by the fragments.
This feature supports only Layer 3 and Layer 4 content rules. For Layer 5 content rules, use the `flow tcp-mss` command. For details about the `flow-tcp-mss` command, see the “Configuring Flow Parameters” section.

Use this feature to support:

- Microsoft Media Server using Microsoft Media Server UDP (MMSU) protocol and other applications that fragment UDP IP packets
- E-mail or other applications that fragment TCP IP packets
- Applications and devices that do not support MTU path discovery
- Network configurations where TCP and UDP IP packets must be fragmented to traverse the network

**Note**
Whenever possible, avoid applications or network configurations that create IP fragments. This feature provides support for those edge conditions where IP fragmentation is unavoidable.

This section describes how to configure flow processing for fragmented IP packets. It includes the following topics:

- What Is IP Packet Fragmentation?
- Enabling Flow Processing for Fragmented IP Packets
- Configuring the Maximum Assembled Size
- Configuring the Minimum Fragment Size
- Resetting IP Fragment Statistics
- Displaying IP Fragment Statistics

**What Is IP Packet Fragmentation?**

An IP fragment is a part of a larger complete IP packet. IP packets require fragmentation when the next-hop network’s maximum transmit unit (MTU) is less than the incoming packet size. The transmitting device divides the packet into smaller pieces that the network medium can accommodate and copies the packet IP header into each fragment. Packets can be fragmented by the source host, intermediate routers, and other network devices.
IP packet fragmentation is generally considered an undesirable condition because fragmentation and subsequent reassembly of packets cause additional CPU and network overhead. However, despite the best efforts of network designers, some fragmentation is inevitable because of the different network media with varying MTUs that support the IP protocol.

For more information about IP packet fragmentation, refer to RFC 791 and RFC 815.

**Configuration Restrictions**

The following TCP applications are not supported when a CSS receives fragmented packets and flow processing for TCP IP fragments is enabled:

- Layer 5 content rules when a client request is fragmented. There is no fallback to a Layer 4 or Layer 3 rule if configured.
- HTTPS client (SSL) with an SSL module for front-end SSL termination.
- HTTPS client (SSL) without an SSL module, with the `advanced-balance-ssl` command configured.
- FTP control channel.
- ArrowPoint cookies.

**Note**
The CSS cannot inspect the UDP/TCP payload of a fragmented IP packet to make a load-balancing decision.

**Enabling Flow Processing for Fragmented IP Packets**

To allow a CSS to flow-process IP fragments, use the `udp-ip-fragment-enabled` or the `tcp-ip-fragment-enabled` command in global configuration mode. By default, this feature is disabled.

**Note**
The `ip-fragment-enabled` command has been deprecated (obsoleted). The CSS automatically converts the `ip-fragment-enabled` command to the `udp-ip-fragment-enabled` command.
To reset the default behavior of the CSS to forward IP fragments, use the no form of the command.

For example, enter:

```bash
(config)# no udp-ip-fragment-enabled
(config)# no tcp-ip-fragment-enabled
```

**Note**

This feature performs content rule-based forwarding using Layer 3 (IP address) and Layer 4 (TCP or UDP port) information in the IP header and the TCP or UDP header. Layer 5 forwarding decisions for IP fragments, based on the packet payload (data), are not supported.

### Configuring the Maximum Assembled Size

The maximum assembled size is the total length of an IP packet if all the IP fragments were assembled into the original packet. Assembled IP packets should be no larger than 64 KB. As the CSS receives the IP fragments, it checks the fragments against the maximum assembled size value. If a fragment IP offset plus the IP payload (data) length is greater than the configured maximum assembled size, the CSS increments the Max Assembled Size error field in the `show ip-fragment-stats` command output and discards the packet. See the “Displaying IP Fragment Statistics” section.

**Note**

To eliminate unnecessary processing overhead, the CSS does not reassemble fragmented IP packets.

To specify the maximum assembled size for TCP and UDP IP fragments, use the `ip-fragment max-assembled-size` command. The syntax of this global configuration mode command is:

```bash
ip-fragment max-assembled-size number
```

The `number` variable specifies the maximum size of an assembled packet in bytes. Enter an integer from 2048 to 65535. The default is 5120 bytes.

For example, enter:

```bash
(config)# ip-fragment max-assembled-size 4096
```
To restore the default maximum IP fragment assembled size to 5120 bytes, use the `no` form of the command.

For example, enter:

```
(config) # no ip-fragment max-assembled-size
```

### Configuring the Minimum Fragment Size

The minimum fragment size is the smallest IP payload in an IP fragment that a CSS accepts. As the CSS receives the IP fragments, it checks the fragments against the minimum fragment size value. If a fragment IP payload length is less than the configured minimum fragment size, the CSS increments the Less Than Min Size error field in the `show ip-fragment-stats` command output and discards the packet. See the “Displaying IP Fragment Statistics” section.

To specify the smallest IP fragment payload for TCP and UDP IP fragments based on your applications, use the `ip-fragment min-fragment-size` command. This command also provides protection against fragment attacks, which can consist of a chain of valid-looking, but very small, fragments.

The syntax of this global configuration mode command is:

```
ip-fragment min-fragment-size number
```

The `number` variable specifies the size of the smallest IP fragment payload that the CSS supports in bytes. Enter an integer from 64 to 1024. The default is 1024 bytes.

For example, enter:

```
(config) # ip-fragment min-fragment-size 256
```

**Note**

Requiring that the minimum fragment size be at least 64 bytes guarantees that the IP header and the TCP or UDP header information is present in the first fragment.

To restore the default minimum IP fragment payload size to 1024 bytes, use the `no` form of the command.

For example, enter:

```
(config) # no ip-fragment min-fragment-size
```
Displaying IP Fragment Statistics

To display the status, statistics, and error counts associated with TCP and UDP IP fragment processing, use the `show ip-fragment-stats` command in any mode.

Table 2-4 describes the fields for the `show ip-fragment-stats` command output.

Table 2-4  Field Descriptions for the `show ip-fragment-stats` Command

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IP Fragment Status</strong></td>
<td></td>
</tr>
<tr>
<td>UDP State</td>
<td>Configured state of the UDP IP fragment feature, Enabled or Disabled.</td>
</tr>
<tr>
<td>TCP State</td>
<td>Configured state of the TCP IP fragment feature, Enabled or Disabled.</td>
</tr>
<tr>
<td>Min Fragment Size</td>
<td>Configured minimum fragment IP payload size.</td>
</tr>
<tr>
<td>Max Assembled Size</td>
<td>Configured maximum assembled IP packet size.</td>
</tr>
<tr>
<td><strong>IP Fragment Statistics</strong></td>
<td></td>
</tr>
<tr>
<td>Packets Tracked</td>
<td>Current, high, and total number of fragmented IP packets that the CSS tracked. This field contains the number of actual packets tracked, not the number of fragments.</td>
</tr>
<tr>
<td>Fragments Buffered</td>
<td>Current, high, and total number of buffered IP fragments from all packets that the CSS tracked.</td>
</tr>
<tr>
<td>Packets Completed</td>
<td>Number of successfully processed IP packets that were fragmented.</td>
</tr>
<tr>
<td>Longest Frag Chain</td>
<td>Longest IP fragment chain that constituted any one fragmented IP packet. An IP fragment chain is the number of fragments that make up the original packet.</td>
</tr>
<tr>
<td>Largest Asm Packet</td>
<td>Largest IP length of an IP fragmented packet that the CSS received.</td>
</tr>
</tbody>
</table>
Table 2-4  Field Descriptions for the show ip-fragment-stats Command (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smallest Fragment</td>
<td>Smallest fragment IP payload length that the CSS received. This field does not include the last fragment in any IP fragment because its payload can be any size.</td>
</tr>
<tr>
<td>IP Fragment Errors</td>
<td></td>
</tr>
<tr>
<td>No Tracking Entry</td>
<td>While receiving a fragment of a new packet, the CSS could not obtain a fragment tracking entry. This error can occur if the CSS memory is low or used completely.</td>
</tr>
<tr>
<td>Could Not Buffer</td>
<td>CSS received a fragment, but could not buffer it because the CSS was low on buffers.</td>
</tr>
<tr>
<td>Duplicate Fragment</td>
<td>CSS detected a duplicate offset or last fragment.</td>
</tr>
<tr>
<td>Validating Fragments</td>
<td>After the CSS received all the IP fragments, it attempted to validate the fragments, but found overlapping offsets, short offsets, or other possible denial of service (DoS) fragment attack conditions.</td>
</tr>
<tr>
<td>Inserting Fragment</td>
<td>While the CSS was inserting fragments into the fragment chain on the tracking entry, it encountered duplicate fragments, fragments of less than the configured minimum fragment size, or a total assembled size greater than the configured maximum assembled size.</td>
</tr>
<tr>
<td>Less Than Min Size</td>
<td>CSS received an IP fragment (not the last fragment) with an IP payload that was less than the configured minimum fragment size.</td>
</tr>
<tr>
<td>Max Assembled Size</td>
<td>After the CSS received a fragment, the calculated total length of the assembled IP packet was greater than the configured maximum assembled size.</td>
</tr>
<tr>
<td>Collection Timeout</td>
<td>While the CSS was waiting to receive IP fragments, too much time elapsed.</td>
</tr>
</tbody>
</table>
Resetting IP Fragment Statistics

To reset the TCP and UDP IP fragment statistics, use the `zero ip-fragment-stats` command in any mode. This command resets the values of the statistics in the IP Fragment Statistics and IP Fragment Errors sections of the `show ip-fragment-stats` command output to zero.

For more information about the `show ip-fragment-stats` command, see the “Displaying IP Fragment Statistics” section.

---

**Table 2-4  Field Descriptions for the show ip-fragment-stats Command (continued)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow Timeout</td>
<td>After the CSS received all fragments of an IP packet and a fragment was sent to flow processing, the entry timed out before the fragment returned.</td>
</tr>
<tr>
<td>IPv4 Header</td>
<td>The CSS received a fragment with an invalid IPv4 header length compared with the total IP fragment size.</td>
</tr>
<tr>
<td>RxQueue Full</td>
<td>The CSS flow-processing receive queue for IP fragments was full. The CSS discarded the IP fragments.</td>
</tr>
</tbody>
</table>
Configuring a CSS to Send a TCP Reset if a VIP Is Unavailable

If a Layer 3 or Layer 4 content rule VIP that a CSS is hosting is unavailable, the CSS, by default:

- Rejects a TCP packet sent to that VIP by a client
- Drops the TCP packet

This behavior can occur when a packet:

- Matches a Layer 3 or Layer 4 content rule that has no active services
- Matches a Layer 3 or Layer 4 content rule with services that have reached their maximum number of connections

If a CSS rejects a TCP packet, the client can retransmit the packet. If no services become available for a matching Layer 3 or Layer 4 content rule, the client application becomes unresponsive and either the connection or the application eventually times out. It takes the application a variable amount of time to time out. The latency caused by this time-out process is unacceptable for some applications.

This feature allows you to configure a CSS to send a TCP RST to the client in response to the TCP packet if the VIP is unavailable. If the application receives the TCP RST, the application stops retransmitting the packet and usually displays an error message about the failed connection attempt.

**Note**

For Layer 5 spoofed connections, if the CSS rejects the content request, it always sends a TCP RST to the client. This behavior has not changed.

Applications where this feature may be useful include:

- Web browsers
- Telnet
- FTP

To configure a CSS to send a TCP RST to a client when a VIP is unavailable, use the `flow tcp-reset-vip-unavailable` command in global configuration mode. The CSS sends the TCP RST only in response to a TCP packet that is destined for a VIP that the CSS is hosting and only if that VIP is unavailable.
For example, enter:

```
(config)# flow tcp-reset-vip-unavailable
```

To return the CSS behavior to the default of dropping the TCP packet if a VIP is unavailable, enter:

```
(config)# no flow tcp-reset-vip-unavailable
```

To display the number of TCP RSTs that a CSS sent because a VIP was unavailable, enter the `show ip statistics` command. For more information on the `show ip statistics` command, refer to the *Cisco Content Services Switch Routing and Bridging Configuration Guide*.

### Configuring the Flow-State Table

A CSS sets up flows for most TCP and UDP traffic. However, for packets that use specific ports, a CSS does not set up flows. The default behavior for a CSS is not to set up flows for the ports listed in Table 2-5.

<table>
<thead>
<tr>
<th>Port</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>67</td>
<td>Bootstrap Protocol (BOOTP) Server</td>
</tr>
<tr>
<td>68</td>
<td>BOOTP Client</td>
</tr>
<tr>
<td>137</td>
<td>NETBIOS Name Service</td>
</tr>
<tr>
<td>138</td>
<td>NETBIOS Datagram Service</td>
</tr>
<tr>
<td>161</td>
<td>Simple Network Management Protocol (SNMP)</td>
</tr>
<tr>
<td>162</td>
<td>SNMP Traps</td>
</tr>
<tr>
<td>520</td>
<td>Routing Information Protocol (RIP)</td>
</tr>
<tr>
<td>8089</td>
<td>Inktomi-UDP</td>
</tr>
</tbody>
</table>
To keep track of the flow states of TCP and UDP ports, a CSS maintains a flow-state table. The 10 default TCP and UDP ports that the CSS preconfigures in the flow-state table are shown in Table 2-6. Of these 10 ports, only DNS (port 53, TCP and UDP) and SIP (port 5060, UDP only) are flow-enabled by default. To set up flows for the other ports in Table 2-6, you must configure them using the flow-state command. All other ports that are not among the 10 default ports are flow-enabled by default.

### Table 2-6 Flow-State Table Default Values

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>NAT-State</th>
<th>Flow-State</th>
<th>Hit-Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>TCP</td>
<td></td>
<td>flow-enable</td>
<td>0</td>
</tr>
<tr>
<td>53</td>
<td>UDP</td>
<td></td>
<td>flow-enable</td>
<td>0</td>
</tr>
<tr>
<td>67</td>
<td>TCP</td>
<td></td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>67</td>
<td>UDP</td>
<td>nat-disable</td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>68</td>
<td>TCP</td>
<td></td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>68</td>
<td>UDP</td>
<td>nat-disable</td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>137</td>
<td>TCP</td>
<td></td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>137</td>
<td>UDP</td>
<td>nat-disable</td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>138</td>
<td>TCP</td>
<td></td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>138</td>
<td>UDP</td>
<td>nat-disable</td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>161</td>
<td>TCP</td>
<td></td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>161</td>
<td>UDP</td>
<td>nat-disable</td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>162</td>
<td>TCP</td>
<td></td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>162</td>
<td>UDP</td>
<td>nat-disable</td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>520</td>
<td>UDP</td>
<td>nat-disable</td>
<td>flow-disable</td>
<td>0</td>
</tr>
<tr>
<td>5060</td>
<td>UDP</td>
<td></td>
<td>flow-enable</td>
<td>0</td>
</tr>
<tr>
<td>8089</td>
<td>UDP</td>
<td>nat-disable</td>
<td>flow-disable</td>
<td>0</td>
</tr>
</tbody>
</table>

You can change the flow states of these preconfigured ports and you can configure 16 additional unique TCP or UDP ports and their flow states. You can also set the Network Address Translation (NAT) state for flow-disabled UDP ports only.
If a CSS receives any traffic destined to its own IP address or VIP address and the port specified by that traffic is flow-disabled and NAT-disabled, the CSS sends an ICMP port unreachable message to the client.

When the CSS receives TCP packets on a flow-disabled port, it does not NAT those packets. In this case, the CSS simply forwards the packets. If a CSS receives such a packet destined to its VIP, the CSS drops the packet.

**Note**
The CSS does not support the NATing of payload data, such as that required for the Real-Time Streaming Protocol (RTSP).

Use the flow-state table:
- For any application that does not use a random client port. For example, if a CSS were to set up a flow for DNS traffic between two authorities that use source port 1024 and destination port 53 repeatedly for a series of requests, all traffic would appear to be the same connection and no load balancing would occur.
- To avoid flow setup overhead for one-time UDP packets.
- To avoid flow setup overhead for known non-NATed packets.

For details on NAT, see [Chapter 5, Configuring Source Groups for Services](#). For details on DNS, refer to the [Cisco Content Services Switch Global Server Load-Balancing Configuration Guide](#).

The CSS supports the Trivial File Transfer Protocol (TFTP) and TFTP-like protocols (where the server selects a random port in its response) only on flow-enabled ports and only when the server is behind a source group (the CSS NATs the server IP address in server-initiated traffic). The CSS does not support these protocols when a client is behind a source group (the CSS NATs the client IP address in client-initiated traffic) or on flow-disabled ports.
Configuring the Flow State of a Port

To set the flow state of any TCP or UDP port, use the `flow-state` command. When you set the flow state of a port to `flow-enable`, the CSS performs content-rule and source-group matching. For flow-disabled UDP ports, you can enable the NAT state independent of the flow state so that the CSS performs NATing and port mapping. In addition to the default ports preconfigured in the table, you can configure a maximum of 16 additional unique TCP or UDP ports using the flow-state table.

The syntax of this global configuration mode command is:

```
flow-state number tcp [flow-enable|flow-disable]
flow-state number udp [flow-enable|flow-disable
{nat-enable|nat-disable}]
```

The options and variables for this global configuration mode command are as follows:

- **number** - Specifies the TCP or UDP port number whose flow state you want to configure. Enter an integer from 1 to 65535.
- **tcp** - Specifies traffic using TCP.
- **udp** - Specifies traffic using UDP.
- **flow-enable** - Enables flows on the specified TCP or UDP port. With this option, the CSS performs full content-rule and source-group matching, including Layer 5 (URL string) content-based load balancing and sticky.
- **flow-disable** - Disables flows on the specified TCP or UDP port. When you disable flows on a port, the CSS does not perform content rule and source group matching. The benefit is that there is no flow setup overhead.
- **nat-enable** - For flow-disabled UDP ports only, enables content-rule and source-group lookups for NAT. With this option, you can use Layer 3 (IP address) and Layer 4 (IP address and destination port) content rules and the sticky table (for example, `sticky-srcip`). However, without the benefit of a flow, the CSS cannot spoof the back-end connection, which is required to make Layer 5 content-based decisions.
- **nat-disable** - For flow-disabled UDP ports only, the CSS does not perform content-rule and source-group lookups for NAT.
Caution

If you configure the `flow-disable` and `nat-disable` options simultaneously on a particular port, content-rule and source-group lookups are no longer available for that port. In this case, the CSS drops TCP packets directed to a VIP address associated with that port. For UDP, the CSS sends an ICMP port unreachable message to the client. The CSS continues to forward packets directed to other IP addresses.

Example 1:

To enable flows for SNMP TCP port 161, enter:

```
(config)# flow-state 161 tcp flow-enable
```

To reset SNMP TCP port 161 to its default value of `flow-disable`, enter:

```
(config)# no flow-state 161 tcp
```

Example 2:

To disable flows for SIP UDP port 5060, enter:

```
(config)# flow-state 5060 udp flow-disable
```

To reset SIP UDP port 5060 to its default value of `flow-enable`, enter:

```
(config)# no flow-state 5060 udp
```

Example 3:

To disable the flow state of SNMP UDP port 162 (SNMP traps) and enable NAT for that port, enter:

```
(config)# flow-state 162 udp flow-disable nat-enable
```

To reset SNMP UDP port 162 to its default settings of `flow-disable` and `nat-disable`, enter:

```
(config)# no flow-state 162 udp
```
Chapter 2 Configuring Flow and Port Mapping Parameters

Configuring the Flow-State Table

Resetting the Flow-State Table Hit Counters

The flow-state table contains hit counters that total the number of hits for each port entry in the table. Use the `zero flow-state-counters` command to reset all the hit counters in the table to zero. For example:

```
(config)# zero flow-state-counters
```

Displaying the Flow-State Table

Use the `show flow-state-table` command to display the flow-state table entries. For the default settings in the flow-state table, see Table 2-6.

Table 2-7 describes the fields in the `show flow-state-table` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Number of the port for which you are displaying flow-state data.</td>
</tr>
<tr>
<td>Protocol</td>
<td>IP protocol (TCP or UDP) associated with the port number.</td>
</tr>
<tr>
<td>NAT-State</td>
<td>Status of network address translation for the port. Possible states are nat-enable, nat-disable, or --------- (state cannot be modified or the field is not applicable in combination with the value in the Flow-State field). The nat-enable and nat-disable states are available only for flow-disabled UDP ports.</td>
</tr>
<tr>
<td>Flow-State</td>
<td>Status of flows for a particular port. Possible states are flow-enable and flow-disable.</td>
</tr>
<tr>
<td>Hit-Count</td>
<td>Number of hits on a particular port.</td>
</tr>
<tr>
<td>*</td>
<td>Indicates that the values in this row of the flow-state table are the default values.</td>
</tr>
</tbody>
</table>
Configuring CSS Port Mapping

This section describes how to globally control the range of port numbers a CSS uses to perform port address translation (PAT) on TCP and UDP source port numbers specified in packets sent to the CSS from clients. The CSS assigns unique port numbers within a configurable range for the source port numbers specified in the packets, then sends the packets to the appropriate server port. When a server initiates a return flow, the packets flow through the CSS. The CSS matches the translated port number with the client that initiated the request and sends the server packets to the appropriate client.

For information about source groups and port mapping, see the “Overview of Source Groups and Port Mapping” section in Chapter 5, Configuring Source Groups for Services.

Overview of Global Port Mapping

Each CSS module (except the SSL module) has one session processor (SP) that is responsible for mastering flows.

- CSS 11501 supports one SP
- CSS 11503 supports a maximum of three SPs
- CSS 11506 supports a maximum of six SPs

The global port mapper in a CSS is called the mega port mapper. The mega port mapper database comprises 16 banks (megamap banks) of 63488 port-map numbers each in each session processor (SP). A CSS uses a source address hash algorithm to select a megamap bank in a particular SP.

For client-side flows, the CSS sends packets to different SPs for flow processing and the flows have access to the source ports in that SP. The CSS performs a simple XOR hash of the TCP or UDP source and destination port numbers to determine the SP that becomes master for that flow. If the port numbers are the same (for example, DNS UDP port 53), then the CSS uses the low order bits of the source and destination IP addresses to calculate the hash value. The CSS uses the hash value to index into a weighted table of SPs and selects the appropriate SP.
When the CSS performs PAT, the master SP for the flow uses a source port from either the global port mapper or a source group, depending on your configuration. (For information about source groups, see Chapter 5, Configuring Source Groups for Services) The CSS chooses a source port so that the hash of it and the destination port will select the same SP for the server-side flow as the SP that mastered the client-side flow.

For the server-side flow from a given destination port, only certain source port numbers hash to the same SP that was used for the client-side flow. For this reason, all ports available to a particular SP are not necessarily eligible for use when establishing the back-end connection. Therefore, the hash algorithm selects only a percentage of the available ports on any one SP.

Each CSS maintains a database of used and available port-map numbers. When a CSS needs to PAT a source port, it uses the next unused port number in its database.

This section includes the following topics:

- Configuring Global Port Mapping
- Displaying Global Port Mapping Statistics
- Configuring No-Flow Port Mapping
- Displaying No-Flow Port Mapping Statistics

### Configuring Global Port Mapping

To control the global PAT for TCP flows on a CSS, use the `global-portmap` command. This command is always enabled.

You can use this command to specify the source-port mapping range on:

- A CSS when you configure a service that uses a nondefault destination port number. A CSS changes a TCP destination port number configured on a service in a content rule when a request hits the content rule and the CSS sends a packet to the selected server. The CSS uses the `global-portmap` command parameters to translate the corresponding client source port number to distinguish it from other clients requesting the same service.

- Redundant Cisco 11500 series CSS peers in an Adaptive Session Redundancy (ASR) configuration. Refer to the Cisco Content Services Switch Redundancy Configuration Guide for information about ASR.
A CSS with back-end server remapping enabled (refer to Chapter 9, Configuring Content Rules).

When you configure a source group, the `portmap` command parameter values take precedence over the `global-portmap` command parameter values. The `portmap disable` command has no effect on TCP flows.

The syntax for this global configuration mode command is:

```
global-portmap base-port number1 range number2
```

The options and variables for this command are:

- **base-port number1** - The starting port number for global port mapping on a CSS. Enter an integer from 2016 to 63456. The default is 2016.

  **Caution** Changing the value of the `number1` variable may cause port conflicts on existing flows.

- **range number2** - The total number of ports in the port-map range that the CSS allocates to each of the 16 megamap banks in each session processor (SP). Each megamap bank in an SP can use the full range of configured ports. Because of the unique source address hash that the CSS uses to select a megamap bank in an SP, more than one SP can use the same port number without a tuple collision.

  **Caution** Changing the value of the `number2` variable may cause port conflicts on existing flows.

Enter an integer from 2048 to 63488. The default is 63488. If you enter a value that is not a multiple of 32, the CSS rounds up the value to the next possible multiple of 32.

**Note** If you enter a port-map range value that exceeds the number of available ports, you get an error. To determine the number of available ports, subtract the starting port number you specify from 65504.
Chapter 2    Configuring Flow and Port Mapping Parameters

Configuring CSS Port Mapping

For example:

```
(config)# global-portmap base-port 3096 range 42308
```

To return the global-portmap command parameters to their default values, enter:

```
(config)# no global-portmap
```

Displaying Global Port Mapping Statistics

Use the `show global-portmap` command to display statistics for global port mapping on a CSS. This command is available in all modes except RMON, URQL, and VLAN configuration modes.

The syntax for this command is:

```
show global-portmap [all-banks [all-sps|slot number1]|number2 [all-sps|slot number1]]
```

The options and variables for this command are as follows:

- **all-banks** - Specifies the display of global port-map information for all port-map banks (0 to 15).
- **all-sps** - Specifies the display of global port-map information for all session processors (SPs) in the CSS.
- **slot number1** - Specifies the chassis slot where the module resides. For a CSS 11503, enter an integer from 1 to 3. For a CSS 11506, enter an integer from 1 to 6.

To display the available active slots in the CSS, enter the `show global-portmap all-banks slot ?` command. If you enter an invalid slot number, the CLI displays values for only the first two parameters listed in Table 2-8.

- **number2** - Specifies the global port-map bank number. Enter an integer from 0 to 15.

To display port mapping statistics for all megamap banks (up to 16) on every active SP in the CSS, enter:

```
(config)# show global-portmap all-banks all-sps
```
To display global port mapping statistics for megamap bank 12 in the SP that resides in slot 3, enter:

```
(config)# show global-portmap 12 slot 3
```

Table 2-8 describes the fields in the `show global-portmap` command output.

### Table 2-8  Field Descriptions for show global-portmap Command

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MegaMap Banks in Use Per SP</td>
<td>The number of global port mapping banks being used in each session processor (SP). There are 16 banks available in each SP. A CSS selects a bank by hashing the source address contained in a packet.</td>
</tr>
<tr>
<td>Configured Base Port</td>
<td>The base-port (starting port number) specified with the <code>global-portmap</code> command or the default of 2016.</td>
</tr>
<tr>
<td>Total Configured Ports</td>
<td>The total number of ports specified with the <code>global-portmap range</code> command or the default of 63488.</td>
</tr>
<tr>
<td>Slot</td>
<td>The number of the slot in the CSS 11503 or CSS 11506 where the specified SP resides.</td>
</tr>
<tr>
<td>MegaMap Bank #</td>
<td>The number of the port mapping bank. Possible values are 0 to 15 for a total of 16 banks for each SP.</td>
</tr>
<tr>
<td>Number Normal Avail Ports</td>
<td>The number of ports available for use by the network address translation algorithm when the source port number is different from the destination port number in a TCP packet.</td>
</tr>
<tr>
<td>Current Mapped Ports</td>
<td>The total number of ports currently in use or mapped.</td>
</tr>
<tr>
<td>Last Normal Mapped Port</td>
<td>The most recent port number used by the network algorithm when the source port number is different from the destination port number in a TCP packet.</td>
</tr>
<tr>
<td>Equal Port Base Port</td>
<td>The starting port number that the network address translation algorithm uses when the source port number is the same as the destination port number in a TCP packet.</td>
</tr>
</tbody>
</table>
Configuring No-Flow Port Mapping

To control the PAT range of DNS UDP source-port numbers greater than 1023 on a CSS, use the `noflow-portmap` command. This command is always enabled. However, before a CSS can use this command, you must use the flow-state command to disable DNS flows on the CSS. See the “Configuring the Flow-State Table” section.

The `portmap` command values configured in a source group take precedence over the `noflow-portmap` command values, unless you configure the `portmap disable` command. Refer to Chapter 3, Configuring Services, for details on configuring the `portmap` commands in a source group.

The syntax for this global configuration mode command is:

```
noflow-portmap base-port number1 range number2
```

The options and variables for this command are:

- `base-port number1` - The starting port number for no-flow (DNS flows are disabled) port mapping on a CSS. Enter an integer from 2016 to 63456. The default is 2016.
Caution
Changing the value of the \textit{number} variable may cause port conflicts on existing flows.

- \texttt{range number} - The total number of ports in the port-map range that the CSS allocates to each session processor (SP). Each SP can use the full range of configured ports.

Caution
Changing the value of the \textit{number} variable may cause port conflicts on existing flows.

Enter an integer from 2048 to 63488. The default is 63488. If you enter a value that is not a multiple of 32, the CSS rounds up the value to the next possible multiple of 32.

Note
If you enter a \texttt{range} value that exceeds the number of available ports, you get an error. To determine the number of available ports, subtract the starting port number from 65504.

For example, to specify a port map range, starting with port 4317, enter:

\begin{verbatim}
(config)# noflow-portmap base-port 4317 range 35421
\end{verbatim}

To reset the starting port number and port-map range to their default values, enter:

\begin{verbatim}
(config)# no noflow-portmap
\end{verbatim}

**Displaying No-Flow Port Mapping Statistics**

Use the \texttt{show noflow-portmap} command to display statistics for no-flow port mapping on a CSS. This command is available in all modes except RMON, URQL, and VLAN configuration modes.

The syntax for this command is:

\begin{verbatim}
show noflow-portmap [all-spslot number]
\end{verbatim}
The options and variables for this command are as follows:

- **all-sps** - Specifies the display of no-flow port-map information for all session processors (SPs) in the CSS.
- **slot number** - The chassis slot number where the module resides. For a CSS 11503, enter an integer from 1 to 3. For a CSS 11506, enter an integer from 1 to 6.

**Note** To display the available active slots in the CSS, enter the `show noflow-portmap slot ?` command. If you enter an invalid slot number, the CLI displays values for only the first two parameters listed in Table 2-9.

For example:

```
(config) # show noflow-portmap slot 3
```

Table 2-9 describes the fields in the `show noflow-portmap` command output.

**Table 2-9 Field Descriptions for show noflow-portmap Command**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configured Base Port</td>
<td>The starting port number specified by the <code>noflow-portmap base-port</code> command or the default of 2016</td>
</tr>
<tr>
<td>Total Configured Ports</td>
<td>The total number of ports specified by the <code>noflow-portmap range</code> command or the default of 63488</td>
</tr>
<tr>
<td>Slot</td>
<td>The number of the slot in the CSS 11503 or CSS 11506 where the specified SP resides</td>
</tr>
<tr>
<td>Number Normal Avail Ports</td>
<td>The number of ports available for use by the network address translation algorithm when the source port number is different from the destination port number in a UDP packet</td>
</tr>
<tr>
<td>Current Mapped Ports</td>
<td>The total number of ports currently in use or mapped</td>
</tr>
<tr>
<td>Last Normal Mapped Port</td>
<td>The most recent port number used by the network address translation algorithm when the source port number is different from the destination port number in a UDP packet</td>
</tr>
</tbody>
</table>
### Table 2-9  Field Descriptions for show noflow-portmap Command (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal Port Base Port</td>
<td>The starting port number that the network address translation algorithm uses when the source port number is the same as the destination port number in a UDP packet.</td>
</tr>
<tr>
<td>Number Equal Avail Ports</td>
<td>The number of ports available for use by the network address translation algorithm when the source port number is the same as the destination port number in a UDP packet.</td>
</tr>
<tr>
<td>High Water Mark</td>
<td>The largest number of ports mapped or in use at one time since the last CSS reboot.</td>
</tr>
<tr>
<td>Last Equal Mapped Port</td>
<td>The last port number used by the network address translation algorithm when the source port number was the same as the destination port number in a UDP packet.</td>
</tr>
<tr>
<td>No Portmap Errors</td>
<td>The number of times that a failure occurred because no ports were available (all ports were mapped).</td>
</tr>
</tbody>
</table>
CHAPTER 3

Configuring Services

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

- Service Configuration Quick Start
- Creating a Service
- Assigning an IP Address to the Service
- Specifying a Port
- Specifying a Protocol
- Specifying a Domain Name
- Specifying an HTTP Redirect String
- Prepending “http://” to a Redirect String or a Domain
- Configuring an Advanced Load-Balancing String
- Configuring a Service HTTP Cookie
- Configuring Weight and Graceful Shutdown
- Specifying a Service Type
- Configuring Service Access
- Bypassing Content Rules on Caches
- Configuring Network Address Translation for Transparent Caches
- Configuring a Service to Bypass a Cache Farm
- Configuring Maximum TCP Connections
- Configuring Keepalives for a Service
Service Configuration Quick Start

Table 3-1 provides a quick overview of the basic steps required to configure a service. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following Table 3-1.

Table 3-1  Service Configuration Quick Start

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter config mode by typing config.</td>
</tr>
<tr>
<td># config</td>
</tr>
<tr>
<td>(config)#</td>
</tr>
<tr>
<td>2. Create services. When you create a service, the CLI enters that service mode, as shown in the command response below. To create additional services, reenter the service command.</td>
</tr>
<tr>
<td>(config)# service serv1</td>
</tr>
<tr>
<td>(config-service[serv1])#</td>
</tr>
<tr>
<td>(config-service[serv1])# service serv2</td>
</tr>
<tr>
<td>(config-service[serv2])#</td>
</tr>
</tbody>
</table>

- Activating a Service
- Suspending a Service
- Showing Service Configurations
- Clearing Service Statistics Counters

For an overview of the association between services, owners, and content rules, see Chapter 1, Content Load-Balancing Overview.

Note

The CSS supports Adaptive Session Redundancy (ASR) on the 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.
3. Assign an IP address to each service. The IP address is the actual IP address of the server.

```
(config-service[serv2])#
(config-service[serv2])# ip address 10.3.6.2
(config-service[serv2])# service serv1
(config-service[serv1])# ip address 10.3.6.1
```

4. Activate each service.

```
(config-service[serv1])# active
(config-service[serv1])# service serv2
(config-service[serv2])# active
(config-service[serv2])# exit
```

5. (Recommended) Display service information to verify your configuration.

```
(config-service[serv2])# show service summary
```

The following running-configuration example shows the results of entering the commands in Table 3-1.

```
!******************************************************** SERVICE ********************************************************
service serv1
  ip address 10.3.6.2
  active

service serv2
  ip address 10.3.6.1
  active
```
Creating a Service

A service can be a destination location or entity that contains and provides Internet content (for example, a server, an application on a server such as FTP, or streaming audio). A service has a name that is associated with an IP address and optionally, a protocol and a port number.

By creating a service, you identify the service and enable the CSS to recognize it. You can then apply content rules to services that allow the CSS to:

- Direct requests for content to the service
- Deny requests for content from the service

Enter a service name from 1 to 31 characters. For example, to create service serv1, enter:

```
(config) service serv1
```

The CSS transitions into the newly created service mode.

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

To delete an existing service, enter:

```
(config) no service serv1
```

Delete service <serv1>, [y/n]: y

Assigning an IP Address to the Service

To enable the CSS to direct requests for content to the appropriate service, you must assign an IP address or range of IP addresses to a service. Assigning an IP address to a service identifies the service to the CSS. When the CSS receives a request for content, it translates the VIP (and potentially, the port) to the service IP address (or addresses) and port.

For example, to assign an IP address to serv1, enter:

```
(config-service[serv1])# ip address 172.16.1.1
```

To restore a service IP address to the default of 0.0.0.0, enter:

```
(config-service[serv1])# no ip address
```
Note

Some services do not require an IP address. Services that does not require an IP address are:

- Services configured with the `ssl-accel` service type
- Services configured with the `redirect` service type
- Services configured with the `bypass-hosttag` command

You must configure these services with a keepalive type of `none`.

---

The `range` option allows you to specify a range of IP addresses starting with the IP address you specified using the `ip address` command. Enter a number from 1 to 65535. The default range is 1. For example, if you enter an IP address of 172.16.1.1 with a range of 10, the IP addresses range from 172.16.1.1 through 172.16.1.10.

For example, enter:

```
(config-service[serv1])# ip address 172.16.1.1 range 10
```

When using the `ip address range` command, use IP addresses that are within the subnet you are using. The CSS does not 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 range as 10.10.10.2 range 400, the CSS will not arp for any IP addresses beyond 10.10.10.254. Using the same example only with a VIP range of 200, the CSS will arp for all IP addresses in the range.

---

Note

The CSS sends keepalives only to the first address in a service range. If you configure a scripted keepalive, it should contain the first address in a service range as one of its arguments.

For the CSS to forward requests to a service on any of the addresses in a range, the CSS must successfully arp for the first address in the range. This behavior is independent of keepalives.
Specifying a Port

The TCP or UDP destination port number is associated with a service. Use the `port` command to specify a service TCP/UDP port number or range of port numbers. Enter the port number as an integer from 0 to 65535. The default is 0 (any).

For example, enter:

```
(config-service[serv1])# port 80
```

To specify a port to be used for keepalives, use the service mode `keepalive port` command.

Use the `range` option to specify a range of port numbers `starting` with the port number you specified using the `port` command. Enter a range number from 1 to 65535. The default range is 1. For example, if you enter a port number of 80 with a range of 10, the port numbers will range from 80 through 89. You can use the `port range` command only on local (default) services.

For example, enter:

```
(config-service[serv1])# port 80 10
```

To set the port to the default of 0 (any), enter:

```
(config-service[serv1])# no port
```

Specifying a Protocol

By default setting, the CSS uses any IP protocol for the service. To specify the service IP protocol that the service is to use, use the `protocol` command. The options for this command are:

- `protocol tcp` - The service uses the TCP protocol suite
- `protocol udp` - The service uses the UDP protocol suite

For example, enter:

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

To set the protocol to the default of `any`, enter:

```
(config-service[serv1])# no protocol
```
Specifying a Domain Name

The CSS uses the configured domain name in the redirect message as the new location for the requested content. The CSS prepends the domain name to the requested URL. If the domain name is not configured, the CSS uses the domain in the host-tag field from the original request. If no host tag is found, the CSS uses the service IP address to generate the redirect. Use the `domain` command to specify the domain name that will be prepended to a requested piece of content when an HTTP redirect service generates a “302 object moved” message for the service.

**Note**
You can use a service redirect domain only on a service type configured to `redirect`. You must specify the `domain` command in order for a redirect service to obtain an applicable HTTP redirect.

**Note**
You cannot configure the `domain` and `(config-service) redirect-string` commands simultaneously on the same service.

**Note**
The `redirect-string` and `(config-service) domain` commands are similar. The CSS returns the `redirect-string` command string as configured. With the `(config-service) domain` command, the CSS prepends the domain to the original requested URL.

Enter the service domain name as an unquoted text string with no spaces and a maximum length of 64 characters.

**Note**
The CSS automatically prepends the domain name with “http://”.

For example, enter:

```
(config-service[serv1])# domain www.arrowpoint.com
```

or

```
(config-service[serv1])# domain 172.16.3.6
```
Specifying an HTTP Redirect String

To clear the redirect domain for this service, enter:

```
(config-service[serv1])# no domain www.arrowpoint.com
```

or

```
(config-service[serv1])# no domain 172.16.3.6
```

The CSS uses the entire configured redirect string as the new location for the requested content. If no string is configured, the CSS prepends the domain configured with the `(config-service) domain` command to the original request. If neither the redirect string nor the domain name is configured, the CSS uses the domain in the host-tag field from the original request combined with the requested HTTP content. If no host tag is found, the CSS uses the IP address of the service to generate the redirect.

Use the `redirect-string` command to specify an HTTP redirect string that the CSS uses when an HTTP redirect service generates a “302 object moved” message for the service. You can configure a redirect string only on a service of type `redirect`.

**Note**

The `redirect-string` and `domain` commands are similar. The CSS returns the `redirect-string` command string exactly as configured. The alternative to using the `redirect-string` command is to configure the `domain` command on the service, where the CSS prepends the configured domain to the originally requested URL. You cannot configure the `redirect-string` and `domain` commands simultaneously on the same service.

The syntax for this service mode command is:

```
redirect-string string
```

Enter the HTTP redirect string as a quoted or an unquoted text string with no spaces and a maximum of 252 characters.
Prepending “http://” to a Redirect String or a Domain

By default, the CSS prepends “http://” to a redirect string or domain. To disable prepending “http://” to a redirect string or domain configured on a service, enter:

```
(config-service[serv1])# no prepend-http
```

Use the `prepend-http` command to prepend “http://” to a redirect string or domain configured for a service.

For example, enter:

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

Configuring an Advanced Load-Balancing String

You can specify an advanced load-balancing method for a content rule that includes stickiness. A content rule is “sticky” when additional sessions from the same user or client are sent to the same service as the first connection, overriding normal load balancing. By default, the advanced balancing method is disabled.

To specify an advanced load-balancing string for a service, use the `string` command. Use this command with the advanced load-balancing methods `url`, `cookie`, or `cookieurl`. For information on advanced load-balancing methods, refer to Chapter 10, Configuring Sticky Parameters for Content Rules.

Enter a string from 1 to 15 characters. For example, enter:

```
(config-service[serv1])# string 172.16.3.6
```

To remove a string from a service, enter:

```
(config-service[serv1])# no string
```
Configuring a Service HTTP Cookie

You can specify an advanced load-balancing method for a content rule that includes stickiness. A content rule is “sticky” when additional sessions from the same user or client are sent to the same service as the first connection, overriding normal load balancing. By default, the advanced balancing method is disabled.

If you are using `advanced-balance cookies, url, or cookieurl` to match an exact string, you must configure the unique string that you want to use for matching each server. Use the `string` command to specify the HTTP cookie for the service. The syntax for this service mode command is:

```
string cookie_name
```

Enter the `cookie_name` as an unquoted text string with no spaces and a maximum of 15 characters.

For example, enter:

```
(config-service[serv1])# string userid3217
```

To remove the cookie for a service, enter:

```
(config-service[serv1])# no string
```

For information on configuring sticky on the CSS, refer to Chapter 10, Configuring Sticky Parameters for Content Rules.

Configuring Weight and Graceful Shutdown

The CSS uses this weight when you configure an ACA or weighted roundrobin load balancing method on a content rule. By default, all services have a weight of 1. A higher weight will bias flows toward the specified service. For background information on ACA load-balancing decisions based on server weight, see the “Using ArrowPoint Content Awareness Based on Server Load and Weight” section later in Chapter 6, Configuring Loads for Services.

To specify the relative weight of the service, use the `weight` command in service mode. To set the weight for a service, enter a number from 0 (graceful shutdown) to 10. The default is 1.
If you want to perform a graceful shutdown of an overloaded service or take a service offline for maintenance, specify a weight of 0 so 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.

**Note**

If you configure the absolute load calculation method on a CSS and then set a weight of zero on a service, the CSS does not include the load of that service in any content rule load that the CSS advertises.

The CSS recognizes content requests that include a location cookie as part of a sticky session. Therefore, even if you add a service with a configured weight of zero as a location service to a content rule, the CSS continues to direct to that service any requests that contain location cookies originating from the service.

For example, to specify a weight of 2, enter:

```
(config-service[serv1])# weight 2
```

To specify a weight of 0 to gracefully shut down a specific service, enter:

```
(config-service[serv1])# weight 0
```

To restore the weight to the default of 1, enter:

```
(config-service[serv1])# no weight
```

If you configure a weight on a service using the `weight` command, and there is a configured Dynamic Feedback Protocol (DFP) agent for the service, the configured weight for the DFP agent takes precedence over the weight configured on a service (weighted round-robin load-balancing method only).

When you add a service to one or more content rules, the CSS applies the service weight, as configured in service mode, to each content rule as a server-specific attribute. To specify a content rule-specific server weight (assuming the content rule is using a weighted load-balancing method), use the `weight` option of either the `add service` command. These two commands override the server-specific weight and apply only to the content rule to which you add the service. For information on using the `add service` command, see the “Specifying a Service Weight” section in Chapter 9, Configuring Content Rules.
Specifying a Service Type

By default, the service type is local. Use the `type` command to specify the type for a service. The syntax and options for this service mode command are:

- **type nci-direct-return** - Specify the service as NAT channel indication for direct return.

  Note: Use the `type nci-direct-return` command to configure NAT peering. For information on NAT peering, see Chapter 12, Configuring Caching.

- **type nci-info-only** - Specify the service as NAT channel indication for information only.

- **type proxy-cache** - Define the service as a proxy cache. This is a cache-specific option. This option bypasses content rules for requests coming from the cache server. Bypassing content rules in this case prevents a loop between the cache and the CSS. For a description of a proxy cache, see Chapter 12, Configuring Caching.

- **type redirect** - Define the service as a remote service to enable the CSS to redirect content requests to the remote service when a local service is not available (for example, the local service has exceeded its configured load threshold). To configure a load threshold for a content rule, use the `load-threshold` command in owner-content mode (see the “Specifying a Load Threshold” section in Chapter 9, Configuring Content Rules). If you have multiple remote services defined as `type redirect`, the CSS uses the roundrobin load-balancing method to load balance requests between them.

  When you add a type redirect service to a content rule, you must also configure a URL to match on the content. For example, “/*” or “/vacations.html”.

- **type redundancy-up** - Specify the router service in a redundant uplink.

- **type rep-cache-redir** - Specify the service as a replication cache with redirect.

- **type rep-store** - Specify the service as a replication store.

- **type rep-store-redir** - Specify the service as a replication store with redirect. No content rules are applied to requests from this service type.
- **ssl-accel** - Specify that this is an SSL termination service for the SSL Acceleration Module. This allows you to:
  - Configure the service as an SSL termination service.
  - Add the SSL proxy list to an SSL termination service through the (config-service) add ssl-proxy-list command.

  For more information on configuring SSL termination, refer to the Cisco Content Services Switch SSL Configuration Guide.

- **ssl-accel-backend** - Specify that this is a service for a back-end SSL server. This allows you to:
  - Configure the service as a back-end SSL service.
  - Add the SSL proxy list to a back-end SSL service through the (config-service) add ssl-proxy-list command.

  For more information on configuring a back-end SSL server, refer to the Cisco Content Services Switch SSL Configuration Guide.

- **ssl-init** - Specify that this is a service for an SSL initiation server. This allows you to:
  - Configure the service as an SSL initiation service.
  - Add the SSL proxy list to an SSL initiation service through the (config-service) add ssl-proxy-list command.

  For more information on configuring an SSL initiation server, refer to the Cisco Content Services Switch SSL Configuration Guide.

- **type transparent-cache** - Specify the service as a transparent cache. This is a cache-specific option. No content rules are applied to requests from this service type. Bypassing content rules in this case prevents a loop between the cache and the CSS. For a description of a transparent cache, see Chapter 12, Configuring Caching.

  For example, to enable the CSS to redirect content requests for serv1, specify redirect in the serv1 content rule:

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

  To restore the service type to the default setting of **local**, enter:

  ```
  (config-service[serv1])# no type
  ```
How the CSS Accesses Server Types

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

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

For information on adding a service to a content rule or adding primary and secondary sorry servers, see Chapter 9, Configuring Content Rules.

Configuring Service Access

When a service offers publishing services, you must associate an FTP access mechanism for moving content during publishing, subscribing, and demand-based replication activities. Use the `access` command to associate an FTP access mechanism with a service. You must use this command for each service that offers publishing services. This command is optional for subscriber services; the subscriber service inherits the access mechanism from the publisher.

When you use this command to associate an FTP access mechanism with a service, the base directory of an existing FTP record becomes the tree root. To maintain coherent mapping between WWW daemons and FTP daemons, make the FTP access base directory equivalent to the WWW daemon root directory as seen by clients. For information on creating an FTP record, refer to the (config) `ftp-record` command in the Cisco Content Services Switch Administration Guide.
Enter the name of the existing FTP record as an unquoted text string with no spaces.

For example, enter:

```
(config-service[serv1])# access ftp arrowrecord
```

To remove a service access mechanism, enter:

```
(config-service[serv1])# no access ftp
```

### Bypassing Content Rules on Caches

By default, no content rules are applied to requests from a proxy or transparent-cache type service. Use the `cache-bypass` command to prevent the CSS from applying content rules to requests originating from a proxy or transparent-cache type service when it processes the requests.

**Note**

For a description of proxy and transparent caching, see Chapter 12, Configuring Caching.

For example, enter:

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

To allow the CSS to apply content rules to requests from a proxy or transparent-cache type service, enter:

```
(config-service[serv1])# no cache-bypass
```

### Configuring Network Address Translation for Transparent Caches

By default, destination Network Address Translation (NAT) for the transparent cache service type is disabled. Use the `transparent-hosttag` command to enable destination NAT for the transparent cache service type.
Currently, you can use the `transparent-hosttag` command only with a CSS operating in a Client Side Accelerator (CSA) environment. For details on CSA, refer to the *Content Services Switch Global Server Load-Balancing Configuration Guide*.

**Note**
For a description of a transparent cache, see Chapter 12, Configuring Caching.

For example, enter:

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

To disable destination NATing for the transparent cache service type, enter:

```
(config-service[serv1])# no transparent-hosttag
```

### Configuring a Service to Bypass a Cache Farm

By default, the CSS bypasses cache for non-cacheable content. Use the `bypass-hosttag` command to allow the CSA on the CSS to bypass a cache farm and establish a connection with the origin server to retrieve non-cacheable content. The domain name from the host-tag field is used to look up the origin IP address on the CSA.

Currently, you can use the `bypass-hosttag` command only with a CSS operating in a CSA environment. For details on CSA, refer to the *Cisco Content Services Switch Global Server Load-Balancing Configuration Guide*.

For example, enter:

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

To disable bypassing cache for non-cacheable content, enter:

```
(config-service[serv1])# no bypass-hosttag
```
Configuring Maximum TCP Connections

By default, there is no limit on the number of TCP connections on a service. To define the maximum number of TCP connections on a service, use the `max connections` command. Enter the maximum number of connections from 6 to 65534. The default is 65534, which indicates that there is no limit on the number of connections.

If you configure the absolute load calculation method on a CSS and a service exceeds the configured maximum number of connections, the CSS does not include the load of that service in any content rule load that the CSS advertises.

For example:

```
(config-service[serv1])# max connections 7
```

To set the maximum TCP connections to the default value, enter:

```
(config-service[serv1])# no max connections
```

Do not use service max connections on UDP content rules. The service connection counters do not increment and remain at 0 because UDP is a connectionless protocol.

Configuring Keepalives for a Service

The default service keepalive is ICMP with a frequency and retry period of 5 seconds, and a maximum failure rate of 3 times. For information on configuring keepalives for a service, see Chapter 4, Configuring Service, Global, and Script Keepalives.
Activating a Service

Once you configure a service, you must activate it to enable the CSS to access it for content requests. Activating a service puts it into the resource pool for load-balancing content requests and starts the keepalive function.

**Note**

Once a service is activated, the following commands cannot be changed for the active service: `ip address`, `port`, `protocol`, `type`, `transparent-hosttag`, and `bypass-hosttag`. If you need to make modifications to an active service, you must first suspend it.

The following command activates service `serv1`:

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

**Note**

The CSS supports one active SSL service for each SSL Acceleration Module in the chassis (one SSL service per slot). You can configure more than one SSL service for a slot, but only a single SSL service can be active at a time. Before you can activate the service, you must add an SSL proxy list to an `ssl-accel` type service and then activate the SSL proxy list.

For information on adding to a content rule, see Chapter 9, Configuring Content Rules. For information on adding a service to a source group, see Chapter 5, Configuring Source Groups for Services.

Suspending a Service

Suspending a service removes it from the pool for future load-balancing content requests. Suspending a service does not affect existing content flows, but it prevents additional connections from accessing the service for its content. You may want to suspend a service prior to performing maintenance on the service.

The following command suspends service `serv1`:

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

**Note**

When you suspend a service, the CSS rebalances the remaining services using the failover setting.
Showing Service Configurations

Before activating a service, you may want to display the service configuration to ensure that all the parameters are correct. The `show service` command enables you to display information for a specific service or all services currently configured in the CSS, depending on the location from where you issue the command.

You can issue the following `show service` commands from any mode:

- `show service` - Display configurations for a service.
- `show service service_name` - Display service information for a specific service.
- `show service summary` - Display a summary of each service.

From a specific service mode, the `show service` command displays configuration information only for that service. For example:

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

When you issue this command from any other mode, it displays configuration information for all services.

To display information for a specific service, use the `show service` command with the service name. For example:

```
# show service serv86
```

The `show service summary` command displays a summary of all service currently configured.

---

**Note**

The connection counters displayed with the `show service` command do not increment and remain at 0 for UDP flows. UDP is a connectionless protocol.

Table 3-2 describes the fields in the `show service` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the service.</td>
</tr>
<tr>
<td>Index</td>
<td>The CSS assigned unique numeric index.</td>
</tr>
</tbody>
</table>
Type the type for the service. If you do not define a type for the service, the default service type is local. The possible types are:

- **nci-direct-return** - A NAT channel indication (NCI) service for NAT peering.
- **nci-info-only** - The service is NAT channel indication for information only.
- **proxy-cache** - The service is a proxy cache. This type bypasses content rules for requests from the cache.
- **redirect** - The service is not directly accessible and requires redirection.
- **redundancy-up** - The service is a redundant uplink.
- **rep-cache-redir** - The service is a replication cache with redirect.
- **rep-store** - The service is a replication store server for hot content.
- **rep-store-redir** - The service is a replication store to which content requests are redirected.
- **ssl-accel** - Specify that this is an SSL acceleration service for an SSL Acceleration Module.
- **transparent-cache** - The service is a transparent cache. No content rules are applied to requests from the cache.

State

The state of the service. The State field displays the service as Alive, Dying, Down, or Suspended. The Dying state reports that a service is failing according to the parameters configured in the following service mode commands: keepalive retryperiod, keepalive frequency, and keepalive maxfailure. When a service enters the Down state, the CSS does not forward any new connections to it (the service is removed from the load-balancing rotation for the content rule). However, the CSS keeps all existing connections to the service (that is, connections to that service are not “torn down”).

### Table 3-2 Field Descriptions for the show service Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>The type for the service. If you do not define a type for the service, the default service type is local. The possible types are:</td>
</tr>
<tr>
<td></td>
<td>- <strong>nci-direct-return</strong> - A NAT channel indication (NCI) service for NAT peering.</td>
</tr>
<tr>
<td></td>
<td>- <strong>nci-info-only</strong> - The service is NAT channel indication for information only.</td>
</tr>
<tr>
<td></td>
<td>- <strong>proxy-cache</strong> - The service is a proxy cache. This type bypasses content rules for requests from the cache.</td>
</tr>
<tr>
<td></td>
<td>- <strong>redirect</strong> - The service is not directly accessible and requires redirection.</td>
</tr>
<tr>
<td></td>
<td>- <strong>redundancy-up</strong> - The service is a redundant uplink.</td>
</tr>
<tr>
<td></td>
<td>- <strong>rep-cache-redir</strong> - The service is a replication cache with redirect.</td>
</tr>
<tr>
<td></td>
<td>- <strong>rep-store</strong> - The service is a replication store server for hot content.</td>
</tr>
<tr>
<td></td>
<td>- <strong>rep-store-redir</strong> - The service is a replication store to which content requests are redirected.</td>
</tr>
<tr>
<td></td>
<td>- <strong>ssl-accel</strong> - Specify that this is an SSL acceleration service for an SSL Acceleration Module.</td>
</tr>
<tr>
<td></td>
<td>- <strong>transparent-cache</strong> - The service is a transparent cache. No content rules are applied to requests from the cache.</td>
</tr>
<tr>
<td>State</td>
<td>The state of the service. The State field displays the service as Alive, Dying, Down, or Suspended. The Dying state reports that a service is failing according to the parameters configured in the following service mode commands: keepalive retryperiod, keepalive frequency, and keepalive maxfailure. When a service enters the Down state, the CSS does not forward any new connections to it (the service is removed from the load-balancing rotation for the content rule). However, the CSS keeps all existing connections to the service (that is, connections to that service are not “torn down”).</td>
</tr>
</tbody>
</table>
### Table 3-2  Field Descriptions for the `show service` Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule</td>
<td>The address, protocol, and port information for the service.</td>
</tr>
<tr>
<td>Redirect Domain</td>
<td>The domain name to be used when an HTTP redirect service generates an OBJECT MOVED message for the service.</td>
</tr>
<tr>
<td>Session Redundancy</td>
<td>Indicates whether Adaptive Session Redundancy (ASR) is enabled or disabled for the service. For details on ASR, refer to the Cisco Content Services Switch Global Server Load-Balancing Configuration Guide.</td>
</tr>
<tr>
<td>SSL-Accel Slot</td>
<td>The slot in the CSS where the SSL module is located. An SSL service requires the SSL module slot number in order to correlate the SSL proxy list to a specific SSL module. For details on SSL, refer to the Cisco Content Services Switch SSL Configuration Guide.</td>
</tr>
<tr>
<td>Session Cache Size</td>
<td>The size of the SSL session ID cache for the service. The cache size is the maximum number of SSL session IDs that can be stored in a dedicated session cache on an SSL module.</td>
</tr>
<tr>
<td>Redundancy Global Index</td>
<td>The unique global index value for ASR assigned to the service using the <code>redundant-index</code> command in service configuration mode.</td>
</tr>
<tr>
<td>Redirect String</td>
<td>The HTTP redirect string to be used when an HTTP redirect service generates an OBJECT MOVED message for the service.</td>
</tr>
</tbody>
</table>
### Table 3-2  Field Descriptions for the show service Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Keepalive | The keepalive type, frequency, maxfailure, and retry period. The possible keepalive types are:  
  - **ftp** - The keepalive method that accesses an FTP server by logging in to an FTP server as defined in an FTP record file.  
  - **http** - An HTTP index page request. By default, HTTP keepalives attempt to use persistent connections. For an HTTP Head keepalive, the response code is also displayed.  
  - **icmp** - An ICMP echo message (default).  
  - **named** - Global keepalive defined in keepalive configuration mode.  
  - **none** - Do not send keepalive messages to the service.  
  - **script** - Script keepalive to be used by the service. The script is played each time the keepalive is issued.  
  - **ssl** - SSL HELLO keepalives for this service. Use this keepalive for all backend services supporting SSL. When the CSS is using an SSL module, use the keepalive type of **none**.  
  - **tcp** - TCP connection handshake request.  
The keepalive frequency value is the interval in seconds between keepalive messages sent to the service. The default is 5. The range is from 2 to 255.  
The keepalive maxfailure value is the number of times the service can fail to respond to a keepalive message before being considered down. The default is 3. The range is from 1 to 10.  
The keepalive retry period value is the interval in seconds between retry messages sent to the service. The default is 5. The range is from 2 to 255. |
### Table 3-2  Field Descriptions for the show service Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Clearing of Stats Counters</td>
<td>The date and time when the State Transitions, Total Connections, or Total Reused Conns. counters were last cleared (reset to 0). The date and time stamp initially shown reflects when the service was activated or 01/01/00 00:00:00 if the service is down.</td>
</tr>
<tr>
<td>Mtu</td>
<td>The size of the largest datagram that can be sent or received on the service.</td>
</tr>
<tr>
<td>State Transitions</td>
<td>The total number of state transitions on the service. If the State Transitions field is 0, the 0 value can be due to a counter reset through either the global configuration mode <code>zero service state-transitions</code> command or the content mode <code>zero state-transitions</code> command. The counter can also be 0 if the service is down, or if the service is alive but no traffic is running.</td>
</tr>
<tr>
<td>Total Local Connections</td>
<td>Total number of TCP connections mastered by the CSS in an ASR configuration.</td>
</tr>
<tr>
<td>Current Local Connections</td>
<td>Number of current active TCP connections on the CSS in an ASR configuration.</td>
</tr>
<tr>
<td>Total Backup Connections</td>
<td>Total number of TCP connections backed up by the CSS for the master CSS in an ASR configuration.</td>
</tr>
<tr>
<td>Current Backup Connections</td>
<td>Number of current TCP connections that the CSS is backing up in an ASR configuration.</td>
</tr>
<tr>
<td>Total Connections</td>
<td>The total number of connections that have been mapped to the service. In an Adaptive Session Redundancy configuration, Total Connections equals the sum of the Total Local Connections and the Total Backup Connections. If the Total Connections field is 0, the 0 value can be due to a counter reset through either the global configuration mode <code>zero service total-connections</code> command or the content mode <code>zero total-connections</code> command. The counter can also be 0 if the service is down, or if the service is alive but no traffic is running.</td>
</tr>
<tr>
<td>Max Connections</td>
<td>The configured maximum number of TCP connections on the service.</td>
</tr>
</tbody>
</table>
### Table 3-2  Field Descriptions for the show service Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Reused Conns.</td>
<td>The total number of connections that were reused for multiple content requests during persistent connections. If the Total Reused Conns field is 0, the 0 value can be due to a counter reset through either the global configuration mode <code>zero service total-reused-connections</code> command or the content mode <code>zero total-reused connections</code> command. The counter can also be 0 if the service is down, or if the service is alive but no traffic is running.</td>
</tr>
<tr>
<td>Weight</td>
<td>The service weight used with load metrics to make load allocation decisions. The weight is used in ArrowPoint Content Awareness (ACA) and weighted roundrobin load balancing decisions. The range is from 1 to 10. The default is 1.</td>
</tr>
<tr>
<td>Load/Average Load</td>
<td>The current and average load for the service.</td>
</tr>
<tr>
<td>DFP</td>
<td>State of the Dynamic Feedback Protocol (DFP). Possible states are Enable or Disable. The DFP state is Disable if either DFP is not configured or DFP is configured and you have configured a weight on a service using the <code>add service weight</code> command in owner-content configuration mode. For details on DFP, see Chapter 7, Configuring Dynamic Feedback Protocol for Server Load Balancing.</td>
</tr>
</tbody>
</table>
Clearing Service Statistics Counters

To clear a specific service statistics counter for all existing CSS services and to set that counter to zero, use the `zero service` command. The reset statistics appear as 0 in the `show service` display. The `zero service` command is available in all modes.

Use the following `zero service` commands from any mode:

- `zero service total-connections` - Set the Total Connections counter to zero for all services
- `zero service total-reused-connections` - Set the Total Reused Conns. counter to zero for all services
- `zero service state-transitions` - Set the State Transitions counter to zero for all services

For example, to clear the Total Connections counter for all services, enter:

```
(config)# zero service total-connections
```

**Note**

To clear statistics counters for all services belonging to a specific content rule, use the `zero` command in content mode. You can also use this command to clear the counters for a specified service associated with the content rule. For details on clearing service statistics associated with a content rule, see Chapter 9, Configuring Content Rules.

Where to Go Next

For information on configuring keepalives, source groups, loads, and Dynamic Feedback Protocol (DFP), see the following chapters:

- Chapter 4, Configuring Service, Global, and Script Keepalives
- Chapter 5, Configuring Source Groups for Services
- Chapter 6, Configuring Loads for Services
- Chapter 7, Configuring Dynamic Feedback Protocol for Server Load Balancing

For information on creating and configuring owners, see Chapter 8, Configuring Owners.
Chapter 3
Configuring Services

Where to Go Next
Configuring Service, Global, and Script Keepalives

When you configure a service on the CSS, the CSS determines the state of the service by sending keepalive messages. By default, the CSS assigns each service with an ICMP keepalive with a frequency and retry period of 5 seconds, and a maximum failure rate of 3 times. To change the default keepalive settings for a service, you can configure individual keepalive attributes for the service or create a keepalive in keepalive mode and apply the service to it.

This chapter describes how to configure service, global, and script keepalives. Information in this chapter applies to all CSS models except where noted.

This chapter contains the following major sections to configure CSS keepalives:

- CSS Keepalive Overview
- Configuring Service Keepalives
- Configuring Global Keepalives
- Configuring Service and Global Keepalive Attributes
- Showing Keepalive Configurations
- Using Script Keepalives with Services
CSS Keepalive Overview

The CSS supports a total of 2048 keepalives. These keepalives include:

- ICMP, HTTP-GET, HTTP-HEAD, TCP, FTP, SSL, and script keepalives configured and assigned to a service through the `config-service keepalive type` command. By default, a service has an ICMP keepalive. Each time you assign one of these keepalives to a service through this command, the CSS counts it as one keepalive. For information on configuring service keepalives, see the “Configuring Service Keepalives” section.

- Global keepalives configured in keepalive configuration mode. You can apply multiple services to a global keepalive reducing the amount of configuration required for each service. The CSS counts a global keepalive as one keepalive regardless of the number of services assigned to it.

  Global keepalives supersede the individual keepalive parameters configured in service mode. For information on configuring global keepalives, see the “Configuring Global Keepalives” section later in this chapter.

The CSS divides the keepalive types into two categories, Class A and Class B keepalives. The CSS supports a maximum of 2048 Class A keepalives. The CSS supports a maximum of 512 Class B keepalives.

Table 4-1 lists the keepalive types in each class, the maximum number of each type, and the maximum number of each keepalive type that can execute concurrently.

<table>
<thead>
<tr>
<th>Class</th>
<th>Type</th>
<th>CSS Maximum</th>
<th>Concurrent Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ICMP</td>
<td>2048</td>
<td>2048</td>
</tr>
<tr>
<td></td>
<td>HTTP-HEAD non-persistent</td>
<td>2048</td>
<td>2048</td>
</tr>
<tr>
<td></td>
<td>SSL (Hello)</td>
<td>2048</td>
<td>2048</td>
</tr>
<tr>
<td></td>
<td>TCP</td>
<td>2048</td>
<td>2048</td>
</tr>
</tbody>
</table>

(The CSS limits 2048 keepalives per Class A.)
## CSS Keepalive Overview

**Caution**

Do not configure more than 2048 total keepalives, including a total of 512 Class B keepalives. Any services assigned to keepalives over the supported total number will not be eligible for content rule selection.

When you configure a keepalive for a service (or associate a service with a global keepalive), the CSS periodically sends a message to the service based on the keepalive frequency to determine the state of the service. See the “Configuring a Keepalive Frequency” section. The CSS considers the service to be alive when a service responds to the keepalive message.

The CSS transitions the service to the dying state when the service fails to respond to a keepalive message. The CSS tests whether the failed service is functional by sending a keepalive message at time intervals based on the retry period. See the “Configuring a Keepalive Retry Period” section.

The CSS transitions the service to the down state if the service fails to respond a maximum number of retries to the keepalive message. See the “Configuring the Maximum Number of Failures for a Keepalive” section. Then the CSS removes the service from the load-balancing algorithm. The CSS continues to test whether the service is functional at time intervals based on the retry period.

Thus, using the default values of a 5-second keepalive frequency interval, a 5-second retry period interval, and maximum of three failures, a service can transition from the alive state to the dead state in 15 seconds; a 5-second interval between a keepalive response and the initial keepalive failure based on the keepalive frequency, and two failures, each occurring at 5-second intervals based on the retry period.

### Table 4-1  Keepalive Class, Types, and Limitations (continued)

<table>
<thead>
<tr>
<th>Class</th>
<th>Type</th>
<th>CSS Maximum</th>
<th>Concurrent Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>FTP</td>
<td>256</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>HTTP-GET persistent and non-persistent</td>
<td>256</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>HTTP-HEAD persistent</td>
<td>256</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Script</td>
<td>256</td>
<td>16</td>
</tr>
</tbody>
</table>
However, if the keepalives are Class B type keepalives, the time for a service to transition from an alive state to the dead state may take longer. This transition delay occurs because the CSS executes smaller numbers of Class B keepalives at the same time. For example, if you configure 256 HTTP-GET keepalives using the default values for frequency, retry period, and maximum failure, and all services fail, the time for all of the services to transition from the alive state to the dead state is 120 seconds; 32 services transitioning in 15 seconds followed by another 32 services until all 256 services have transitioned.

Configuring Service Keepalives

A service keepalive is the keepalive configured for a specific service. As you configure a service, you can configure its keepalive attributes. To configure keepalive attributes for a service, access Service configuration mode for the service and use the *keepalive* command. For information, see the “Configuring Service and Global Keepalive Attributes” section.

If you want to apply a CSS service to a global keepalive, see the “Configuring Global Keepalives” section.

After you configure a service including its keepalive attributes, you can activate the service. Activating a service puts it into the resource pool for load-balancing content requests and starts the keepalive function. For example, to activate service *serv1*, enter:

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

Configuring Global Keepalives

A global keepalive allows you to configure keepalive attributes and apply multiple services to the keepalive. As long as one service is alive, the global keepalive service is alive. By having a single keepalive configuration for more than one service, you can reduce the amount of time to configure each service. Also the keepalive counts as one keepalive no matter how many services you apply to it.

Table 4-2 provides a quick overview of the basic steps required to configure a global keepalive. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following Table 4-2.
### Table 4-2  Global Keepalive Configuration Quick Start

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter global configuration mode by typing <code>config</code>.</td>
</tr>
</tbody>
</table>
|   ```yaml
   # config
   (config)#
   ``` |
| 2. Create a global keepalive and enter the keepalive configuration mode. See the  |
|    “Creating and Naming a Global Keepalive” section.                                     |
|   ```yaml
   (config) keepalive keepimages
   (config-keepalive[keepimages])#
   ``` |
| 3. Specify the IP address where the keepalive messages are sent. See the  |
|    “Configuring a Global Keepalive IP Address” section.                                  |
|   ```yaml
   (config-keepalive[keepimages]) ip address 192.168.7.6
   ``` |
| 4. Specify the type of keepalive message assigned to a keepalive. See the  |
|    “Configuring a Keepalive Type” section.                                               |
|   ```yaml
   (config-keepalive[keepimages]) type http
   ``` |
| 5. Specify the HTTP keepalive method assigned to the global keepalive. See the  |
|    “Configuring the HTTP Keepalive Method” section.                                      |
|   ```yaml
   (config-keepalive[keepimages]) method get
   ``` |
| 6. Specify the content information for an HTTP global keepalive. See the  |
|    “Configuring a Keepalive URI” section.                                                 |
|   ```yaml
   (config-keepalive[keepimages]) uri "/index.html"
   ``` |
| 7. Activate the global keepalive.                                                        |
|   ```yaml
   (config-keepalive[keepimages]) active
   ``` |
| 8. Associate a service with a global keepalive.                                           |
|   ```yaml
   (config-service[imageserver1]) keepalive type named keepimages
   ``` |
| 9. (Recommended) Use the `show keepalive` command to verify the global  |
|    keepalive configuration. See the “Showing Keepalive Configurations”  |
|    section.                                                                             |
|   ```yaml
   (config-keepalive[keepimages]) show keepalive
   ``` |
| 10. (Optional) Use the `show service` command to verify the basic keepalive  |
|    configuration on the service.                                                        |
|    ```yaml
    (config-service[imageserver1]) show service
    ``` |
The following running-configuration example shows the results of entering the commands in Table 4-2 as shown in bold and any related commands.

```
server2
  ip address 10.3.6.1
  keepalive type named keepimages
  active

keepalive keepimages
  ip address 192.168.7.6
  type http
  method get
  uri "/index.html"
  active
```

The following sections provides information on:

- Creating and Naming a Global Keepalive
- Configuring a Global Keepalive IP Address
- Configuring a Global Keepalive Description
- Activating the Global Keepalive
- Suspending a Global Keepalive
- Associating a Service with a Global Keepalive

For information on configuring the keepalive attributes, see the “Configuring Service and Global Keepalive Attributes” section.

**Creating and Naming a Global Keepalive**

To create and name a global keepalive, use the `keepalive` command to access keepalive mode. You can access keepalive mode from circuit, global, interface, and IP configuration modes. The prompt changes to (config-keepalive [name]). You can also use this command from keepalive mode to access another keepalive.

Enter the name of the new keepalive you want to create or the name of an existing keepalive. Enter an unquoted text string with no spaces and a length of 1 to 31 characters. To see a list of existing keepalive names, use the `keepalive ?` command.

For example, to create the global keepalive keepimages, enter:

```
(config)# keepalive keepimages
```
When you access this mode, the prompt changes to (config-keepalive [keepimages]).

(config-keepalive[keepimages])#

To remove an existing keepalive, enter:

(config)# no keepalive keepimages

### Configuring a Global Keepalive IP Address

The CSS sends global keepalives to a service that monitors the state of services assigned to it. Use the `ip address` command to specify the IP address where the keepalive messages are sent. Enter the IP address in dotted-decimal notation.

For example, to enter an IP address for keepalive keepimages, enter:

(config-keepalive[keepimages])# ip address 192.168.7.6

### Configuring a Global Keepalive Description

 optionally, you can provide a description for the global keepalive. To specify the description, use the `description` command. Enter the description as a quoted text string with a maximum of 64 characters, including spaces.

For example, to enter a description for the global keepalive keepimages, enter:

(config-keepalive[keepimages])# description "This keepalive is for the image servers"

To delete a description, enter:

(config-keepalive[keepimages])# no description

### Activating the Global Keepalive

Activating a keepalive enables the CSS to start sending keepalive messages to the IP address. Use the `active` command to activate the global keepalive.

For example, to activate the global keepalive keepimages, enter:

(config-keepalive[keepimages])# active
## Suspending a Global Keepalive

To deactivate the keepalive, use the `suspend` command.

For example, enter:

```
(config-keepalive[keepimages])# suspend
```

## Associating a Service with a Global Keepalive

To associate a service with a global keepalive, use the `keepalive type named` command. The service maintains the global keepalive attributes when you add the service to content rules.

For example, to associate `imageserver1` with global keepalive `keepimages`, enter:

```
(config-service[imageserver1])# keepalive type named keepimages
```

## Configuring Service and Global Keepalive Attributes

The following sections describe the attributes you can configure for keepalives:

- Configuring a Keepalive Frequency
- Configuring a Keepalive Retry Period
- Configuring the Maximum Number of Failures for a Keepalive
- Configuring a Keepalive Type
- Configuring a TCP Keepalive with Graceful Socket Close (FIN)
- Configuring a Keepalive Port Number
- Configuring the HTTP Keepalive Method
- Configuring a Keepalive HTTP Response Code
- Configuring a Keepalive URI
- Configuring a Keepalive Hash Value
Chapter 4  Configuring Service, Global, and Script Keepalives

Configuring a Keepalive Frequency

The keepalive frequency specifies the interval in seconds between keepalives messages sent to a service. Specify a frequency from 2 to 255 seconds. The default is 5 seconds.

**Note**
When configuring the CSS for FTP keepalives, do not configure the keepalive frequency or the keepalive retryperiod to a value less than 15 seconds. Note that the CSS does not prevent you from configuring smaller values. Also, the default value for the keepalive frequency or the keepalive retryperiod is five seconds. You must use the frequency and retryperiod commands to override the defaults.

**Note**
The timeout value for a keepalive is related to the configured keepalive frequency. For versions 7.20.1.04 and greater, the timeout is 2 seconds less than the keepalive frequency with a minimum of 1 second. From version 5.20 up to version 7.20.1.04, the timeout is one second less than the keepalive frequency.

- To configure the keepalive frequency for a specific service, use the service mode `keepalive frequency` command. For example, to configure a frequency of 15 seconds, enter:

  ```bash
  (config-service[serv1])# keepalive frequency 15
  ```

  To reset the frequency to its default value of 5, enter:

  ```bash
  (config-service[serv1])# no keepalive frequency
  ```

- To configure the frequency for a global keepalive, use the keepalive mode `frequency` command to specify the time between keepalive messages sent to the IP address.

  For example, to set the frequency time to 10 seconds, enter:

  ```bash
  (config-keepalive[keepimages])# frequency 10
  ```

  To reset the frequency to its default value of 5, enter:

  ```bash
  (config-keepalive[keepimages])# no frequency
  ```
Configuring a Keepalive Retry Period

When a service has failed to respond to a given keepalive message (the service has transitioned to the dying state), the retry period specifies how frequently the CSS tests the service to see if it is functional. Enter the retry period as an integer from 2 to 255 seconds. The default is 5 seconds.

Note When configuring the CSS for FTP keepalives, do not configure the keepalive frequency or the keepalive retryperiod to a value less than 15 seconds. Note that the CSS does not prevent you from configuring smaller values. Also, the default value for the keepalive frequency or the keepalive retryperiod is five seconds. You must use the frequency and retryperiod commands to override the defaults.

- To configure the keepalive retry period for a service, use the service mode keepalive retryperiod command. For example, to configure a retry period of 60 seconds, enter:

  (config-service[serv1])# keepalive retryperiod 60

  To reset the retry period to its default value of 5, enter:

  (config-service[serv1])# no keepalive retryperiod

- To configure the retry period for a global keepalive, use the keepalive mode retryperiod command. For example, to configure a retry period of 60 seconds, enter:

  (config-keepalive[keepimages])# retryperiod 60

  To reset the retry period to its default value of 5, enter:

  (config-keepalive[keepimages])# no retryperiod
Configuring the Maximum Number of Failures for a Keepalive

The maximum failures is the number of times a service can fail to respond to a keepalive message before the CSS considers it offline. Specify a maximum failure number from 1 to 10. The default is 3.

- To configure the keepalive maximum failure value for a service, use the service mode `keepalive maxfailure` command. For example, to configure the maximum failure value to 5, enter:

  ```
  (config-service[serv1])# keepalive maxfailure 5
  ```

  To reset the maximum failure number to its default value of 3, enter:

  ```
  (config-service[serv1])# no keepalive maxfailure
  ```

- To configure the maximum failure value for a global keepalive, use the keepalive mode `maxfailure` command. For example, to configure a value of 7, enter:

  ```
  (config-keepalive[keepimages])# maxfailure 7
  ```

  To reset the maximum failure number to its default value of 3, enter:

  ```
  (config-keepalive[keepimages])# no maxfailure
  ```

Configuring a Keepalive Type

The keepalive type specifies the type of keepalive message assigned to the keepalive. The keepalive types include ICMP, HTTP-GET, HTTP-HEAD, TCP, FTP, SSL, and script keepalives. For a service keepalive, a named keepalive type allows you to apply the service to a configured global keepalive.

- To specify the type of keepalive message for a service, use the service mode `keepalive type` command, if any, appropriate for a service or to associate a service with a global keepalive. For example, to set serv1 keepalive type to `ftp`, enter:

  ```
  (config-service[serv1])# keepalive type ftp
  ```
To specify the keepalive type for a global keepalive, use the keepalive mode `type` command. For example, to set the global keepalive keepimages to `type tcp`, enter:

```
(config-keepalive[keepimages])# type tcp
```

Each time you assign a keepalive type to a service or global keepalive, the CSS counts it as one keepalive.

**Caution**

Do not configure more than 2048 total keepalives, including a total of 512 Class B keepalives. Any services assigned to keepalives over the supported total number will not be eligible for content rule selection.

The options for the `keepalive type` or `type` command are:

- `ftp ftp_record` - Keepalive method in which the CSS logs in to an FTP server as defined in the FTP record file. Enter the name of the existing FTP record for an FTP server as an unquoted text string with no spaces. To create an FTP record, use the `(config) ftp-record` command.

  The FTP keepalive type is a Class B type. The CSS supports a maximum of 256 FTP keepalives and concurrently executes a maximum of 32 keepalives of this type at a time.

  When configuring the CSS for FTP keepalives, do not configure the keepalive frequency or the keepalive retryperiod to a value less than 15 seconds. Note that the CSS does not prevent you from configuring smaller values. Also, the default value for the keepalive frequency or the keepalive retryperiod is five seconds. You must use the `keepalive frequency` and `keepalive retryperiod` commands to override the defaults.

- `http` - A persistent HTTP index page request. By default, HTTP keepalives attempt to use persistent connections.

  For configuring the method for the HTTP keepalive type, see the “Configuring the HTTP Keepalive Method” section. The HTTP-HEAD persistent, and HTTP-GET persistent keepalive types are a Class B types. Of each of these types, the CSS supports a maximum of 256 keepalives and concurrently executes a maximum of 32 keepalives at a time.

  If an HTTP persistent keepalive fails to make a persistent connection, then it attempts to make a non-persistent connection. If the non-persistent connection succeeds, then the keepalive succeeds. At the next interval, the keepalive attempts a persistent connection.
Chapter 4  Configuring Service, Global, and Script Keepalives

Configuring Service and Global Keepalive Attributes

• **http non-persistent** - A non-persistent HTTP index page request. This command disables the default persistent behavior.

  For configuring the method for the HTTP keepalive type, see the “Configuring the HTTP Keepalive Method” section. The HTTP-GET non-persistent keepalive type is a Class B type. Of this type, the CSS supports a maximum of 256 keepalives and concurrently executes a maximum of 32 keepalives at a time.

  The HTTP-HEAD non-persistent keepalive type is a Class A type. The CSS supports a maximum of 2048 HTTP-HEAD non-persistent keepalives and concurrently executes a maximum of 2048 keepalives of this type at a time.

• **icmp** - An ICMP echo message (ping). This is the default keepalive type.

  The ICMP keepalive type is a Class A type. The CSS supports a maximum of 2048 ICMP keepalives and concurrently executes a maximum of 2048 keepalives of this type at a time.

• **named name** - (service mode only) Associates the service with a previously defined global keepalive.

  Before using this command, ensure that the global keepalive is activated through the **(config-keepalive) active** command. Assigning a service to a global keepalive overrides any keepalive properties you assigned in service mode. For information on creating a global keepalive, see the “Configuring Global Keepalives” section.

• **none** - Do not send keepalive messages to a service.

• **script script_name** (“arguments”) **[use-output]** - Defines a script keepalive to be used by the service. The script is played each time the keepalive is issued. Enter the name of an existing script keepalive.

  The optional *arguments* variable passes arguments into the keepalive script. Enter a quoted text string with a maximum of 128 characters including spaces.

  The **use-output** option allows the script to parse the output for each executed command. This optional keyword allows the use **grep** and file direction within a script. By default, the script does not parse the output. For details on using script keepalives, see the “Using Script Keepalives with Services” section later in this chapter.

  The script keepalive type is a Class B type. The CSS supports a maximum of 256 script keepalives and concurrently executes a maximum of 16 keepalives of this type at a time.
Configuring Service and Global Keepalive Attributes

Note

To preserve CSS system resources, use script keepalives only when needed. If an ICMP or HTTP keepalive message is sufficient to validate the service, then use that type of message instead of a script keepalive.

- **ssl** - SSL HELLO keepalives for this service. Use this keepalive for all backend services supporting SSL. The CSS sends a client HELLO to connect the SSL server. After the CSS receives a HELLO from the server, the CSS closes the connection with a TCP RST.
  
The SSL keepalive type is a Class A type. The CSS supports a maximum of 2048 SSL keepalives and concurrently executes a maximum of 2048 keepalives of this type at a time.
  
  When the 11500 series CSS is using an SSL module, use the keepalive type of **none**. The SSL module is an integrated device in the CSS and does not require the use of keepalive messages for the service.

- **tcp** - A TCP session that determines service viability through a 3-way handshake and reset; SYN, SYN-ACK, ACK, RST-ACK. By default, the CSS sends a RST to close the socket on a server port for TCP keepalives. If your servers require a graceful closing of a socket using a FIN, you can use a keepalive to send a FIN to close a socket by using the **tcp-close fin** command. For more information on the **tcp-close** command, see the “Configuring a TCP Keepalive with Graceful Socket Close (FIN)” section.
  
The TCP keepalive type is a Class A type. The CSS supports a maximum of 2048 TCP keepalives and concurrently executes a maximum of 2048 keepalives of this type at a time.

### Configuring a TCP Keepalive with Graceful Socket Close (FIN)

By default and in compliance with RFC 1122, the CSS sends a reset (RST) to close the socket on a server port for TCP keepalives. A RST is faster than a FIN, because a RST requires only one packet, while a FIN can take up to four packets. If your servers require a graceful closing of a socket using a FIN, you can configure a keepalive to send a FIN to close a socket.
• To configure a keepalive to send a FIN to close a socket, use the service mode `keepalive tcp-close fin` command. For example, enter:

```plaintext
(config-service[serv1])# keepalive tcp-close fin
```

To reset the keepalive to send a RST, enter:

```plaintext
(config-service[serv1])# keepalive tcp-close rst
```

• To configure a global keepalive to send a FIN to close a socket, use the keepalive mode `tcp-close fin` command. For example, enter:

```plaintext
(config-keepalive[keepimages])# tcp-close fin
```

To reset the keepalive to send a RST, enter:

```plaintext
(config-keepalive[keepimages])# tcp-close rst
```

### Configuring a Keepalive Port Number

By default, the port number for keepalives is based on the keepalive type. If the keepalive type is:

- HTTP or TCP - The default port number is 80
- FTP - The port number is 21 and is not configurable
- SSL - The port number is 443
- ICMP - The port number is the number for the service

You can configure a port number from 0 to 65535.

• To specify the keepalive port number for a service, use the service mode `keepalive port` command. For example, to specify port 8080 as the keepalive port, enter:

```plaintext
(config-service[serv1])# keepalive port 8080
```

To reset the keepalive port to its default value, enter:

```plaintext
(config-service[serv1])# no keepalive port
```

• To specify a port for a global keepalive, use the keepalive mode `port` command. For example, to specify port 8080, enter:

```plaintext
(config-keepalive[keepimages])# port 8080
```
To reset the keepalive port to its default value, enter:

```
(config-keepalive[keepimages])# no port
```

## Configuring the HTTP Keepalive Method

By default, when you configure an HTTP keepalive type, the CSS uses an HTTP-HEAD method. The CSS issues an HTTP-HEAD method to the service and a 200 OK status is required. The CSS does not compute a reference hash value for this type of keepalive. If the 200 OK status is not returned, the CSS considers the service down.

You can also configure an HTTP GET method. The CSS issues an HTTP GET method to the service, computes an MD5 (Message Digest Algorithm Version 5) hash value on the page, and stores the hash value as a reference hash. Subsequent GETs require a 200 OK status (HTTP command completed OK response) and the hash value to equal the reference hash value. If the 200 OK status is not returned, or if the 200 OK status is returned but the hash value is different from the reference hash value, the CSS considers the service down.

When you specify the content information of an HTTP Uniform Resource Identifier (URI) for an HTTP keepalive, the CSS calculates a hash value for the content. If the content information changes, the hash value no longer matches the original hash value and the CSS assumes that the service is down. To prevent the CSS from assuming that a service is down due to a hash value mismatch, specify the keepalive method as HTTP HEAD.

For information of configuring an HTTP response code, see the “Configuring a Keepalive HTTP Response Code” section. For information of configuring an HTTP URI, see the “Configuring a Keepalive URI” section.

- To specify the HTTP keepalive method for a service, use the service mode `keepalive method` command. For example, to specify the HTTP GET method, enter:

  ```
  (config-service[serv1])# keepalive method get
  ```

  To reset the HTTP keepalive method to HTTP HEAD, enter:

  ```
  (config-service[serv1])# keepalive method head
  ```
To specify the HTTP keepalive method for a global keepalive, use the keepalive `method` command. For example, to specify the HTTP GET keepalive method, enter:

```
(config-keepalive[keepimages])# method get
```

To reset the HTTP keepalive method to HTTP HEAD, enter:

```
(config-keepalive[keepimages])# method head
```

If you change the keepalive method on an active service, make sure that you suspend and reactivate the service for the change to take effect.

---

**Note**

By default, HTTP keepalives attempt to use persistent connections. If an HTTP persistent keepalive fails to make a persistent connection, then it attempts to make a non-persistent connection. If the non-persistent connection succeeds, then the keepalive succeeds. At the next interval, the keepalive attempts a persistent connection.

---

### Configuring a Keepalive HTTP Response Code

By default, when the CSS issues an HTTP-HEAD keepalive, the CSS expects a response code of 200 in the response packet from the server it is querying. For HTTP-HEAD non-persistent keepalives, you can configure the CSS to expect a non-200 response code (for example, a 302 redirect response code). Enter the response code as an integer from 100 to 999.

- To specify the keepalive response code for a service, use the service mode `keepalive http-rspcode` command. For example, to specify a response code of 302, enter:

  ```
  (config-service[serv1])# keepalive http-rspcode 302
  ```

  To reset the response code to its default value of 200, enter:

  ```
  (config-service[serv1])# no keepalive http-rspcode
  ```

- To specify the response code for a global keepalive, use the `http-rspcode` command. For example, to specify a response code of 302, enter:

  ```
  (config-keepalive[keepimages])# http-rspcode 302
  ```
To reset the response code to its default value of 200, enter:

```
(config-keepalive[keepimages])# no http-rspcode
```

## Configuring a Keepalive URI

When you configure an HTTP keepalive type, the CSS uses the URI string to determine if the service is alive. By default, the CSS uses the URI string to the root directory,”/”. For an HTTP Get, the CSS uses the URI information to calculate the hash value. You can specify the URI content information for an HTTP keepalive.

**Note**

When you specify the content information of a URI for an HTTP keepalive, the CSS calculates a hash value for the content. If the content information changes, the hash value no longer matches the original hash value and the CSS assumes that the service is down. To prevent the CSS from assuming that a service is down due to a hash value mismatch, define **keepalive method** as **head**. The CSS does not compute a hash value for this type of keepalive. If you specify a Web page with changeable content and do not specify the head keepalive method, you must suspend and reactivate the service each time the content changes.

Enter the content information of the URI as a quoted text string with a maximum of 64 characters. Do not include the host information in the string. The CSS derives the host information from the service IP address and the keepalive port number.

- To specify the HTTP keepalive content information for a service, use the service mode **keepalive uri** command. For example, enter:

  ```
  (config-service[serv1])# keepalive uri "/index.html"
  ```

  To clear the content information for the keepalive, enter:

  ```
  (config-service[serv1])# no keepalive uri
  ```

- To specify the HTTP keepalive content information for a global keepalive, use the **uri** command. For example, to specify the content information for the global keepalive, enter:

  ```
  (config-keepalive[keepimages])# uri "/index.html"
  ```

  To clear the content information assigned to this keepalive, enter:

  ```
  (config-keepalive[keepimages])# no uri
  ```
Chapter 4 Configuring Service, Global, and Script Keepalives

Configuring Service and Global Keepalive Attributes

Configuring a Keepalive Hash Value

By default, the CSS uses the MD5 (Message Digest Algorithm Version 5) hash for an HTTP GET keepalive. The CSS compares the hash value against the computed hash value of all HTTP GET responses. A successful comparison causes the keepalive to maintain an Alive state.

For a service keepalive, use the service mode `keepalive hash` command to override the default MD5 hash. To configure the hash value for a service keepalive:

1. Configure the keepalive. The example below creates a keepalive GET to a test page.

   ```
   (config)# service serv1
   (config-service[serv1])# ip address 10.0.3.21
   (config-service[serv1])# keepalive type http
   (config-service[serv1])# keepalive method get
   (config-service[serv1])# keepalive uri "/testpage.html"
   (config-service[serv1])# keepalive hash "1024b91e516637aaf9ffca21b4b05b8c"
   (config-service[serv1])# active
   ```

2. Display the hash value using the `show keepalive` command. For example, enter:

   ```
   (config-service[serv1])# show keepalive
   Keepalives:
   Name: serv1          Index: 0          State: ALIVE
   Description: Auto generated for service serv1  Address: 10.0.3.21  Port: 80
   Type: HTTP:GET:/testpage.html  Hash: 1024b91e516637aaf9ffca21b4b05b8c
   Frequency: 5  Max Failures: 3  Retry Frequency: 5
   Dependent Services:
   ```

3. Use the hash value from the keepalive display to configure the keepalive hash. Enter the MD5 hash as a quoted hexadecimal string with a maximum of 32 characters. For example, enter:

   ```
   (config-service[serv1])# keepalive hash "1024b91e516637aaf9ffca21b4b05b8c"
   ```
An excerpt of the service configuration from the running-config is as follows:

```
service serv1
  ip address 10.0.3.21
  keepalive type http
  keepalive method get
  keepalive uri "/testpage.html"
  keepalive hash "1024b91e516637aaf9ffca21b4b05b8c"
  active
```

To clear a hash value and return to the default hash value, enter:

```
(config-service[serv1])# no keepalive hash
```

For a global keepalive, use the `hash` command to override the default MD5 hash for an HTTP GET keepalive. To configure the hash value for a global keepalive:

1. Configure the global keepalive. For example, enter:

```
(config-keepalive[keepimages])# method get
(config-keepalive[keepimages])# uri "/testpage.html"
(config-keepalive[keepimages])# hash
  "1024b91e516637aaf9ffca21b4b05b8c"
```

2. Configure the service. For example, enter:

```
(config)# service imageserver1
(config-service[imageserver1])# ip address 10.0.3.21
(config-service[imageserver1])# keepalive type named keepimages
(config-service[imageserver1])# active
```

3. Display the hash value using the `show keepalive` command. For example, enter:

```
(config-keepalive[keepimages])# show keepalive
```

Keepalives:

- **Name:** imageserver1
  - **Index:** 0
  - **State:** ALIVE
  - **Description:** Auto generated for service serv1
  - **Address:** 10.0.3.21
  - **Port:** 80
  - **Type:** HTTP GET:/testpage.html
  - **Hash:** 1024b91e516637aaf9ffca21b4b05b8c
  - **Frequency:** 5
  - **Max Failures:** 3
  - **Retry Frequency:** 5
  - **Dependent Services:**
4. Use the hash value from the keepalive display to configure the keepalive hash. Enter the MD5 hash value as a quoted hexadecimal string with a maximum of 32 characters. For example, enter:

```
(config-keepalive[keepimages])# hash
"1024b91e516637aaf9ffca21b4b05b8c"
```

An excerpt of the service configuration from the running-config is as follows:

```
service imageserver1
  ip address 10.0.3.21
  keepalive type http
  keepalive method get
  keepalive uri "/testpage.html"
  keepalive hash "1024b91e516637aaf9ffca21b4b05b8c"
active
```

To clear a hash value and return to the default hash value, enter:

```
(config-keepalive[keepimages])# no hash
```

**Showing Keepalive Configurations**

To display keepalive information for a service, use the `show service` command. For more information on this command and what it displays, see the “Showing Service Configurations” section in Chapter 3, Configuring Services.

To display global keepalive configurations, use the `show keepalive` command. To display a list of existing keepalives, use the `show keepalive ?` command.

---

**Note**

Two sessions (for example, SSH, console or Telnet) can access keepalive data at the same time. If one session views the data through the `show keepalive` command when the other session reconfigures the keepalive data by clearing a service or a keepalive, the CSS may abort the `show` command and display the following message:

```
Command Aborted!!! Configuration changed. Please reissue command.
```
This command provides the following options:

- **show keepalive** - Display information for all keepalives
- **show keepalive keepalive_name** - Display information for a specific keepalive
- **show keepalive-summary** - Display summary information for all keepalives

For example, enter:

```
(config)# show keepalive
```

Keepalives:

- **Name**: keepimages  **Index**: 1  **State**: ALIVE (ICP Check)
  - **Description**: This keepalive is for image servers
  - **Address**: 172.16.1.7  **Port**: 80
  - **Type**: HTTP:HEAD-302:/index.html
  - **Frequency**: 5
  - **Max Failures**: 3
  - **Retry Frequency**: 5
  - **Dependent Services**: imageserver1

- **Name**: rualive  **Index**: 2  **State**: ALIVE
  - **Description**: Auto generated for service serv2
  - **Address**: 172.16.1.8  **Port**: 80
  - **Type**: HTTP:HEAD:/index.html
  - **Frequency**: 5
  - **Max Failures**: 3
  - **Retry Frequency**: 5
  - **Dependent Services**: serv2

```
(config)# show keepalive-summary
```

Keepalives:

- **Alive1**: DOWN  192.25.1.7
- **Alive2**: ALIVE  192.25.1.8

Table 4-3 describes the fields in the **show keepalive** command output.

**Table 4-3  Field Descriptions for the show keepalive Command Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the keepalive.</td>
</tr>
<tr>
<td>Index</td>
<td>The CSS-assigned unique index value for each keepalive</td>
</tr>
</tbody>
</table>
Using Script Keepalives with Services

Script keepalives are scripts that you can create to provide custom keepalives for your specific service requirements. To create the scripts, use the rich CSS Scripting Language that is included in your CSS software. For details on using the CSS Scripting Language, including using socket commands and examples of keepalive scripts, refer to the Cisco Content Services Switch Administration Guide.

Currently, a CSS provides keepalives for FTP, HTTP, ICMP, SSL, and TCP. For information on configuring keepalive messages, see the “CSS Keepalive Overview” section earlier in this chapter.

Table 4-3  Field Descriptions for the show keepalive Command Output

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>The state of the keepalive. The possible states are Down, Alive, Dying, Suspended, and No Services.</td>
</tr>
<tr>
<td>Description</td>
<td>The description for the keepalive.</td>
</tr>
<tr>
<td>Address</td>
<td>The IP address where the keepalive messages are sent.</td>
</tr>
<tr>
<td>Port</td>
<td>The port number for the keepalive.</td>
</tr>
<tr>
<td>Type</td>
<td>The type of keepalive message assigned to the keepalive. The possible types are FTP, HTTP, ICMP, script, SSL, TCP, or named. For an HTTP Head keepalive, the response code is also displayed.</td>
</tr>
<tr>
<td>Frequency</td>
<td>The time, in seconds, between keepalive messages sent to the IP address. The range is from 2 to 255. The default is 5.</td>
</tr>
<tr>
<td>Max Failures</td>
<td>The configured number of times the IP address can fail to respond to a keepalive message before being considered down. The range is from 1 to 10. The default is 3.</td>
</tr>
<tr>
<td>Retry Frequency</td>
<td>The retry period, in seconds, to send messages to the keepalive IP address. The range is from 2 to 255. The default is 5.</td>
</tr>
<tr>
<td>Dependent Services</td>
<td>Services currently configured to use the keepalive. This is mainly used for named keepalive types.</td>
</tr>
</tbody>
</table>
Using script keepalives allow you to extend the CSS keepalive functionality beyond the default keepalives. For example, you can develop a script specifically to connect a CSS to a Post Office Protocol 3 (POP3) mail server.

Once you create a script offline, you can upload it to the CSS and configure the script keepalive option on a service.

The CSS supports a maximum of 256 script keepalives. If you specify a script to parse the output for each executed command, you can configure only 16 keepalives that use script output.

**Note**

You can also configure a script keepalive without having the corresponding script present on the CSS. In this case, a constant Down state remains on the service until you upload the appropriate script to the CSS. This allows you to develop and implement a configuration before uploading all the scripts to the CSS.

### Script Keepalive Considerations

When you configure a script keepalive, follow the same general guidelines as those for keepalive types, with the exceptions noted in these sections. For details on keepalives, see the “CSS Keepalive Overview” section earlier in this chapter.

- The CSS provides scripted keepalives to support the need for keepalives operations that cannot be handled using non-scripted keepalives. Cisco recommends that you limit I/O operations in a scripted keepalive to socket operations used to probe network connectivity to a server and for determining application health on a server. Although the scripting language supports file I/O on the CSS hard drive or flash drive, Cisco recommends that you do not use file I/O operations within scripted keepalives. Extensive file I/O operations within scripted keepalives may cause services to transition. File system access is allowed in scripts executed from the CLI or from the command scheduler.

- The CSS Scripting Language allows you to pass 128 characters in a quoted argument. Assuming an average of seven characters per argument (plus a space delimiter), you can potentially use a maximum of 16 arguments in one script.
The CSS executes each line in a script keepalive. If your application requires numerous script keepalives (for example, greater than 60), keep each script as short and concise as possible. A smaller script yields much faster script execution results than a larger size script. To maximize CSS system performance, avoid complex protocols or extensive scripts (for example, no database queries, not performing a full login with validation), which can take the CSS longer to execute.

- Use the script naming convention of *ap-kal-type*, so that when you press Tab or “?” , you can easily see the keepalive scripts available for use. For example, an SMTP script would be named ap-kal-smtp. The script name can have a maximum of 32 characters. The arguments must be in a quoted text string with a maximum of 128 characters.

- For the configured script keepalive to find the corresponding script, the script must reside in the */<current running version>/script* directory. When you configure a script keepalive, use only script names. (A CSS does not accept path names.) If the script is present elsewhere on the CSS, the script keepalive assumes it does not exist.

To see a complete list of all scripts available in the */<current running version>/script* directory, press the Tab key or “?”. Optionally, you can type a script name not found in the list, then you can upload the script later. You can manipulate scripts using the archive, clear, and copy commands. You can also upload a script from a local hard drive to the /script directory on the CSS, or download a script from the /script directory on the CSS to a local hard drive.

- Because many scripts have a multistep process such as connecting, sending a request, and waiting for a specific type of response, configure a higher frequency time value for script keepalives than for standard keepalives. A time interval of 10 seconds or higher ensures that the script keepalive has enough time to finish. Otherwise, state transitions may occur more often than is usual.

- The CSS sends keepalives only to the first address in a service range. If you configure a service with a range of IP addresses and configure a script keepalive with an IP address to it, the address must be the first address in a service range.
Because a CSS reads an entire script into memory, there is a maximum script keepalive size of 200 KB (approximately 6,000 lines). If a script exceeds this limit, it will not load. This should be more than adequate for all applications. For example, the script keepalives included with your CSS software are approximately 1 KB. To further conserve CSS memory, services can share a common script keepalive so that only one instance of the script needs to reside in memory. However, you must configure the script keepalive for each service where you want the script to run.

Note

For a large number of services that use script keepalives, use a smaller subset of global keepalives to handle the work for them. For information on global keepalives, see the “Configuring Global Keepalives” section earlier in this chapter.

Configuring Script Keepalives

Script keepalives are scripts that you can create to provide custom keepalives for your specific service requirements. Use the `keepalive type script` command to configure script keepalives. The syntax for this service configuration mode command is:

```
keepalive type script script_name {"arguments"} {use-output}
```

Enter the name of an existing script keepalive. The optional `arguments` variable passes arguments into the keepalive script. Enter a quoted text string with a maximum of 128 characters including spaces.

The optional `use-output` keyword allows the script to parse the output for each executed command. This optional keyword allows the use of `grep` and `file` direction within a script. You can configure a maximum of 16 script keepalives (out of a maximum of 255 script keepalives) to use script output. By default, the script does not parse the output.

For example, to configure a script keepalive named `ap-kal-httplist`, enter:

```
(config-service[serv1])# keepalive type script ap-kal-httplist
"10.10.102.105 /default.htm"
```
In the previous example, the `keepalive` command configures the serv1 service keepalive to be of type script with the script name ap-kal-httplist and the arguments “10.10.102.105 /default.htm”. The output is not parsed by the script.

To disable a script keepalive on a service, enter:

```
(config-service[serv1])# keepalive type none
```

**Viewing a Script Keepalive in a Service**

When you add a script keepalive to a service, the CSS recognizes that the script is the keepalive for the service in the `show service` screen. The script name appears in the Keepalive field, and any potential arguments appear directly below in the Script Arguments field. If there are no script arguments, then the Script Arguments field does not appear.

For example, enter:

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

```
Name: serv1                    Index: 1
  Type: Local                  State: Alive
  Rule (10.10.102.105 ANY ANY)
  Session Redundancy: Disabled
  Redirect Domain:
  Redirect String:
  Keepalive: (SCRIPT ap-kal-httplist 10 3 5)
  Script Arguments: "10.10.102.105 /default.htm"
  Script Error: None
  Script Run Time: 1 second
  Script Using Output Parsing: No
  Last Clearing of Stats Counters 03/15/2002 13:45:01
  Mtu:                         1500
  Connections:                 0
  Total Connections:           0
  Weight:                      1
  Load:                        2
```

**Note**

If a script keepalive terminates with an error, you can use the Script Error and Script Run Time fields to help troubleshoot the problem.
Using Script Keepalives with Services

You can also use the `show running-config` command to display the script keepalive and its arguments.

For example, enter:

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

service serv1
 ip address 10.10.102.105
 keepalive frequency 10
 keepalive type script ap-kal-httplibst "10.10.102.105/default.htm"
 active
```

The example above shows the script keepalive and arguments that have been configured on a service. If no arguments are specified in the script, then the quoted text following the script name will not appear.

**Script Keepalive Status Codes**

A script can return a status code of zero or non-zero. On a return of non-zero, the CSS flags the service state as Dying or Down; on a return of zero, the CSS flags the service state as Alive. For example, enter:

```
! Connect to the remote host
socket connect host einstein port 25 tcp
! Purposely fail
exit script 1
```

Because the above script fails when it executes the `exit` command, the script returns a non-zero value. By default, the script will fail with a syntax error if the `connect` command fails. Be sure to check the logic of your scripts to ensure that the CSS returns the correct value.

**Script Keepalives and Upgrading WebNS Software**

When you upgrade the WebNS software in your CSS, the upgrade process creates a new `<current running version>/script` directory. You must copy your custom scripts (including custom script keepalives) to the new `<current running version>/script` directory so that the CSS can find them.
Use the following procedure to ensure that your custom script keepalives operate properly after upgrading the software.

1. Upgrade the WebNS software in your CSS. Refer to the Cisco Content Services Switch Administration Guide.

2. Copy the scripts from the old /<current running version>/script directory to the new /<current running version>/script directory.

3. Reboot the CSS.
Using Script Keepalives with Services
Configuring Source Groups for Services

A source group is a collection of local servers that initiate flows from within the local web farm. The CSS enables you to treat a source group as a virtual server with its own source IP address to which all IP addresses of services configured in the group will be translated. For example, if you configure several streaming audio transmitters as a group, the CSS will process flows from the group members and give them all the same source IP address.

This chapter describes how to configure source groups for services.

- Overview of Source Groups and Port Mapping
- Source Group Configuration Quick Start
- Creating a Source Group
- Configuring the Source Group
- Activating and Suspending a Source Group
- Configuring Source Group Port Mapping
- Configuring Source Groups and ACLs
- Configuring a Source Group for FTP Connections
- Configuring Source Groups to Allow Servers to Resolve Domain Names Using the Internet
- Showing Source Groups
- Clearing Source Group Counters

Information in this chapter applies to all CSS 11500 models except where noted.
Overview of Source Groups and Port Mapping

When you configure a source group, a CSS provides network address translation (NAT) of source IP addresses and port address translation (PAT) of source ports. NAT and PAT add a measure of security to your network by not exposing private network addresses and ports to the public side of a CSS. To NAT source IP addresses and source ports for flows originating from a server (server-side) on the private side of the CSS, add existing services to a source group. To NAT source IP addresses and source ports for flows originating from a client (client-side) on the public side of the CSS, add existing services to a source group as destination services. You can also configure access control lists (ACLs) to perform source NATing. For information about ACLs, refer to the Cisco Content Services Switch Security Configuration Guide.

Each CSS module (except the SSL module) has one session processor (SP) that is responsible for mastering flows.

- CSS 11501 supports one SP
- CSS 11503 supports a maximum of three SPs
- CSS 11506 supports a maximum of six SPs

The default number of source ports available for a single source group is 63488 (65533 minus the named ports). With one source group configured, the CSS allocates the total number of ports proportionally among all the SPs in the CSS chassis according to the SP relative weight value. To display the relative weight value of an SP, enter the show chassis session-processors command as described in the Cisco Content Services Switch Administration Guide. The SP relative weight value is not configurable.

For client-side flows, the CSS sends packets to different SPs for flow processing and the flows have access to the source ports in that SP. The CSS performs a simple XOR hash of the TCP or UDP source and destination port numbers to determine the SP that becomes master for that flow. If the port numbers are the same (for example, DNS UDP port 53), then the CSS uses the low order bits of the source and destination IP addresses to calculate the hash value. The CSS uses the hash value to index into a weighted table of SPs and selects the appropriate SP.

When the CSS performs PAT, the master SP for the flow uses a source port from either a source group or the global port mapper, depending on your configuration. (For information about global port mapping, see the “Configuring Global Port Mapping” section in Chapter 2, Configuring Flow and Port Mapping Parameters.)
The CSS chooses a source port so that the hash of the source port and the destination port will cause the CSS to select the same SP for the server-side flow as the SP that mastered the client-side flow.

For the server-side flow from a given destination port, only certain source port numbers hash to the same SP that was used for the client-side flow. For this reason, all ports available to a particular SP are not necessarily eligible for use when establishing the back-end connection. Therefore, the hash algorithm selects only a percentage of the available ports on any one SP.

To make more available source ports eligible for flows or to provide additional source ports for each SP, use one of the following methods:

- Configure a VIP address range for port mapping using the `portmap vip-address-range` command. For each additional VIP address that you configure for port mapping, you add one more port mapper to your configuration with another 63488 available ports. This method requires that you configure a destination service on a source group. For details, see “Configuring a VIP Address Range for Port Mapping” section.

- Configure services on different destination ports (vary the destination port) to broaden the hash across the SPs and allow a larger percentage of available ports to be eligible for port mapping. This strategy works by making the hashing algorithm less restrictive in the sense that now more source ports can be used to satisfy the hashing equations. Use this method when you cannot use the `vip-address-range` command because of limited server-side address space. For each additional destination port that you configure, the CSS receives an additional set of eligible source ports to use for port mapping as shown in the second column of Table 5-1. This method has the following requirements:
  - Configure your web server to listen on multiple ports (for example ports 80, 81, 82, and so on)
  - For each destination port, configure a new service on the CSS
  - Add the services to a content rule
  - Add the services as destination services to a source group

- Configure multiple source groups to provide an additional 63488 ports for each source group, which the CSS also distributes among the SPs in the same manner as described earlier in this section. This method requires that you:
  - Configure multiple IP addresses on your web server (IP aliases)
  - Create a new service on the CSS for each server IP address
Overview of Source Groups and Port Mapping

- Add each service to a unique source group as a destination service
- Add the services to a content rule

Table 5-1 illustrates how the number of eligible ports in a CSS 11506 decreases as you increase the number of installed modules (SPs) and how you can dramatically increase the number of eligible ports by configuring a VIP address range for port mapping. In all cases, the CSS is configured with one service in one source group with a single destination port for all flows (for example, port 80). The numbers of eligible ports in Table 5-1 are approximate and are used for illustration only. Your results may vary depending on your configuration.

Table 5-1  Adding Modules (SPs) to a CSS 11506 Decreases the Number of Eligible Source Ports While Adding VIP Addresses for Port Mapping Increases the Number of Eligible Source Ports

<table>
<thead>
<tr>
<th>Number of Modules (SPs)</th>
<th>Number of Eligible Source Ports for the Chassis port-map vip-address-range = 1</th>
<th>Number of Eligible Source Ports for the Chassis port-map vip-address-range = 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>63488</td>
<td>634880</td>
</tr>
<tr>
<td>2</td>
<td>33728</td>
<td>337280</td>
</tr>
<tr>
<td>3</td>
<td>21824</td>
<td>218240</td>
</tr>
<tr>
<td>4</td>
<td>16616</td>
<td>166160</td>
</tr>
<tr>
<td>5</td>
<td>13144</td>
<td>131440</td>
</tr>
<tr>
<td>6</td>
<td>11408</td>
<td>114080</td>
</tr>
</tbody>
</table>

Table 5-2 shows that, by increasing the number of destination ports, even in a fully-loaded CSS 11506 (six SPs), you can dramatically increase the number of source ports that are eligible for port mapping. You can even more dramatically increase the number of eligible source ports by configuring a higher VIP address range for port mapping. In this example, the destination ports were chosen consecutively.
By comparing row six in Table 5-1 with row 1 in Table 5-2, you can see that increasing the number of destination ports to 10 more than doubles the number of source ports eligible for port mapping.

Note that it is algorithmically significant which destination ports you select to increase the number of eligible source ports and it is not a linear relationship. You may need to select several ranges of destination ports to produce the maximum number of eligible source ports.

Adaptive Session Redundancy (ASR) requires that both CSSs have the same number of SPs installed in the same relative order (skipping slots is acceptable) in each chassis. This requirement allows the port mapper to use the same port-selection algorithm used in a non-ASR configuration. There is no further restriction on the number of eligible source ports in an ASR configuration. For more information about ASR, refer to the Cisco Content Services Switch Redundancy Configuration Guide.

### Source Group Configuration Quick Start

Use the procedure in Table 5-3 to configure a source group for TCP/UDP traffic. To configure a source group for FTP traffic, see the next section. Note that each source group requires a content rule that contains the same services and virtual IP address (VIP) as the source group.
Table 5-3  Source Group Configuration Quick Start

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Create the source group. Source group names can be a maximum of 31 characters. The following example creates a source group ftpgroup.</td>
</tr>
<tr>
<td>(config)# group ftpgroup</td>
</tr>
<tr>
<td>The CLI transitions into config-group mode where you can configure attributes for the source group and activate it.</td>
</tr>
<tr>
<td>(config-group[ftpgroup])#</td>
</tr>
<tr>
<td>2. Configure the source group VIP address to which all service IP addresses will be translated. For example, enter:</td>
</tr>
<tr>
<td>(config-group[ftpgroup])# vip address 172.16.36.58</td>
</tr>
<tr>
<td>You can assign the same VIP address to multiple source groups, but only one of the source groups can be active at a time.</td>
</tr>
<tr>
<td>3. Add previously defined services to the source group. For example, enter:</td>
</tr>
<tr>
<td>(config-group[ftpgroup])# add service server1</td>
</tr>
<tr>
<td>(config-group[ftpgroup])# add service server2</td>
</tr>
<tr>
<td>4. Activate the source group.</td>
</tr>
<tr>
<td>(config-group[ftpgroup])# active</td>
</tr>
<tr>
<td>Because a VIP address can belong to only one active source group at a time, the CSS will not allow you to activate a second source group that contains the same VIP address as the one in the active source group.</td>
</tr>
</tbody>
</table>
The following running-configuration example shows the results of entering the commands in Table 5-3.

```
!*************************** GROUP ***************************
group ftpgroup
  vip address 172.16.36.58
  add service server1
  add service server2
  active

!*************************** OWNER ***************************
owner arrowpoint
  content ftpsource1
    add service server2
  vip address 172.16.36.58
  active
```

The following running-configuration example shows the results of entering the commands in Table 5-3.

```
!*************************** GROUP ***************************
group ftpgroup
  vip address 172.16.36.58
  add service server1
  add service server2
  active

!*************************** OWNER ***************************
owner arrowpoint
  content ftpsource1
    add service server2
  vip address 172.16.36.58
  active
```
Creating a Source Group

Group configuration mode allows you to configure a maximum of 255 source groups on a CSS. To access group configuration mode, use the group command from any mode except ACL and boot configuration modes. The syntax for this command is:

```
  group groupname
```

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

For example, enter:

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

To view a list of existing source groups, enter:

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

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

To remove a source group, enter:

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

Configuring the Source Group

This section describes how to configure a source group.

- Configuring a VIP Address for a Source Group
- Configuring a Service on a Source Group
- Adding a Destination Service to the Source Group

For information on configuring source group port mapping, see the “Configuring Source Group Port Mapping” section. After you configure a source group, you can activate it, as described in the “Activating and Suspending a Source Group” section.
To make certain modifications to an active source group, you must first suspend the source group using the `suspend` command. Such modifications include:
changing the IP address to 0 or using the `no ip address` command, adding or removing a service or destination service, or using the `portmap` command.

**Configuring a VIP Address for a Source Group**

When a CSS performs NAT, it substitutes a VIP for the source IP address in flows originating from one of the group’s sources or destined to one of the group’s destinations if you configured the service with the `add destination service` command. NATing provides a measure of security by preventing the source IP address from being exposed on the Internet. You can assign the same VIP address to multiple source groups, but only one of the source groups can be active at a time.

Use the `vip address` command to specify the base VIP address for the group. The syntax for this group configuration mode command is:

```
vip address ip_or_host {range number}
```

The options and variables for this command are:

- `ip_or_host` - IP address or name for the group. 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).
- `range number` - (Optional) Defines the range of IP addresses for the group. Enter a `number` from 1 to 65353. The default is 1. The `ip_or_host` variable is the first address in the range.

When you configure the base VIP address of a source group, be sure to leave enough address space for expansion in case the CSS uses all configured port-map entries and you need to increase the VIP address range used for port mapping. See the “Configuring a VIP Address Range for Port Mapping” section.

For example enter:

```
(config-group[ftpgroup])# vip address 172.16.36.58 range 3
```
Configuring a Service on a Source Group

To NAT source IP addresses and source ports for flows originating from a server (server-side) on the private side of the CSS, add existing services to a source group. You can configure a maximum of 64 services per source group.

A service may belong to only one group at a time. When the source group is active and the same service is selected through a content rule, ACL preferred service or sorry service, the source group is used to NAT (Network Address Translation) the source address. The service must be active in order for it to perform source address NATing for the source group.

Be aware that you cannot use a service with:

- The same name in other source groups or use the destination service list within the same source group
- The same address as a source service on another source group

To add previously defined services to the source group, use the add service command. For example, enter:

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

To remove a previously configured service from a source group, use the remove service command. For example, to remove service server1 from the source group, enter:

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

Adding a Destination Service to the Source Group

To NAT source IP addresses and source ports for flows originating from a client (client-side) on the public side of the CSS, add existing services to a source group as destination services. You can configure a maximum of 64 services per source group. Be aware that:

- You cannot use a service with the same name in other source groups or use the source service list within the same source group.
- You can use services with duplicate addresses among destination services because the actual service is chosen through content rule selection.
The destination service must be active and must be added to a content rule for it to perform destination source address NATing for the source group (see Chapter 9, Configuring Content Rules).

**Note**

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

Use the `add destination service` command to add a destination service to a source group. For example, enter:

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

To remove a previously configured destination service from a source group, use the `remove` command.

```
(config-group[ftpgroup])# remove destination service server2
```

### Activating and Suspending a Source Group

When you activate a source group, the CSS uses it to NAT (Network Address Translation) the source IP address. After you configure a source group, you can activate it. Because a VIP address can belong to only one active source group at a time, the CSS does not allow you to activate a second source group that contains the same VIP address as the one in the active source group.

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

Suspend the source group when you need to change its configuration. The group and its attributes remain the same but no longer have an effect on flow creation. Use the `suspend` command to suspend a source group. For example, enter:

```
(config-group[ftpgroup])# suspend
```
Configuring Source Group Port Mapping

By default, PAT or port mapping is enabled for source groups on source ports greater than 1023. The CSS translates such source ports to a range starting at 2016. The following sections provide information about how to change the default PAT behavior of the CSS:

- Configuring the Starting Port Number
- Configuring the Total Number of Ports in a Port-Map Range
- Configuring a VIP Address Range for Port Mapping
- Disabling Port Mapping

Before configuring an active source group, make sure that you suspend it.

Configuring the Starting Port Number

By default, the base port (starting port number) for the CSS is 2016. The `portmap base-port` command defines the base port for the CSS. You can enter a base port value from 2016 to 63456. For example, to configure a base port of 3354, enter:

```
(config-group[ftpgroup])# portmap base-port 3354
```

To reset the base port to its default value of 2016, use the `no portmap base-port` command. For example, enter:

```
(config-group[ftpgroup])# no portmap base-port
```

Configuring the Total Number of Ports in a Port-Map Range

The CSS allocates the total number of configured ports proportionally among all the SPs in the CSS chassis according to the session processor relative weight value. To display the relative weight value of a session processor, enter the `show chassis session-processors` command as described in the Cisco Content Services Switch Administration Guide.
The more modules you add to the CSS chassis, the less session processing each module performs and the fewer ports the CSS makes available to each module. To display the number of ports that the CSS allocates to each module, enter the `show group portmap` command as described in the “Showing Source Groups” section. For more information about the port mapping behavior of the CSS, see the “Configuring Source Group Port Mapping” section.

By default, the total number of ports in the port-map range for the entire CSS is 63488. This default value should be fine for most applications. To define the total number of ports in the port-map range, use the `portmap number-of-ports` command. Enter a number from 2048 to 63488. If you enter a value that is not a multiple of 32, the CSS rounds up the value to the next possible multiple of 32. For example, to configure the total number of ports to 2048, enter:

```text
(config-group[ftpgroup])# portmap number-of-ports 2048
```

To reset the number of ports to the default value, use the `no portmap number-of-ports` command. For example, enter:

```text
(config-group[ftpgroup])# no portmap number-of-ports
```

### Configuring a VIP Address Range for Port Mapping

For each source group that you configure, a maximum of 63488 (the default) source ports are available for port mapping. However, not all available ports are eligible for flows. For details about source groups and port mapping, see the “Overview of Source Groups and Port Mapping” section.

To increase the number of available ports for port mapping, you can configure the port mapper with additional VIP addresses by specifying a range of VIPs. For each additional VIP address that you configure, the CSS creates a new port mapper to manage the available ports for that VIP. When the CSS performs PAT, the source group roundrots among all the configured port mappers and the selected port mapper chooses the next eligible port for a given VIP.

Note that configuring a VIP address range for port mapping is different from a Virtual Web Hosting (VWH) configuration where you configure a VIP address range on a source group, not the port mapper. In a VWH configuration, there is only one port mapper available. For information about VWH, see the “Configuring Virtual Web Hosting” section in Chapter 9, Configuring Content Rules.
The CLI enforces the following configuration restrictions:

- You cannot configure virtual Web hosting and a port mapper VIP address range in the same source group. For information about virtual Web hosting, see the “Configuring Virtual Web Hosting” section in Chapter 9, Configuring Content Rules.

- You cannot configure a service (using the `add service` command) and a port mapper VIP address range in the same source group. For information about the `add service` command, see the “Configuring a Service on a Source Group” section.

- You cannot configure a port mapper VIP address range in a source group that is used by an ACL. The reverse is also true. For information about ACLs, refer to the Cisco Content Services Switch Security Configuration Guide.

- You can configure a maximum of 255 port mappers on one CSS. You can reach this limit by configuring any of the following:
  - A port-map VIP address range of 255 on one source group
  - A port-map VIP address range of 1 on 255 source groups
  - A combination of port-map VIP address ranges configured on a number of source groups that total 255 port mappers

To configure additional VIP addresses for the port mapper of a source group, use the `portmap vip-address-range` command in group configuration mode. The syntax of this command is:

```
portmap vip-address-range number
```

The `number` variable indicates a range of VIP addresses starting with the address specified by the `vip address` command in group configuration mode. Enter an integer from 1 to 255. The default is 1. For information about configuring a VIP address for a source group using the `vip address` command, see the “Configuring a VIP Address for a Source Group” section.

---

**Note**

When you configure the base VIP address of a source group, be sure to leave enough address space for expansion in case the CSS uses all configured port-map entries and you need to increase the VIP address range used for port mapping. See the “Configuring a VIP Address for a Source Group” section.
If you observe no-portmap errors, configure the `portmap vip-address-range` command and set the range to a value greater than that required to support the maximum number of active connections that you anticipate for your application.

With a VIP range of 255, the maximum number of eligible ports on an SCM in a fully populated CSS 11506 chassis is 63240. For other SPs or chassis configurations, the number of ports is greater.

For example, to configure the port mapper of a source group with three VIP addresses, enter:

```
(config-group[ftpgroup])# portmap vip-address-range 3
```

If the configured VIP for the source group is 192.168.44.3, then, after entering the above `portmap vip-address-range` command, the three available VIPs for the port mapper would be:

- 192.168.44.3
- 192.168.44.4
- 192.168.44.5

To reset the VIP address range to the default value of 1, enter:

```
(config-group[ftpgroup])# no portmap vip-address-range
```

### Disabling Port Mapping

By default, the CSS NATs source IP addresses and PATs source ports for a configured source group. If you configure the `portmap disable` command in a source group, the CSS performs NAT on the source IP addresses but does not perform PAT on the source ports of UDP traffic that matches on that source group.

For UDP applications with high-numbered assigned ports (for example, SIP and WAP), we recommend that you preserve those port numbers by configuring destination services in source groups instead of using the `portmap disable` command. Destination services cause the CSS to NAT the client source ports, but not the destination ports. For information about configuring destination services, see Chapter 3, Configuring Source Groups for Services.
If you disable flows for a UDP port using the flow-state table and configure the `portmap disable` command in a source group, traffic for that port that matches on the source group may be returned to the client on an unrecognizable port number. For information about the flow-state table, see Chapter 2, Configuring Flow and Port Mapping Parameters.

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

The `portmap disable` command does not affect TCP flows.

To disable port mapping, enter:

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

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

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

## Configuring Source Groups and ACLs

For the CSS to perform NAT for traffic destined to the Internet and not to perform NAT for local traffic, you can use ACLs with source groups to make the decision based on the destination IP address in the ACL.

In the following example, clients on 10.0.1.0 and 10.0.2.0 private subnets want to communicate with each other without the source group NATing their traffic. Three VLANs exist, one for each subnet (VLAN1 and VLAN2) and a VLAN to the Internet through the source group (VLAN3).
1. Create a source group and activate it. In this example, the source group is named outbound and has a VIP address of 192.168.1.10.

   (config) # group outbound
   Create group <outbound>, [y/n]: y
   (config-group[outbound]) # vip address 192.168.1.10
   (config-group[outbound]) # active

   Note that the VIP address in the source group must be a public address allowing the routing of response traffic to the CSS. The address can be an IP address in the same subnet as the IP address configured for the VLAN3 circuit (but not the same IP address), or a different public IP address that the routers in the network have static routes pointing to the CSS.

2. Create an ACL that allows the clients on the private subnet to communicate to each other. The following ACL and clause allows clients on 10.0.1.0 subnet to communicate with clients on 10.0.2.0 subnet without the source group using NATing because the CSS uses the bypass option to route the traffic and bypass all rules configured on the CSS.

   (config) # acl 1
   Create ACL <1>, [y/n]: y
   (config-acl[1]) # clause 2 bypass any 10.0.1.0 255.255.255.0
   destination 10.0.2.0 255.255.255.0

3. Add a clause to direct all other traffic from the clients on the 10.0.1.0 subnet to the source group, allowing the source IP address to use NAT to connect to 192.168.1.10.

   (config-acl[1]) # clause 10 permit any 10.0.1.0 255.255.255.0
   destination any sourcegroup outbound

4. Add a clause 1 to permit the keepalives for the services on the CSS.

   (config-acl[1]) # clause 1 permit icmp any destination any

5. Apply the ACL to VLAN1.

   (config-acl[1]) # apply circuit-(VLAN1)
   (config-acl[1]) # exit
Configuring Source Groups for FTP Connections

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

**Note**

When you use an FTP content rule with a configured VIP address range, be sure to configure the corresponding source group with the same VIP address range (see Chapter 9, Configuring Content Rules).

To configure FTP sessions to a VIP:

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

6. If you want to allow traffic from the servers on VLAN2 to the source group but also allow the servers to communicate with VLAN1 without using a NAT IP address, configure the following ACL for VLAN2.

```config
(config) # acl 2
Create ACL <2>, [y/n]:y
(config-acl[2]) # clause 2 bypass any 10.0.2.0 255.255.255.0
destination 10.0.1.0 255.255.255.0
(config-acl[2]) # clause 10 permit any 10.0.2.0 255.255.255.0
destination any sourcedroup outbound
(config-acl[2]) # apply circuit-(VLAN2)
(config-acl[2]) # exit
```

7. For inbound traffic from the Internet, configure an ACL for VLAN3.

```config
(config) # acl 3
Create ACL <3>, [y/n]:y
(config-acl[3]) # clause 1 permit any any destination any
(config-acl[3]) # apply circuit-(VLAN3)
(config-acl[3]) # exit
```

8. Globally enable all ACLs on the CSS.

```config
(config) # acl enable
```
2. Configure a source group defining the same VIP and services as configured in the content rule.

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

The following running-config example shows source group ftp_group.

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

---

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

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

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

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

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

2. Create a content rule to process DNS replies. The content rule to process DNS replies is in addition to the content rules you created to process Web traffic. The content rule example below enables the CSS to NAT inbound DNS replies from the public VIP address (192.168.200.200) to the server’s private IP address (10.0.3.251).

   The following example creates content rule dns1 with a public VIP 192.168.200.200 and adds server Server1.

   ```
   (config-owner[arrowpoint.com])# content dns1
   (config-owner-content[arrowpoint.com-dns1])# vip address 192.168.200.200
   (config-owner-content[arrowpoint.com-dns1])# add service Server1
   (config-owner-content[arrowpoint.com-dns1])# active
   ```

3. Create a source group to process DNS requests. The source group enables the CSS to NAT outbound traffic source IP addresses from the server’s private IP address (10.0.3.251) to the public VIP address (192.168.200.200).

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

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

   The following example creates source group dns1 with public VIP address 192.168.200.200 and adds the service Server1.

   ```
   (config)# group dns1
   (config-group[dns1])# vip address 192.168.200.200
   (config-group[ dns1])# add service Server1
   (config-group[ dns1])# active
   ```
Showing Source Groups

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

- `show group` - Displays all source group configurations.
- `show group group_name` - Displays the source group configuration specified by `group_name`. You cannot specify a group name in Group mode.
- `show group group_name portmap` - Displays detailed port mapping information for each SP in a CSS.
- `show group group_name portmap all` - Displays detailed port mapping information about each SP in a CSS for all VIP addresses of the source group port mapper.
- `show group group_name portmap ip_address` - Displays detailed port mapping information about each SP in a CSS for the specified VIP address of the source group port mapper.

For example, enter:

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

Table 5-4 describes the fields in the `show group` command output.

### Table 5-4  Field Descriptions for the `show group` Command Output

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>Name of the group, whether the group is activated (Active) or suspended (Suspend), and the source IP address for the group.</td>
</tr>
<tr>
<td>Portmap VIP Range</td>
<td>Number of configured VIP addresses that the port mapper can use for NAT and the address range.</td>
</tr>
<tr>
<td>Session Redundancy</td>
<td>Indicates whether ASR is enabled or disabled for the source group. For details on ASR, refer to the <em>Cisco Content Services Switch Redundancy Configuration Guide</em>.</td>
</tr>
</tbody>
</table>
### Table 5-4  Field Descriptions for the show group Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redundancy Global Index</td>
<td>The unique global index value for Adaptive Session Redundancy assigned to the source group using the <code>redundant-index</code> command in group configuration mode.</td>
</tr>
<tr>
<td>Associated ACLs</td>
<td>Any ACLs associated with the group.</td>
</tr>
<tr>
<td>Source/Destination Services</td>
<td>The source or destination services of the source group.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the service.</td>
</tr>
<tr>
<td>Hits</td>
<td>The number of content accessed (hit) on the service. This field is incremented for traffic from a group server going out from the source group. Traffic coming into the group does not increment the counter.</td>
</tr>
<tr>
<td>State</td>
<td>The state of the service. The possible states are Alive, Dying, or Dead.</td>
</tr>
<tr>
<td>DNS Load</td>
<td>The DNS load for the service. A load of 255 indicates that the service is down. An eligible load range is from 2 to 254.</td>
</tr>
<tr>
<td>Trans</td>
<td>The number of times that the state of the service has transitioned.</td>
</tr>
<tr>
<td>Keepalive</td>
<td>The keepalive type of the service. The possible types are FTP, HTTP, ICMP, NAMED, SCRIPT, or TCP.</td>
</tr>
<tr>
<td>Conn</td>
<td>The number of connections currently on the service.</td>
</tr>
<tr>
<td>Flow Timeout Multiplier</td>
<td>Number of 16-second multiples that a flow remains idle before the CSS reclaims the flow resources, as configured with the <code>flow-timeout-multiplier</code> command. For details on the <code>flow-timeout-multiplier</code> command, refer to the Chapter 2, Configuring Flow and Port Mapping Parameters.</td>
</tr>
<tr>
<td>Group Service Total Counters</td>
<td>The counters for the group.</td>
</tr>
</tbody>
</table>
### Table 5-4  Field Descriptions for the show group Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hits/Frames/Bytes</td>
<td>The number of group hits, frames, and bytes. This field is incremented for traffic from a group server going out from the source group. Traffic coming into the group does not increment the counter.</td>
</tr>
<tr>
<td>Connection Total/Current</td>
<td>The total number of connections and the current number of connections for the group.</td>
</tr>
<tr>
<td>FTP Control Total/Current</td>
<td>The total number of FTP control channels that were mapped and monitored by the CSS, and the current number of those connections that are mapped.</td>
</tr>
<tr>
<td>SP Port Map Info</td>
<td>The port map information for each SP in the CSS. Includes the status of the <code>portmap</code> command (Enabled or Disabled).</td>
</tr>
<tr>
<td>Configured Base Port</td>
<td>The configured starting port number.</td>
</tr>
<tr>
<td>Configured Ports per VIP</td>
<td>The total number of ports on each VIP address in the CSS. If the number is not a multiple of 32, the CSS rounds the number up to the next multiple of 32.</td>
</tr>
<tr>
<td>Slot</td>
<td>The slot in the CSS chassis where the module resides.</td>
</tr>
<tr>
<td>Subslot</td>
<td>The subslot in the module where the SP resides.</td>
</tr>
<tr>
<td>Ports Avail to this SP</td>
<td>The total number of source ports available to the SP.</td>
</tr>
<tr>
<td>VIP Address</td>
<td>The configured VIP address of the port mapper. For the <code>show group portmap</code> command, the CSS displays “all” if there are multiple configured VIPs. For the all command option or for a specified VIP address, the fields in the <code>show group portmap</code> screen contain information specific to individual port mappers.</td>
</tr>
<tr>
<td>Current Mapped Ports</td>
<td>The total number of ports currently in use for flows.</td>
</tr>
<tr>
<td>Last Mapped Port</td>
<td>The port number that the CSS used for the most recent NAted flow. Use this field with the Last Mapped VIP field to obtain the latest NAT information.</td>
</tr>
</tbody>
</table>
Table 5-4 Field Descriptions for the show group Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Water Mark</td>
<td>The highest number of ports that this source group has had concurrently mapped since the last group was activated. This counter may not be equal to the sum of all individual port mapper high water marks because the high water marks for each port mapper may occur at different times.</td>
</tr>
<tr>
<td>Current Ctrl Channels</td>
<td>The total number of FTP control channels that the CSS is currently NATing.</td>
</tr>
<tr>
<td>No Portmap Errors</td>
<td>The number of times no port could be allocated by the port mapper.</td>
</tr>
<tr>
<td>Last Mapped VIP</td>
<td>The VIP address that the CSS used in the most recently NATed flow. This is the same as the VIP Address field for the all command option or a specified VIP address option. Use this field with the Last Mapped Port field to obtain the latest NAT information.</td>
</tr>
</tbody>
</table>

Clearing Source Group Counters

To reset the statistics displayed by the show group command to zero, use the zero all command.

For example, enter:

(config-group[ftpgroup])# zero all
Configuring Loads for Services

A service becomes ineligible to receive flows when its load number exceeds the configured load threshold. This chapter contains the following sections on how to configure relative and absolute load for services:

- Configuring Relative Load for Services
- Configuring the Absolute Load Calculation Method

Information in this chapter applies to all CSS models except where noted.

Configuring Relative Load for Services

The following sections describe how to configure relative load for services:

- Relative Load Overview
- Configuring Relative Load
- Showing Global Service Loads
Relative Load Overview

Relative load is a mechanism that the CSS uses to express the current load experienced by a service. The CSS calculates relative load by using the variances in normalized response times from client to service to determine a service’s load number. A service with a heavier processing load would be biased toward a more significant, larger load number. For details on configuring absolute load, see the “Configuring the Absolute Load Calculation Method” section.

To configure global load parameters for the eligibility and ineligibility of CSS services, use the load report, load teardown timer, and load ageout timer commands (discussed later in this section).

Note
Use relative load in a GSLB environment when the configurations and traffic patterns of all CSSs in the peer mesh are very similar.

You can adjust relative load calculations by changing the load step size, which is the difference, in milliseconds, between load numbers. The CSS can determine the load step dynamically, or you can configure the initial load step using the load step command.

The load on a service has a range of 2 to 255, with an eligible load of 2 to 254. An eligible service is an active service that can receive flows. A service with a load of 255 is offline.

A service becomes ineligible to receive flows when its load number exceeds the configured load threshold. The CSS uses the configured ageout timer value to return the service to the eligible state.

For the CSS to consider the service loads as different, response times of the services must differ by the configured load step or greater. If the response times differ by less than the configured load step, the CSS considers the services to have the same load.

Note
Redirect services have load numbers associated with them, but the load numbers are either 2 (available) or 255 (unavailable).
Figure 6-1 shows servers A, B, and C with response times of 100 ms, 1100 ms, and 120 ms, respectively. One group of servers has load step configured to 10 ms. The second group of servers has load step configured to 100 ms.

**Figure 6-1 Load Calculation Example with Three Servers**

<table>
<thead>
<tr>
<th>Server Name</th>
<th>Normalized Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverA</td>
<td>100 ms</td>
</tr>
<tr>
<td>serverB</td>
<td>1100 ms</td>
</tr>
<tr>
<td>serverC</td>
<td>120 ms</td>
</tr>
</tbody>
</table>

For the servers set to the 10 ms load step, the difference in response time between:

- ServerA and serverB is 1000 ms. Because this value is greater than the configured load step of 10 ms, the CSS considers the server loads to be different.
- ServerA and serverC is 20 ms. Because this value is greater than the configured load step of 10 ms, the CSS considers the server loads to be different.
For the servers set to 100 ms load step, the difference in response time between:

- ServerA and serverB is 1000 ms. Because this value is greater than the configured load step of 100 ms, the CSS considers the server loads to be different.
- ServerA and serverC is 20 ms. Because this value is less than the configured load step of 100 ms, the CSS considers servers A and C to be the same load.

Increasing the load step causes the load for servers to be closer to each other. Decreasing the load step causes the load for servers to be further from each other.

To enable you to configure an accurate load threshold for a server, you can calculate a load number for a server. To calculate a server load number:

1. Take the difference between the server with the lowest response time and the server for which you want to determine a load number.
2. Divide the difference by the configured load step.
3. Add this number to the calculated load step of the server with the lowest response time, which is always 2.

For example, to calculate the load number for serverC with the 10 ms load step:

1. Take the difference in server response time between serverA and serverC (20 ms).
2. Divide it by the configured load step (10 ms). The result equals 2.
3. Add 2 to serverA’s (server with lowest response time) calculated load of 2 to determine serverC’s calculated load of 4.

**Configuring Relative Load**

The following sections describe how to configure load:

- Relative Load Configuration Quick Start
- Configuring Global Load Reporting
- Configuring the Relative Load Step
- Configuring the Global Load Threshold
- Configuring the Load Teardown Timer
- Configuring the Load Ageout Timer
Relative Load Configuration Quick Start

Table 6-1 provides a quick overview of the basic steps required to configure relative load for services. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following Table 6-1.

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter config mode by typing <code>config</code>.</td>
</tr>
</tbody>
</table>
| # config
  (config)# |
| 2. Enable the CSS to generate teardown reports and to derive load numbers. |
| (config)# load reporting |
| 3. Set the relative load step, which is the difference, in milliseconds, between load numbers. |
| (config)# load step 100 dynamic |
| 4. Define the global load number. The CSS uses this number to determine whether a service is eligible to receive flows. |
| (config)# load threshold 25 |
| 5. Set the maximum time interval, in seconds, between teardown reports |
| (config)# load teardown-timer 120 |
| 6. Set the time interval, in seconds, in which the CSS times out stale load information for a service. |
| (config)# load ageout-timer 180 |
| 7. (Recommended) Use the `show load` command to verify your configuration. |
| (config)# show load |
The following running-configuration example shows the results of entering the commands in Table 6-1.

```
!***************************************************************************
load teardown-timer 120
load ageout-timer 180
load step 100 dynamic
load threshold 25

Configuring Global Load Reporting

A teardown report is a summary of response times for services when flows are being torn down. The CSS uses the teardown report to derive the load number for a service. By default, load reporting is enabled on the CSS. This command applies to both relative load and absolute load. Use the `load reporting` command to enable load reporting; the CSS generates teardown reports and derives load numbers.

If you are not concerned about load reporting, disable it and it may increase performance (depending on flows and load reporting already occurring). To disable load reporting, enter:

```
(config)# no load reporting
```

To reenable load reporting, enter:

```
(config)# load reporting
```

Configuring the Relative Load Step

By default, the CSS starts at a load step of 10 ms and then dynamically calculates the load step as it accumulates minimum and maximum response times for the services. Use the `load step` command to set the relative load step, which is the difference, in milliseconds, between load numbers. Load numbers have a range from 2 to 254.

When you configure the load step to reduce the flows to a slower service, consider the differences in response times between services. For example:

- Increasing the load step causes the load for services to be closer to each other, thus increasing the number of flows to a slower service.
- Decreasing the load step causes the load for services to be further from each other, decreasing the flows to a slower service.
The options and syntax for this global configuration mode command are:

- **load step ms dynamic** (default) - Set the initial load step. The CSS uses the default of 10 ms as the initial load step, modifying it after the CSS collects sufficient response-time information.

- **load step ms static** - Set a constant load step. The CSS uses this load step value instead of making dynamic calculations.

Enter the load step, in milliseconds, from 10 to 1000000000. The default is 10 ms. For example, to set the load step to 100 ms, enter:

```plaintext
(config)# load step 100
```

To set the load step to the default of 10 ms, enter:

```plaintext
(config)# no load step
```

### Configuring the Global Load Threshold

The CSS uses the global load number to determine whether a service is eligible to receive flows. Use the **load threshold** command to define the global load number. If the service load exceeds the threshold, the service becomes ineligible to receive flows until the CSS ages the service into the eligible state. This command applies to both relative load and absolute load.

Enter the threshold as a number from 2 to 254. The default is 254, which is the maximum threshold services can reach before becoming unavailable. To view the global load on services, use the **show load** command (see Table 6-2 for details).

For example, to set the load threshold to 25, enter:

```plaintext
(config)# load threshold 25
```

If you do not configure a load threshold for the content rule with the **(config-owner-content) load-threshold** command, the rule inherits the global load threshold.

To set the load threshold to the default of 254, enter:

```plaintext
(config)# no load threshold
```
If you configure the absolute load calculation method on a CSS and a service exceeds its configured global load threshold, the CSS does not include the load of that service in any content rule load that the CSS advertises.

**Configuring the Load Teardown Timer**

A teardown report is a summary of response times for services when flows are being torn down. The CSS uses the teardown report to derive the load number for a service. This command applies to both relative load and absolute load.

When the CSS has sufficient teardown activity for a service, it generates a teardown report and the teardown timer is reset. If a teardown report is not triggered at the end of the teardown timer interval due to insufficient activity, the CSS generates a teardown report based on its current activity. If there is no activity, no report is generated and the timer resets.

Use the `load teardown-timer` command to set the maximum time between teardown reports. The teardown timer is the number of seconds between teardown reports. Enter an integer from 0 to 1000000000. The default is 20. The value of 0 disables the timer.

The teardown timer is overridden when a service is reset. After 10 teardown reports are recorded, the timer is reset to its configured value.

For example, to set the teardown timer to 120 seconds, enter:

```
(config)# load teardown-timer 120
```

To reset the teardown time interval to its default of 20 seconds, enter:

```
(config)# no load teardown-timer
```

**Configuring the Load Ageout Timer**

By default, the CSS times out stale load information for a service at time interval of 60 seconds. When the ageout timer interval expires, the CSS erases the information and resets the service load to 2. Load information is stale when the teardown report number recorded on a service has not incremented during the ageout time interval because no flows (long or short) are being torn down on the service. This command applies to both relative load and absolute load.
At the beginning of the time interval, the ageout timer saves the number of the current teardown report. When the CSS generates a new teardown report, the report number in the CSS increments and any services in the report will save this number. At the end of the ageout time interval, the CSS compares the initial teardown number, saved at the beginning of the time interval, with the current teardown number saved by each service. If the number of a service is less than or equal to the timer number, the load information is stale. The CSS erases it and the service load is reset to 2.

Use the `load ageout-timer` command to set the time interval, in seconds, in which the CSS times out stale load information for a service. Enter the ageout timer as the number of seconds to time out load information for a service. Enter an integer from 0 to 1000000000. The default is 60. A value of 0 disables the timer.

For example, enter:

```
(config)# load ageout-timer 180
```

To set the ageout time to the default of 60, enter:

```
(config)# no load ageout-timer
```

### Showing Global Service Loads

Use the `show load` command to display the global load configuration and service load information. For example, enter:

```
(config)# show load
```

Table 6-2 describes the fields in the `show load` command output.
### Table 6-2 Field Descriptions for the show load Command Output

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global load</td>
<td>The configured state of load reporting (enabled or disabled). Reporting is disabled by default.</td>
</tr>
<tr>
<td>information</td>
<td></td>
</tr>
<tr>
<td>Step Size</td>
<td>The configured method in which the load step size is calculated:</td>
</tr>
<tr>
<td></td>
<td>• Dynamic indicates that the CSS calculates the step size.</td>
</tr>
<tr>
<td></td>
<td>• Static indicates that the configured step size is used.</td>
</tr>
<tr>
<td>Configured</td>
<td>The configured load step. The value is the difference, in milliseconds, between load numbers. If the step size method is dynamic, this is the initial load step. The CSS modifies the value after it collects sufficient response time information from the services.</td>
</tr>
<tr>
<td>Actual</td>
<td>The actual load step. The value is the difference, in milliseconds, between load numbers. If the step size method is configured, the actual value will be the same as that in the Configured field.</td>
</tr>
<tr>
<td>Threshold</td>
<td>The configured global load number that the CSS uses to determine whether a service is eligible to receive flows. The range is from 2 to 254. The default is 254.</td>
</tr>
<tr>
<td>Ageout-Timer</td>
<td>The configured time interval, in seconds, in which stale load information for a service is timed out. When the ageout timer interval expires, the CSS erases the information and resets the service load to 2. The range is an integer from 0 to 1000000000. The default is 60. A value of 0 disables the timer.</td>
</tr>
<tr>
<td>Teardown-timer</td>
<td>The maximum time between teardown reports. The range is from 0 to 1000000000. The default is 20. A value of 0 disables the timer.</td>
</tr>
<tr>
<td>Configured</td>
<td>The configured maximum time between teardown reports. The range is from 0 to 1000000000. The default is 20. A value of 0 disables the timer.</td>
</tr>
</tbody>
</table>
Chapter 6  Configuring Loads for Services

Configuring the Absolute Load Calculation Method

Configure the absolute load calculation method on a CSS to enhance the way the CSS determines service load, either locally, or in a global server load balancing (GSLB) environment. This method is an alternative to the relative load-calculation algorithm and calculates the load on a service without normalizing load values against the fastest services on the CSS. Consider using absolute load instead of relative load when you have a single CSS serving multiple applications, or when you are using GSLB to balance between multiple CSSs.

The section contains the following subsections:

- Overview of Calculating Absolute Load
- Configuration Requirements and Restrictions
- Configuring Load Calculation
- Using the load absolute-sensitivity Command
- Configuring Load Variance
- Displaying Relative Load Statistics
- Displaying Absolute Load Calculation Ranges

Table 6-2  Field Descriptions for the show load Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>The actual time between teardown reports.</td>
</tr>
<tr>
<td>Service Name</td>
<td>The name of the service.</td>
</tr>
<tr>
<td>Average Load Number</td>
<td>The average load number for the service.</td>
</tr>
</tbody>
</table>

Configuring the Absolute Load Calculation Method

Configure the absolute load calculation method on a CSS to enhance the way the CSS determines service load, either locally, or in a global server load balancing (GSLB) environment. This method is an alternative to the relative load-calculation algorithm and calculates the load on a service without normalizing load values against the fastest services on the CSS. Consider using absolute load instead of relative load when you have a single CSS serving multiple applications, or when you are using GSLB to balance between multiple CSSs.

The section contains the following subsections:

- Overview of Calculating Absolute Load
- Configuration Requirements and Restrictions
- Configuring Load Calculation
- Using the load absolute-sensitivity Command
- Configuring Load Variance
- Displaying Relative Load Statistics
- Displaying Absolute Load Calculation Ranges
Overview of Calculating Absolute Load

Calculating absolute load numbers for services may allow the CSS to make more intelligent load-balancing decisions than using relative load numbers. Absolute load only takes into account the actual observed load on a service, whereas relative load compares services to the service with the fastest response time.

Absolute load also allows you to configure the response times that correlate with values within the CSS load number scale. Unlike the relative load number scale, where all the load numbers between 2 and 254 represent equal steps or increases in response times, absolute load creates 16 different divisions or ranges within the CSS load number scale. Ranges are groups of consecutive load numbers that share a common step size (delta) between numbers.

Note

Regardless of which load calculation method you choose, be sure that all CSSs in a GSLB environment have very similar configurations.

This feature provides a default set of 16 ranges with a configurable sensitivity option that you can use to modify the upper boundary of the load number scale while adjusting the step sizes (granularity) within the ranges. In general, the better the granularity between load numbers, the better load balancing a CSS performs. However, if the granularity is too fine, the slower servers will be excluded from the load number scale and load numbers will be meaningless for these load-balancing decisions. Keeping the ranges within the load number scale allows some fine granularity for faster servers and coarser granularity for slower servers, while accommodating both short-lived and long-lived flows.

A CSS calculates the average response time for a service based on the measured lifetime of flows to that service. The CSS filters the response values for deviation and damps them to avoid sudden changes. The average response time is then mapped to the absolute load ranges.

For example, suppose a site has two groups of services serving two different types of applications. Group A supports application A, which involves mainly short-lived, quick connections; Group B supports application B, which is much more server-intensive and takes longer to complete. Further, it should never take more than 200 ms for a service handling application A to respond, but it could take up to 200,000 ms for services handling application B to respond. Rather than grouping these services together and using a response time much too large for application A, absolute load allows the CSS to use ranges within the load number scale to better handle load monitoring and balancing for each application.
Configuration Requirements and Restrictions

Observe the following configuration requirements and restrictions when you configure your services.

- You must configure the **load reporting** command to enable the CSS to derive loads on services. See the “Configuring Global Load Reporting” section.

- If you are using absolute load calculations in a GSLB configuration, the values of load absolute sensitivity should be the same for all participating sites.

- If you decide to change an existing configuration to use absolute load instead of relative load, it is possible that the CSS load-balancing behavior will change. The CSS may report some service load numbers differently; any configured load thresholds may affect these load numbers.

- If you plan to combine absolute load calculation with the GSLB least-loaded algorithm, we recommend that you set the load variance to 0. This ensures that the CSS always uses load numbers to determine the least-loaded site.

Absolute Load Configuration Quick Start

Table 6-3 provides a quick overview of the basic steps required to configure absolute load. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following Table 6-3.

<table>
<thead>
<tr>
<th>Task and Command Example</th>
<th>Absolute Load Configuration Quick Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter config mode by typing <code>config</code>.</td>
<td>Table 6-3 provides a quick overview of the basic steps required to configure absolute load. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following Table 6-3.</td>
</tr>
<tr>
<td># <code>config</code></td>
<td></td>
</tr>
<tr>
<td><code>(config)#</code></td>
<td>2. Specify the absolute load method, which the CSS uses to assign load numbers to all configured services. See the “Configuring Load Calculation” section.</td>
</tr>
<tr>
<td><code>(config)# load calculation absolute</code></td>
<td></td>
</tr>
</tbody>
</table>
The following running-configuration example shows the results of entering the commands in Table 6-3.

```
!*************************** GLOBAL ***************************
dns-server zone load variance 0
load calculation absolute
```
Configuring Load Calculation

By default, the CSS uses the relative load calculation method to assign load numbers to all configured services. This method assigns load numbers to services based on a comparison with the fastest local service. Use the `load calculation` command to specify the calculation method that the CSS uses to assign load numbers to all configured services. The syntax for this global configuration mode command is:

```
load calculation relative|absolute
```

The options are:

- **relative** (default) - Specifies that the CSS assigns load numbers to services based on a comparison with the fastest local service. For details about relative load, see the “Relative Load Overview” section.

- **absolute** - Specifies that the CSS assigns load numbers to services based strictly on pure response times.

For example, to configure the absolute load calculation method, enter:

```
(config)# load calculation absolute
```

To return the load calculation method to the default of **relative**, enter:

```
(config)# no load calculation
```

**Note**

In a GSLB environment with the absolute load calculation method configured, if a service exceeds its maximum connections limit, exceeds the local load threshold, or has a configured weight of 0 (to gracefully shut down), a CSS does not consider the load for that service in the calculation of reported load average for one or more content rules. This behavior results in more accurate load average reporting for APP, kal-ap, and kal-ap-vip. For information about services, see Chapter 3, Configuring Services. For details about APP, kal-ap, and kal-ap-vip, refer to the *Cisco Content Services Switch Global Server Load-Balancing Configuration Guide*. 
Using the `load absolute-sensitivity` Command

By default, the absolute load calculation method uses an internal load number scale designed to support a wide range of configurations and applications. However, you can adjust the absolute load number scale to suit your configuration.

Configuring load `absolute-sensitivity`

Increasing the CSS `load absolute-sensitivity` value increases the upper boundary of the maximum response time and the step size (granularity) of the absolute load number scale, thereby reducing the load value for a given service response time. Conversely, decreasing the `load absolute-sensitivity` value decreases upper boundary of the maximum response time and the step size (granularity) of the absolute load number scale, thereby increasing the load value for a given service response time.

Use the `load absolute-sensitivity` command to modify the absolute load number scale. The syntax for this global configuration mode command is:

```
load absolute-sensitivity number
```

The `number` variable specifies the sensitivity of the absolute load number scale. Enter an integer from 1 to 25. The default is 21.

For example, to configure a load sensitivity of 18, enter:

```
(config)# load absolute-sensitivity 18
```

To return the `load absolute-sensitivity` to the default value of 21, enter:

```
(config)# no load absolute-sensitivity
```

For `number` values from 1 to 20, the absolute load number ranges are linear, which means that the step sizes are equal among all the ranges. For values from 21 to 25, the ranges are nonlinear, which means different ranges have different step sizes that increase as the range number increases. For details, see the “Displaying Absolute Load Calculation Ranges” section later in this chapter.
Optimizing the Absolute Load Number Scale

As an experienced user, you can optimize the absolute load number scale to more closely resemble the actual load numbers and maximum response times of your configured services. Before you attempt to modify the absolute load number scale, read this procedure in its entirety to familiarize yourself with the steps. To optimize the absolute load number scale:

1. Use the `show load` command to gather information about the load numbers and response times of your configured services. Capture and print out or write down the statistics from the `show load` command output. See the “Displaying Relative Load Statistics” section later in this chapter.

2. Use the data you gathered in Step 1 to determine if you have services whose peak average response times correspond approximately with the maximum response time associated with a load of 254, as displayed with the `show load absolute` command. See the “Displaying Absolute Load Calculation Ranges” section later in this chapter.

3. Expand the absolute response time range if you do have such services and the high load values are unexpected. Do this by gradually increasing the `load absolute-sensitivity` value in increments of one, thereby reducing the load number for those services. You may find it desirable to repeat this step until the target service load values reach the middle of the absolute load number scale.

4. Condense the absolute response time range if your peak average service response times tend to cluster around lower load number range. Do this by gradually decreasing the `load absolute-sensitivity` value in decrements of two, thereby increasing the load number for those services.

5. Monitor the results of each change you make to the `load absolute-sensitivity` value by observing the `show load absolute` command output. See the “Displaying Absolute Load Calculation Ranges” section later in this chapter.

6. Repeat Steps 3, 4, and 5 until you are satisfied with the load number and response time results for each configured service.

7. Be sure to allow sufficient load number differentiation among all your services for best load-balancing result. Check to ensure that all services are represented on the absolute load number scale and that services are not clustered around a particular load number range.
8. Test the new configuration by running traffic through the CSS and checking the load-balancing results with the `show rule owner_name content_rule_name services` and `show service` commands. If necessary, repeat this entire procedure.

### Configuring Load Variance

Load variance is a configured value that represents a range of load numbers among sites or zones that the CSS considers to be similar for the least-loaded algorithm in a DNS load-balancing decision. For example, if you configure a load variance of 50, and the load difference among three sites is 50 or less, the CSS calculates the minimum response time for each site, then selects the site with the fastest service, ignoring the similar load values.

**Note**

For GSLB, we recommend that you set the same load variance value on all CSSs in a peer mesh. If you configure the absolute load calculation method, we recommend that you configure a load variance of 0. See the “Configuring Load Calculation” section.

To set the deterministic difference in peer load numbers that a CSS considers to be similar for the least-loaded algorithm in a zone-based DNS load-balancing decision, use the `dns-server zone load variance` command. For the `number` variable, enter an integer from 0 to 254. The default is 50. Use the `no dns-server zone load variance` command to restore the load variance to the default of 50. For more information on the `dns-server zone` command, refer to the *Cisco Content Services Switch Global Server Load-Balancing Configuration Guide*.
Displaying Relative Load Statistics

Use the `show load` command to display the load calculation information for each service configured on your CSS.

Table 6-4 describes the service-specific fields in the `show load` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Name</td>
<td>Name of the configured service</td>
</tr>
<tr>
<td>Average Load Number</td>
<td>Accumulated average load number for the service identified in the Service Name field. Values range from 2 to 255 and indicate a position on the load number scale. A load of 255 indicates that the service is unavailable.</td>
</tr>
<tr>
<td>Average Response Time</td>
<td>Accumulated average response time, in milliseconds, for the service identified in the Service Name field. The displayed value indicates the response time measured from flow setup to flow teardown.</td>
</tr>
<tr>
<td>Peak Average Response Time</td>
<td>Highest Average Response Time, in milliseconds, reported for each configured service.</td>
</tr>
</tbody>
</table>

Use the Average Response Time and the Peak Average Response Time values when you configure services and their associated load and when monitoring configured services. These two fields appear in the `show load` command output regardless of the configured load calculation method.
After monitoring traffic, use the `show load` command to determine whether the `absolute-sensitivity` value needs to be modified for your configuration. Observe the peak response times of all the servers and determine the worst performing service. By comparing the worst server response time to the associated response time of the load number 254, you can determine whether the load number scale needs to be expanded.

**Note**
You can reset the current values of Average Response Time and Peak Average Response Time by toggling load reporting using the `no load reporting` and the `load reporting` commands. Be sure that `load reporting` is enabled when you are finished. The CSS requires that the `load reporting` command be enabled to calculate loads for services.

### Displaying Absolute Load Calculation Ranges

Use the `show load absolute` command to display absolute load number ranges. This command displays all load numbers and their associated maximum response times based upon the currently configured value for `load absolute-sensitivity` (see the “Configuring load absolute-sensitivity” section). The `show load absolute` command also displays the ranges and their calculated step sizes for load numbers within a range.
Table 6-5 displays the `show load absolute` command output based on the `load absolute-sensitivity` default value of 21.

### Table 6-5  Output for the `show load absolute` Command (`load absolute-sensitivity = 21`)

<table>
<thead>
<tr>
<th>Range Number</th>
<th>Load Numbers</th>
<th>Step Size (ms)</th>
<th>Maximum Response Time (ms)</th>
<th>Maximum Response Time (h:m:s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-15</td>
<td>2</td>
<td>32</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>2</td>
<td>16-31</td>
<td>4</td>
<td>96</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>3</td>
<td>32-47</td>
<td>8</td>
<td>224</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>4</td>
<td>48-63</td>
<td>16</td>
<td>480</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>5</td>
<td>64-79</td>
<td>32</td>
<td>992</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>6</td>
<td>80-95</td>
<td>64</td>
<td>2016</td>
<td>0: 0: 2</td>
</tr>
<tr>
<td>7</td>
<td>96-111</td>
<td>128</td>
<td>4064</td>
<td>0: 0: 4</td>
</tr>
<tr>
<td>8</td>
<td>112-127</td>
<td>256</td>
<td>8160</td>
<td>0: 0: 8</td>
</tr>
<tr>
<td>9</td>
<td>128-143</td>
<td>512</td>
<td>16,352</td>
<td>0: 0:16</td>
</tr>
<tr>
<td>10</td>
<td>144-159</td>
<td>1024</td>
<td>32,736</td>
<td>0: 0:32</td>
</tr>
<tr>
<td>11</td>
<td>160-175</td>
<td>2048</td>
<td>65,504</td>
<td>0: 1: 5</td>
</tr>
<tr>
<td>12</td>
<td>176-191</td>
<td>4096</td>
<td>131,040</td>
<td>0: 2:11</td>
</tr>
<tr>
<td>13</td>
<td>192-207</td>
<td>8192</td>
<td>262,112</td>
<td>0: 4:22</td>
</tr>
<tr>
<td>14</td>
<td>208-223</td>
<td>16,384</td>
<td>524,256</td>
<td>0: 8:44</td>
</tr>
<tr>
<td>15</td>
<td>224-239</td>
<td>32,768</td>
<td>1,048,544</td>
<td>0:17:28</td>
</tr>
<tr>
<td>16</td>
<td>240-254</td>
<td>65,536</td>
<td>2,031,584</td>
<td>0:33:51</td>
</tr>
</tbody>
</table>
Table 6-6 describes the fields in the `show load absolute` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range Number</td>
<td>Numbers from 1 to 16 representing the Load Number ranges</td>
</tr>
<tr>
<td>Load Numbers</td>
<td>Numbers from 2 to 254 of the CSS load scale segmented into 16 ranges</td>
</tr>
<tr>
<td>StepSize</td>
<td>Difference between response times for load numbers within a range</td>
</tr>
<tr>
<td>Maximum Response</td>
<td>Maximum response time, measured from flow setup to flow teardown, permitted in a range</td>
</tr>
</tbody>
</table>

The load number scale starts at 2 and ends at 255, where the value of 255 means a service is unavailable. Within the load number scale, there are 16 equal-sized ranges. The response time boundaries of each range are based on deriving a step size and the number of steps within a range. The stepsizes differ among ranges, with stepsizes getting larger as load numbers increase. This scheme provides finer granularity to faster services where it is needed and provides coarser granularity to slower services.
Table 6-7 displays the `show load absolute` command output based on a `load absolute-sensitivity` value of 22. Notice that both the Step Size and the Maximum Response Time values have increased for each range.

<table>
<thead>
<tr>
<th>Range Number</th>
<th>Load Numbers</th>
<th>Step Size (ms)</th>
<th>Maximum Response Time (ms)</th>
<th>Maximum Response Time (h:m:s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-15</td>
<td>4</td>
<td>60</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>2</td>
<td>16-31</td>
<td>8</td>
<td>188</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>3</td>
<td>32-47</td>
<td>16</td>
<td>444</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>4</td>
<td>48-63</td>
<td>32</td>
<td>956</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>5</td>
<td>64-79</td>
<td>64</td>
<td>1980</td>
<td>0: 0: 1</td>
</tr>
<tr>
<td>6</td>
<td>80-95</td>
<td>128</td>
<td>4028</td>
<td>0: 0: 4</td>
</tr>
<tr>
<td>7</td>
<td>96-111</td>
<td>256</td>
<td>8124</td>
<td>0: 0: 8</td>
</tr>
<tr>
<td>8</td>
<td>112-127</td>
<td>512</td>
<td>16,316</td>
<td>0: 0:16</td>
</tr>
<tr>
<td>9</td>
<td>128-143</td>
<td>1024</td>
<td>32,700</td>
<td>0: 0:32</td>
</tr>
<tr>
<td>10</td>
<td>144-159</td>
<td>2048</td>
<td>65,468</td>
<td>0: 1:5</td>
</tr>
<tr>
<td>11</td>
<td>160-175</td>
<td>4096</td>
<td>131,004</td>
<td>0: 2:11</td>
</tr>
<tr>
<td>12</td>
<td>176-191</td>
<td>8192</td>
<td>262,076</td>
<td>0: 4:22</td>
</tr>
<tr>
<td>13</td>
<td>192-207</td>
<td>16,384</td>
<td>524,220</td>
<td>0: 8:44</td>
</tr>
<tr>
<td>14</td>
<td>208-223</td>
<td>32,768</td>
<td>1,048,508</td>
<td>0: 17:28</td>
</tr>
<tr>
<td>15</td>
<td>224-239</td>
<td>65,536</td>
<td>2,097,084</td>
<td>0: 34:57</td>
</tr>
<tr>
<td>16</td>
<td>240-254</td>
<td>131,072</td>
<td>4,063,164</td>
<td>1: 7:43</td>
</tr>
</tbody>
</table>
Table 6-8 displays the `show load absolute` command output based on a load `absolute-sensitivity` value of 1. This value represents the smallest (finest) granularity allowed between service response times and the load numbers that represent them. Notice that the step size remains constant (linear) for all ranges.

<table>
<thead>
<tr>
<th>Range Number</th>
<th>Load Numbers</th>
<th>Step Size (ms)</th>
<th>Maximum Response Time (ms)</th>
<th>Maximum Response Time (h:m:s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-15</td>
<td>1</td>
<td>16</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>2</td>
<td>16-31</td>
<td>1</td>
<td>32</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>3</td>
<td>32-47</td>
<td>1</td>
<td>48</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>4</td>
<td>48-63</td>
<td>1</td>
<td>64</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>5</td>
<td>64-79</td>
<td>1</td>
<td>80</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>6</td>
<td>80-95</td>
<td>1</td>
<td>96</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>7</td>
<td>96-111</td>
<td>1</td>
<td>112</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>8</td>
<td>112-127</td>
<td>1</td>
<td>128</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>9</td>
<td>128-143</td>
<td>1</td>
<td>144</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>10</td>
<td>144-159</td>
<td>1</td>
<td>160</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>11</td>
<td>160-175</td>
<td>1</td>
<td>176</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>12</td>
<td>176-191</td>
<td>1</td>
<td>192</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>13</td>
<td>192-207</td>
<td>1</td>
<td>208</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>14</td>
<td>208-223</td>
<td>1</td>
<td>224</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>15</td>
<td>224-239</td>
<td>1</td>
<td>240</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>16</td>
<td>240-254</td>
<td>1</td>
<td>255</td>
<td>0: 0: 0</td>
</tr>
</tbody>
</table>
Table 6-9 displays the `show load absolute` command output based on a `load absolute-sensitivity` value of 2. The step size remains constant for all ranges, but its value has increased. The maximum response time associated with each range has also increased.

**Table 6-9  Output for the show load absolute Command (load absolute-sensitivity = 2)**

<table>
<thead>
<tr>
<th>Range Number</th>
<th>Load Numbers</th>
<th>Step Size (ms)</th>
<th>Maximum Response Time (ms)</th>
<th>Maximum Response Time (h:m:s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-15</td>
<td>2</td>
<td>30</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>2</td>
<td>16-31</td>
<td>2</td>
<td>62</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>3</td>
<td>32-47</td>
<td>2</td>
<td>94</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>4</td>
<td>48-63</td>
<td>2</td>
<td>126</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>5</td>
<td>64-79</td>
<td>2</td>
<td>158</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>6</td>
<td>80-95</td>
<td>2</td>
<td>190</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>7</td>
<td>96-111</td>
<td>2</td>
<td>222</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>8</td>
<td>112-127</td>
<td>2</td>
<td>254</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>9</td>
<td>128-143</td>
<td>2</td>
<td>286</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>10</td>
<td>144-159</td>
<td>2</td>
<td>318</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>11</td>
<td>160-175</td>
<td>2</td>
<td>350</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>12</td>
<td>176-191</td>
<td>2</td>
<td>382</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>13</td>
<td>192-207</td>
<td>2</td>
<td>414</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>14</td>
<td>208-223</td>
<td>2</td>
<td>446</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>15</td>
<td>224-239</td>
<td>2</td>
<td>478</td>
<td>0: 0: 0</td>
</tr>
<tr>
<td>16</td>
<td>240-254</td>
<td>2</td>
<td>508</td>
<td>0: 0: 0</td>
</tr>
</tbody>
</table>
Using ArrowPoint Content Awareness Based on Server Load and Weight

The ArrowPoint Content Awareness (ACA) load-balancing algorithm balances traffic between a group of servers. You can configure the CSS to make ACA load-balancing decisions based on:

- Server load
- Server weight and load

Using ACA Based on Server Load

ACA determines the best service for each content request based on server load and size of the content being requested. ACA estimates the file size based on previous requests for the same content. A service with a lower load receives more flows than a service with a higher load.

Using ACA Based on Server Weight and Load

Server weight is a mechanism to express the processing capabilities of a server. Weights allow you to configure the CSS to prefer one group of servers over another. When you configure weights, the number of hits per server is relative to the weight configured on that server. A higher weight will bias flows toward the specified server. For example, in Figure 6-1, ServerA with a weight of two is hit twice as often as ServerB which has a weight of one. ServerC has a weight of 10 and is hit 10 times as often as ServerB. All servers with the same weight are hit equally in a roundrobin manner.

The CSS can use a server’s weight in tandem with server load to determine server availability. When you configure ACA on a content rule to use both weight and load, the CSS calculates the number of requests per weight level based on the number of servers with that weight. The CSS then balances the requests among the servers based on their individual loads. The number of requests per weight level is equal to weight level times the number of servers times 10. The CSS then increments the weight level and uses the same mechanism to balance requests among the servers in the next weight level.
For information on configuring weight for a service, see the “Configuring Weight and Graceful Shutdown” section in Chapter 3, Configuring Services. Also see the “Specifying a Service Weight” section in Chapter 9, Configuring Content Rules.

### Configuring the Load Command for Use with ACA

To configure a load on a service and bypass the CSS load calculation method (relative or absolute), use the `load` command in service configuration mode. Use this command with the ACA load-balancing method when you want to take into account server load parameters, for example:

- CPU utilization
- Free memory
- Application threads
- Other server tasks

You can set the `load` command value with your application or server using SNMP or the CSS XML interface. For information about ACA, see the “Using ArrowPoint Content Awareness Based on Server Load and Weight” section. For information about SNMP and the XML interface, refer to the *Cisco Content Services Switch Administration Guide*.

**Caution**

Before you can use the `load` command on a service, you must disable load reporting by entering the `no load reporting` command in global configuration mode. Do not reenable load reporting. If you do, the load value you entered with the `load` command will no longer apply to the service. To recover, you must disable load reporting again and reenter the `load` command on the service at the CLI.

The load command has the following syntax:

```
load number
```

The `number` variable is the load value that you assign to a service. A service with a higher load number receives fewer hits than a service with a lower load number. The CSS considers a service with a load of 254 as unavailable, and, therefore, the service receives no hits. Enter an integer from 2 to 254. The default is 2.
For example, to configure a load of 50, enter:

```
(config-service[server1])# load 50
```

Use the `no` form of the command to reset the load value to the default of 2. For example, enter:

```
(config-service[server1])# no load
```

To display the configured value for the `load` command, use the `show load` command. For details about the `show load` command, see the “Showing Global Service Loads” section.
The Dynamic Feedback Protocol (DFP) is a mechanism that allows load-balanced servers (both local and remote) to dynamically report changes in their status and their ability to provide services to a CSS. A status report sent to a CSS from a server contains a relative weight/number of connections to define the load and availability of each server. A CSS incorporates server feedback into the load-balancing decision process in order to:

- Obtain server availability information
- Identify load imbalances over multiple sites
- Distribute traffic more evenly

This chapter contains the following major sections:

- DFP Overview
- Configuring a DFP Agent
- Maintaining a Consistent Weight Range Among Services
- Displaying Configured DFP Agents
- Displaying Services Supported by Configured DFP Agents
- Displaying DFP Information

Information in this chapter applies to all CSS models except where noted.
DFP Overview

The DFP manager (running on the CSS as a task and part of the load manager) is responsible for establishing TCP connections with the DFP agents that reside on each server. A DFP manager can communicate simultaneously with a maximum of 127 DFP agents. DFP agents can be software running on the actual server itself or may be separate hardware devices that collect and consolidate information from one or more servers for load-balancing purposes. DFP agents are available from a number of third-party sources.

DFP agents collect relative weights from the load-balanced servers and periodically send new or adjusted weights to the DFP manager in the form of load vectors. The CSS load manager distributes the incoming connections or services (local or remote) to the servers in the order of weight assigned to the load-balanced servers. The load manager uses the reported weights to choose the best available server, resulting in optimal performance of servers and less response time.

*Note*

If you configure a weight on a service using the `add service weight` command in owner-content configuration mode, the configured weight takes precedence over the service weight reported by the DFP agent for that content rule. In turn, the DFP-reported weight takes precedence over the weight configured on a service in service configuration mode.

The CSS uses load-balancing algorithms such as roundrobin, weighted roundrobin, Arrowpoint Content Aware (ACA), and least connections to distribute the incoming connections or service requests. Weighted roundrobin can take advantage of the server weights reported by the DFP agents.

The weighted roundrobin load-balancing method uses weight to specify how many consecutive connections to give to the highest-weighted server before moving on to the next highest-weighted server. As a server’s load changes, the DFP agent recalculates the weight for each server and reports the updated weights to the DFP manager, thereby influencing how the load manager distributes the service requests. For more information on CSS server load-balancing, refer to Chapter 9, Configuring Content Rules.
Chapter 7 Configuring Dynamic Feedback Protocol for Server Load Balancing

DFP Overview

The following sections provides information on:

- Functions of a DFP Agent
- Types of DFP Messages
- DFP System Flow

Functions of a DFP Agent

A DFP agent reports server weight/connection information to the DFP manager. Multiple DFP agents can exist on a server platform. An agent provides several benefits to the load-balancing process. A DFP agent can inform the CSS that the server:

- Is congested
- Is under-utilized
- Should not be used for load balancing for a period of time

Types of DFP Messages

The following messages are defined for communication between the DFP agent and the DFP manager in the CSS:

- The preference information message reports the status or weight of an IP server and is sent from the DFP agent to the DFP manager.
- The server state message, sent from the DFP manager to the agent, informs the agent that the load manager has decided to take the server in or out of service.
- The DFP parameters send configuration information from the DFP manager to the agent. Currently, the only configuration parameter passed is the keepalive interval.

DFP messages consist of a DFP header called a signal header followed by message vectors. Vectors are optional commands that exist in the defined messages. Each message vector contains a vector header, which is the first part of each vector in the DFP message, followed by data specific to the defined vector. The vector header allows the DFP manager or the DFP agent to discard any vectors or commands that it does not understand.
Defined vectors for DFP include:

- **Security Vector** - Allows each DFP message to be verified.
- **Load Vector** - Contains the actual load information being reported for the real servers and represents the servers' preferred capability.
- **Keepalive Vector** - Part of the DFP connection configuration. The keepalive vector allows the load manager to inform the DFP agent of the minimum time interval by which the agent must send information over the DFP connection to the CSS.

If a CSS receives a message that contains a vector type that it does not understand, the CSS discards the unknown vector.

### DFP System Flow

When you configure a DFP agent on a CSS, the DFP manager initiates a single TCP connection with the DFP agent (regardless of the number of servers the agent supports) with the parameters specified in the DFP agent configuration. The DFP manager sends a keepalive vector in a DFP message to change the default keepalive time if required.

After the connection is established, the DFP agent periodically sends update information in the form of a load-vector. If an agent has no information to send, it still must send an empty DFP packet to prevent the connection from being torn down.

If a DFP agent is responsible for collecting information from multiple servers, the servers are grouped by their port number and protocol type, and a separate load vector is required for each grouping. A DFP agent can report weights for a maximum of 128 servers in a single weight report. Upon receiving information about an adjusted weight, the DFP manager updates the weights of the server reported in the list of load-balanced servers.

If DFP is disabled, a CSS uses the weight configured on a service in owner-content configuration mode using the `add service weight` command (for that content rule only) or the weight configured on the service in service configuration mode, in that order. If no weight is configured on the service, the CSS uses a default weight of 1 to load balance the service. If a connection between a DFP agent and the DFP manager closes because of a timeout, a CSS uses the default weight for load balancing until the DFP manager reestablishes the connection with the DFP agent and obtains a new weight report.
If the configured DFP agent supports MD5 security, you can specify a shared key text string in the DFP manager. MD5 encryption is a one-way hash function that provides strong encryption protection. The CSS provides an MD5 secure connection between the DFP manager and the DFP agent on the server. In this secure environment, the CSS discards DFP messages from the server unless the messages contain the MD5 code.

Figure 7-1 summarizes the relationship between the DFP manager (in the CSS) and a DFP agent.

Configuring a DFP Agent

To configure a DFP agent listening for DFP connections on a particular IP address and TCP port combination on a server and to enable the DFP manager on the CSS, use the `dfp` command. You can configure a maximum of 127 DFP agents for the DFP manager in the CSS. Use the `no dfp` command to disable the DFP agent connection to a particular IP address.

The syntax for the `dfp` command is:

```
dfp ip_or_host {port} {key "secret"|[des-encrypted encrypted_key]"encrypt_key"}] {timeout seconds} {retry count} {delay time} {max-agent-wt weight}
```

Figure 7-1  Example of DFP Manager to DFP Agents System Flow
Chapter 7 Configuring Dynamic Feedback Protocol for Server Load Balancing

Configuring a DFP Agent

The variables and options are:

- **ip_or_host** - The IP address or host name of the configured DFP agent. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1) or a mnemonic host name (for example, myhost.mydomain.com).

- **port** - (Optional) The server TCP port that the configured DFP agent uses to listen for connections from the CSS DFP manager. Valid entries are 0 to 65535. The default is 14001.

  **Note**  
  Do not configure a service TCP keepalive to connect to the same port that the DFP agent uses to listen for connections from the DFP manager. This type of configuration causes the built-in DFP keepalive to fail.

- **key “secret”** - (Optional) An MD5 security key used for encryption to provide a secure data exchange between the CSS DFP manager and the DFP agents. MD5 encryption is a one-way hash function that provides strong encryption protection. Enter the secret as a case-sensitive quoted text string (maximum of 64 characters). It can include any printable ASCII character except tabs.

  For DFP to function properly, ensure that you configure the same key on each DFP agent that you configured on the DFP manager. If the key on an agent does not match the key on the DFP manager, no connection will be established and the DFP agent will not be able to send a weight report to the CSS. In this case, when the DFP manager fails to establish a connection with an agent for a given key, the CSS logs the following informational message in SYSLOG: Secret key might not be same as DFP agent’s key. Check secret key.

- **des-encrypted** - (Optional) Specifies that a Data Encryption Standard (DES) key follows.

- **encrypted_key** - The DES key that the CSS previously encrypted. The CSS does not reencrypt this key. The CSS saves the key in the running-config the same as you entered it. Enter an unquoted case-sensitive text string with no spaces and a maximum of 128 characters.

- **“encrypt_key”** - The DES encryption key that you want the CSS to encrypt. The CSS saves the encrypted key in the running-config as you entered it. Enter a quoted case-sensitive text string with no spaces and a maximum of 64 characters.
- **timeout seconds** - (Optional) The maximum inactivity time period (the keepalive time) for the connection between the CSS DFP manager and the server DFP agent. If the inactivity time period exceeds the timeout value, the DFP manager closes the connection. The DFP manager attempts to reopen the connection as often as specified by the value of the **retry** option. The range is from 1 to 10000 seconds. The default is 3600 seconds (1 hour).

- **retry count** - (Optional) The number of times the CSS DFP manager tries to reopen a connection with the server DFP agent. The range is from 0 (for continuous retries) to 65535. The default is 3 retry attempts.

- **delay time** - (Optional) The delay time, in seconds, between each attempt to reestablish a connection. Valid entries are 1 (immediately) to 65535 seconds (18 hours). The default value is 5 seconds.

- **max-agent-wt value** - (Optional) Maximum value of the weight reported by a DFP agent. A CSS uses this option to scale the reported weight when the weight range of a DFP agent does not match the weight range of the DFP manager. For example, the DFP manager weight range is 0 to 255. If a DFP agent reports weight in the range 0 to 16, the CSS scales up the agent-reported weight to match the weight range of the DFP manager. If an agent reports weight in the range 0 to 65535, the CSS scales down the agent-reported weight to match the weight range of the DFP manager.

If a DFP agent reports a weight greater than the maximum configured weight, then the CSS rejects the weight report and does not use the weight in load-balancing decisions. In this case, the CSS also logs an error in SYSLOG. Enter an integer from 1 to 65535. The default is 255.

For example, the following command configures the DFP manager to communicate with the DFP agent at the specified address running with the following options and variables:

- DFP agent IP address - 192.168.1.2
- Port - 14001 (default)
- MD5 security key - “hello”
- Connection timeout - 6000 seconds
- Number of connection retries - 3
- Delay between connection retries - 60 seconds

```
(config)# dfp 192.168.1.2 14001 key "hello" timeout 6000 retry 3
delay 60
```
To disable the DFP agent, enter:

```
(config)# no dfp 192.168.1.2
```

### Maintaining a Consistent Weight Range Among Services

The CSS has a weight range of 1 through 10; the DFP manager has a weight range of 0 through 255. Because of this difference in weight ranges, you may need to manually adjust the weights configured on the DFP agent for different services to maintain the same service weight range that exists outside of the DFP.

For example, suppose that you configure on the same content rule three services (serv1, serv2, and serv3) with weights of 1, 2, and 5, respectively. If the DFP agent reports a weight of 20 for serv1, serv1 will now receive 20 connections for every 2 connections on serv2 and 5 connections on serv3. This configuration places a disproportionate load on serv1, especially if serv2 and serv3 represent fast servers with plenty of unused resources.

To solve this problem and to maintain the same weight range for all three services, you can do either of the following:

- Force the DFP agent to report a weight in the range of 1 to 10 for serv1
- Have the DFP agent report weights for all three services to maintain the same weight range
Displaying Configured DFP Agents

For reporting purposes, you can view the configured DFP agents on a CSS using the `show dfp` command. This command displays a list of all DFP agents or the DFP agents at the specified IP address or host name arranged by their IP addresses, the port number on which the agent is connected to the DFP manager, the current state of the DFP agent, the keepalive time for the DFP TCP connection, and the DES-encrypted key of the agent, if any.

The syntax for this command is:

```
show dfp ip_or_host
```

The `ip_or_host` variable allows you to specify the DFP agent or agents running at a particular IP address or host name.

For example, to display configuration information for all DFP agents, enter:

```
# show dfp
```

Table 7-1 describes the fields in the `show dfp` command output.

**Table 7-1  Field Descriptions for the show dfp Command Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address</td>
<td>The IP address of the configured DFP agent.</td>
</tr>
<tr>
<td>Port</td>
<td>The port number of the configured DFP agent. The default is 14001.</td>
</tr>
<tr>
<td>State</td>
<td>The state of the DFP agent. Possible states are Active, Dead, or Connecting.</td>
</tr>
<tr>
<td>KAL</td>
<td>The configured maximum inactivity time, in seconds, for the TCP connection between the DFP manager and the DFP agent. When this time elapses, the CSS tears down the connection.</td>
</tr>
<tr>
<td>MD5 Key</td>
<td>The DES-encrypted key of the DFP agent, if configured.</td>
</tr>
</tbody>
</table>
Displaying Services Supported by Configured DFP Agents

To view the individual weights of load-balanced services reported by a configured DFP agent, use the `show dfp-reports` command. This command groups the weights by the port number of reported services, the type of protocol, and the IP address of servers.

The syntax for this command is:

```
show dfp-reports {ip_or_host {port number {protocol text {ip ip_or_host}}}}
```

The options and variables for this command are:

- `ip_or_host` - The IP address or host name of the configured DFP agent. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1) or a mnemonic host name (for example, myhost.mydomain.com).

- `port number` - (Optional) The port number for the load-balanced server or service. Valid entries are 0 to 65535. The default is 14001.

- `protocol text` - (Optional) The type of protocol for the load-balanced server or service. Possible values are TCP, UDP, HTTP, or FTP.

- `ip ip_or_host` - (Optional) The IP address or host name of the load-balanced server or service. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1) or a mnemonic host name (for example, myhost.mydomain.com).

The following example shows the weight report by a DFP agent configured at 192.168.1.2, for server 192.168.1.3. Weights are first grouped by port number of reported servers, and then by protocol.

```
# show dfp-reports 192.168.1.2 port 80 protocol tcp ip 192.168.1.3
```
Table 7-2 describes the fields in the `show dfp-reports` command output.

### Table 7-2  Field Descriptions for the show dfp-reports Command Output

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>The name of the configured service for which the DFP agent is reporting</td>
</tr>
<tr>
<td>Weight</td>
<td>The last weight reported by the DFP agent for the service</td>
</tr>
<tr>
<td>Time-Stamp</td>
<td>The month, day, and time of the last-received report</td>
</tr>
<tr>
<td># of Reports</td>
<td>The total number of reports</td>
</tr>
</tbody>
</table>

### Displaying DFP Information

To display DFP information, see the following sections:

- Using the `show service` Command
- Using the `show rule services` Command

### Using the show service Command

Use the `show service` command to display service-specific information. The `show service` command output includes a DFP field that indicates the state of DFP. Possible states are Enable or Disable.

The state is Enable when DFP is configured and there is no weight configured on the service in owner-content configuration mode. The state is Disable if DFP is not enabled or if DFP is enabled and you have configured a service weight in owner-content configuration mode using the `add service weight` command.

For details on the `show service` command, see the “Showing Service Configurations” section in Chapter 3, Configuring Services.
Using the show rule services Command

Use the show rule services command in owner-content mode to display weights configured for services in service mode, owner-content mode, and DFP, as well as other service-related information. The output of the command includes the weight assigned to each service preceded by a code letter. The code letters have the following meanings:

- D, the weight reported by a DFP agent
- R, the weight configured for a service using the add service weight command in owner-content mode
- S, the weight configured for a service using the weight command in service mode

For details on the show rule services command, see Chapter 9, Configuring Content Rules.
Configuring Owners

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

This chapter contains the following major sections:

- Owner Configuration Quick Start
- Creating an Owner
- Configuring an Owner DNS Balance Type
- Specifying Owner Address
- Specifying Owner Billing Information
- Specifying Case
- Specifying Owner DNS Type
- Specifying Owner E-Mail Address
- Removing an Owner
- Showing Owner Information

For information on how service, owners and content rules work together, see Chapter 1, Content Load-Balancing Overview.
Owner Configuration Quick Start

Table 8-1 provides a quick overview of the steps required to configure owners. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI command, see the sections following Table 8-1.

Table 8-1  Owner Configuration Quick Start

<table>
<thead>
<tr>
<th>Task and Command Example</th>
<th>Command Example</th>
</tr>
</thead>
</table>
| 1. Enter config mode by typing config. | `# config
(config)#` |
| 2. Create an owner. | `(config)# owner arrowpoint
(config-owner[arrowpoint])#` |
| 3. Specify the owner e-mail address. | `(config-owner[arrowpoint])# email-address bobo@arrowpoint.com` |
| 4. Specify the owner mailing address. | `(config-owner[arrowpoint])# address "373 grand ave usa"` |
| 5. Specify the owner billing information. | `(config-owner[arrowpoint])# billing-info "finance"` |
| 6. (Recommended) Display owner information to verify your owner configuration. | `(config-owner[arrowpoint])# show owner` |

The following running-configuration example shows the results of entering the commands in Table 8-1.

```
!*****************************************************************************
** OWNER **
*****************************************************************************
owner arrowpoint
   email-address bobo@arrowpoint.com
   address "373 grand ave usa"
   billing-info "finance"
```
Creating an Owner

An owner is generally the person or company who contracts the web hosting service to host their web content and allocate bandwidth as required. Use the `owner` command to create an owner for a content rule. When you create an owner, you enable the CSS to identify the entity (for example, person, company name, or other meaningful title) that owns content rules. The CSS can contain many owners and maintain a configurable profile for each owner.

When creating an owner, you may want to use the owner’s DNS (Domain Name Service) name. Enter the owner name as an unquoted text string from 1 to 31 characters in length. The following example creates the owner arrowpoint:

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

Once you create an owner, the CLI enters owner mode.

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

To remove an owner, use the no `owner` command. When you remove an owner, you also remove all content rules created for the owner. For example, enter:

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

Configuring an Owner DNS Balance Type

By default, the content rule will use the DNS load-balancing method assigned to the owner. The DNS load-balancing method configured for the owner applies to all of the owner’s content rules. To set a different method to a specific content rule, use the `(config-owner-content) dnsbalance` command.

Use the `dnsbalance` command to determine where to resolve a request for a domain name to an IP address. The syntax and options for this owner mode command are:

- `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 faster response time is considered the least-loaded site.
Specifying Owner Address

To enter an address for an owner, use the `address` command in owner mode. Enter a quoted text string with a maximum of 128 characters.

For example, enter:

```
(config-owner[arrowpoint])# address "373 granite ave usa"
```

To delete an owner address, enter:

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

Specifying Owner Billing Information

To enter billing information for an owner, use the `billing-info` command in owner mode. Enter the billing information assigned to an owner as a quoted text string with a maximum length of 128 characters. For example, enter:

```
(config-owner[arrowpoint])# billing-info "finance"
```

To delete an owner billing address, enter:

```
(config-owner[arrowpoint])# no billing-info
```
Specifying Case

To define whether or not the CSS employs case-sensitivity when matching content requests to an owner’s content rule, use the `case` command. The default is case insensitive.

**Note**

You must reboot the CSS for the `case` command to take effect.

For example, a client requests content from arrowpoint/index.html. If owner arrowpoint is configured for:

- **case sensitive**, the request must match content index.html exactly
- **case insensitive**, the request can be any combination of uppercase and lowercase letters (for example, Index.html, INDEX.HTML)

To configure owner arrowpoint content rules to be case-sensitive, enter:

```
(config-owner[arrowpoint])# case sensitive
```

To return to the default, enter:

```
(config-owner[arrowpoint])# case insensitive
```

Specifying Owner DNS Type

To set the peer name exchange policy for a specific owner, use the `dns` command. The default is none, which does not set a peer name exchange policy. For information on configuring DNS, refer to the *Cisco Content Services Switch Global Server Load-Balancing Configuration Guide*.

The syntax and options for this owner mode command are:

- **dns accept** - Accept all content rules proposed by the CSS peer
- **dns push** - Push (send) all content rules onto the CSS peer
- **dns both** - Accept all content rules proposed by the CSS peer and push all rules onto the CSS peer

For example, enter:

```
(config-owner[arrowpoint])# dns push
```
Specifying Owner E-Mail Address

To enter an e-mail address for an owner, use the `email-address` command in owner mode. For example, enter:

```
(config-owner[arrowpoint])# email-address bobo@arrowpoint.com
```

To remove an owner e-mail address, enter:

```
(config-owner[arrowpoint])# no email-address
```

Removing an Owner

**Caution** Removing an owner also deletes the content rules associated with it.

To remove an owner, use the `no owner` command from config mode. To remove an owner, you must first exit from the owner mode. You cannot be in the owner mode that you wish to remove.

For example, to remove an owner, enter:

```
(config)# no owner arrowpoint
```
Showing Owner Information

The `show owner` command enables you to display owner information for an owner. An owner is an entity that owns Web content and is using the CSS to manage access to that content.

You can issue the following `show owner` commands from the specified command modes to display configuration information and statistics for an owner:

- `show owner {owner_name [statistics]}` - Display configuration information and statistics for an owner.

  This command is available in ACL, Circuit, Global, Group, Interface, Service, SuperUser, and User modes. The `show owner` command displays configuration information for all owners. The `show owner owner_name` command displays configuration information for a specified owner. The `statistics` option displays the statistics for the owner.

- `show owner [statistics]` - Display configuration information and statistics for the current owner, or for the owner of the current content rule. This command is available in Owner and Content mode. The `show owner` command with no options displays configuration information only. The `statistics` option displays the statistics for the current owner.

For example, to display configuration information for a specific owner from the ACL, Circuit, Global, Group, Interface, Service, SuperUser, or User modes, enter:

```
# show owner test.com
```

To display configuration information for the owner in Owner mode, enter:

```
(config-owner[test.com])# show owner
```

Table 8-2 describes the fields in the `show owner` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the owner.</td>
</tr>
<tr>
<td>Billing Info</td>
<td>The billing information about the owner.</td>
</tr>
<tr>
<td>Address</td>
<td>The postal address for the owner of the Web-hosting service.</td>
</tr>
</tbody>
</table>
Chapter 8  Configuring Owners

Table 8-2  Field Descriptions for the show owner name Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email Address</td>
<td>The e-mail address for the owner.</td>
</tr>
<tr>
<td>DNS Policy</td>
<td>The peer DNS exchange policy for the owner. The possible policies are:</td>
</tr>
<tr>
<td></td>
<td>• accept - Accept all content rules proposed by the CSS peer.</td>
</tr>
<tr>
<td></td>
<td>• push - Advertise the owner and push all content rules onto the CSS peer.</td>
</tr>
<tr>
<td></td>
<td>• both - Advertise the owner and push all content rules onto the CSS peer,</td>
</tr>
<tr>
<td></td>
<td>and accept all content rules proposed by the CSS peer.</td>
</tr>
<tr>
<td></td>
<td>• none - The default DNS exchange policy for the owner. The owner is hidden</td>
</tr>
<tr>
<td></td>
<td>from the CSS peer.</td>
</tr>
<tr>
<td>Case Matching</td>
<td>Indicates the matching of content requests to the owner’s rules is</td>
</tr>
<tr>
<td></td>
<td>case-sensitive or insensitive.</td>
</tr>
</tbody>
</table>

To display statistics for an owner from the ACL, Circuit, Global, Group, Interface, Service, SuperUser, or User modes, enter:

```
# show owner test.com statistics
```

To display statistics for the owner from either Owner or Content mode, enter:

```
(config-owner[test.com])# show owner statistics
```
Table 8-3 describes the fields in the `show owner name statistics` command output.

### Table 8-3  Field Descriptions for the show owner name statistics Command Output

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS Policy</td>
<td>The peer DNS exchange policy for the owner. The possible policies are:</td>
</tr>
<tr>
<td></td>
<td>• accept - Accept all content rules proposed by the CSS peer.</td>
</tr>
<tr>
<td></td>
<td>• push - Advertise the owner and push all content rules onto the CSS peer.</td>
</tr>
<tr>
<td></td>
<td>• both - Advertise the owner and push all content rules onto the CSS peer, and accept all content rules proposed by the CSS peer.</td>
</tr>
<tr>
<td></td>
<td>• none - The default DNS exchange policy for the owner. The owner is hidden from the CSS peer.</td>
</tr>
<tr>
<td>Hits</td>
<td>Number of connections processed under the rules of the owner.</td>
</tr>
<tr>
<td>Bytes</td>
<td>Total number of bytes transferred that matched the rules of the owner.</td>
</tr>
<tr>
<td>Frames</td>
<td>Total frames transferred that matched the rules of the owner.</td>
</tr>
<tr>
<td>Redirects</td>
<td>Total number of flows that have been redirected due to persistent connections or stickiness.</td>
</tr>
</tbody>
</table>
Showing Owner Information

The `show summary` command enables you to display a summary of the following owner information for all owners or a specific owner:

- Owners
- Content rules
- Services
- Service hits

You can issue the following `show summary` commands from any mode:

- `show summary` - Display a summary of all owner information
- `show summary owner_name` - Display a summary of owner information for a specific owner

For example, enter:
```
(config)# show summary
```
Table 8-4 describes the fields in the show summary command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Bypass Counters</td>
<td></td>
</tr>
<tr>
<td>No Rule Bypass Count</td>
<td>The number of times that a flow passes through even though it did not match one of the existing content rules.</td>
</tr>
<tr>
<td>ACL Bypass Count</td>
<td>The number of times that the ACL immediately sends traffic to its destination, bypassing the content rule.</td>
</tr>
<tr>
<td>URL Prams Bypass Count</td>
<td>The number of times that content requests match on content rules that have param-bypass set to enable. The CSS forwards the content requests to the origin server.</td>
</tr>
<tr>
<td>Cache Miss Bypass Count</td>
<td>The number of times that TCP connections from the cache servers bypassed content rules so the cache server could access the origin server for the requested content.</td>
</tr>
<tr>
<td>Garbage Bypass Count</td>
<td>The number of times that the CSS examined content requests and deemed them unrecognizable or corrupt. As a result, the CSS forwards the content request to the origin server rather than the cache server.</td>
</tr>
<tr>
<td>Owner</td>
<td>The owner name.</td>
</tr>
<tr>
<td>Content Rules</td>
<td>The rule associated with the owner.</td>
</tr>
<tr>
<td>State</td>
<td>The state of the rule (active or suspended).</td>
</tr>
<tr>
<td>Services</td>
<td>The services associated with the rule.</td>
</tr>
<tr>
<td>Service Hits</td>
<td>The number of hits on the service.</td>
</tr>
</tbody>
</table>

Where to Go Next

Once you create and configure an owner, see Chapter 9, Configuring Content Rules, for information on configuring content rules. Content rules instruct the CSS on how to handle requests for the owner’s content. You create and configure a content rule within a specific owner mode. This method ensures that the configured content rule applies only to a specific owner.
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 Extension Qualifier Lists
- Configuring URL Qualifier Lists
- Specifying a Uniform Resource Locator
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.

For information on how service, owners and content rules work together, see Chapter 1, Content Load-Balancing Overview.
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
9. 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 11, 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/index* or /test/index.html) 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 9-1 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 9-1.

Ensure that you have already created and configured a service and owner for the content rules. The command examples in Table 9-1 create a Layer 3 content rule for owner arrowpoint.

Table 9-1  Content Rule Configuration Quick Start

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter config mode by typing <strong>config</strong>.</td>
</tr>
<tr>
<td># config</td>
</tr>
<tr>
<td>(config)#</td>
</tr>
<tr>
<td>2. Enter the owner mode for which you wish to create content rules.</td>
</tr>
<tr>
<td>(config)# owner arrowpoint</td>
</tr>
<tr>
<td>3. Create the content rule for the owner.</td>
</tr>
<tr>
<td>(config-owner[arrowpoint])# content rule1</td>
</tr>
</tbody>
</table>

The CSS enters the owner-content rule mode.

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

```bash
(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).

```bash
(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).

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

5. Specify a load-balancing type.

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

6. Add previously configured services to the content rule.

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

7. Activate the content rule.

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

8. Display the content rules (optional).

```bash
(config-owner-content[arrowpoint-rule1]# show rule
```
Chapter 9   Configuring Content Rules

Naming and Assigning a Content Rule to an Owner

The following running-configuration example shows the results of entering the commands in Table 9-1.

```
!******************************************************************************
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 "//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.

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.
Chapter 9  Configuring Content Rules

Configuring a Virtual IP Address

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 12, 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).

- **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 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
```

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 9-1 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 9-1   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.

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


For example, enter:

```bash
(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:

```plaintext
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:

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

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.
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
```


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:

```plaintext
(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:

```plaintext
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
```
### Configuring Domain Qualifier Lists

**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 9-2 describes the fields in the `show dql` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the DQL</td>
</tr>
<tr>
<td>Index</td>
<td>The CSS unique index which identifies the DQL</td>
</tr>
<tr>
<td>Description</td>
<td>The description for the DQL</td>
</tr>
<tr>
<td>Index</td>
<td>The DQL unique index number for this domain</td>
</tr>
<tr>
<td>Domain</td>
<td>The name of the domain associated with the index number</td>
</tr>
<tr>
<td>Description</td>
<td>The description for the domain</td>
</tr>
</tbody>
</table>

### 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 9-3 for the steps required to configure virtual Web hosting.

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter config mode by typing <code>config</code>.</td>
</tr>
<tr>
<td>(config)#</td>
</tr>
<tr>
<td>2. Create a service.</td>
</tr>
<tr>
<td>(config)# service serv1</td>
</tr>
<tr>
<td>(config-service[serv1])#</td>
</tr>
<tr>
<td>3. Assign an IP address to the service and define the IP address range. Enter a number from 1 to 65535.</td>
</tr>
<tr>
<td>When using the <code>ip address range</code> 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.</td>
</tr>
<tr>
<td>(config-service[serv1])# ip address 10.3.6.1 range 200</td>
</tr>
</tbody>
</table>
Table 9-3  Virtual Web Hosting Configuration Quick Start (continued)

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Configure other service rules as needed (for example, protocol, keepalive parameters).</td>
</tr>
<tr>
<td>(config-service[serv1])# protocol tcp</td>
</tr>
<tr>
<td>(config-service[serv1])# keepalive type http</td>
</tr>
<tr>
<td>(config-service[serv1])# keepalive method get</td>
</tr>
<tr>
<td>(config-service[serv1])# keepalive uri “/index.html”</td>
</tr>
<tr>
<td><strong>Note</strong> 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.</td>
</tr>
<tr>
<td>5. Activate the service.</td>
</tr>
<tr>
<td>(config-service[serv1])# active</td>
</tr>
<tr>
<td>6. Create the content rule.</td>
</tr>
<tr>
<td>(config-owner[arrowpoint])# content rule1</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint-rule1])#</td>
</tr>
<tr>
<td>7. Configure a VIP. You can define a VIP range only if you do not plan to configure a DQL.</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint-rule1])# vip address 192.168.3.6 range 10</td>
</tr>
<tr>
<td>When using the <strong>vip address range</strong> 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.</td>
</tr>
<tr>
<td>8. Configure other content rule commands as needed (for example, port, protocol, and add a service).</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint-rule1])# port 80</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint-rule1])# protocol tcp</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint-rule1])# add service serv1</td>
</tr>
<tr>
<td>9. Activate the content rule.</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint-rule1])# active</td>
</tr>
<tr>
<td>10. Create a source group.</td>
</tr>
<tr>
<td>(config)# group group1</td>
</tr>
<tr>
<td>(config-group[group1])#</td>
</tr>
</tbody>
</table>
The following running-configuration example shows the results of entering the commands in Table 9-3.

```plaintext
!**************************************** SERVICE ****************************************
service serv1
  ip address 10.3.6.1 range 200
  protocol tcp
  keepalive type http
  keepalive method get
  keepalive url "/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"
```

Table 9-3 Virtual Web Hosting Configuration Quick Start (continued)

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>11. Configure a VIP address range on the source group.</td>
</tr>
<tr>
<td>(config-group[group1])# vip address 192.168.5.7 range 10</td>
</tr>
<tr>
<td>12. Add the services that you want to be part of the group.</td>
</tr>
<tr>
<td>(config-group[group1])# add service serv1</td>
</tr>
<tr>
<td>13. Activate the source group.</td>
</tr>
<tr>
<td>(config-group[group1])# active</td>
</tr>
<tr>
<td>14. If you have not configured a VIP range on a content rule, you can create a DQL.</td>
</tr>
<tr>
<td>(config)# dql pet_domains</td>
</tr>
<tr>
<td>(config-dql[pet_domains])#</td>
</tr>
<tr>
<td>15. Add domains to the DQL you created.</td>
</tr>
</tbody>
</table>
| (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 |
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

A Layer 5 content rule supports the HTTP CONNECT, GET, HEAD, POST, PUSH, and PUT methods. In addition, 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: RFC 2068 - OPTIONS, TRACE and RFC 2518 - PROPFIND, PROPPATCH, MKCOL, MOVE, LOCK, UNLOCK, COPY, DELETE.
Adding Services 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.
Adding Services to a Content Rule

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 `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
```
Adding Services to a Content Rule

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 affect 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 slowserv
```
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:

```bash
(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:

```bash
(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:

```bash
(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:

```bash
(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. 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.
Configuring Load Balancing

- **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** If you are using the **domainhash** load-balancing method with proxy cache services, you may see duplicate sites across caches because the CSS balances on the first GET request in a persistent connection unless the subsequent GET request does not match a rule with the same proxy service specified. If you are concerned with duplicate hits across caches, reset persistence to remap and disable persistence on the rule. Issue the *(config)* persistence reset remap command globally and the *(config-owner-content)* no persistent command on the content rule.

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

  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 weights 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.

A Layer 5 content rule supports the HTTP CONNECT, GET, HEAD, POST, PUSH, and PUT methods. In addition, 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: RFC 2068 - OPTIONS, TRACE and RFC 2518 - PROPFIND, PROPPATCH, MKCOL, MOVE, LOCK, UNLOCK, COPY, DELETE. In a transparent caching environment (for example, no VIP address on a Layer 5 content rule), the CSS bypasses these HTTP methods, and they are forwarded to the destination server.

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
```
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 8, 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 9-4 describes the fields in the `show domain hotlist` command output.

**Table 9-4  Field Descriptions for the show domain hotlist Command Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hotlist</td>
<td>Enable the domain hot list. The domain hot list is disabled by default.</td>
</tr>
<tr>
<td>Size</td>
<td>The configured maximum number of domain entries contained in the hot list. The range is from 1 to 100. The default is 10.</td>
</tr>
<tr>
<td>Interval</td>
<td>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.</td>
</tr>
<tr>
<td>Threshold</td>
<td>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.</td>
</tr>
<tr>
<td># Hot Domains</td>
<td>The total number of hot domains.</td>
</tr>
<tr>
<td>Hits</td>
<td>The number of hits for a hot domain.</td>
</tr>
<tr>
<td>Domain</td>
<td>The name of the hot domain associated with the Hits field.</td>
</tr>
</tbody>
</table>
Configuring a Domain Hotlist

A domain hotlist lists the most accessed domains on a CSS during a user-defined period of time. Use the `domain` command to enable the domain hotlist 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:

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

To disable the domain hot list, enter:

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

To display the domain hot list and its configuration, use the `show domain hotlist` command (see Table 9-4).

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 9, Configuring Content Rules.
Showing EQL Extensions and Descriptions

To display a list of existing EQLs names, use `eql ?` command.
For example, enter:
```
(config)# eql ?
```

To display the extensions configured for a specific EQL including any descriptions, use the `show eql` command and the EQL name. For example, enter:
```
(config)# show eql graphics
```

Table 9-5 describes the fields in the `show eql` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQL</td>
<td>The name of the EQL and its description, if configured</td>
</tr>
<tr>
<td>Extensions</td>
<td>The extensions of content requests associated with the EQL and their descriptions, if configured</td>
</tr>
</tbody>
</table>

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 9-6 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>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Create a URQL.</td>
</tr>
<tr>
<td>(config)# urql videos</td>
</tr>
<tr>
<td>(config-urql[videos])#</td>
</tr>
<tr>
<td>2. Optionally, describe the URQL.</td>
</tr>
<tr>
<td>(config-urql[videos])# description &quot;cooking streaming video&quot;</td>
</tr>
</tbody>
</table>
Table 9-6  URQL Configuration Quick Start (continued)

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Configure the URLs you want to group in the URQL:</td>
</tr>
<tr>
<td>a. Specify the URL entry.</td>
</tr>
<tr>
<td>(config-urql[videos])# url 10</td>
</tr>
<tr>
<td>b. Define the URL.</td>
</tr>
<tr>
<td>(config-urql[videos])# url 10 url &quot;/cooking/cookies.avi&quot;</td>
</tr>
<tr>
<td>c. Optionally, describe the URL</td>
</tr>
<tr>
<td>(config-urql[videos])# url 10 description &quot;making cookies&quot;</td>
</tr>
<tr>
<td>4. Designate the domain name of the URLs in a URQL. For example:</td>
</tr>
<tr>
<td>(config-urql[videos])# domain &quot;www.arrowpoint.com&quot;</td>
</tr>
<tr>
<td>5. Add the URQL to a content rule using the owner-content url command.</td>
</tr>
<tr>
<td>(config-owner-content[chefsbest-recipes])# url urql videos</td>
</tr>
</tbody>
</table>

The following running-configuration example shows the results of entering the commands in Table 9-6.

```
!*************************** 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"

candidate
    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.
```
Configuring URL Qualifier Lists

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 9, 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:

```sh
(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.

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

Showing URQLs

To display a list of URQLs, enter:

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

To display all configured URQLs, enter:

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

To display a specific URQL, enter:

```sh
(config)# show urql videos
```
Table 9-7 describes the fields in the `show urql` command output.

**Table 9-7  Field Descriptions for the show urql Command Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the URQL</td>
</tr>
<tr>
<td>Description</td>
<td>The configured description for the URQL</td>
</tr>
<tr>
<td>Domain</td>
<td>The domain name or address of the URLs associated with the URQL</td>
</tr>
<tr>
<td>Create Type</td>
<td>The create type (static or dynamic)</td>
</tr>
<tr>
<td>State</td>
<td>The state of the URQL (Active or Suspended)</td>
</tr>
<tr>
<td>Rules Associated</td>
<td>The number of rules associated with the URQL</td>
</tr>
</tbody>
</table>

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

**Table 9-8  Field Descriptions for a Specified URQL**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URQL Domain</td>
<td>The domain name or address of the URLs associated with the URQL</td>
</tr>
<tr>
<td>Number of entries configured</td>
<td>The number of URL entries in the URQL</td>
</tr>
<tr>
<td>URL</td>
<td>The URL</td>
</tr>
<tr>
<td>Description</td>
<td>The description associated with the URL</td>
</tr>
<tr>
<td>Create Type</td>
<td>The create type (static or dynamic)</td>
</tr>
<tr>
<td>State</td>
<td>The state of the URL (Active or Suspended)</td>
</tr>
<tr>
<td>CSD Entries</td>
<td>The number of Content Server Database (CSD) entries</td>
</tr>
</tbody>
</table>
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**

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.

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/*"
  ```

To use stickiness based on Secure Sockets Layer (SSL) session ID, set the URL, set the port to 443 with the `port` command and enable stickiness with the `advanced-balance ssl` command. Then specify an SSL application type.
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.
  - **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.

For example, “/announcements/new/*”. The *eql_name* is the name of the EQL. To see a list of EQLs, use the **eql ?** command.
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 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. In addition, for UDP flows, the command allows the CSS to purge the flow cache of the UDP flow so that another request gets remapped to a different IP address, if necessary, without attempting to use the previously mapped IP address. 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.

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
```
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 9-9 describes the fields in the `show remap` command output.

**Table 9-9  Field Descriptions for the show remap Command Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group SFP Port Map</td>
<td>This field is currently not used.</td>
</tr>
<tr>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>Persistence Reset</td>
<td>The configured persistence reset method when resetting a connection to a</td>
</tr>
<tr>
<td>Method</td>
<td>new back-end service. The possible methods are:</td>
</tr>
<tr>
<td></td>
<td>- <strong>redirect</strong> - Causes an HTTP redirection when resetting a connection to</td>
</tr>
<tr>
<td></td>
<td>a new back-end service. An HTTP redirection resets both sides of the</td>
</tr>
<tr>
<td></td>
<td>connection.</td>
</tr>
<tr>
<td></td>
<td>- <strong>remap</strong> - Uses a back-end remapping operation when resetting a</td>
</tr>
<tr>
<td></td>
<td>connection to a new back-end service.</td>
</tr>
<tr>
<td>Bypass Persistence</td>
<td>The configured bypass persistence setting. The possible settings are:</td>
</tr>
<tr>
<td></td>
<td>- <strong>disable</strong> - The CSS performs either a service remapping or HTTP</td>
</tr>
<tr>
<td></td>
<td>redirection operation to reset a bypassed service when a content request</td>
</tr>
<tr>
<td></td>
<td>matches a content rule, but a previous request caused the bypass.</td>
</tr>
<tr>
<td></td>
<td>- <strong>enable</strong> - The CSS does not perform remapping or redirection to reset</td>
</tr>
<tr>
<td></td>
<td>the connection and continues to bypass a service. By default, bypass</td>
</tr>
<tr>
<td></td>
<td>persistence is enabled.</td>
</tr>
</tbody>
</table>
Defining Failover

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

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 9-2** 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 9-2 ServerB Configured for Failover Next**
As shown in Figure 9-3, if ServerC fails, the CSS sends ServerC content requests to ServerA because no other services were configured after ServerC.

**Figure 9-3  ServerC Configured for Failover Next**

Figure 9-4 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 9-4 and Figure 9-5 use the alphabet to illustrate division balance.

**Figure 9-4  Suspended or Failed Service Configured for Failover Linear**
Figure 9-5 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.

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 HTTP CONNECT, GET, HEAD, POST, PUSH, and PUT methods. In addition, 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: RFC 2068 - OPTIONS, TRACE and RFC 2518 - PROPFIND, PROPPATCH, MKCOL, MOVE, LOCK, UNLOCK, COPY, DELETE.
Chapter 9  Configuring Content Rules

Specifying an Application Type

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.
- **http** (default) - Process HTTP data streams.
- **realaudio-control** - Process RealAudio Control 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 10, 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. 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**.

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
```
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 9-10 describes the fields in the `show content` command output.

**Note**

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

### Table 9-10 Field Descriptions for the show content Command Output

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pieces of Content for Slot</td>
<td>The chassis slot number in which the module resides.</td>
</tr>
<tr>
<td>Subslot</td>
<td>The module slot number in which the Session Processor resides.</td>
</tr>
<tr>
<td>Total Content</td>
<td>The total number of content entries.</td>
</tr>
<tr>
<td>Index</td>
<td>Unique index for known content in the CSD.</td>
</tr>
<tr>
<td>&lt;address&gt;</td>
<td>The IP address of the content.</td>
</tr>
<tr>
<td>Protocol</td>
<td>The IP Protocol of the content.</td>
</tr>
<tr>
<td>Port</td>
<td>Protocol port of the content.</td>
</tr>
</tbody>
</table>
Chapter 9      Configuring Content Rules

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:

- `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

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

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

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
```

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 9-11 describes the fields in the show rule command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the content rule.</td>
</tr>
<tr>
<td>Owner</td>
<td>The owner of the rule.</td>
</tr>
<tr>
<td>Author</td>
<td>The author (Local CSS or remote CSS peer) of the rule.</td>
</tr>
</tbody>
</table>
### Table 9-11  Field Descriptions for the show rule Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>A CSS assigned unique index for the rule. The number is based in the order that the rule was created.</td>
</tr>
<tr>
<td>State</td>
<td>The state of the rule (active or suspend).</td>
</tr>
<tr>
<td>Type</td>
<td>The application type associated with the rule. The possible values are:</td>
</tr>
<tr>
<td>bypass</td>
<td>Bypasses the matching of the content rule and sends the request directly to the origin server</td>
</tr>
<tr>
<td>http</td>
<td>Processes HTTP data streams (default)</td>
</tr>
<tr>
<td>ftp-control</td>
<td>Processes FTP data streams</td>
</tr>
<tr>
<td>realaudio-control</td>
<td>Processes RealAudio Control data streams</td>
</tr>
<tr>
<td>ssl</td>
<td>Processes Secure Sockets Layer (SSL) protocol data streams</td>
</tr>
<tr>
<td>L3</td>
<td>Destination IP address.</td>
</tr>
<tr>
<td>L4</td>
<td>Destination protocol and port.</td>
</tr>
<tr>
<td>URL</td>
<td>The URL for the content.</td>
</tr>
<tr>
<td>URQL</td>
<td>The name of the associated URL Qualifier list.</td>
</tr>
<tr>
<td>EQL</td>
<td>The name of the associated EQL.</td>
</tr>
<tr>
<td>DQL</td>
<td>The name of the associated DQL.</td>
</tr>
<tr>
<td>Header Field Group</td>
<td>The name of the associated header-field group.</td>
</tr>
<tr>
<td>Total Bytes</td>
<td>Total bytes to the content rule.</td>
</tr>
<tr>
<td>Total Frames</td>
<td>Total frames to the content rule.</td>
</tr>
<tr>
<td>Total Redirects</td>
<td>Total redirects by the content rule (when the redirect command is configured for a content rule). This field increments whenever a request for content is redirected to an alternate location.</td>
</tr>
</tbody>
</table>
### Table 9-11  Field Descriptions for the show rule Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Rejects</td>
<td>Total rejects by the content rule. This field increments when all services for a content rule are unavailable.</td>
</tr>
<tr>
<td>Overload Rejects</td>
<td>Total rejects on the content rule due to overload on the rule’s available services.</td>
</tr>
<tr>
<td>Balance</td>
<td>The load-balancing algorithm for the content rule. The possible values are:</td>
</tr>
<tr>
<td></td>
<td>• <strong>ACA</strong> - ArrowPoint Content Awareness algorithm. The CSS correlates content request frequency with the server’s cache sizes to improve cache hit rates for that server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>destip</strong> - Destination IP address division. The CSS directs all client requests with the same destination IP address to the same service.</td>
</tr>
<tr>
<td></td>
<td>• <strong>domain</strong> - Domain name division. The CSS uses the domain name in the request URI to direct the client request to the appropriate service.</td>
</tr>
</tbody>
</table>
**Table 9-11  Field Descriptions for the show rule Command Output (continued)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Balance (continued)** | • **domainhash** - 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.  
  • **leastconn** - Least connections. The CSS chooses a running service that has the least number of connections.  
  • **roundrobin** - Roundrobin algorithm (default).  
  • **srcip** - Source IP address division. The CSS directs all client requests with the same source IP address to the same service.  
  • **url** - URL division. The CSS uses the URL (omitting the leading slash) in the redirect URL to direct the client requests to the appropriate service.  
  • **urllash** - 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.  
  • **weightedrr** - 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. |
Advanced Balance

The advanced load-balancing method for the content rule, including stickiness. The possible values are:

- **none** - Disables the advanced-balancing method for the rule. This is the default setting.
- **arrowpoint-cookie** - 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.
- **cookies** - 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.
- **cookieurl** - 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.
- **sticky-srcip** - 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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Balance</td>
<td>The advanced load-balancing method for the content rule, including stickiness. The possible values are:</td>
</tr>
<tr>
<td>none</td>
<td>Disables the advanced-balancing method for the rule. This is the default setting.</td>
</tr>
<tr>
<td>arrowpoint-cookie</td>
<td>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.</td>
</tr>
<tr>
<td>cookies</td>
<td>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.</td>
</tr>
<tr>
<td>cookieurl</td>
<td>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.</td>
</tr>
<tr>
<td>sticky-srcip</td>
<td>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.</td>
</tr>
</tbody>
</table>
### Table 9-11  Field Descriptions for the show rule Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Advanced Balance (continued) | • **sticky-srcip-dstport** - 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.  
  • **ssl** - 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.  
  • **url** - 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. |
| 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. |
Table 9-11  Field Descriptions for the show rule Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sticky No Cookie Found Action</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>• loadbalance - The CSS uses the configured balanced method when no cookie is found in the client request. This is the default setting.</td>
</tr>
<tr>
<td></td>
<td>• redirect &quot;URL&quot; - 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.</td>
</tr>
<tr>
<td></td>
<td>• reject - The CSS rejects the client request when no cookie is found in the request.</td>
</tr>
<tr>
<td></td>
<td>• service name - The CSS sends the no cookie client request to the specified service when no cookie is found in the request.</td>
</tr>
</tbody>
</table>
Sticky Server Down Failover

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:

- **Balance** - The failover method uses a service based on the configured load-balancing method (default).
- **Redirect** - 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.
- **Reject** - The failover method rejects the content request.
- **Sticky-srcip** - The failover method uses a service based on the client IP address. This is dependent on the sticky configuration.
- **Sticky-srcip-dstport** - The failover method uses a service based on the client IP address and the server destination port. This is dependent on the sticky configuration.

### ArrowPoint Cookie Path

The pathname where you want to send the ArrowPoint cookie. The default path of the cookie is “/”.

### ArrowPoint Cookie Expiration

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.

### ArrowPoint Cookie CSS/Browser Expired

Indicates whether the **arrowpoint-cookie browser-expire** 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.”
Table 9-11  Field Descriptions for the show rule Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrowPoint Cookie</td>
<td>Specifies whether the <strong>arrowpoint-cookie expire-services</strong> 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.</td>
</tr>
<tr>
<td>Service</td>
<td></td>
</tr>
<tr>
<td>ArrowPoint Cookie</td>
<td>Specifies whether the <strong>advanced-balance arrowpoint-cookie</strong> 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.</td>
</tr>
<tr>
<td>Advanced</td>
<td></td>
</tr>
<tr>
<td>ArrowPoint Cookie</td>
<td>Specifies the format of the ArrowpointCookie expiration time, whether the RFC 2822-compliant format is enabled or disabled. The <strong>arrowpoint-cookie rfc2822-compliant</strong> 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”).</td>
</tr>
<tr>
<td>Format</td>
<td></td>
</tr>
</tbody>
</table>
Table 9-11  Field Descriptions for the show rule Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Match Criteria</td>
<td>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.</td>
</tr>
</tbody>
</table>
| String Range           | 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.  
  • The range is from 1 to 1999. The default starting byte position is 1.  
  • The range is from 2 to 2000. The default ending byte position is 100. |
| 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, ASCII 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 9-11 Field Descriptions for the show rule Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Process-Len</td>
<td>The number of bytes, after the end of the prefix designated by the <strong>string prefix</strong> command and skipping the bytes designated by the <strong>string skip-length</strong> command, that the string operation will use. The range is from 0 to 64. The default is 0.</td>
</tr>
<tr>
<td>String Operation</td>
<td>The method to choose a destination server for a string result; derived from the settings of the string criteria commands. The possible values are:</td>
</tr>
<tr>
<td></td>
<td>- <strong>match-service-cookie</strong> - 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.</td>
</tr>
<tr>
<td></td>
<td>- <strong>hash-a</strong> - Apply a basic hash algorithm on the hash string to generate the hash key.</td>
</tr>
<tr>
<td></td>
<td>- <strong>hash-crc32</strong> - Apply the CRC32 algorithm on the hash string to generate a hash key.</td>
</tr>
<tr>
<td></td>
<td>- <strong>hash-xor</strong> - Perform an Exclusive OR (XOR) on each byte of the hash string to derive the final hash key.</td>
</tr>
<tr>
<td>Location-Cookie</td>
<td>The format (NAME=VALUE) of the location cookie string.</td>
</tr>
<tr>
<td>Location-Cookie</td>
<td>The expiration date and time of the location cookie. This value tells the client browser when the cookie will expire.</td>
</tr>
<tr>
<td>Cookie-Domain</td>
<td>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.</td>
</tr>
<tr>
<td>Redirect</td>
<td>Text used to build an HTTP 302 redirect message that is sent to the client when the rule is matched.</td>
</tr>
</tbody>
</table>
### Table 9-11  Field Descriptions for the show rule Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistence</td>
<td>Indicates whether or not a persistent connection with a server is maintained. By default, persistence is enabled.</td>
</tr>
<tr>
<td>Param-Bypass</td>
<td>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.</td>
</tr>
<tr>
<td>Session Redundancy</td>
<td>Indicates whether ASR is enabled or disabled on the rule. For details on ASR, refer to the <em>Cisco Content Services Switch Redundancy Configuration Guide</em>.</td>
</tr>
<tr>
<td>Redund Glb Index</td>
<td>The unique global index value for Adaptive Session Redundancy assigned to the content rule using the <code>redundant-index</code> command in owner-content configuration mode.</td>
</tr>
<tr>
<td>IP Redundancy</td>
<td>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.</td>
</tr>
<tr>
<td>Flow Timeout Multiplier</td>
<td>Number of seconds that a flow remains idle before the CSS reclaims the flow resources, as configured with the <code>flow-timeout-multiplier</code> command. For details on the <code>flow-timeout-multiplier</code> command, see Chapter 2, Configuring Flow and Port Mapping Parameters.</td>
</tr>
</tbody>
</table>
Table 9-11  Field Descriptions for the show rule Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Services</td>
<td>Content rule services to configuration and statistic information, as follows:</td>
</tr>
<tr>
<td>Local Load Threshold</td>
<td>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.</td>
</tr>
<tr>
<td>PrimarySorryServer</td>
<td>The primary service to be used when all other services for the content rule are unavailable.</td>
</tr>
<tr>
<td>SecondSorryServer</td>
<td>The secondary service to be used when all other services for the content rule are unavailable.</td>
</tr>
<tr>
<td>Name</td>
<td>The names of the services.</td>
</tr>
<tr>
<td>Hits</td>
<td>The number of content accesses on the service.</td>
</tr>
<tr>
<td>Wgt</td>
<td>The weight for the service used when you configure ACA, weighted roundrobin, and 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:</td>
</tr>
<tr>
<td></td>
<td>• D = Weight reported by DFP</td>
</tr>
<tr>
<td></td>
<td>• R = Weight configured for a service using the add service weight command in owner-content mode</td>
</tr>
<tr>
<td></td>
<td>• S = Weight configured for a service using the weight command in service mode</td>
</tr>
<tr>
<td>State</td>
<td>The state of the service.</td>
</tr>
<tr>
<td>Ld</td>
<td>The service load. The range is from 2 to 255; 255 indicates that the service is unavailable.</td>
</tr>
<tr>
<td>KAlive</td>
<td>The service keepalive type.</td>
</tr>
</tbody>
</table>
Table 9-11  Field Descriptions for the show rule Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conn</td>
<td>The number of connections currently mapped to the service.</td>
</tr>
<tr>
<td>DNS</td>
<td>The number of times that the CSS DNS resolver chose the service as the answer to a DNS client query.</td>
</tr>
<tr>
<td>DNS Names</td>
<td>Domain Name System names.</td>
</tr>
<tr>
<td>DNS TTL</td>
<td>The Time to Live value, in seconds, which determines how long the DNS client remembers the IP address response to the query.</td>
</tr>
<tr>
<td>DNS Balance</td>
<td>Where the CSS resolves a request for a domain name into an IP address. The possible values are:</td>
</tr>
<tr>
<td></td>
<td>• leastloaded - 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.</td>
</tr>
<tr>
<td></td>
<td>• Preferlocal - 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.</td>
</tr>
<tr>
<td></td>
<td>• roundrobin - 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.</td>
</tr>
<tr>
<td></td>
<td>• useownerdnssbalance - 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.</td>
</tr>
</tbody>
</table>
Table 9-11  Field Descriptions for the show rule Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hotlist</td>
<td>Indicates whether or not hot list is enabled.</td>
</tr>
<tr>
<td>Size</td>
<td>The total number of hot-list entries that is maintained for the rule. The range is from 1 to 100. The default is 10.</td>
</tr>
<tr>
<td>Type</td>
<td>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.</td>
</tr>
<tr>
<td>Threshold</td>
<td>The hit count per interval threshold below which content is not considered hot. The range is from 0 to 65535. The default is 0.</td>
</tr>
<tr>
<td>Interval</td>
<td>The interval, in minutes, for refreshing the hot list. The range is from 1 to 60. The default is 1.</td>
</tr>
<tr>
<td>Associated ACLs</td>
<td>The ACLs associated with a content rule.</td>
</tr>
<tr>
<td>TCP RST Client If Service Unreachable</td>
<td>Whether or not the flow-reset-reject 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 flow-reset-reject command is disabled.</td>
</tr>
</tbody>
</table>

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 servi
```

**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 10, Configuring Sticky Parameters for Content Rules.
Configuring Sticky Parameters for Content Rules

This chapter describes how to configure sticky parameters for content rules. The information in this chapter applies to all CSS models, except where noted. This chapter contains the following major sections:

- Sticky Overview
- Configuring Sticky on the CSS
- Specifying an Advanced Load-Balancing Method for Sticky Content
- Configuring SSL-Layer 4 Fallback
- Configuring Sticky Serverdown Failover
- Configuring Sticky Mask
- Configuring Sticky Inactive Timeout
- Configuring Sticky Content for SSL
- Configuring String Range
- Specifying a String Operation
- Enabling or Disabling String ASCII Conversion
- Configuring the Handling of Multiple String Matches
- Specifying End-of-String Characters
- Specifying a String Prefix
- Specifying a String Process Length
- Specifying a String Skip Length
Sticky Overview

During a session, the CSS maintains an association between a client and a server. This association is referred to as *stickiness*. Stickiness enables transactions over the Web when the client must remain on the same server for the entire session. Depending on the content rule, the CSS “sticks” a client to an appropriate server after the CSS has determined which load-balancing method to use.

If the CSS determines that a client is already stuck to a particular service, then the CSS places the client request on that service, regardless of the load balancing criteria specified by the matched content rule. If the CSS determines that the client is not stuck to a particular service, it applies normal load balancing to the content request.

Client *cookies* uniquely identify clients to the services providing content. A cookie is a small data structure used by a server to deliver data to a Web client and request that the client store the information. In certain applications, the client returns the information to the server to maintain the state between the client and the server.

When the CSS examines a request for content and determines through content rule matching that the content is sticky, it examines any cookie or URL present in the content request. The CSS uses this information to place the content request on the appropriate server.

The total number of entries in the CSS sticky table depends on the size of the CPU memory.

- The CSS 11501 supports a 128K sticky table (288 MB of CPU memory).
- The CSS 11503 and 11506 supports either a 128K or 32K sticky table, depending on whether the System Control module (SCM) has 288 MB or 144 MB of memory. With 288 MB of memory, the CSS supports a 128K sticky table. With 144 MB of memory, the CSS supports a 32K sticky table.
The size of the sticky table means that once 128K or 32K simultaneous users are on the site, the table wraps and the first users become “unstuck.”

The following sections describe stickiness and its uses:

- Why Use Stickiness?
- Using Layer 3 Sticky
- Using Layer 4 Sticky
- Using Layer 5 Sticky

Why Use Stickiness?

When customers visit an e-commerce site, they usually start out by browsing the site, the Internet equivalent of window shopping. Depending on the application, the site may require that the customer become “stuck” to one server once the connection is established, or the application may not require this until the customer starts to build a shopping cart.

In either case, once the customer adds items to the shopping cart, it is important that all of the customer’s requests get directed to the same server so that all the items are contained in one shopping cart on one server. An instance of a customer's shopping cart is typically local to a particular Web server and is not duplicated across multiple servers.

E-commerce applications are not the only types of applications that require stickiness. Any Web application that maintains client information may require stickiness, such as banking applications or online trading.

Because the application must distinguish each user or group of users, the CSS needs to determine how a particular user is stuck to a specific Web server. The CSS can use a variety of methods, including:

- Source IP address
- Source IP address and destination port
- String found in a cookie or a URL
- SSL session ID

The e-commerce application itself dictates which of these methods is appropriate for a particular e-commerce vendor.
Using Layer 3 Sticky

If an application requires that a user be stuck for the entire session, use Layer 3 sticky, which sticks a user to a server based on the user's IP address. The total number of entries in the sticky table depends on the size of the CPU memory (128K sticky table with 288 MB of CPU memory or a 32K sticky table with 144 MB of CPU memory).

If the volume of your site is such that you will have more than 128K or 32K users at a time, or if a large percentage of your customers come to you through a mega-proxy, then consider using either a different sticky method (for example, the advanced-balance method cookies, cookieurl, or url), or increasing your sticky mask.

Note

If you use the sticky-inact-timeout command to specify the inactivity timeout period on a sticky connection, when the sticky table becomes full and none of the entries have expired from the sticky table, the CSS rejects subsequent needed sticky requests.

The default sticky mask is 255.255.255.255, which means that each entry in the sticky table is an individual IP address. Some mega-proxies allow one user to use several different IP addresses in a range of addresses over the life of one session. This use of multiple addresses for one session can cause some of the TCP connections to get stuck to one server, and other TCP connections to a different server for the same transaction. The result is possibly losing some items from the shopping cart. To avoid this problem, use one of the more advanced methods of sticking. If you cannot, Cisco Systems recommends using a sticky mask of 255.255.240.0.

Using Layer 4 Sticky

Layer 4 sticky functions identically to Layer 3 sticky, except that it sticks based on a combination of source IP address, protocol, and destination port. Layer 4 sticky also uses a sticky table and has the same limitations as Layer 3 sticky.

If the CSS sees the same IP address with two different destination ports, it will use two entries. You can also apply sticky mask to Layer 4 sticky.

If you are concerned about whether your site can handle all of the simultaneous sessions, then consider using the Layer 5 advanced-balanced methods of arrowpoint-cookie, cookie, cookieurl, or url.
Using Layer 5 Sticky

Layer 5 sticky uses a combination of destination IP address, protocol, port, and URL that may or may not contain an HTTP cookie or a domain name. Layer 5 sticky can function based on a sticky string in a cookie or URL, or based on an SSL version 3 session ID. The advanced-balanced methods such as arrowpoint-cookie, cookie, cookieurl, and url do not use a sticky table to keep track of IDs. The advanced-balance ssl method for SSL sticky does use a sticky table.

Note

If you use the sticky-inact-timeout command to specify the inactivity timeout period on a sticky connection, when the sticky table becomes full and none of the entries have expired from the sticky table, the CSS rejects subsequent new sticky requests. If the sticky-inact-timeout command is specified for a Layer 5 content rule using SSL sticky, the SSL sessions continue even if the sticky table is full but the CSS does not maintain stickiness on the new sessions.

Configuring Sticky on the CSS

Configuring sticky on the CSS requires you to:

- Determine the sticky method you want to use according to the requirements of the site (for example, Layer 3, Layer 4, or one of the string methods)
- Configure a failover method

If you use advanced-balance methods cookies, url, or cookieurl, you must also:

- Determine whether you want to use an exact string match or a hash, and then configure that function.
- Determine how you want to delimit (configure) the string.

To configure sticky on the CSS:

1. Configure the sticky method using the advanced-balance command and its options. The advanced-balance command options are described in “Specifying an Advanced Load-Balancing Method for Sticky Content” later in this chapter.
   - To configure Layer 3 sticky, use advanced-balance sticky-srcip in the content rule. If necessary, change the sticky mask from the default of 255.255.255.255.
To configure Layer 4 sticky, use `advanced-balance sticky-srcip-dstport` in the content rule. If necessary, change the sticky mask from the default of 255.255.255.255.

To configure sticky cookies, use `advanced-balance cookies` in the content rule.

To configure sticky URL, use `advanced-balance url` in the content rule.

To configure sticky cookies with URLs, use `advanced-balance cookieurl` in the content rule.

2. Configure a failover method. Use the `sticky-serverdown-failover` command to define what will happen if a sticky string is found but the associated service has failed or is suspended. The sticky failover default is for the CSS to use the configured load-balancing method. The `sticky-serverdown-failover` options are described in the “Configuring Sticky Serverdown Failover” section later in this chapter.

If you configured an advanced-balance method of `sticky-srcip` or `sticky-srcip-dstport`, no further steps are required.

If you configured the advanced-balance methods `cookies`, `url`, or `cookieurl`, complete Steps 3 and 4.

3. If you are using `advanced-balance cookies`, `url`, or `cookieurl`, determine whether you want to use an exact string match or a hash.

To use an exact string match:

a. Enter the `string operation match-service-cookie` command (this is the default for the `string operation` command).

b. For each service configuration, use the service mode `string` command to configure the unique string that you want to use for matching each server.

   For example, you have three servers and you want the string matching to be serverid111 for service1, serverid112 for service2, and serverid113 for service3. Configure the Web server applications to use these strings when they set cookies or pass parameters.

   For information on the `string operation match-service-cookie` command, see the “Specifying a String Operation” section later in this chapter.

To use the hash algorithm:

a. Enter the `string operation` command in the content rule.
b. Select an option (hash-a, hash-crc32, or hash-xor) depending on the hash method you wish to use. Hashing requires that each server can accept cookies set by all other servers.

Cisco TAC recommends using either hash-xor or hash-crc32, depending on your string possibilities. If the strings are completely dissimilar, use hash-xor. If the strings are similar, use hash-crc32. For example, if your string values are abc1, abc2, and abc3, the hash-xor method cannot provide you with enough variance in the hash values (that is, abc1 and abc2 may end up on the same server because they may hash to the same value).

For information on the string operation hash options, see the “Specifying a String Operation” section later in this chapter.

4. If you are using advanced-balance cookies, url, or cookieurl, determine how you want to delimit (configure) the string. Use the following owner-content string commands to delimit the string:

- **string range** - Defining the string range enables you to limit the size of the search. By default the CSS searches the first 100 bytes of the cookie, URL, or parameters in the URL depending on the method. If you know where in the cookie or URL the string is likely to appear, define the string range accordingly. The range is from 1 to 2000. The default is 1 to 100. The string range options are described in the “Configuring String Range” section later in this chapter.

- **string eos-char** - A maximum of 3 ASCII characters that delimit the end of the string within the string range. Use this option when the string length varies. Note that string process-length overrides string eos-char. If you do not configure either option, the CSS uses a maximum of 100 bytes for the delimiter.

- **string prefix** - The CSS uses the string prefix (maximum of 30 characters) to locate the string within the string range of the cookie or URL. If the string prefix is specified, but not found, the CSS uses the normal balance method.

- **string process-length** - Specifies the number of bytes within the string range after the end of the prefix plus the skip-length that is used to determine the string. Use this option when the string length is fixed.

- **string skip-length** - Specifies the number of bytes to skip after the end of the prefix within the string range. The range is 0 to 64.
For example, if you are using ipaddr=192.168.3.6&, then use the **string prefix** “ipaddr=” and the **string eos-char** “&” because the IP addresses vary in length.

For example, if you are using server ID=server111, then use the **string prefix** “server ID=” and a **string process-length** of 8 because the string length does not vary in length.

**Table 10-1** describes sticky rules and how they apply to content rules.

**Table 10-1 Applying Sticky Rules to Content Rules**

<table>
<thead>
<tr>
<th>Rule Type</th>
<th>Sticky Configuration</th>
<th>Stickiness Based on...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 3 content rule</td>
<td>advanced-balance sticky-srcip</td>
<td>Source IP address using a sticky mask.</td>
</tr>
<tr>
<td>Layer 4 content rule</td>
<td>advanced-balance sticky-srcip-dstport</td>
<td>Source IP address and destination port using a sticky mask.</td>
</tr>
<tr>
<td>Layer 5 content rule not using a sticky string</td>
<td>advanced-balance sticky-srcip-dstport</td>
<td>Source IP address and destination port using a sticky mask.</td>
</tr>
<tr>
<td>Layer 5 content rule using a sticky string</td>
<td>advanced-balance cookies or advanced-balance cookieurl</td>
<td>Searching for a sticky string in the cookie or URL. If the CSS does not find the sticky string in the cookie or URL, the CSS load-balances each request among the available servers.</td>
</tr>
<tr>
<td>Layer 5 content rule with SSL</td>
<td>advanced-balance ssl</td>
<td>SSL v3 session ID. If no session ID is present, the CSS uses the source IP address and destination port to maintain stickiness.</td>
</tr>
</tbody>
</table>
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 Chapter 9, Configuring Content Rules.

### Specifying an Advanced Load-Balancing Method for Sticky Content

A content rule is “sticky” when additional sessions from the same user or client are sent to the same service as the first connection, overriding normal load balancing. By default, the advanced balancing method is disabled.

Use the `advanced-balance` command to specify an advanced load-balancing method for a content rule that includes stickiness. The `advanced-balance` command options `cookies`, `cookieurl`, and `url` use strings for sticking clients to servers. These options are beneficial when the sticky table limit is too small for your application requirements because the string methods do not use the sticky table.

The syntax and options for this content mode command are:

- **advanced-balance arrowpoint-cookie** - Enables the content rule to stick a client to a server based on the unique service identifier information of the selected server in the arrowpoint cookie. Configure the service identifier by using the `string` command. For information on configuring the arrowpoint cookie, see the “Configuring an Arrowpoint Cookie” section later in this chapter. You can use this option with any Layer 5 content rule.

  **Note** If you are using the `arrowpoint-cookie` option of the `advanced-balance` command, do not configure string match criteria or use the `sticky-no-cookie-found-action` or `sticky-serverdown-failover` commands.
• **advanced-balance cookies** - Enables the content rule to stick a client to a 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. A content rule with a sticky configuration set to `advanced-balance cookies` requires all clients to enable cookies on their browser.

When a client makes an initial request, they do not have a cookie. But once they go to a server that is capable of setting cookies, they receive the cookie from the server. Each subsequent request contains the cookie until the cookie expires. A string in a cookie can be used to stick a client to a server. The service mode `string` command enables you to specify where the CSS should locate the string within the cookie.

The CSS processes the cookie using:

- An exact match that you set up when you configure the services.
- Data for a hash algorithm. For more information, see the “Comparing Hash Method with Match Method” section later in this chapter.

• **advanced-balance cookieurl** - Same as the `advanced-balance cookies` command, but if the CSS cannot find the cookie header in the HTTP packet, this type fails over to look up the URL extensions (that is, the portion after the “?” in the URL) based on the same string criteria. You must specify a port in the content rule to use this option. The CSS then spoofs the connection.

This option is useful if a Microsoft IIS web server is used with Cookie Munger, which dynamically places the session state information in the cookie header or URL extension, depending on whether the client can accept cookies.

Some client applications do not accept cookies. When a site depends upon the information in the cookie, administrators sometimes modify the server application so that it appends the cookie data to the parameters section of the URL. The parameters typically follow a “?” at the end of the main data section of the URL.

The `advanced-balance cookieurl` command sticks a client to a server based on locating the configured string in the:

- Cookie, if a cookie exists
- Parameters section of the URL, if no cookie exists

The string can either be an exact match or be hashed.
Specifying an Advanced Load-Balancing Method for Sticky Content

- **advanced-balance none** - Disables the advanced-balancing method for a content rule (default).
- **advanced-balance sticky-srcip** - 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, Layer 4, or Layer 5 content rules.
- **advanced-balance sticky-srcip-dstport** - 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 Layer 5 content rules.
- **advanced-balance sip-call-id** - Enables the content rule to stick a client to a server based on Session Initiation Protocol (SIP) Call-ID. The application type must be **sip** for the content rule and the protocol must be UDP. For more information about SIP, see the “Configuring Session Initiation Protocol Load Balancing” section.
- **advanced-balance ssl** - Enables the content rule to stick the client to the server based on the Secure Socket 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 then spoofs the connection.

Sites where encryption is required for security purposes often use SSL. SSL contains session IDs, and the CSS can use these session IDs to stick the client to a server. For the CSS to successfully provide SSL stickiness, the application must be using SSL version 3 session IDs. Sticky SSL uses the sticky table. If you are concerned about the number of concurrent sessions, and not concerned about security, you should consider using the **cookies**, **cookieurl**, or **url** options.

**Note**

Use the **ssl-l4-fallback disable** command when you want to disable the CSS from inserting the Layer 4 hash value, which is based on the source IP address and destination address pair, into the sticky table. This may be necessary in a lab environment when testing SSL with a small number of clients and servers, where some retransmissions might occur. In this case, you would not want to use the Layer 4 hash value because it will skew the test results. See the “Configuring SSL-Layer 4 Fallback” section later in this chapter for details.

Do not issue the **ssl-l4-fallback disable** command if SSL version 2 is in use on the network.
Specifying an Advanced Load-Balancing Method for Sticky Content

- **advanced-balance url** - 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 then spoofs the connection.

  The `advanced-balance url` command is similar to the `advanced-balance cookies` command. It can use either an exact match method or a hash algorithm. The string can exist anywhere in the URL.

- **advanced-balance wap-msisdn** - Enables a Layer 5 content rule to stick a client to a server based on the MSISDN header field in an HTTP request. MSISDN is the header field for wireless clients using the Wireless Application Protocol (WAP). The MSISDN field value can contain the client’s telephone number or user ID, which uniquely identifies the client. This command is especially useful for clients using e-commerce applications.

  **Note**  
  We recommend that you configure `advanced-balance wap-msisdn` only on a Layer 5 content rule (a rule configured with a URL statement).

If the MSISDN header is present in an HTTP request, the CSS generates a hash value (key) based on the value in the MSISDN header field. The CSS uses the key to look up an entry in the sticky table. If an entry exists in the sticky table, the CSS sends the client to the sticky server indicated by the table entry.

If an entry does not exist in the sticky table, the CSS:

a. Generates a new entry in the sticky table (similar to Layer 3, Layer 4, and SSL sticky)

b. Load balances the request to a server

c. Stores the selected server and the key (hashed value of the MSISDN header) in the sticky entry

The CSS looks up the same table entry and sends the client to the same server for subsequent requests from the same client.

If the MSISDN header field is not present in an HTTP request, the CSS load-balances the client request based on the configured load-balancing method. The default load-balancing method is roundrobin.
In the following example, TCP port 80 traffic destined for 192.168.128.151 is stuck to either server1 or server2 based on the contents of the MSISDN HTTP header field.

```
owner arrowpoint
  content ruleWapSticky
    vip address 192.168.128.151
    protocol tcp
    port 80
    url "/*"
    add service server1
    add service server2
    advanced-balance wap-msisdn
    active
```

For example, to specify `advanced-balance wap-msisdn` for content rule `rule1`, enter:

```
(config-owner-content[arrowpoint-rule1])# advanced-balance wap-msisdn
```

**Note**
You can use the `advanced-balance wap-msisdn` command alone or with the MSISDN header field type. For a configuration example using both, see the “Configuring Wireless Users for E-Commerce Applications” section later in this chapter.

To disable the advanced load-balancing method, enter:

```
(config-owner-content[arrowpoint-rule1])# advanced-balance none
```

### Configuring SSL-Layer 4 Fallback

Insertion of the Layer 4 hash value into the sticky table occurs when more than three frames are transmitted in either direction (client-to-server, server-to-client) or if SSL version 2 is in use on the network. If either condition occurs, the CSS inserts the Layer 4 hash value into the sticky table, overriding the further use of the SSL version 3 session ID. Use the `ssl-l4-fallback disable` command when you want to prevent the CSS from inserting the Layer 4 hash value, based on the source IP address and destination address pair, into the sticky table (the default CSS operation).
The `ssl-l4-fallback` command is applicable only when the `advanced-balance ssl` method is specified for a content rule, which forces the content rule to stick to a server based on SSL version 3 session ID. The use of the `ssl-l4-fallback` command may be necessary in a lab environment when testing SSL with a small number of clients and servers, where some retransmissions might occur. In this case, you would not want to use the Layer 4 hash value because it will skew the test results.

**Note**
The `ssl-l4-fallback` command is a global configuration mode command and affects all contents rules using the `advanced-balance ssl` method.

The options for this global configuration mode command include:

- **ssl-l4-fallback enable** - The CSS inserts the Layer 4 hash value into the sticky table (default setting).
- **ssl-l4-fallback disable** - The CSS does not insert the Layer 4 hash value into the sticky table and continues to look for SSL version 3 session IDs.

**Note**
Do not issue the `ssl-l4-fallback disable` command if SSL version 2 is in use on the network.

For example, to disable the CSS from inserting the Layer 4 hash value into the sticky table, enter:

```
(config)# ssl-l4-fallback disable
```

To reset the CSS back the default action of inserting a Layer 4 hash value into the sticky table, enter:

```
(config)# ssl-l4-fallback enable
```
Chapter 10 Configuring Sticky Parameters for Content Rules

Configuring Sticky Serverdown Failover

The sticky failover default method is for the CSS to use the configured load-balancing method. Use the `sticky-serverdown-failover` command to define what will happen if a sticky string is found but the associated service has failed or is suspended.

**Note**

If you are using the `arrowpoint-cookie` option of the `advanced-balance` command, do not configure string match criteria, the `sticky-no-cookie-found-action` command, or the `sticky-serverdown-failover` command.

The syntax and options for this content mode command are:

- **sticky-serverdown-failover balance** - Sets the failover method to use a service based on the configured load-balancing method.

- **sticky-serverdown-failover redirect** - Sets the failover method to use the redirect string configured on a content rule. This command supports a 252-character redirect string (URL). For more information on redirect strings, see the “Redirecting Requests for Content” section in Chapter 9, Configuring Content Rules. If you do not configure a redirect string on a content rule, the load-balancing method is used.

- **sticky-serverdown-failover reject** - Rejects the content request.

- **sticky-serverdown-failover sticky-srcip** - Sets the failover method to use a service based on the client source IP address.

- **sticky-serverdown-failover sticky-srcip-dstport** - Sets the failover method to use a service based on the client source IP address and the server destination port.

For example, to set the sticky failover method to `sticky-srcip`, enter:

```
(config-owner-content[arrowpoint-rule1])sticky-serverdown-failover
sticky-srcip
```

To set the sticky failover method to its default setting of using the configured load-balancing method, enter:

```
(config-owner-content[arrowpoint-rule1])# no
sticky-serverdown-failover
```
Configuring Sticky Mask

A client IP address uniquely identifies the client to the CSS. During normal client-server sessions, the IP address is maintained throughout the connection. However, if the connection is lost (for example, due to a dense proxy failover) and the client reconnects with a different IP address, the CSS needs to reconnect the client to the same server that is preserving the client information (for example, information from a shopping cart or financial session).

Use the `sticky-mask` command to mask a group of client IP addresses in order to preserve the client connection state when the client’s source IP address changes. The sticky mask specifies which portion of the client IP address the CSS will mask. The default sticky subnet mask is 255.255.255.255.

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# sticky-mask 255.255.255.0
```

To restore the sticky subnet mask to the default of 255.255.255.255, enter:

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

Configuring Sticky Inactive Timeout

By default, new sticky connection uses the oldest used sticky entry. A sticky association could exist for a time depending on the sticky traffic load on the CSS. Use the `sticky-inact-timeout` command to specify the inactivity timeout period on a sticky connection for a content rule before the CSS removes the sticky entry from the sticky table. When you configure this period, the CSS keeps the sticky entry in the sticky table for the specified amount of time. The CSS does not reuse this entry until the time expires. If the sticky table is full and none of the entries has expired, the CSS rejects the new sticky request. If the `sticky-inact-timeout` command is specified for a Layer 5 content rule using SSL sticky, the SSL sessions continue even if the sticky table is full; however, the CSS does not maintain stickiness on the new sessions.

When the sticky connection expires, the CSS uses the configured load-balancing method to choose an available server for the request.

When this feature is disabled, the new sticky connection uses the oldest used sticky entry. A sticky association could exist for a time depending on the sticky traffic load on the CSS.
The syntax for this command is:

```
sticky-inact-timeout minutes
```

Enter the number of minutes of inactivity, from 0 to 65535. The default value is 0, which means this feature is disabled. For example, enter:

```
(config-owner-content[arrowpoint-rule1])# sticky-inact-timeout 9
```

To disable the sticky connection inactivity timeout feature, enter:

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

## Configuring Sticky Content for SSL

To use stickiness based on SSL version 3 session ID, configure a specific SSL Layer 5 rule for a service. To configure an SSL Layer 5 rule for a service:

- Set the port to 443 using the \( \text{(config-owner-content) port} \) command.
- Enable the content rule to be sticky based on SSL using the \( \text{(config-owner-content) advanced-balance ssl} \) command.
- Specify the SSL application type using the \( \text{(config-owner-content) application ssl} \) command.

**Note**

We recommend that the \( \text{application ssl} \) command always be configured in conjunction with the \( \text{advanced-balance ssl} \) command. The \( \text{application ssl} \) command causes the CSS to spoof a connection so that you see the response come back from the server. The \( \text{advanced-balance ssl} \) command causes the CSS to look for the SSL session ID coming from the server and stick the client to the server based on that session ID. Once a flow is set up, the \( \text{application ssl} \) command then causes the CSS to treat the flow as a Layer 4 flow and does not inspect the flow for Layer 5 data in order to prevent the CSS from misinterpreting encrypted data.
For example, the following owner portion of a startup-config shows a content rule configured for SSL. Note that url “/*” command in this example is optional. The combination of the application ssl and advanced-balance ssl commands promotes the rule to Layer 5.

```
!*************************** OWNER ***************************!
owner arrowpoint
content L5sslsticky
vip address 192.3.6.58
add service server87
add service server88
balance aca
protocol tcp
port 443
url “/*”
advanced-balance ssl
application ssl
active
```

**Configuring String Range**

By specifying the starting and ending byte positions within a cookie, URL, or URL extension that the CSS uses to search for the specified string, the CSS processes the information located only within this range. This limits the amount of information that the CSS has to process when examining each cookie, URL, or URL extension, enhancing its performance. By default, the string range is the first 100 bytes of the cookie, URL, or parameters in the URL.

**Note**

If the starting position is beyond the cookie, URL, or URL extension, the CSS does not perform the string function. When the ending position is beyond the cookie, URL, or URL extension, the string processing stops at the end of the corresponding header.

Use the `string-range` command to specify the starting and ending byte positions within a cookie, URL, or URL extension that the CSS uses to search for the specified string. Enter the `start_byte` variable as the starting byte position of the cookie, URL, or URL extension after the header. Enter an integer from 1 to 1999. The default is 1. Ensure that the starting byte position is less than the end byte.
Specifying a String Operation

Enter the *end_byte* variable as the ending byte position of the cookie, URL, or URL extension. Enter an integer from 2 to 2000. The default is 100. Ensure that the ending byte position is more than the start byte position.

If you are using **advanced-balance**:

- **cookies** - The CSS starts counting after “Cookie: ” (including the space after the colon).
- **url** - The CSS starts counting after the “/”.
- **cookieurl** - The CSS starts counting after the “Cookie: “ string. If the CSS does not find “Cookie: “ in the HTTP request, it starts counting after the “?” in the URL of the same request.

For example, enter:

    (config-owner-content[arrowpoint-rule1])# string-range 35 to 55

To restore the string range to the default of 1 to 100, enter:

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

**Specifying a String Operation**

To determine the method to choose a destination server for a string result, use the **string operation** command. The CSS derives the string result from the settings of the **string** criteria commands within the string range. You can choose a server by using the configured balance method or by using the hash key generated by the specified sticky hash type. If the Web servers:

- Are only capable of accepting the cookies that they set, then you must use the exact match method
- Can accept any cookies that are set by either a cookie server or other servers, then you may use the hash method

**Note**

If you are using the **arrowpoint-cookie** option of the **advanced-balance** command, do not configure string match criteria, the **sticky-no-cookie-found-action** command, or the **sticky-serverdown-failover** command.
Comparing Hash Method with Match Method

When an application uses the exact match method, once a client makes a request to a particular server, the server is responsible for providing the client with a string unique to the server to use for future requests. Typically, if a server receives a string in a request that was set by another server, that string causes an error. In an exact match, the CSS looks for the unique string. If it finds an exact match, then the server is used. If no match is found, the CSS uses the configured load-balancing method to select a server for the client.

When an application uses one of the hash algorithms, all of the servers are capable of accepting any strings set by other servers. The model was designed so you could set up a site where the initial login would send a client to a Web server that assigns cookies to clients. When the CSS receives the first request from a client with the cookie string, it performs the hash operation on the string and chooses a server accordingly. The hash algorithm ensures that a particular string is always sent to a specific server, but it does not have to be a predefined server, as with an exact match.

Using the string operation hash algorithms may allow the Web server application to be used without being modified. When you use the string operation match-service-cookie method, you must modify the Web server application so that each server generates a unique string. The hash algorithms may be able to take advantage of strings already generated by the servers.

The syntax and options for this content mode command are:

- **string operation match-service-cookie** - Chooses a server by matching a service cookie in the sticky string. This is the default setting. When a match is not found, the CSS chooses the server by using the configured balance method (for example, roundrobin).

- **string operation [hash-a|hash-crc32|hash-xor]** - Chooses a server by using the hash key generated by the designated hash method. When using advanced balance cookies with a hash algorithm, all servers in the same domain must accept cookies regardless of which server created the cookie. This enables all servers configured on the Layer 5 rule to process cookies passed in an HTTP request.

The hash method keywords are:

- **hash-a** - Apply a basic hash algorithm on the hash string to generate the hash key
Chapter 10 Configuring Sticky Parameters for Content Rules

Specifying a String Operation

- **hash-crc32** - Apply the CRC32 algorithm on the hash string to generate a hash key
- **hash-xor** - Exclusive OR (XOR) each byte of the hash string to derive the final hash key

If the selected server is out of service, the CSS performs a rehash to choose another server.

TAC recommends using either **hash-xor** or **hash-crc32** depending on your string possibilities. If the strings are completely dissimilar, use **hash-xor**. If the strings are similar, use **hash-crc32**. For example, if your string values are abc1, abc2, and abc3, the **hash-xor** method cannot provide you with enough variance in the hash values (that is, abc1 and abc2 may end up on the same server because they may hash to the same value).

For example, to set the string operation to choose a server by using the string operation **hash-crc32** algorithm, enter:

```
(config-owner-content[arrowpoint-rule1])# string operation hash-crc32
```

To reset the string operation to its default setting of choosing a server by matching a service cookie in the sticky string, enter:

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

The CSS derives a string result from the following string criteria commands:

- string ascii-conversion
- string match
- string eos-char
- string prefix
- string process-length
- string skip-length
Enabling or Disabling String ASCII Conversion

By default, ASCII conversion of escaped special characters within the specified sticky string range before applying any processing to the string is enabled. Use the `string ascii-conversion` command to enable or disable the ASCII conversion.

For example, to disable ASCII conversion of escaped special characters, enter:

```
(config-owner-content[arrowpoint-rule1])# string ascii-conversion disable
```

To reenable the ASCII conversion of escaped special characters to its default setting, use the `no` form of the command or the `enable` option. For example, enter:

```
(config-owner-content[arrowpoint-rule1])# no string ascii-conversion
(config-owner-content[arrowpoint-rule1])# string ascii-conversion enable
```

Configuring the Handling of Multiple String Matches

By default, if the CSS determines that the incoming string matches multiple configured service strings, the CSS matches the most specific (longest) string. In the following example, the CSS service configuration is:

```
service s1
  string pear

service s2
  string grape

service s3
  string banana
```

If the incoming string is grapebananapear, the string match of the most specific string is banana.

Using the `string match` command, you can also configure the CSS to:

- Match the first string in the incoming string by using the `first-string-match` keyword. Enter:

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

  In the case of the previous example, the string match is grape.
Look at each service in the order of its index entry until there is a match by using the `first-service-match` keyword. Enter:

```
(config-owner-content[arrowpoint-rule1])# string match first-service-match
```

In the case of the previous example, the first-service-string match is pear.

**Note**

Use `string match` command with the `advanced-balance cookies/cookiesurl/url` command.

To reset the default behavior of matching the most specific string, enter:

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

### Specifying End-of-String Characters

To specify up to three ASCII characters as the delimiters for the sticky string within the string range, use the `string eos-char` command. For example, in a cookie header, a semicolon (:) is usually used as a delimiter; in a URL extension, an ampersand (&) is often used as a delimiter.

The CSS uses the `string eos-char` value if the `(config-owner-content) string process-length` command is not configured. The `(config-owner-content) string process-length` command has higher precedence. If neither command is configured, the CSS uses the maximum of 100 bytes for the final string length. Enter the sticky string end-of-string characters as a quoted text string with a maximum of three characters.

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# string eos-char ";;"
```

To clear the end of string characters, enter:

```
(config-owner-content[arrowpoint-rule1])# no string eos-char
```
Specifying a String Prefix

If you do not configure the string prefix, the string functions start from the beginning of the string range for the cookie, URL, or URL extension, depending on the sticky type. By default, the string range is the first 100 bytes of the cookie, URL, or parameters in the URL. If the string prefix is configured but is not found in the string range, the CSS uses the load-balancing method you defined in the `sticky-serverdown-failover` command.

Use the `string prefix` command to specify the string prefix located in the string range. Enter the string prefix as a quoted text string with a maximum of 30 characters. The default is no prefix (“”).

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# string prefix "UID=
```

To clear the string prefix, enter:

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

Specifying a String Process Length

To specify how many bytes, after the end of the prefix within the string range designated by the `string prefix` command and skipping the bytes designated by the `string skip-length` command, the string action will use, use the `string process-length` command. This command has higher precedence than the `string eos-char` command. If neither command is configured, the CSS uses the maximum of 100 bytes for the final string action. Enter the number of bytes from 0 to 252. The default is 0.

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# string process-length 16
```

To set the number of bytes to its default setting of 0, enter:

```
(config-owner-content[arrowpoint-rule1])# no string process-length
```
Specifying a String Skip Length

To specify how many bytes to skip after the end of prefix within the string range to find the string result, use the `string skip-length` command. Enter the number of bytes from 0 to 64. The default is 0. For example, enter:

```
(config-owner-content[arrowpoint-rule1])# string skip-length 3
```

To set the number of bytes to its default setting of 0, enter:

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

Configuring Sticky-No-Cookie-Found-Action

To specify the action the CSS should take for a sticky cookie content rule when it cannot locate the cookie header or the specified cookie string, use the `sticky-no-cookie-found-action` command.

**Note**

If you intend to use the `advanced-balance arrowpoint-cookie` command, do not configure the `sticky-no-cookie-found-action` command. They are not compatible.

The options for the `sticky-no-cookie-found-action` command are:

- `loadbalance` (default) - The CSS uses the configured balance method when no cookie is found in the client request.
- `redirect "URL"` - Redirects the client request to a specified URL string when no cookie is found in the client request. When using this option, you must also specify a redirect URL. Specify the redirect URL as a quoted text string from 0 to 252 characters.
- `reject` - Rejects the client request when no cookie is found in the request.
- `service name` - Sends the no cookie client request to the specified service when no cookie is found in the request.

For example, enter:

```
(config-owner-content[arrowpoint-rule1])#
sticky-no-cookie-found-action redirect
"http://www.lmi.com/nocookie.html"
```
To reset `sticky-no-cookie-found-action` to the default of `loadbalance`, enter:

```
(config-owner-content[arrowpoint-rule1])# no sticky-no-cookie-found-action
```

## Configuring Sticky Parameters for E-Commerce and Other Internet Applications

By configuring sticky parameters for e-commerce applications, you can instruct the CSS how to process client requests that do not contain cookies when the requests are destined to a content rule that is sticking based on a string within a cookie. You can also instruct the CSS how to process wireless users by integrating HTTP header load balancing with the `advanced-balance wap-msisdn` command. For applications that use the CSS sticky table, you can remove a sticky table entry after a defined period of activity.

The following sections describe how to configure sticky parameters for e-commerce and other Internet applications:

- Configuring an advanced-balance arrowpoint-cookie
- Configuring an Arrowpoint Cookie
- Configuring a Location Cookie
- Configuring Wireless Users for E-Commerce Applications
- Configuring Session Initiation Protocol Load Balancing

### Configuring an advanced-balance arrowpoint-cookie

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, use the `advanced-balance arrowpoint-cookie` command. Configure the service identifier by using the `(config-service) string` command.
Chapter 10 Configuring Sticky Parameters for Content Rules

Configuring Sticky Parameters for E-Commerce and Other Internet Applications

Configuring Sticky Parameters for E-Commerce and Other Internet Applications

Note

If you are using the arrowpoint-cookie option of the advanced-balance command, do not configure string match criteria, the sticky-no-cookie-found-action command, or the sticky-serverdown-failover command. They are not compatible with advanced-balance arrowpoint-cookie command.

You do not need to configure string match criteria. For information on configuring the arrowpoint-generated cookie, see the “Configuring an Arrowpoint Cookie” section. You can use this option with any Layer 5 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.

For example, to specify advanced-balance arrowpoint-cookie for content rule1, enter:

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

To disable the advanced load-balancing method, enter:

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

Configuring an Arrowpoint Cookie

The CSS generates the arrowpoint cookie transparently for a client, the client stores it and returns it in subsequent requests, and the CSS later uses it to maintain the client-server stickiness. This cookie contains the sticky information itself and does not refer to a sticky table.

Use the arrowpoint-cookie command to configure the arrowpoint cookie name, path, and expiration time. If you configure the arrowpoint-cookie method in a content rule, the CSS always checks for the existence of the arrowpoint cookie when it receives a client request. If this cookie does not exist, the CSS performs server load balancing and generates an arrowpoint cookie.
When the CSS finds the cookie in the client request, it unscrambles the cookie data and then validates it. Then, the CSS checks the cookie expiration time. If the cookie has expired, the CSS sends a new cookie containing the information about the server where the client was stuck. This appears as an uninterrupted connection.

If the cookie format is valid, the CSS ensures the consistency between the cookie and the CSS configuration. When all the validations are passed, the CSS forwards the client request to the server indicated by the server identifier. Otherwise, the CSS treats this request as an initial request.

The options for this content mode command are:

- **arrrowpoint-cookie name** - Specifies a unique cookie identifier
- **arrrowpoint-cookie path** - Sets the cookie path to a configured path
- **arrrowpoint-cookie expiration** - Sets an expiration time, which the CSS compares with the time associated with the cookie
- **arrrowpoint-cookie browser-expire** - Allows the browser to expire the cookie
- **arrrowpoint-cookie expire services** - Expires the service information when the cookie expires

### Configuring an Arrowpoint Cookie Name

To configure a unique cookie identifier with a maximum of four alphanumeric characters, use the `arrrowpoint-cookie name` command. With this option, you can configure multiple CSSs to inject cookies without the potential of one CSS overwriting another CSS cookie.

The syntax of this owner-content configuration mode command is:

```
arrrowpoint-cookie name name
```

Enter a unique string consisting of 1 to 31 alphanumeric characters for the `name` variable. The default is ARPT.

**Caution**

When you configure a new cookie name on a content rule, the CSS no longer recognizes any pre-existing cookie name configured on that rule. Therefore, any existing stickiness is lost.
Configuring Sticky Parameters for E-Commerce and Other Internet Applications

Chapter 10    Configuring Sticky Parameters for Content Rules

For example:

```
(config-owner-content[arrowpoint-rule1])# arrowpoint-cookie name abc5678
```

The cookie resulting from this command would appear as:
```
abc5678=000000SservicenameT0x_0xC1234
```

To reset the Arrowpoint cookie name to the default of ARPT, enter:

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

**Configuring an Arrowpoint Cookie Path**

By default, the CSS sets the default path attribute of the cookie to a slash (/). Use the `arrowpoint-cookie path` command to set the arrowpoint-cookie path to a configured path.

The syntax of this owner-content configuration mode command is:

```
arrowpoint-cookie path "path_name"
```

Enter the `path_name` where you want to send the cookie. Enter a quoted text string with a maximum of 99 characters. The default path of the cookie is “/”.

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# arrowpoint-cookie path "/cgi-bin/"
```

To reset the cookie path to its default of “/”, enter:

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

**Configuring an Arrowpoint-Cookie Expiration Time**

If the arrowpoint cookie has expired, the CSS sends a new cookie that includes the server where the client was stuck. The sending of the new cookie allows for the appearance of an uninterrupted connection. If you do not set an expiration time, the cookie expires when the client exits the browser.

Use the `arrowpoint-cookie expiration` command to set an expiration time, which the CSS compares with the time associated with the arrowpoint cookie. The syntax of this owner-content mode configuration command is:

```
arrowpoint-cookie expiration dd:hh:mm:ss
```
The variables are:

- **dd** - Number of days. Valid numbers are from 00 to 99.
- **hh** - Number of hours. Valid numbers are from 00 to 99.
- **mm** - Number of minutes. Valid numbers are from 00 to 99.
- **ss** - Number of seconds. Valid numbers are from 00 to 99.

**Note**

Do not use all zeros for days, hours, minutes, and seconds. This value is invalid.

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# arrowpoint-cookie expiration 08:04:03:06
```

To reset the expiration time to when the client exits the browser, enter:

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

### Configuring Arrowpoint-Cookie Browser Expire

To allow the browser to expire the arrowpoint cookie based on the expiration time, use the **arrowpoint-cookie browser-expire** command. To configure the expiration time, see the previous section. The syntax of this owner-content configuration mode command is:

```
arrowpoint-cookie browser-expire
```

For example, enter:

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

To allow the CSS to expire the cookie, enter:

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

**Note**

When the cookie expires, all sticky information is lost.
Configuring Arrowpoint-Cookie Expire Services

By default, when the arrowpoint cookie expires, the CSS sends a new cookie with the server information from the expired cookie. Use the `arrowpoint-cookie expire-services` command to expire service information when the cookie expires before sending a new cookie. The syntax of this owner-content configuration mode command is:

```
arrowpoint-cookie expire-services
```

For example, enter:

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

To reset the default behavior, enter:

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

Configuring an Arrowpoint Cookie Domain

Use the `cookie-domain` command to configure a domain name for the arrowpoint cookie. For details, see the “Configuring a Domain Name for the Location Cookie” section later in this chapter.

Configuring a Location Cookie

Occasionally, when a client is stuck to a particular CSS and server in a multisite network configuration, a subsequent DNS request may resolve to a different IP address, which sends the client to a different CSS and server. This different resolution can be a problem in an e-commerce application, especially when a client already has items in a shopping cart. To ensure that a CSS returns the client to the original CSS in a multisite environment, configure a Location Cookie.

**Note**

The CSS recognizes content requests that include a location cookie as part of a sticky session. Therefore, even if you add a service with a configured weight of zero as a location service to a content rule, the CSS continues to direct to that service any requests that contain location cookies originating from the service.
Overview

Location Cookie injects a user-defined NAME=VALUE cookie pair string (configured on the CSS specific to a site) into the response packet from the server. On subsequent connections after cookie injection, if a new DNS resolution sends the client to a different CSS, the new CSS attempts to match the NAME=VALUE pair to the values in a configured content rule. If the CSS cannot match the NAME=VALUE pair, the CSS compares the VALUE portion of the cookie to information in location services configured in a content rule. The CSS uses the information in the location services to return the client to the original site.

There are two methods that a CSS can use to return a client to the original site, depending on the type of location service that you configured on the CSS. The first method uses a standard service (pass-through service) to pass all traffic through the current CSS, then back to the original CSS. You must configure this same service as a destination service in a source group to ensure that all return traffic from the original CSS is NATed through the current CSS before going to the client.

The second method uses a redirect service to send a 302 redirect to the client, thereby forcing the client back to the original site. The 302 redirect uses any URL PUT and appends it with http:// unless you configure the service configuration mode no prepend http command. The URL points to a specific file or default file in a directory. Redirected services can also use the service configuration mode domain command to redirect from HTTP to HTTPS. Note that a 302 redirect changes all HTTP methods to HTTP GETs. For example, if a client is performing a POST operation and the current CSS redirects the client to the original site, the client will lose all data in the POST method.

Note

In addition to the IP address of the main site, each site must also have a unique DNS entry; for example, site1.work.com. These site-specific DNS entries guarantee that a client will receive a unique response from the DNS server for each site and prevent a client from being redirected to an IP address.
Location Cookie Quick Start

Table 10-2, Table 10-3, and Table 10-4 provide a quick overview of the steps required to configure the Location Cookie feature on each CSS in a multisite configuration. This section includes quick start procedures for three sample sites. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following the tables.

Table 10-2 Location Cookie Configuration Quick Start for Site1

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter global configuration mode.</td>
</tr>
<tr>
<td># config</td>
</tr>
<tr>
<td>(config)#</td>
</tr>
<tr>
<td>2. Configure local services as needed. For details on configuring services, see Chapter 3, Configuring Services. For example:</td>
</tr>
<tr>
<td>(config)# service localServ1</td>
</tr>
<tr>
<td>(config-service[localServ1])# ip address 192.168.2.3</td>
</tr>
<tr>
<td>(config-service[localServ1])# active</td>
</tr>
<tr>
<td>(config-service[localServ1])# service localServ2</td>
</tr>
<tr>
<td>(config-service[localServ2])# ip address 192.168.2.4</td>
</tr>
<tr>
<td>(config-service[localServ2])# active</td>
</tr>
<tr>
<td>3. Configure a redirect service that the CSS will use as a location service.</td>
</tr>
<tr>
<td>(config-service[localServ2])# service site2</td>
</tr>
<tr>
<td>(config-service[site2])# ip address 192.158.128.209</td>
</tr>
<tr>
<td>(config-service[site2])# string site2</td>
</tr>
<tr>
<td>(config-service[site2])# type redirect</td>
</tr>
<tr>
<td>(config-service[site2])# active</td>
</tr>
<tr>
<td>4. Configure a standard service that the CSS will use as a location service.</td>
</tr>
<tr>
<td>(config-service[site2])# service site3</td>
</tr>
<tr>
<td>(config-service[site3])# ip address 192.148.128.209</td>
</tr>
<tr>
<td>(config-service[site3])# string site3</td>
</tr>
<tr>
<td>(config-service[site3])# active</td>
</tr>
<tr>
<td>5. Configure an owner. For details on configuring an owner, see Chapter 8, Configuring Owners.</td>
</tr>
<tr>
<td>(config)# owner ArrowPoint</td>
</tr>
<tr>
<td>(config-owner[ArrowPoint])#</td>
</tr>
</tbody>
</table>
6. Create a content rule for the owner. For details on configuring a content rule, see Chapter 9, Configuring Content Rules.

```
(config-owner[ArrowPoint])# content locCookie
(config-owner-content[ArrowPoint-locCookie])#
```

7. Configure the following commands on the content rule.

```
(config-owner-content[ArrowPoint-locCookie])# vip address 192.168.128.209
(config-owner-content[ArrowPoint-locCookie])# add service localServ1
(config-owner-content[ArrowPoint-locCookie])# add service localServ2
(config-owner-content[ArrowPoint-locCookie])# protocol tcp
(config-owner-content[ArrowPoint-locCookie])# port 80
(config-owner-content[ArrowPoint-locCookie])# url "/**"
(config-owner-content[ArrowPoint-locCookie])# location-cookie
  name work value site1
  cookie-domain ".work.com"
(config-owner-content[ArrowPoint-locCookie])# add
  location-service site2
(config-owner-content[ArrowPoint-locCookie])# add
  location-service site3
(config-owner-content[ArrowPoint-locCookie])# active
```

8. Create a source group and add service site3 as a destination service. For details on configuring a source group, see Chapter 3, Configuring Services.

```
(config)# group site1
(config-group[site1])# add destination service site3
(config-group[site1])# vip address 192.168.128.210
(config-group[site1])# active
```

9. Use the `show rule sticky` command to verify the location cookie configuration.

```
# show rule sticky
```
The following running-config example shows the results of entering the commands described in Table 10-2 for site 1.

```
!************************** SERVICE **************************
service localServ1
  ip address 192.168.2.3
  active

service localServ2
  ip address 192.168.2.4
  active

service site2
  ip address 192.158.128.209
  string site2
  type redirect
  active

service site3
  ip address 192.148.128.209
  string site3
  active

!*************************** OWNER ***************************
owner ArrowPoint

  content locCookie
    vip address 192.168.128.209
    add service localServ1
    add service localServ2
    protocol tcp
    port 80
    url "/*"
    location-cookie name work value site1
    cookie-domain ".work.com"
    add location-service site2
    add location-service site3
    active

!*************************** GROUP ***************************
group site1
  add destination service site3
  vip address 192.168.128.210
  active
```
1. Enter global configuration mode.
   # config
   (config)#

2. Configure local services as needed. For example:

   (config)# service localServ1
   (config-service[localServ1])# ip address 192.158.2.3
   (config-service[localServ1])# active
   (config-service[localServ1])# service localServ2
   (config-service[localServ2])# ip address 192.158.2.4
   (config-service[localServ2])# active

3. Configure a standard service for each of the other sites that the CSS will use as location services.

   (config-service[localServ2])# service site1
   (config-service[site1])# ip address 192.168.128.209
   (config-service[site1])# string site1
   (config-service[site1])# active
   (config-service[site1])# service site3
   (config-service[site3])# ip address 192.148.128.209
   (config-service[site3])# string site3
   (config-service[site3])# active

4. Configure an owner.

   (config)# owner ArrowPoint
   (config-owner[ArrowPoint])#

5. Create a content rule for the owner.

   (config-owner[ArrowPoint])# content locCookie
   (config-owner-content[ArrowPoint-locCookie])#
Chapter 10  Configuring Sticky Parameters for Content Rules

Configuring Sticky Parameters for E-Commerce and Other Internet Applications

Table 10-3  Location Cookie Configuration Quick Start for Site2 (continued)

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. Configure the following commands on the content rule.</td>
</tr>
<tr>
<td>(config-owner-content[ArrowPoint-locCookie])# vip address 192.158.128.209</td>
</tr>
<tr>
<td>(config-owner-content[ArrowPoint-locCookie])# add service localServ1</td>
</tr>
<tr>
<td>(config-owner-content[ArrowPoint-locCookie])# add service localServ2</td>
</tr>
<tr>
<td>(config-owner-content[ArrowPoint-locCookie])# protocol tcp</td>
</tr>
<tr>
<td>(config-owner-content[ArrowPoint-locCookie])# port 80</td>
</tr>
<tr>
<td>(config-owner-content[ArrowPoint-locCookie])# url &quot;/**&quot;</td>
</tr>
<tr>
<td>(config-owner-content[ArrowPoint-locCookie])# location-cookie name work value site2 .work.com &quot; .work.com&quot;</td>
</tr>
<tr>
<td>(config-owner-content[ArrowPoint-locCookie])# cookie-domain</td>
</tr>
<tr>
<td>(config-owner-content[ArrowPoint-locCookie])# add location-service site1</td>
</tr>
<tr>
<td>(config-owner-content[ArrowPoint-locCookie])# add location-service site3</td>
</tr>
<tr>
<td>(config-owner-content[ArrowPoint-locCookie])# active</td>
</tr>
<tr>
<td>7. Create a source group and add service site1 and service site3 as destination services.</td>
</tr>
<tr>
<td>(config)# group site2</td>
</tr>
<tr>
<td>(config-group[site2])# add destination service site1</td>
</tr>
<tr>
<td>(config-group[site2])# add destination service site3</td>
</tr>
<tr>
<td>(config-group[site2])# vip address 192.158.128.210</td>
</tr>
<tr>
<td>(config-group[site2])# active</td>
</tr>
<tr>
<td>8. Use the show rule sticky command to verify the location cookie configuration.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td># show rule sticky</td>
</tr>
</tbody>
</table>
The following running-config example shows the results of entering the commands described in Table 10-3 for site 2.

```
!************************** SERVICE **************************
service localServ1
  ip address 192.158.2.3
  active

service localServ2
  ip address 192.158.2.4
  active

service site1
  ip address 192.168.128.209
  string site1
  active

service site3
  ip address 192.148.128.209
  string site3
  active

!*************************** OWNER ***************************
owner ArrowPoint
  content locCookie
    vip address 192.158.128.209
    add service localServ1
    add service localServ2
    protocol tcp
    port 80
    url "/*"
    location-cookie name work value site2
    cookie-domain ".work.com"
    add location-service site1
    add location-service site3
    active

!*************************** GROUP ***************************
group site2
  add destination service site1
  add destination service site3
  vip address 192.158.128.210
  active
```
Table 10-4 Location Cookie Configuration Quick Start for Site3

Task and Command Example

1. Enter global configuration mode.
   
   # config
   (config)#

2. Configure local services as needed. For example:
   
   (config)# service localServ1
   (config-service[localServ1])# ip address 192.148.2.3
   (config-service[localServ1])# active
   (config-service[localServ1])# service localServ2
   (config-service[localServ2])# ip address 192.148.2.4
   (config-service[localServ2])# active

3. Configure a standard service that the CSS will use as a location service.
   
   (config-service[localServ2])# service site1
   (config-service[site1])# ip address 192.168.128.209
   (config-service[site1])# string site1
   (config-service[site1])# active

4. Configure a redirect service that the CSS will use as a location service.
   
   (config-service[site1])# service site2
   (config-service[site2])# ip address 192.158.128.209
   (config-service[site2])# string site2
   (config-service[site2])# type redirect
   (config-service[site2])# active

5. Configure an owner.
   
   (config)# owner ArrowPoint
   (config-owner[ArrowPoint])#

6. Create a content rule for the owner.
   
   (config-owner[ArrowPoint])# content locCookie
   (config-owner-content[ArrowPoint-locCookie])#
7. Configure the following commands on the content rule.

```plaintext
(config-owner-content[ArrowPoint-locCookie])# vip address 192.148.128.209
(config-owner-content[ArrowPoint-locCookie])# add service localServ1
(config-owner-content[ArrowPoint-locCookie])# add service localServ2
(config-owner-content[ArrowPoint-locCookie])# protocol tcp
(config-owner-content[ArrowPoint-locCookie])# port 80
(config-owner-content[ArrowPoint-locCookie])# url "/*"
(config-owner-content[ArrowPoint-locCookie])# location-cookie name work value site3
(config-owner-content[ArrowPoint-locCookie])# cookie-domain ".work.com"
(config-owner-content[ArrowPoint-locCookie])# add location-service site1
(config-owner-content[ArrowPoint-locCookie])# add location-service site2
(config-owner-content[ArrowPoint-locCookie])# active
```

8. Create a source group and add service site1 as a destination service.

```plaintext
(config)# group site3
(config-group[site1])# add destination service site1
(config-group[site1])# vip address 192.148.128.210
(config-group[site1])# active
```

9. Use the `show rule sticky` command to verify the location cookie configuration.

```plaintext
# show rule sticky
```
The following running-config example shows the results of entering the commands described in Table 10-4 for site 3.

```
!************************** SERVICE **************************
service localServ1  
ip address 192.148.2.3  
active

service localServ2  
ip address 192.148.2.4  
active
service site1  
ip address 192.168.128.209  
string site1  
active

service site2  
ip address 192.158.128.209  
string site2  
type redirect  
active

!*************************** OWNER ***************************
owner ArrowPoint  
content locCookie  
vip address 192.148.128.209  
add service localServ1  
add service localServ2  
protocol tcp  
port 80  
url "/"  
location-cookie name work value site3  
cookie-domain ".work.com"  
add location-service site1  
add location-service site2  
active

!*************************** GROUP ***************************
group site3  
add destination service site1  
vip address 192.148.128.210  
active
```
Configuring the location-cookie Command

To configure the NAME=VALUE cookie string and expiration time for the local site, use the **location-cookie** command. A CSS injects this cookie string into responses from a server.

**Note**
Location cookie requires a Layer 5 (L5) content rule. At a minimum, you need to configure url “/*” on the rule, as shown in the “Location Cookie Quick Start” section.

The syntax for this owner-content configuration mode command is:

```plaintext
location-cookie name text value text [expiration dd:hh:mm:ss]
```

The options and variables for this command are:

- **name text** - The first part of the NAME=VALUE cookie string. Enter an unquoted text string from 1 to 31 characters.
- **value text** - The second part of the NAME=VALUE cookie string. Enter an unquoted text string from 1 to 31 characters.
- **expiration dd:hh:mm:ss** - (Optional) Expiration date and time of the Location Cookie. This value indicates to the client browser when the cookie will expire based on a relative time from cookie generation. Enter a date and time in the following format:
  - **dd** - Number of days. Valid numbers are from 00 to 99.
  - **hh** - Number of hours. Valid numbers are from 00 to 99.
  - **mm** - Number of minutes. Valid numbers are from 00 to 99.
  - **ss** - Number of seconds. Valid numbers are from 00 to 99.

For example:

```
(config-owner-content[ArrowPoint-rule1])# location-cookie name work value site1 00:02:30:00
```

To remove the location cookie, enter:

```
(config-owner-content[ArrowPoint-rule1])# no location-cookie name
```
Configuring Sticky Parameters for Content Rules

Chapter 10      Configuring Sticky Parameters for Content Rules

Configuring Sticky Parameters for E-Commerce and Other Internet Applications

Configuring 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. For example, if you specify a cookie domain name of .work.com, the browser will send back the Location Cookie to all sites that end with .work.com, including site1.work.com, site2.work.com, and site3.work.com.

Use the `cookie-domain` command to configure a domain name for the Location Cookie. The syntax for this owner-content configuration mode command is:

```
cookie-domain name
```

The `name` variable specifies the name of the domain for the Location Cookie. Enter a quoted text string from 1 to 64 characters.

For example:

```
(config-owner-content[ArrowPoint-rule1])# cookie-domain "work.com"
```

To remove the cookie domain name, enter:

```
(config-owner-content[ArrowPoint-rule1])# no cookie-domain
```

Configuring Location Services

To add services to the content rule that the CSS uses to locate the site where the client was originally stuck, use the `add location-service` command.

The syntax for this owner-content configuration mode command is:

```
add location-service service_name
```

The `name` variable specifies the name of a standard service or redirect service that you want to add to the content rule for Location Cookie matching. Enter a name from 1 to 31 characters.

For example, enter:

```
(config-owner-content[ArrowPoint-rule1])# add location-service sitel
```

To remove the sitel location service, enter:

```
(config-owner-content[ArrowPoint-rule1])# remove location-service sitel
```
You can configure a maximum of 10 location services; either standard services or redirect services. These services do not count toward the 64-service maximum per content rule and do not participate in the load-balancing algorithm of the content rule.

**Note**

You cannot add a location service to a content rule if another location service on the rule is configured with the same cookie string.

A redirect service configured as a location service must have an IP address that is the same as the VIP address of the location cookie content rule configured on the redirected site. You must also define a string on any service used as a location service. This string must match the VALUE portion of the Location Cookie. For information on configuring a string on a service, see the “Configuring an Advanced Load-Balancing String” section.

Also, you must configure as a destination service in a source group any standard service that you configure as a location service in the content rule, as shown in the “Location Cookie Quick Start” section.

### Examples of Location Cookie Flow

The following examples illustrate the two mechanisms that a CSS uses to return a client to the CSS and server that serviced the original request. See the “Location Cookie Quick Start” section for specific CSS site configuration information.

**Example 1 - Returning a client to the original site using a location service (pass-through method)**

1. The client sends a lookup request for my.work.com to the DNS server.
2. The DNS server responds with the CSS Site1 VIP address of 192.168.128.209.
3. The client sends a GET request to 192.168.128.209 with no cookie.
4. The CSS injects the location cookie work=site1 to the response from the local server. Client-server interaction proceeds normally.

5. At some point in the future, the client sends another lookup request for my.work.com to the DNS server.
6. This time the DNS server responds with the CSS Site2 VIP address of 192.158.128.209.

7. The client sends a GET request to 192.158.128.209 (CSS Site2) with a location cookie of work=site1. This cookie string matches the configured location cookie name (work), but not the cookie value (site2). CSS Site2 searches through the list of location services checking the configured strings against the value in the cookie. In this case, a match is made on service site1.
8. Service site1 is configured as a destination service on the source group site2. This service matches on the locCookie content rule configured on CSS Site2. CSS Site2 forwards packets from the client to CSS Site1.

9. Because the client already sent a cookie with the GET request in Step 7, there is no need to inject another cookie. Content rule processing continues with no changes. The server response goes back through CSS Site1 to CSS Site2 to the client. The client is now stuck to the original site (site1.work.com) through CSS Site2.

Example 2 - Returning a client to the original site using a redirect service

1. The client sends a lookup request for my.work.com to the DNS server.

2. The DNS server responds with the CSS Site2 VIP address of 192.158.128.209.

3. The client sends a GET request to 192.158.128.209 (Site2) with no cookie.

4. The CSS injects the location cookie work=site2 to the response from the local server. Client-server interaction proceeds normally.
5. At some point in the future, the client sends another lookup request for my.work.com to the DNS server.

6. This time, the DNS server responds with the CSS Site3 VIP address of 192.148.128.209.

7. The client sends a GET request to 192.148.128.209 (CSS Site3) with a location cookie of work=site2. This cookie string matches the configured location cookie name (work) on CSS Site3, but not the cookie value (site3). CSS Site3 searches through the list of location services checking the configured strings against the value in the cookie. In this case, a match is made on service site2.

8. Service site2 is a redirect service, so CSS Site3 sends the client a 302 redirect to site2.work.com.
9. The client sends a lookup request for site2.work.com to the DNS server.

10. The DNS server responds with the CSS Site2 VIP address of 192.158.128.209.

11. The client sends a GET request to 192.158.128.209 (CSS Site2) with the cookie work=site2.

12. The cookie name and value match on the locCookie content rule configured on CSS Site2. Because the client already sent a cookie with the GET request in Step 11, there is no need to inject another cookie. Content rule processing continues with no changes. The client is now permanently stuck to the original site (site2.work.com).
Chapter 10      Configuring Sticky Parameters for Content Rules

Configuring Sticky Parameters for E-Commerce and Other Internet Applications

Displaying Location Cookie Information

To display location cookie information, use the `show rule sticky` command. For details, see the “Showing Sticky Attributes” section.

Configuring Wireless Users for E-Commerce Applications

Wireless clients use the Wireless Application Protocol (WAP) to access Internet content. When a wireless client sends a request for content, the WAP protocol gateway (a device that translates requests from the WAP protocol stack to the WWW protocol stack) generates the MSISDN field and adds it to the HTTP header.

Use the `advanced-balance wap-msisdn` command with the MSISDN header field to configure wireless users for e-commerce applications. For details on the `advanced-balance wap-msisdn` command, see the “Specifying an Advanced Load-Balancing Method for Sticky Content” section earlier in this chapter. For details on the MSISDN header field, see the “Configuring a Header-Field Entry” section in Chapter 11, Configuring HTTP Header Load Balancing.

In the following example, TCP port 80 traffic destined for VIP 192.168.128.151 that contains the string “012” in the MSISDN HTTP header field will match content rule rule012. The CSS will stick this traffic to either server1 or server2 based on the entire contents of the MSISDN field.

TCP port 80 traffic destined for 192.168.128.151 that does not contain the string “012” in the MSISDN HTTP header field, but has the field in the header, will match content rule ruleNo012. The CSS will use roundrobin to load balance the traffic across server21 and server22.

TCP port 80 traffic destined for 192.168.128.151 that does not contain the MSISDN HTTP header field will match content rule ruleNoWap. The CSS will use roundrobin to load balance the traffic across server31 and server32.

```
header-field-group wap012
  header-field 1 wap-msisdn contain "012"

header-field-group wapNo012
  header-field 1 wap-msisdn not-contain "012"
```
Configuring Session Initiation Protocol Load Balancing

Session Initiation Protocol (SIP) is an application-layer control protocol that functions as a signaling mechanism between user devices and media servers. SIP is a peer-to-peer protocol where end-devices (the User Agent Clients) initiate interactive communications sessions with SIP servers. These sessions can include Internet multimedia conferences, Internet telephone calls (voice-over-IP), and multimedia distribution. Examples of client devices include hardware, software, handheld IP telephones, and personal digital assistants (PDAs).

The session Call-ID is a unique call identifier that is contained in the SIP messages sent from the client to the SIP proxy server. Stickiness by Call-ID is particularly important for call stateful services that use the Call-ID to identify current SIP sessions and make decisions based on the content of the message.
If the CSS finds the SIP Call-ID in the SIP messages sent from the client to the SIP server, the CSS generates a key (hash value) based on the SIP Call-ID. The CSS uses the key to look up an entry in the sticky table. If the entry exists, the CSS sends the client to the sticky server indicated by the table entry. If the entry does not exist, the CSS creates a new sticky entry, hashes the SIP Call-ID value into a key, and saves the key in the entry.

The CSS supports the following SIP methods:

- **INVITE** - Indicates that the user or service is being invited to participate in a SIP session
- **ACK** - Confirms that the client has received a final response to an INVITE request
- **OPTIONS** - The server is being queried about its capabilities
- **BYE** - Indicates to the server that the client wants to release the call
- **CANCEL** - Cancels a pending request with the same Call-ID, To, From, and Cseq header field values, but does not affect a completed request
- **REGISTER** - Registers the address listed in the To header field with a SIP server

**Configuration Requirements and Restrictions**

The following requirements and restrictions apply to SIP load-balancing configurations on a CSS:

- The CSS supports SIP over UDP only.
- If you want UDP responses from the SIP proxy server to return to the client through the CSS and be NATed, configure destination services in the source group. Destination services NAT the client IP address to the CSS VIP, forcing return packets from the server to flow through the CSS and back to the client.
- When you enter the `application sip` content configuration mode command, the CSS automatically configures the SIP port as 5060 and the protocol as UDP. If you remove the SIP port from your configuration, the activation of the SIP content rule fails and the CSS sends an error message explaining the reason for the failure. To ensure that the port is configured, enter the `show rule` command.
- The `application sip` command is not compatible with the `url` command and the `url`, `urlhash`, `domain`, and `domainhash` load-balancing methods.
By default, the CSS sets up flows for SIP. You can disable SIP flows using the `flow-state 5060 udp flow-disable nat-enable` global configuration mode command. Doing so prohibits the concept of different call-IDs in the same flow. Stickiness behaves the same as in a flow-enabled configuration. To restore SIP flows, enter the `no flow-state 5060 udp` command.

### SIP Load Balancing Configuration Quick Start

Table 10-5 provides a quick overview of the steps required to configure the SIP load balancing feature on each CSS in a multisite configuration. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following the table.

Table 10-5  **SIP Configuration Quick Start**

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter global configuration mode.</td>
</tr>
<tr>
<td># config</td>
</tr>
<tr>
<td>(config)#</td>
</tr>
<tr>
<td>2. Configure a service for the SIP proxy server. For details on configuring services, see Chapter 3, Configuring Services. For example:</td>
</tr>
<tr>
<td>(config)# service sipServer</td>
</tr>
<tr>
<td>(config-service[sipServer])# ip address 192.168.2.3</td>
</tr>
<tr>
<td>(config-service[sipServer])# active</td>
</tr>
<tr>
<td>3. Configure an owner. For details on configuring an owner, see Chapter 8, Configuring Owners.</td>
</tr>
<tr>
<td>(config)# owner sipOwner</td>
</tr>
<tr>
<td>(config-owner[sipOwner])#</td>
</tr>
<tr>
<td>4. Create a SIP content rule for the owner. For details on configuring a content rule, see Chapter 9, Configuring Content Rules.</td>
</tr>
<tr>
<td>(config-owner[sipOwner])# content sipRule</td>
</tr>
<tr>
<td>(config-owner-content[sipOwner-sipRule])#</td>
</tr>
</tbody>
</table>
5. Configure the following commands on the content rule. For information about the `application sip` command, see the “Specifying an Application Type” section in Chapter 9, Configuring Content Rules. For information about the `advanced-balance sip-call-id` command, see the “Specifying an Advanced Load-Balancing Method for Sticky Content” section earlier in this chapter.

Note: When you enter the `application sip` command, the CSS automatically configures the protocol as UDP and the port number as 5060 if you have not already configured a protocol and a port.

```
{config-owner-content[sipOwner-sipRule]}# vip address 192.168.128.191
{config-owner-content[sipOwner-sipRule]}# application sip
{config-owner-content[sipOwner-sipRule]}# advanced-balance sip-call-id
{config-owner-content[sipOwner-sipRule]}# add service sipServer
{config-owner-content[sipOwner-sipRule]}# active
```

6. If your application requires that the client receive the server response from the VIP (CSS), then you need to configure a source group. In this case, source groups work only if the server is configured to use the “received=” field. Otherwise, you can skip this step. For details on configuring a source group, see Chapter 5, Configuring Source Groups for Services.

```
(config)# group sipGroup
{config-group[sipGroup]}# vip address 192.168.1.228
{config-group[sipGroup]}# add destination service sipServer
{config-group[sipGroup]}# active
```

7. (Recommended) Use the `show rule sticky`, `show sticky-table all-sticky`, `show sticky-table call-id-sticky`, and the `show sticky-stats` commands to verify your SIP configuration.

```
# show rule sticky
# show sticky-table all-sticky
# show sticky-table call-id-sticky
# show sticky-stats
```
The running-config example below shows the results of entering the commands described in Table 10-5.

```
!************************** CIRCUIT **************************
circuit VLAN1
  ip address 192.168.1.191 255.255.255.0
  ip address 192.168.2.191 255.255.254.0
!************************** SERVICE **************************
service sipServer
  ip address 192.168.2.3
  active

!*************************** OWNER ***************************
owner sipOwner
  content sipRule
    vip address 192.168.128.191
    protocol udp
    port 5060
    application sip
    advanced-balance sip-call-id
    add service sipServer
    active

!*************************** GROUP ***************************
group sipGroup
  vip address 192.168.1.228
  add destination service sipServer
  active
```

**Showing Sticky Attributes**

To display the sticky attributes for a content rule, use the `show rule` command with the `sticky` option. The syntax for the `show rule` command is:

```
show rule {owner_name {content_rule_name
  {acl|all|dns|header-field|hot-list|services|statistics|sticky}}} }
```

This command is available in SuperUser, User, global configuration mode, or owner configuration mode.

For example, enter:

```
(config)# show rule sipOwner sipRule sticky
```
For example, in content configuration mode, enter:

```
(config-owner-content[sipOwner-sipRule])# show rule sticky
```

Table 10-6 describes the fields in the `show rule sticky` command output.

**Table 10-6  Field Descriptions for the show rule sticky Command Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance</td>
<td>The load-balancing algorithm for the content rule.</td>
</tr>
<tr>
<td></td>
<td>The possible values are:</td>
</tr>
<tr>
<td></td>
<td>• ACA - Arrowpoint Content Awareness algorithm. The CSS correlates content request</td>
</tr>
<tr>
<td></td>
<td>frequency with the server’s cache sizes to improve cache hit rates for that server.</td>
</tr>
<tr>
<td></td>
<td>• destip - Destination IP address division. The CSS directs all client requests with the same destination IP address to the same service.</td>
</tr>
<tr>
<td></td>
<td>• domain - Domain name division. The CSS uses the domain name in the request URI to direct the client request to the appropriate service.</td>
</tr>
<tr>
<td></td>
<td>• domainhash - 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.</td>
</tr>
</tbody>
</table>
Chapter 10  Configuring Sticky Parameters for Content Rules

Showing Sticky Attributes

Table 10-6  Field Descriptions for the show rule sticky Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance (continued)</td>
<td>• <strong>leastconn</strong> - Least connections. The CSS chooses a running service that has the least number of connections.</td>
</tr>
<tr>
<td></td>
<td>• <strong>roundrobin</strong> - Roundrobin algorithm (default).</td>
</tr>
<tr>
<td></td>
<td>• <strong>srcip</strong> - Source IP address division. The CSS directs all client requests with the same source IP address to the same service.</td>
</tr>
<tr>
<td></td>
<td>• <strong>url</strong> - URL division. The CSS uses the URL (omitting the leading slash) in the redirect URL to direct the client requests to the appropriate service.</td>
</tr>
<tr>
<td></td>
<td>• <strong>urlhash</strong> - Internal CSS hash algorithm based on the URL string. The CSS uses the algorithm to hash the entire URL string. The CSS uses the hash result to choose the server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>weightedrr</strong> - Weighted roundrobin algorithm. The CSS uses the roundrobin algorithm but weighs some services more heavily than others, depending on the weight configured on the service. You can configure or change the weight of a service when you add it to the rule. The content rule-configured service weight overrides the service-configured weight only for that content rule.</td>
</tr>
</tbody>
</table>
The advanced load-balancing method for the content rule, including stickiness. The possible values are:

- **arrowpoint-cookie** - Enables the content rule to stick the client to the server based on the unique service identifier information of the selected server in the cookie.

- **cookies** - 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.

### Table 10-6 Field Descriptions for the show rule sticky Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Balance</td>
<td>The advanced load-balancing method for the content rule, including stickiness. The possible values are:</td>
</tr>
<tr>
<td></td>
<td>• <strong>arrowpoint-cookie</strong> - Enables the content rule to stick the client to the server based on the unique service identifier information of the selected server in the cookie.</td>
</tr>
<tr>
<td></td>
<td>• <strong>cookies</strong> - 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.</td>
</tr>
</tbody>
</table>
### Table 10-6  Field Descriptions for the show rule sticky Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advanced Balance</strong></td>
<td><strong>cookieurl</strong> - This is the same as the <code>advanced-balance cookies</code> option, 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. You can use this option with any Layer 5 HTTP content rule.</td>
</tr>
<tr>
<td></td>
<td><strong>none</strong> - Disables the advanced-balancing method for the rule. This is the default setting.</td>
</tr>
<tr>
<td></td>
<td><strong>sip call-ID</strong> - Enables the content rule to stick a client to a server based on Session Initiation Protocol (SIP) session Call-ID. The application type must be SIP for the content rule and the protocol must be UDP.</td>
</tr>
<tr>
<td></td>
<td><strong>sticky-srcip</strong> - 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.</td>
</tr>
<tr>
<td></td>
<td><strong>sticky-srcip-dstport</strong> - 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.</td>
</tr>
<tr>
<td></td>
<td><strong>ssl</strong> - Enables the content rule to stick the client to the server based on the Secure Socket 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 then spoofs the connection.</td>
</tr>
</tbody>
</table>
url - 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 then spoofs the connection.

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 - 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:

- **loadbalance** - The CSS uses the configured balanced method when no cookie is found in the client request. This is the default setting.
- **redirect “URL”** - The CSS redirects the client request to a specified URL string when no cookie is 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.
- **reject** - The CSS rejects the client request when no cookie is found in the request.
- **service name** - The CSS sends the no cookie client request to the specified service when no cookie is found in the request.

### Table 10-6  Field Descriptions for the show rule sticky Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Balance (continued)</td>
<td>- url: 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 then spoofs the connection.</td>
</tr>
<tr>
<td>Sticky Mask</td>
<td>The subnet mask used for stickiness. The default is 255.255.255.255.</td>
</tr>
<tr>
<td>Sticky Inactivity timeout</td>
<td>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.</td>
</tr>
<tr>
<td>Sticky No Cookie Found Action</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>- loadbalance: The CSS uses the configured balanced method when no cookie is found in the client request. This is the default setting.</td>
</tr>
<tr>
<td></td>
<td>- redirect “URL”: The CSS redirects the client request to a specified URL string when no cookie is 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.</td>
</tr>
<tr>
<td></td>
<td>- reject: The CSS rejects the client request when no cookie is found in the request.</td>
</tr>
<tr>
<td></td>
<td>- service name: The CSS sends the no cookie client request to the specified service when no cookie is found in the request.</td>
</tr>
</tbody>
</table>
Table 10-6  Field Descriptions for the show rule sticky Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sticky Server Down Failover</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Balance</strong> - The failover method uses a service based on the configured load balancing method (default).</td>
</tr>
<tr>
<td></td>
<td>- <strong>Redirect</strong> - 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.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Reject</strong> - The failover method rejects the content request.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Sticky-srcip</strong> - The failover method uses a service based on the client IP address. This is dependent on the sticky configuration.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Sticky-srcip-dstport</strong> - The failover method uses a service based on the client IP address and the server destination port. This is dependent on the sticky configuration.</td>
</tr>
<tr>
<td>ArrowPoint Cookie Path</td>
<td>The pathname where you want to send the cookie. The default path of the cookie is “/”.</td>
</tr>
<tr>
<td>ArrowPoint Cookie Expiration</td>
<td>The expiration time that the CSS compares with the time associated with the cookie. If you do not set an expiration time, the cookie expires when the client exits the browser.</td>
</tr>
<tr>
<td>ArrowPoint Cookie CSS/Browser Expired</td>
<td>Indicates whether the <code>arrowpoint-cookie browser-expire</code> command is enabled to allow the browser to expire the cookie based on the expiration time. If the command is enabled, the field displays “Browser” in place of “CSS.” The default is “CSS.”</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ArrowPoint Cookie Service</td>
<td>Specifies whether the <code>arrowpoint-cookie expire-services</code> command has been entered 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.</td>
</tr>
<tr>
<td>ArrowPoint Cookie Advanced</td>
<td>Specifies whether the <code>advanced-balance arrowpoint-cookie</code> command has been entered 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 cookie.</td>
</tr>
<tr>
<td>ArrowPoint Cookie Format</td>
<td>Specifies the format of the arrowpoint-cookie expiration time, whether the RFC 2822-compliant format is enabled or disabled. The <code>arrowpoint-cookie rfc2822-compliant</code> command configures the arrowpoint-cookie 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”).</td>
</tr>
</tbody>
</table>

Table 10-6  Field Descriptions for the show rule sticky Command Output (continued)
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: 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.
  - The range is from 1 to 1999. The default starting byte position is 1.
  - The range is from 2 to 2000. The default ending byte position is 100.

- 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 as the delimiters for the sticky string.

- String Ascii-Conversion: Specifies whether to enable or disable the ASCII conversion of Escape-sequence special characters within the specified sticky range before applying any processing to the string. By default, ASCII 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>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Match Criteria</td>
<td>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.</td>
</tr>
</tbody>
</table>
| String Range           | 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.  
  - The range is from 1 to 1999. The default starting byte position is 1.  
  - The range is from 2 to 2000. The default ending byte position is 100. |
| 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 as the delimiters for the sticky string.                                                                                   |
| String Ascii-Conversion| Specifies whether to enable or disable the ASCII conversion of Escape-sequence special characters within the specified sticky range before applying any processing to the string. By default, ASCII 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 10-6  Field Descriptions for the show rule sticky Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Process-Len</td>
<td>The number of bytes, after the end of the prefix designated by the string prefix command and skipping the bytes designated by the string skip-length command, that the string operation will use. The range is from 0 to 64. The default is 0.</td>
</tr>
<tr>
<td>String Operation</td>
<td>The method to choose a destination server for a string result as derived from the settings of the string criteria commands. The possible values are:</td>
</tr>
<tr>
<td></td>
<td>• match-service-cookie - 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.</td>
</tr>
<tr>
<td></td>
<td>• hash-a - Apply a basic hash algorithm on the hash string to generate the hash key.</td>
</tr>
<tr>
<td></td>
<td>• hash-crc32 - Apply the CRC32 algorithm on the hash string to generate a hash key.</td>
</tr>
<tr>
<td></td>
<td>• hash-xor - Exclusive OR (XOR) each byte of the hash string to derive the final hash key.</td>
</tr>
<tr>
<td>Location-Cookie</td>
<td>The format (NAME=VALUE) of the location cookie string.</td>
</tr>
<tr>
<td>Location-Cookie</td>
<td>The expiration date and time (dd:hh:mm:ss) of the location cookie. This value tells the client browser the time the cookie expires.</td>
</tr>
<tr>
<td>Expiration</td>
<td>A domain name for the location cookie (for example, .site.com). The cookie domain name allows your browser to send the cookie back to any site that ends with the domain name that you specify.</td>
</tr>
</tbody>
</table>
The `show sticky-table` command displays the contents of the CSS sticky table based on the advanced load-balancing method for a content rule. Use the following `show sticky-table` commands from any mode to display the sticky information contained in the CSS sticky table:

- `show sticky-table all-sticky` - Displays all Layer 3, Layer 4, SIP Call-ID, SSL, and WAP MSISDN sticky entries
- `show sticky-table l3-sticky` - Displays the Layer 3 entries
- `show sticky-table l4-sticky` - Displays the Layer 4 entries
- `show sticky-table sip-callid-sticky` - Displays the SIP Call-ID entries
- `show sticky-table ssl-sticky` - Displays the SSL entries
- `show sticky-table wap-sticky` - Displays the WAP MSISDN entries

To display sticky configurations for content, use the `show rule sticky` command in content mode. For details on the `show rule sticky` command, see the “Showing Sticky Attributes” section.

To display all sticky entries contained in the CSS sticky table, enter:

```
(config)# show sticky-table all-sticky
```

Table 10-7 describes all of the fields in the `show sticky-table all-sticky` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Sticky List on Slot n, Subslot n</td>
<td>Identifies the slot and subslot numbers of the SP in the CSS. If there are multiple SPs, the sticky table information is divided by slot number.</td>
</tr>
<tr>
<td>Entries for Page</td>
<td>Indicates the number of the page of information from the sticky table. The show screen displays a maximum of 100 sticky table entries for a page. The default is page 1.</td>
</tr>
<tr>
<td>Entry Number</td>
<td>The row number of the entry displayed from the sticky table.</td>
</tr>
</tbody>
</table>
### Table 10-7 Field Descriptions for the show sticky-table all-sticky Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Hash Value  | A key generated by the CSS for the different sticky data types. The hash value is a representation of the client-specific information inserted into the sticky table, functioning as an index or entry key into the sticky table. The CSS generates the following hash values for the various sticky data types:  
  - Layer 3 - Source IP address  
  - Layer 4 - Combination of source IP address and destination port  
  - SIP - Session Initiation Protocol (SIP) Call-ID (CID)  
  - SSL - SSL version 3.0 session ID (SID)  
  - WAP - MSISDN header field                                                                                                                                                           |
| Rule Index  | A CSS-assigned unique numeric index for the rule. This is the index displayed using the `show rule summary` command.                         |
| Rule State  | The state of the rule: ACT (active) or SUSP (suspended).                                                                                      |
| Srv Index   | The CSS-assigned unique numeric index for the service. This is the index displayed in the `show service summary` command.                    |
| Srv State   | The state of the service. The State field displays the service as Alive, Dying, Down, or Suspended.                                           |
| Time (Sec) Elapsed | Indicates the elapsed time since the entry in the sticky table was last referenced and has been idle since that point in time. The counter starts at 0 and increments until the sticky table entry is used again. |
| Hit Cnt     | The number of times the CSS received a transaction from the client for the entry in the CSS sticky table.                                      |
### Table 10-7  Field Descriptions for the show sticky-table all-sticky Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Col Cnt</td>
<td>The number of times the CSS received a transaction from different clients for an entry in the CSS sticky table with the same hash value. This field is used only with the <code>show sticky-table ssl-sticky</code> command.</td>
</tr>
<tr>
<td>Elem Type</td>
<td>The sticky type associated with the content rule. The possible element types include:</td>
</tr>
<tr>
<td></td>
<td>• Layer 3</td>
</tr>
<tr>
<td></td>
<td>• Layer 4</td>
</tr>
<tr>
<td></td>
<td>• SIP - Session Initiation Protocol (SIP) Call-ID (CID)</td>
</tr>
<tr>
<td></td>
<td>• SSL - SSL version 3.0 session ID (SID)</td>
</tr>
<tr>
<td></td>
<td>• WAP - MSISDN header field</td>
</tr>
<tr>
<td>Inact Cfg (Min)</td>
<td>The inactivity timeout period configured for the content rule associated with the entry. This field indicates the length of idle time the sticky entry is held in the sticky table. A value of 0 (the default, which means the feature is disabled) indicates that the entry is not timed out of the sticky table. The CSS removes the entry from the sticky table if it is the least used entry in the table, the sticky table becomes full, and a new entry needs to be added to the table.</td>
</tr>
<tr>
<td>Total Number of Entries Found</td>
<td>The total number of the queried entries found in the sticky table. This total value can also be based on a specific sticky data type (Layer 3, Layer 4, SIP, SSL, or WAP).</td>
</tr>
</tbody>
</table>
Chapter 10 Configuring Sticky Parameters for Content Rules

Showing Sticky Table Configurations

Showing Layer 3 Sticky Table Information

Use the `show sticky-table l3-sticky` command to display the Layer 3 sticky entries contained in the CSS sticky table. Layer 3 sticks a user to a server based on the source IP address.

The syntax for this global configuration mode command is:

```
show sticky-table l3-sticky [page {value}]|ipaddress {ip_address sticky_mask}
```

The `show sticky-table l3-sticky` command supports the following options and variables:

- **page value** - Shows Layer 3 sticky entries for a specific page in the sticky table, at 100 entries per page. Enter a value from 1 to 5000 to select the page of entries you want to view from the sticky table. To determine the page you want to display, take the Total Number of Used Entries Found value in the `show sticky-stats` command output and divide by 100 (entries per page).

- **ipaddress ip_address sticky_mask** - The IP address of the Layer 3 sticky table entry to be shown. Enter the IP address in dotted-decimal notation (for example, 192.168.2.5). Specify the sticky mask from the content rule for this IP address in dotted-decimal notation (for example, 255.255.255.0). The default sticky mask of a content rule is 255.255.255.255.

For example, to display Layer 3 sticky entries from page 60 in the sticky table, enter:

```
(config)# show sticky-table l3-sticky page 60
```

For example, to display Layer 3 sticky entries from a specific IP address and sticky mask in the sticky table, enter:

```
(config)# show sticky-table l3-sticky ipaddress 192.168.2.5
  255.255.255.255
```

See Table 10-7 for a description of the fields in the `show sticky-table l3-sticky` command output.
Showing Layer 4 Sticky Table Information

Use the `show sticky-table l4-sticky` command to display the Layer 4 sticky entries contained in the CSS sticky table. Layer 4 sticky functions identically to Layer 3 sticky, except that it sticks a user to a server based on a combination of source IP address and destination port.

The syntax for this global configuration mode command is:

```
show sticky-table l4-sticky [page {value}|ipaddress {ip_address sticky_mask} {port}]
```

The `show sticky-table l4-sticky` command supports the following options and variables:

- **page value** - Shows Layer 4 sticky entries for a specific page in the sticky table, at 100 entries per page. Enter a value from 1 to 5000 to select the page of entries you want to view from the sticky table. To determine the page you want to display, take the Total Number of Used Entries Found value in the `show sticky-stats` command output and divide by 100 (entries per page).

- **ipaddress ip_address sticky_mask** - The IP address of the Layer 3 sticky table entry to be shown. Enter the IP address in dotted-decimal notation (for example, 192.168.2.5). Specify the sticky mask from the content rule for this IP address in dotted-decimal notation (for example, 255.255.255.0). The default sticky mask of a content rule is 255.255.255.255.

- **port** - Destination port of the entry to be shown.

For example, to display Layer 4 sticky entries from a specific IP address and sticky mask in the sticky table for destination port 80, enter:

```
(config)# show sticky-table l4-sticky ipaddress 192.168.2.5 255.255.255.255 80
```

See Table 10-7 for a description of the fields in the `show sticky-table l4-sticky` command output.
Showing SIP Call-ID Sticky Table Information

Use the `show sticky-table sip-callid-sticky` command to display the entries contained in the sticky table based on session Call-ID. Call-ID is a unique call identifier contained in the SIP messages sent from the client to the SIP server.

The syntax for this global configuration mode command is:

```
show sticky-table sip-callid-sticky [page {value}]|Call-ID {sip_callid}
```

The `show sticky-table sip-callid-sticky` command supports the following options and variables:

- **page value** - Shows SIP Call-ID sticky entries for a specific page in the sticky table, at 100 entries per page. Enter a value from 1 to 5000 to select the page of entries you want to view from the sticky table. To determine the page you want to display, take the Total Number of Used Entries Found value in the `show sticky-stats` command output and divide by 100 (entries per page).

- **Call-ID sip_callid** - Specifies a specific Call-ID to display from the sticky table. You can locate the Call-ID number by performing a packet trace.

For example, to display sticky entries for a specific Call-ID in the sticky table, enter:

```
(config)# show sticky-table sip-callid-sticky 12345600@here.com
```

See Table 10-7 for a description of the fields in the `show sticky-table sip-callid-sticky` command output.
Showing SSL Sticky Table Information

Use the `show sticky-table ssl-sticky` command to display the SSL entries contained in the sticky table.

The syntax for this global configuration mode command is:

```
show sticky-table ssl-sticky [rule {index} {page [value]}] | [time [number] [page [value]]] | [sid [text]] | [collision] [page [value]]
```

The `show sticky-table ssl-sticky` command supports the following options and variables:

- **rule index** - Displays the SSL entries in the sticky table for the content rule. Enter the index number for the SSL sticky content rule. You can locate the index number for the content rule in the `show rule summary` command.

- **page value** - Shows entries for a specific page in the sticky table, at 100 entries per page. Enter a value from 1 to 5000 to select the page of entries you want to view from the sticky table. To determine the page you want to display, take the Total Number of Used Entries Found value list in the `show sticky-stats` command output and divide by 100 (entries per page).

- **time number** - Specifies the window of elapsed time (in seconds) in which to display entries from the sticky table. All sticky entries in the table that were referenced within the specified time appear in the show output. Enter the time in seconds.

- **sid text** - Displays the entries in the sticky table based on SSL Session ID (SID). Enter the SID value as a hexadecimal ASCII string without the 0x prefix. You can locate the SID number by performing a packet trace.

- **collision** - Displays the entries in the sticky table that have a collision count (Col Cnt) greater than 0.

For example, to show SSL entries in the sticky table based on content rule index number and page number in the sticky table, enter:

```
(config)# show sticky-table ssl-sticky rule 4 page 33
```

See Table 10-7 for a description of the fields in the `show sticky-table ssl-sticky` command output.
Showing Sticky Table Configurations

Chapter 10 Configuring Sticky Parameters for Content Rules

Showing WAP Sticky Table Information

Use the `show sticky-table wap-sticky` command to display the entries contained in the sticky table based on the MSISDN header field. MSISDN is the header field for wireless clients using the Wireless Application Protocol (WAP).

The syntax for this global configuration mode command is:

```
show sticky-table wap-sticky [page {value}]|msisdn {msisdn_header}
```

The `show sticky-table wap-sticky` command supports the following options and variables:

- **page** value - Shows MSISDN sticky entries for a specific page in the sticky table, at 100 entries per page. Enter a value from 1 to 5000 to select the page of entries you want to view from the sticky table. To determine the page you want to display, take the Total Number of Entries Found value list in the `show sticky-table all-sticky` command output or the `show sticky-stats` command output and divide by 100 (entries per page).

- **msisdn msisdn_header** - Specifies the MSISDN header field to display from the sticky table. Enter the `msisdn_header` as a text string. The MSISDN header field typically contains the wireless phone numbers. You can locate the MSISDN header by performing a packet trace.

For example, to show MSISDN sticky entries in the sticky table based on MSISDN header, enter:

```
(config)# show sticky-table wap-sticky msisdn 6079979410
```

See Table 10-7 for a description of the fields in the `show sticky-table wap-sticky` command output.
Showing Sticky Connection Statistics

The `show sticky-stats` command displays a summary of sticky connection statistics for the CSS.

To display sticky configurations for content, you can also use the `show rule sticky` command in content mode. For details on the `show rule sticky` command, see the “Showing Sticky Attributes” section.

For example:

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

Table 10-8 describes the fields in the `show sticky-stats` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of New Sticky Entries</td>
<td>The total number of unique entries used in the sticky table. Every time an entry is created in the sticky table that is unique, the counter increments.</td>
</tr>
<tr>
<td>Total Number of Sticky Table Hits</td>
<td>The total number of times the CSS received a request from a client that matches an entry in the CSS sticky table. Every time the CSS receives a client entry and the hash value exists in the sticky table, the counter increments. The CSS performs a lookup in the sticky table. If no match is found, the entry is considered to be a new sticky count.</td>
</tr>
<tr>
<td>Total Number of Sticky Rejects (No Entry)</td>
<td>The total number of times that the CSS rejects sticky requests. When the sticky table becomes full and none of the entries have expired from the sticky table, the CSS rejects subsequent sticky requests.</td>
</tr>
<tr>
<td>Total Number of Sticky Collisions</td>
<td>The total number of times the CSS receives a request from a client for an entry in the CSS sticky table with the same hash value and the load-balancing server cannot be resolved.</td>
</tr>
<tr>
<td>Total Number of Available Sticky Entries</td>
<td>The total number of available sticky entries in the sticky table.</td>
</tr>
</tbody>
</table>
### Field Descriptions for the show sticky-stats Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of Used Sticky Entries</td>
<td>The total number of entries currently used in the sticky table.</td>
</tr>
<tr>
<td></td>
<td>The CSS supports a 128K sticky table (with 288 MB of CPU memory) or a 32K sticky table (with 144 MB of CPU memory).</td>
</tr>
<tr>
<td>Total Number of L3 Sticky Entries</td>
<td>The total number of Layer 3 sticky entries in the sticky table.</td>
</tr>
<tr>
<td>Total Number of L4 Sticky Entries</td>
<td>The total number of Layer 4 sticky entries in the sticky table.</td>
</tr>
<tr>
<td>Total Number of SSL Sticky Entries</td>
<td>The total number of SSL session ID sticky entries in the sticky table.</td>
</tr>
<tr>
<td>Total Number of WAP Sticky Entries</td>
<td>The total number of WAP MSISDN header sticky entries in the sticky table.</td>
</tr>
<tr>
<td>Total Number of SIP Sticky Entries</td>
<td>The total number of SIP Call-ID sticky entries in the sticky table.</td>
</tr>
</tbody>
</table>
Chapter 10 Configuring Sticky Parameters for Content Rules

Showing Sticky Connection Statistics
This chapter describes how to configure HTTP header load balancing by creating an HTTP header field group and configuring HTTP header fields. Information in this chapter applies to all CSS models except where noted.

This chapter contains the following major sections:

- HTTP Header Load-Balancing Overview
- HTTP Header Load Balancing Configuration Quick Start
- Creating a Header Field Group
- Describing the Header Field Group
- Configuring a Header-Field Entry
- Associating a Header Field Group with a Content Rule
- Showing a Content Rule Header Field Group Configuration
- Showing Header Field Groups
- Header Field Group Configuration Examples

Note: You must enable service remapping in order for HTTP header load balancing to work properly. For information on the service remapping feature, see Chapter 9, Configuring Content Rules.
HTTP Header Load-Balancing Overview

Configuring HTTP header load balancing enables the CSS to inspect incoming content requests for HTTP header fields. HTTP header load balancing allows the CSS to make load-balancing decisions based on the HTTP header field information and then direct content requests to the servers designed to handle the type of content being requested.

The CSS can direct content requests to specific servers based on different types of browsers or different representations of the same content that has been modified for end users. For example, a client running a hand-held personal organizer may want the same content as a client using a PC, but with fewer graphics. Users may want to see content in only a particular language.

Using HTTP header load balancing eliminates the need to duplicate various forms of the same content across all of the servers, thus freeing up valuable server space. In addition to dividing the server farm for different types of clients, you can also use HTTP header load balancing to bypass noncacheable traffic and prioritize client browser traffic from search engine services.

Using HTTP Header Load Balancing in a Content Rule

Using 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. The HTTP header field group makes the content match more specific. 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. This hierarchy for Layer 5 rules is defined below.

The CSS uses this order of precedence to process requests for the content, with 1 being the highest match and 4 being the lowest match.

1. Domain name, IP address, protocol, port, URL, HTTP header field group
2. IP address, protocol, port, URL, HTTP header field group
3. Domain name, protocol, port, URL, HTTP header field group
4. Protocol, port, URL, HTTP header field group
HTTP Header Load Balancing Configuration Quick Start

Table 11-1 provides a quick overview of the steps required to create and configure HTTP header load balancing. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the HTTP header load-balancing configuration options, see the sections following Table 11-1.

Ensure that you have already created and configured a service and owner for the content rules. The command examples in Table 11-1 create HTTP load balancing for owner arrowpoint and content rule rule1.

Table 11-1 HTTP Load Balancing Configuration Quick Start

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter config mode by typing <code>config</code>.</td>
</tr>
<tr>
<td>2. Create a header field group. This example creates the group ppilot.</td>
</tr>
<tr>
<td>3. Describe the header field group (optional).</td>
</tr>
<tr>
<td>4. Configure header field entries by defining a header, field, name, field type, and operator.</td>
</tr>
<tr>
<td>5. Associate the header field group with a content rule.</td>
</tr>
<tr>
<td>6. (Recommended) Display the header field group to verify your configuration.</td>
</tr>
</tbody>
</table>
The following running-configuration example shows the results of entering the commands in Table 11-1.

```
!********************* HEADER FIELD GROUP ********************
header-field-group ppilot
    description "ppilot content"
    header-field palm1 user-agent contain "MSIE" 20

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

content rule1
    vip address 192.1.1.100
    protocol tcp
    port 80
    add service server1
    header-field-rule ppilot
```

### Creating a Header Field Group

Header field group configuration mode allows you to create a header field group. A header field group contains a list of user-defined header field entries used by the CSS content rule lookup process. A group can contain several header-field entries.

**Note**

The CSS supports a maximum number of 1024 header field groups, with a maximum of 4096 header field entries.

**Note**

When there is more than one header field entry in a group, each header field entry must be successfully matched before the CSS uses the associated content rule.

To create a header field group or to access header field group configuration mode, use the `header-field-group` command from any configuration mode except boot and RMON modes.

The prompt changes to (config-header-field-group [group_name]). You can also use this command in header-field-group mode to access another group.
Describing the Header Field Group

The syntax for this mode-transition command is:

```
header-field-group group_name
```

Enter the `group_name` of the header-field group you want to create. You must define a unique name for each header field group so different content rules can use the groups. Enter a text string with a maximum of 32 characters. To see an existing list of header-field groups, use the `header-field-group ?` command.

For example, enter:

```
(config)# header-field-group ppilot
(config-header-field-group[ppilot])#
```

To remove a header-field group, use the `no header-field-group` command. For example, enter:

```
(config)# no header-field-group ppilot
```

Describing the Header Field Group

To provide a description for a header field group, use the `description` command. The syntax for this command is:

```
description "text"
```

Enter the text as a quoted text string with a maximum length of 64 characters. For example,

```
(config-header-field-group[ppilot])# description "ppilot content"
```

To remove a description for a header-field group, enter:

```
(config-header-field-group[ppilot])# no description
```
Configuring a Header-Field Entry

Configure a header-field entry in a header-field group to specify a field in an HTTP header and an operator to perform a function on that field. When the CSS receives an HTTP content request, it inspects the HTTP header field specified in the header-field entry and performs the function on that field specified in the operator variable. The CSS uses the results of the header-field operation to load balance all subsequent packets in the flow.

A header field entry contains a header field name, field type to be used, an operation to be performed, the header-string to be searched for, and an optional search length.

If a header field group contains multiple header field entries, a content request must match each entry for the rule to be used.

Note

The CSS supports a maximum number of 1024 header field groups, with a maximum of 4096 header field entries.

Use the header-field command to define a header field entry in a header field group. The syntax for this command is:

```
header-field name field_type {custom_string} operator {header_string {search_length}}
```

The variables and options are:

- `name` - The name uniquely identifies the header field entry. Enter the name as a string from 1 to 31 characters. You must define a header field entry name because the CSS can use the same field type multiple times in a header field group.
- `field_type` - The field type includes one of the following:
  - `user-agent` - Information about the user agent, for example a software program originating the request. This information is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations.
– **language** - The ISO code for the language in which the document is written. The language code is an ISO 3316 language code with an optional ISO639 country code to specify a national variant.

– **host** - The Internet host and port number of the resource being requested, as obtained from the original URI given by the user or referring resource. The Host field value MUST represent the naming authority of the origin server or gateway given by the original URL.

– **cache-control** - Directives that must be obeyed by all caching mechanisms along the request/response chain. The directives specify behavior intended to prevent caches from adversely interfering with the request or response.

– **pragma** - Pragma directives understood by servers to whom the directives are relevant. The syntax is the same as for other multiple-value fields in HTTP, for example, the `accept` field, a comma-separated list of entries, for which the optional parameters are separated by semicolons.

– **encoding** - The encoding mechanism used.

– **charset** - The character sets are acceptable for the response. This field allows clients capable of understanding more comprehensive or special-purpose character sets to signal that capability to a server that can representing documents in those character sets.

– **connection** - Options for the connection.

– **referer** - The address (URI) of the resource from which the URI in the request was obtained.

– **accept** - A semicolon-separated list of representation schemes (content type metainformation values) that will be accepted in the response to this request.

– **request-line** - When you attempt to access an Internet resource using your browser (for example, http://www.cisco.com), the browser issues a request for the resource in an HTTP request message. The request line contains the HTTP method (GET, HEAD, or PUSH), the request URI, and the HTTP version. A uniform resource identifier (URI) consists of a string of alphanumeric and sometimes special characters that identify a resource on the Internet. The request line is a required HTTP request message field.
For example, suppose an HTTP request contains the following URI:
/cgi-bin/some-app.pl?session=123456789123456789&user=CiscoUser
&action=LoadBalanceMe&foo=bar

By creating a header field group and header field rules, you can configure
a CSS to make a content rule selection based on a string in the URI. For
example, you can configure a CSS to make a content rule selection based
on the string LoadBalanceMe in the above URI using the following
configuration:

```
header-field-group url
  header-field urlString request-line contain "LoadBalanceMe"
  owner arrowpoint
  content rule UrlString
    vip address 192.168.128.151
    protocol tcp
    port 80
    url "/**"
    add service server1
    add service server2
  header-field-rule url
    active
  content rule2
    vip address 192.168.128.151
    protocol tcp
    port 80
    url "/**"
    add service server21
    add service server22
    active
```

- **cookies** - The configured string found in the HTTP header that the CSS
  uses to stick the client to the server.

- **msisdn** - The header field type for Wireless Application Protocol
  (WAP). HTTP requests from certain wireless gateways contain the
  MSISDN field in the HTTP header. By configuring the **msisdn** header
  field type in a header field group, you can load balance wireless requests.
  See the “Example 3. Wireless configuration that load balances HTTP
  requests based on the MSISDN header field” section later in this chapter.

You can use this option alone or with the **advanced-balance
wap-msisdn** sticky command. See the “Specifying an Advanced
Load-Balancing Method for Sticky Content” section in Chapter 10,
Configuring Sticky Parameters for Content Rules.
- **custom** - Field type keyword that indicates a user-defined header field. Use the **custom** header field with the **custom_string** variable to perform HTTP header matching on the Name: value in the HTTP header field.

- **custom_string** - A case-insensitive, alphanumeric string used with the **custom** field type. Enter a quoted alphanumeric string from 1 to 31 characters. You cannot use the following ASCII characters: control characters (decimal 0 through 31), DEL (decimal 127), and special characters (,, >, @, ,, ,, ,,, /, [, ], ?, =, {, }, SP, and HT. You can define a maximum of 16 unique custom header fields for each CSS.

**Note** You cannot configure a custom header field that is identical to one of the currently predefined header field tags.

The **custom** header field uses the current header-field matching rules; that is, it does not add any new matching rules. See the “Example 4. Configuration that load-balances HTTP requests based on user-defined header fields” section.

For example:

```
header-field customtag2 custom "Peak" contain "CD"
```

- **operator** - Enter one of the following operators:

  - **exist|not-exist** - Use the **exist** and **not-exist** operators to check whether a specified header field exists in a content request header.

  - **equal|not-equal** {"header_string"} - Use the **equal** and **not-equal** operators to match a defined **header_string** to the contents of the specified header field, and to determine whether it is equal to the header string. Enter the **header_string** as a case-insensitive, quoted text string with a maximum of 31 characters including spaces.

  - **contain|not-contain** {"header_string"} {search_length} - Use the **contain** and **not-contain** operators to match the configured **header_string** to a substring in the contents of the specified field type, and to determine whether its contents contain the **header_string**. Enter the **header_string** as a case-insensitive, quoted text string with a maximum of 31 characters including spaces.

    You may include an optional **search_length** to define the header field portion to be used for the operation. If you do not define a search length, the CSS uses the entire header field (delimited by a CR and LF) for the operation. To define the search length, enter a number from 0 to 1024.
Associating a Header Field Group with a Content Rule

For example, enter:

```
(config-header-field-group[ppilot])# header-field palm1 user-agent contain "MSIE" 20
```

```
(config-header-field-group[ppilot])# header-field palm2 user-agent contain "palm"
```

To remove a header field entry, use the **no header-field** command. For example, enter:

```
(config-header-field-group[ppilot])# no header-field palm1
```

---

**Note**

To completely delete a custom header field from a CSS and to make it available for reuse, you must remove all instances of that custom header field from all header-field groups on the CSS using the **no header-field** command.

---

**Associating a Header Field Group with a Content Rule**

To associate a header field group with a content rule, and optionally assign a weight value to the header field group, use the **header-field-rule** command. Use weights to allow the CSS to prefer one content rule over a similar content rule. For example, you want to load balance French clients to a specific server, and you also want to differentiate the clients using Microsoft Internet Explorer from those using Netscape Navigator. If it is more important to direct the French clients to a specific server than to direct them to a server based on whether they are using Internet Explorer or Netscape Navigator, then you need to weight the “French” content rule higher than the “Internet Explorer/Netscape” content rule.

---

**Note**

The CSS supports only one header field group for each content rule.

The syntax for this content mode command is:

```
header-field-rule name {weight number}
```
The variables are:

- **name** - The name of the header field group used with the content rule. To see a list of groups, use the `header-field-rule ?` command.
- **weight number** - The weight you want to assign to the header field group. Enter a number from 0 to 1024. The default weight is 0.

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# header-field-rule french weight 3
```

To remove the header field group from the content rule, enter:

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

**Showing a Content Rule Header Field Group Configuration**

Use the `show rule header-field` command to display information about the header field group associated with a content rule. For example, to display information about the header-field rule and group associated with a specific content rule, enter:

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

**Showing Header Field Groups**

Use the `show header-field-group` command to display the configuration for all header field groups or a specific group. This command is available in all modes.

The syntax and options for this command are:

- **show header-field-group** - Displays a summary of all configured header field groups
- **show header-field-group all** - Displays detailed information about all configured header field groups
- **show header-field-group name** - Displays detailed information about a specific header field group
For example, to show a summary of all configured header field groups, enter:

```config
(config)# show header-field-group
```

Table 11-2 describes the fields in the `show header-field-group` command output.

**Table 11-2  **Field Descriptions for the show header-field-group Command Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header field group</td>
<td>The name of the header-field group</td>
</tr>
<tr>
<td>Description</td>
<td>The configured description for the header-field group</td>
</tr>
</tbody>
</table>

## Header Field Group Configuration Examples

When configuring header field groups, it is good practice to configure rules to be specific in rule matching (as shown in configuration Example 2). If the rules are not specific enough, the CSS may match a client request to the first rule it finds, and the first-matched rule may change on subsequent requests.

This section contains the following configuration examples:

- Example 1. Configuration that is ambiguous in rule-matching capabilities
- Example 2. Configuration that broadens the rule-matching capabilities
- Example 3. Wireless configuration that load balances HTTP requests based on the MSISDN header field
- Example 4. Configuration that load-balances HTTP requests based on user-defined header fields

### Example 1. Configuration that is ambiguous in rule-matching capabilities

Example 1 shows a configuration that is ambiguous. If a client request specifies the language as French and the user-agent as Netscape, this request may match equally to ruleA2 or ruleA3. In this example, the rule matching may not be consistent. One way to solve the ambiguity between ruleA2 and ruleA3 is to use different weight values (not shown in the configuration example). If you assign a weight value of 10 to header field group B when you associate it with ruleA2, the CSS will always use ruleA2 as a match to the client request. Another method is to configure more specific rules as shown in configuration Example 2.
Chapter 11  Configuring HTTP Header Load Balancing

Header Field Group Configuration Examples

! ****************** HEADER FIELD GROUP ******************

header-field-group A
  header-field ua1 language equal "en"

header-field-group B
  header-field ua2 language equal "fr"

header-field-group C
  header-field-group ua3 user-agent contain "Netscape"

! ****************** OWNER ******************

owner arrowpoint
  content ruleA
    protocol tcp
    vip address 192.168.128.151
    port 80
    url "/*"
    add service server1
    add service server2

  content ruleA1
    protocol tcp
    vip address 192.168.128.151
    port 80
    url "/*"
    header-field-rule A
    add service server11
    add service server12

  content ruleA2
    protocol tcp
    vip address 192.168.128.151
    port 80
    url "/*"
    header-field-rule B
    add service server21
    add service server22

  content ruleA3
    protocol tcp
    vip address 192.168.128.151
    port 80
    url "/*"
    header-field-rule C
    add service server31
    add service server32
Example 2. Configuration that broadens the rule-matching capabilities

Example 2 shows the same configuration as Example 1, only modified to broaden the rule-matching capabilities. Each content rule is specific. The client request specifying the language as French and the user-agent as Netscape will match only content rule ruleA2.

! ***************** HEADER FIELD GROUP *****************

header-field-group A
  header-field ua1 language equal "en"
  header-field ua2 user-agent contain "Netscape"

header-field-group B
  header-field ua3 language equal "fr"
  header-field ua4 user-agent contain "Netscape"

header-field-group C
  header-field ua5 language equal "en"
  header-field ua6 user-agent not-contain "Netscape"

header-field-group D
  header-field ua7 language equal "fr"
  header-field ua8 user-agent not-contain "Netscape"

! ****************** OWNER **********************

owner arrowpoint
  content ruleA
    protocol tcp
    vip address 192.168.128.151
    port 80
    url "/**"
    add service server1
    add service server2

  content ruleA1
    protocol tcp
    vip address 192.168.128.151
    port 80
    url "/**"
    header-field-rule A
    add service server11
    add service server12
Example 3. Wireless configuration that load balances HTTP requests based on the MSISDN header field

Example 3 shows a configuration that makes load-balancing decisions based on whether a client is a wireless client. Wireless devices use the Wireless Application Protocol (WAP). When a wireless client sends a request for content, the WAP protocol gateway (a device that translates requests from the WAP protocol stack to the WWW protocol stack) generates the MSISDN field and adds it to the HTTP header. You can test for the presence of the MSISDN header field using the `exist` and `not-exist` operators in the header field entry of a header field group. Then, you can make load-balancing decisions based on the presence or absence of the MSISDN header field. For details on configuring the MSISDN header field type, see the “Configuring a Header-Field Entry” section earlier in this chapter.
In the following example, any TCP port 80 traffic destined for VIP 192.168.128.151 that has the MSISDN field in the HTTP header will match on the content rule ruleWap. Any TCP port 80 traffic destined for 192.168.128.151 that does not have the MSISDN field in the HTTP header will match on the 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
```

Note You can use the MSISDN header field with the `advanced-balance wap-msisdn` command to configure wireless users for e-commerce applications. For details on configuring a wireless user, see the “Configuring Wireless Users for E-Commerce Applications” section in Chapter 10, Configuring Sticky Parameters for Content Rules.

**Example 4. Configuration that load-balances HTTP requests based on user-defined header fields**

Example 4 shows a configuration that enables a CSS to make load-balancing decisions based on custom header fields. You can define a maximum of 16 unique custom header fields on one CSS. However, you can define more than one custom header field in each header field group. If you configure an identical custom header field in more than one header field group, the custom header field counts as only one of the 16 maximum custom header fields that you can configure on that CSS.
In the following example, two unique custom header fields are configured. Any TCP port 80 traffic destined for VIP 192.168.128.15 with the tag “Acme” and with the tag “Peak” that contains the string “CD” will match on the content rule HTTPRule1. Any TCP port 80 traffic destined for VIP 192.168.128.15 with the tag “Peak” that contains the string “CD” will match on the content rule HTTPRule2. The CSS finds the best match based on all the commands configured in the content rules. For more information about configuring content rules, see Chapter 9, Configuring Content Rules.

header-field-group group1
  header-field customtag1 custom "Acme" exist
  header-field customtag2 custom "Peak" contain "CD"

header-field-group group2
  header-field customtag1 custom "Peak" contain "CD"

owner arrowpoint
  content HTTPRule1
    vip address 192.168.128.15
    protocol tcp
    port 80
    url "/**"
    add service server1
    add service server2
    header-field-rule group1
    active

  content HTTPRule2
    vip address 192.168.128.15
    protocol tcp
    port 80
    url "/**"
    add service server3
    add service server4
    header-field-rule group2
    active
This chapter provides an overview of the CSS caching feature and describes how to configure it for operation. Information in this chapter applies to all CSS models, except where noted.

The chapter includes the following major sections:

- Caching Overview
- Caching Configuration Quick Start
- Configuring Caching Content Rules
- Configuring Network Address Translation Peering

Caching Overview

Increasing demand for information on the Internet causes congestion and long delays in retrieving information. Because much of the same information is retrieved over and over again, saving and storing this information can satisfy subsequent requests with more efficiency and less bandwidth.

Saving and storing information locally is known as caching. With Web caching, copies of recently requested content are stored temporarily on a cache server in locations that are topologically closer to the client. The content is then readily available to be reused for subsequent client requests for the same content.
By storing content locally, you:

- Optimize network resources
- Conserve network bandwidth
- Reduce Internet congestion
- Improve network response time and overall service quality

**Content Caching**

You can make Web caching cost-effective and more reliable by deploying content caching in your network. By creating content rules to utilize your cache servers, the CSS acts as a cache front-end device by:

- Examining network traffic for Web content requests
- Bypassing the cache automatically for non-cacheable content
- Distributing content requests to maximize cache hits on services
- Bypassing the cache or redistributing content requests among the remaining cache services if a cache service fails

When a client requests content, the CSS:

- Intercepts the request for content
- Applies content intelligence by parsing the HTTP request header to distribute content requests to the cache servers

The CSS then either:

- Directs the request to the appropriate cache based on the load-balancing method you specify in the content rule (for example, destination IP address)
- Bypasses the cache servers and forwards the request to the origin server if the content is noncacheable
Caching Overview

When the CSS directs the request to the cache server, the cache server either returns the requested content (if it has a local copy) or sends a new request for the content through the CSS to the origin server hosting the content. When the cache sends a new request for content and receives a reply from the origin server, it returns the response to the client. If the content is cacheable, the cache saves a copy of the content for future requests.

When the requested content is found on a local cache server, the request is known as a cache hit. When the requested content is not local and the cache initiates a new request for the content, the request is known as a cache miss.

The following sections provide CSS content caching examples:

- Using Proxy Caching
- Using Reverse Proxy Caching
- Using Transparent Caching
- Using Cache Clustering

Using Proxy Caching

With proxy caching, each client is configured with the IP address of the proxy cache to which clients send content requests. You may also configure a URL for browsers to identify the location of the proxy configuration file for automatic proxy configuration. Each client’s content request is sent directly to the proxy cache IP address. The cache returns the requested content if it has a local copy, or else it sends a new request to the origin server for the information.

If all cache servers are unavailable in a proxy cache configuration, the client request does not pass to the origin server because clients are configured with the proxy cache VIP.
Figure 12-1 shows an example of using a CSS in a proxy cache configuration.

Using Reverse Proxy Caching

In a reverse proxy cache configuration, the proxy server is configured with an Internet-routable IP address. Clients are directed to the proxy server based on a Domain Name System (DNS) resolution of a domain name. To a client, the reverse proxy server appears like a Web server.

In a regular proxy cache configuration, the proxy server acts as a proxy for the client. In the reverse proxy configuration, the reverse proxy server acts as a proxy for the server. Also, a reverse proxy cache caches specific content, whereas proxy and transparent caches cache frequently requested content. Reverse proxy caches serve two primary functions:

- Replication of content to geographically dispersed areas
- Replication of content for load balancing
Figure 12-2 shows an example of a CSS 11506 and CSS 11503s in a reverse proxy cache configuration.

**Figure 12-2 Reverse Proxy Cache Configuration Example**

![Diagram of CSS 11506 and CSS 11503s in a reverse proxy cache configuration]

**Using Transparent Caching**

Transparent caching deploys cache servers that are transparent to the browsers. You do not have to configure browsers to point to a cache server. Cache servers duplicate and store inbound Internet data previously requested by clients.

When you configure transparent caching on the CSS, the CSS intercepts and redirects outbound client requests for Internet data to the cache servers on your network. The cache returns the requested content if it has a local copy, or else it sends a new request to the origin server for the information.
If all cache servers are unavailable in a transparent cache configuration, the CSS allows all client requests to progress to the origin servers.

A transparent caching configuration:

- Reduces network congestion caused by HTTP traffic
- Increases network efficiency
- Decreases the time required to fulfill a client request by accessing locally stored information rather than obtaining the same information across the Internet

Figure 12-3 shows an example of a typical transparent cache configuration.

**Figure 12-3  Transparent Cache Configuration Example**

![Transparent Cache Configuration Example Diagram](image-url)
Using Cache Clustering

Multiple caches deployed at a single location is referred to as cache clustering. Cache clustering provides:

- Scalability
- Redundancy
- Transparency
- Simplified administration

Figure 12-4 shows an example of using content caching in a cache cluster configuration.

Figure 12-4  Cache Cluster Configuration Example
Caching Configuration Quick Start

Table 12-1 provides the steps to configure service serv1 as a caching service. Each step includes the CLI command required to complete the task. Ensure that you have configured services, owners, and content rules prior to configuring CSS caching.

Note

When using content caching, the keepalive type must be ICMP (default setting).

For a complete description of each caching command, see the sections following Table 12-1.

Table 12-1 Caching Configuration Quick Start

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Specify a service type (type local, type proxy-cache, type redirect, type transparent-cache). The default is local.</td>
</tr>
<tr>
<td>(config-service[serv1])# type transparent-cache</td>
</tr>
<tr>
<td>2. Create an Extension Qualifier List (EQL) where you specify which content types the CSS caches.</td>
</tr>
<tr>
<td>(config)# eql graphics</td>
</tr>
<tr>
<td>(config-eql[graphics])#</td>
</tr>
<tr>
<td>3. Describe the EQL by entering a quoted text string with a maximum length of 63 characters.</td>
</tr>
<tr>
<td>(config-eql[graphics])# description &quot;This EQL specifies cacheable graphic files&quot;</td>
</tr>
<tr>
<td>4. Specify the extension for content you want the CSS to cache. Enter a text string from 1 to 8 characters.</td>
</tr>
<tr>
<td>(config-eql[graphics])# extension jpeg</td>
</tr>
<tr>
<td>Optionally, you may provide a description of the extension type. Enter a quoted text string with a maximum length of 64 characters.</td>
</tr>
<tr>
<td>(config-eql[graphics])# extension gif &quot;This is a graphics file&quot;</td>
</tr>
<tr>
<td>(config-eql[graphics])# exit</td>
</tr>
<tr>
<td>(config)#</td>
</tr>
</tbody>
</table>
Chapter 12  Configuring Caching

Caching Configuration Quick Start

Table 12-1  Caching Configuration Quick Start (continued)

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>5.</strong> Specify the EQL in a content rule to match all content requests with the desired extensions.</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint.com-rule1])# url &quot;/*&quot; eql graphics</td>
</tr>
<tr>
<td><strong>6.</strong> Configure the load-balancing method for the cache content rule. The default is round robin.</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint.com-rule1])# balance domain</td>
</tr>
<tr>
<td><strong>7.</strong> Specify a failover type to define how the CSS handles content requests when a service fails (bypass, next). The default is linear.</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint.com-rule1])# failover bypass</td>
</tr>
<tr>
<td><strong>8.</strong> Display the EQL configuration.</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint.com-rule1])# show eql</td>
</tr>
<tr>
<td><strong>9.</strong> Display the content rule to show the cache configuration.</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint.com-rule1])# show rule</td>
</tr>
</tbody>
</table>

The following running-configuration example shows the results of entering the commands in Table 12-1.

```
!************************** SERVICE **************************
service serv1
    type transparent-cache
    ip address 192.168.100.100
    active

!************************** EQL ******************************
eql graphics
    extension .jpg
    description "This EQL specifies cacheable graphic files"
    extension jpeg
    extension gif "This is a graphics file"

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

Configure caching using content rules. When you are creating caching content rules, the additional configuration requirements involve:

- Specifying a service type that supports caching
- Specifying a failover type for the cache servers
- Configuring a load-balancing algorithm that supports caching
- Configuring EQLs to identify file extensions that the CSS should direct to the cache services

Note

If you are running the Inktomi Traffic Server on a system that does not listen in promiscuous mode and want to bypass the Inktomi Adaptive Redirect module (that is, you want to send traffic directly to port 8080 instead of port 80), specify the CSS service type as `type proxy-cache`. Configuring the CSS service type to `type proxy-cache` causes the CSS to perform full Network Address Translation (NAT) when directing traffic to the Traffic Server.

Specifying a Service Type

The CSS enables you to specify the following cache-specific service types using the `type` command. The default service type is local.

- `type nci-direct-return` - Specifies the service as NAT Channel indication for direct return. Use with reverse proxy cache and NAT peering.
- `type nci-info-only` - Specifies the service as NAT Channel indication for information only. Use with reverse proxy cache and NAT peering.


- **type proxy-cache** - Specifies the service as a proxy cache. This option bypasses content rules for requests coming from the cache server. In this case, bypassing content rules prevents a loop between the cache and the CSS.

- **type rep-cache** - Specifies the service as a replication cache.

- **type rep-cache-redir** - Specifies the service as a replication cache with redirect.

- **type transparent-cache** - Specifies the service as a transparent cache. No content rules are applied to requests from this service type. Bypassing content rules in this case prevents a loop between the cache and the CSS.

For example, to specify service serv1 as a proxy cache, enter:

```
(config-service[serv1])# type proxy-cache
```

The CSS recognizes and forwards the following HTTP methods directly to the destination server in a transparent caching environment. However, the CSS does not load balance these methods.

- RFC 2068: OPTIONS, TRACE
- RFC 2518: PROPFIND, PROPPATCH, MKCOL, MOVE, LOCK, UNLOCK, COPY, DELETE

**Note**

To enable the CSS to redirect a request to a remote service when a request for content matches the rule, you must specify a URL for the content rule.

### Specifying a Failover Type

By default, the CSS uses a linear failover method, which distributes the content requests to the failed service evenly among the remaining services.

To define how the CSS handles content requests when a cache 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 (default keepalive is ICMP). The CSS uses the keepalive settings to monitor the cache services to determine server health and availability. See Chapter 3, Configuring Services for more information on the `keepalive` command.
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.com-rule1])# failover bypass
```

To restore the default failover method of linear, enter:

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

**Figure 12-5** 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 12-5  Cache Services Configured for Failover Next Example 1**
As shown in Figure 12-6, if ServerC fails, the CSS sends ServerC content requests to ServerA because no other services were configured after ServerC.

**Figure 12-6 Cache Services Configured for Failover Next Example 2**

![Diagram of cache services configured for failover linear](image)

**Figure 12-7** shows three cache services configured for **failover linear** (the default). If you suspend ServerB or if it fails, the CSS does not rebalance the services. It evenly distributes ServerB cache workload between servers A and C.

Note that **Figure 12-7** and **Figure 12-8** use the alphabet to illustrate division balance.

**Figure 12-7 Suspended or Failed Cache Service Configured for Failover Linear**

![Diagram of suspended or failed cache service](image)
Figure 12-8 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 12-8  Removing a Cache Service Configured for Failover Linear**

### Configuring Load Balancing

To specify the load-balancing algorithm for a content rule, use the `balance` command. This command is available in content configuration mode. 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 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** If you are using the `domainhash` load-balancing method with proxy cache services, you may see duplicate sites across caches because the CSS balances on the first GET request in a persistent connection unless the subsequent GET request does not match a rule with the same proxy service specified. If you are concerned about duplicate hits across caches, reset persistence to remap and disable persistence on the rule. Issue the `(config) persistence reset remap` command globally and the `(config-owner-content) no persistent` command on the content rule.

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

• **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 on 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.
### Configuring Caching Content Rules

- **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**
A Layer 5 content rule supports the HTTP CONNECT, GET, HEAD, POST, PUSH, and PUT methods. The CSS recognizes and forwards the following HTTP methods directly to the destination server in a transparent caching environment. Note that the CSS does not load balance these HTTP methods. RFC 2068: OPTIONS, TRACE; RFC 2518: PROPFIND, PROPPATCH, MKCOL, MOVE, LOCK, UNLOCK, COPY, DELETE.

In a transparent caching environment (for example, no VIP address on a Layer 5 content rule), the CSS bypasses these HTTP methods, and they are forwarded to the destination server.

For example, to specify weighted roundrobin 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
```

### Configuring a Double-Wildcard Caching Content Rule

When you want to optimize Layer 3 and Layer 4 TCP/IP traffic, configure a content rule for transparent caching without specifying the VIP address and port number. This configuration may be particularly useful in a wireless environment where there is intelligence built into the backend server.

If all other matching criteria in the content rule are met by the client request, a request with any VIP or port will match the rule. This is called a double-wildcard caching rule. You still need to specify the protocol in the rule. Typically, use this type of rule when you are load-balancing services of type `transparent-cache`. However, you can configure this type of rule with other service types as well.
Note: If you have a configuration that requires a double-wildcard rule, be aware that the client request will match on this rule when the client attempts to connect directly to a server IP address.

Enabling Content Requests to Bypass Caches

The following sections describe how to enable content requests to bypass caches:

- Using the **param-bypass** Command
- Using the **cache-bypass** Command
- Using the **bypass-hosttag** Command

Using the **param-bypass** Command

The terminators “#” and “?” 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 not cacheable.

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
```
Using the cache-bypass Command

By default, a CSS does not apply content rules to requests from a proxy or transparent-cache type service going to the origin server when the cache does not contain the requested content. Use the no cache-bypass command to allow the application of content rules to requests originating from a proxy or transparent cache. Use the cache-bypass command to restore the default behavior of the CSS after you have issued the no cache-bypass command.

For example, to allow the CSS to apply content rules to requests from a proxy or transparent-cache type service, enter:

```
(config-service[serv1])# no cache-bypass
```

To restore the CSS default behavior after issuing the no cache-bypass command, enter:

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

Using the bypass-hosttag Command

By default, the CSS disables the bypassing of a cache farm for noncacheable content. Use the bypass-hosttag command to allow a CSS configured as a Client Side Accelerator (CSA) to bypass a cache farm and establish a connection with the origin server to retrieve noncacheable content. The domain name from the host-tag field is used to look up the origin IP address on the CSA.

Note

Use the bypass-hosttag command only with a CSS operating in a CSA environment. For details on CSA, refer to the Cisco Content Services Switch Global Server Load-Balancing Configuration Guide.

For example, enter:

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

To disable bypassing cache for noncacheable content, enter:

```
(config-service[serv1])# no bypass-hosttag
```
Configuring Network Address Translation for Transparent Caches

By default, the CSS disables destination NATing for the transparent cache service type. Use the `transparent-hosttag` command to enable destination Network Address Translation (NAT) for the transparent cache service type. This command NATs the destination address of the client’s packet (forwarded by the CSS to the cache) to the origin server IP address for the requested domain. Using this command ensures that the cache always has the current origin server IP address based on periodic DNS lookups that the CSS performs for all accelerated domains.

The alternative is to manually configure all origin server IP addresses on the cache, which may or may not support static configuration. Also, statically configured IP addresses can become obsolete if the origin server IP address changes. For caches that support DNS resolution and use the DNS response to fetch content or that support configuration of origin server IP addresses, `transparent-hosttag` is not required but is recommended.

**Note**

You can use the `transparent-hosttag` command only with a CSS operating in a Client Side Accelerator (CSA) environment. For details on CSA, refer to the *Cisco Content Service Switch Global Server Load-Balancing Configuration Guide*.

For example, enter:

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

To disable destination NATing for the transparent cache service type, enter:

```
(config-service[serv1])# no transparent-hosttag
```
Configuring Network Address Translation Peering

NAT peering allows clients to connect to remote Web sites through CSSs and have the return traffic use the shortest network path back to the client. The forward path from the client to the server is through TCP connections between two CSSs, but the reverse path from the server to the client may take the shortest network route rather than traversing back through the CSSs.

Note

NAT peering requires the CSS Enhanced feature set license.

NAT peering allows the CSS to:

- Forward client connections to a remote CSS
- Perform the final translation at the remote CSS, which allows return traffic packets to flow to the client through any network path
- Preserve the client IP address when forwarding traffic to the origin server

Note

Adaptive Session Redundancy (ASR) does not support NAT peering. For details on ASR, refer to the Cisco Content Services Switch Global Server Load-Balancing Configuration Guide.

To perform NAT transformations on a TCP flow, the client-side CSS forwards traffic to the server-side CSS through a NAT channel. This channel uses a special TCP option called the NAT Channel Indication (NCI) option. This option indicates to the server-side CSS that NAT parameters are in use, and contains the original source and destination IP addresses, and TCP port numbers. This option also has a spoof bit to indicate that part of the flow has been spoofed and the rest of the forward path must be established before the destination CSS can use the information in the packet to perform the NAT transformations for the reverse path.

Note

Spoofing occurs when a CSS requires information from the HTTP request, (such as host tag, filename, file extension) in order to make a load-balancing decision.
The server-side CSS preserves the client address and port. This allows the origin server to maintain statistics based on the original traffic source addressing data, and allows the return path to be independent of the forwarding path.

Figure 12-9 shows an example of NAT peering. The steps that follow describe this example.

**Figure 12-9  NAT Peering Configuration Example**


2. The client-side CSS matches the request to its content rule, which specifies a service located on the server-side CSS (CSS2, VIP2 200.200.200.200). The server-side CSS service is configured for service type `nci-direct-return`. This service type informs the client-side CSS to include the NCI option in the TCP packet sent to server-side CSS. If a Layer 5 rule is matched, the spoof bit in the NCI option is set.
3. The client-side CSS sends the TCP packet to the server-side CSS. Source address group mapping maps the Client A source address and port to those from the client-side CSS. The TCP packet contains the client-side CSS source information, the server-side CSS destination information, and the original source and destination information from Client A.

4. The server-side CSS determines whether the spoof bit has been set in the packet. If the bit is set, the CSS stores the NAT information until the connection is spoofed. The server-side CSS sets up the forward and return paths. The server-side CSS then matches the request from the client-side CSS on a content rule.

Note
The server-side CSS (in Figure 12-9) would use the NCI option in a packet if the VIP rule is directed at a local, proxy-cache, or transparent cache service.

5. The server-side CSS sends the request to the origin server with the destination IP address translated to the origin server IP address and the source IP address translated to the client IP address.

6. The origin server responds directly back to Client A. As the packet flows through the server-side CSS, that CSS translates the source IP address to the CSS1 VIP. The destination IP address is the client IP address.

**Configuring NAT Peering**

All NAT peering configuration occurs on the client-side CSS. During the configuration consider the following:

- When you configure the NCI service as `nci-direct-return`, the service must be directed to the VIP on the server-side CSS to indicate an endpoint for the connection. The server-side CSS always uses the `nci-direct-return` option to modify the source address and port that the server sees. When the nci-direct-return service is used on the client-side, the return path is modified to directly return to the client.

- When you are specifying an NCI service type, you must specify:
  - `type nci-direct-return` to represent a VIP on another CSS
  - `type nci-info-only` for any Web server
Table 12-2 describes the steps necessary to configure NAT peering using command examples based on the configuration in Figure 12-9. Because NAT peering applies to Layer 3 as well as Layer 5 rules, the port, protocol, and URL rule examples shown in Table 12-2 are optional.

**Table 12-2  NAT Configuration Quick Start**

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. On the client-side CSS (CSS1), create content rules to configure the server-side CSS (CSS2) as a service.</td>
</tr>
<tr>
<td>a. Create service CSS2.</td>
</tr>
</tbody>
</table>
  CSS1 (config)# service CSS2 |
| b. Configure CSS2 VIP as the service IP address. |  
  CSS1 (config-service[CSS2])# ip address 200.200.200.200 |
| c. Configure CSS2 as a service type nci-direct-return. |  
  CSS1 (config-service[CSS2])# type nci-direct-return |
| d. Activate the content rule. |  
  CSS1 (config-service[CSS2])# active |
2. On the client-side CSS (CSS1), create content rules with the criteria required for the client-side CSS (CSS1) to forward traffic to the server-side CSS (CSS2).
   a. Create an owner.
      ```
      CSS1 (config)# owner boston.com
      ```
   b. Name the content rule and assign it the owner.
      ```
      CSS1 (config-owner[boston.com])# content rule1
      ```
   c. Configure the CSS1 VIP.
      ```
      CSS1 (config-owner-content[boston.com-rule1])# vip address 195.195.195.195
      ```
   d. Configure port and protocol.
      ```
      CSS1 (config-owner-content[boston.com-rule1])# port 80
      CSS1 (config-owner-content[boston.com-rule1])# protocol tcp
      ```
   e. Define the URL.
      ```
      CSS1 (config-owner-content[boston.com-rule1])# url "//bostoninfo.html/
      ```
   f. Add CSS2 as the service.
      ```
      CSS1 (config-owner-content[boston.com-rule1])# add service CSS2
      ```
   g. Activate the rule.
      ```
      CSS1 (config-owner-content[boston.com-rule1])# active
      ```

---

**Table 12-2  NAT Configuration Quick Start (continued)**

**Task and Command Example**

2. On the client-side CSS (CSS1), create content rules with the criteria required for the client-side CSS (CSS1) to forward traffic to the server-side CSS (CSS2).
   a. Create an owner.
      ```
      CSS1 (config)# owner boston.com
      ```
   b. Name the content rule and assign it the owner.
      ```
      CSS1 (config-owner[boston.com])# content rule1
      ```
   c. Configure the CSS1 VIP.
      ```
      CSS1 (config-owner-content[boston.com-rule1])# vip address 195.195.195.195
      ```
   d. Configure port and protocol.
      ```
      CSS1 (config-owner-content[boston.com-rule1])# port 80
      CSS1 (config-owner-content[boston.com-rule1])# protocol tcp
      ```
   e. Define the URL.
      ```
      CSS1 (config-owner-content[boston.com-rule1])# url "//bostoninfo.html/
      ```
   f. Add CSS2 as the service.
      ```
      CSS1 (config-owner-content[boston.com-rule1])# add service CSS2
      ```
   g. Activate the rule.
      ```
      CSS1 (config-owner-content[boston.com-rule1])# active
      ```
3. On the client-side CSS (CSS1), create a source group for the client traffic. CSS1 will translate the Client A IP address to the IP address defined in the source group. To configure a source group:

   a. Create the source group.
      ```
      CSS1 (config)# group boston
      ```
   
   b. Define the CSS1 VIP as the IP address into which the Client A IP address will be translated.
      ```
      CSS1 (config-group[boston])# vip 195.195.195.195
      ```
   
   c. Activate the source group.
      ```
      CSS1 (config-group[boston])# active
      ```

4. On the client-side CSS (CSS1), create an access control list (ACL) clause to specify which source IP addresses use the source group. Note that clause 20 is a required clause that permits all other traffic. Without clause 20, all traffic not defined in clause 10 is denied.

   ```
   CSS1 (config)# acl 1
   CSS1 (config-acl[1])# clause 10 permit tcp any destination content boston.com/rule1 sourcegroup boston
   CSS1 (config-acl[1])# clause 20 permit any any destination any apply circuit-(VLAN1)
   ```

5. On the server-side CSS (CSS2), configure the origin server connected to CSS2.

   a. Create origin server serv1.
      ```
      CSS2 (config)# service serv1
      ```
   
   b. Configure an IP address for serv1.
      ```
      CSS2 (config-service[serv1])# ip address 10.3.6.58
      ```
   
   c. Activate the server.
      ```
      CSS2 (config-service[serv1])# active
      ```
The following running-configuration example shows the results of entering the client-side CSS commands in Table 12-2.

```
---
!************************** SERVICE ****************************
service CSS2
 ip address 200.200.200.200
 type nci-direct-return
 active
---
```
The following running-configuration example shows the results of entering the server-side CSS commands in Table 12-2.
CHAPTER 13

Configuring Content Replication

This chapter describes how to configure demand-based content replication and content staging and replication.

Note

The Demand-Based Content Replication and the Content Staging and Replication features require the CSS Enhanced feature set license.

The information in this chapter applies to all CSS models, except where noted. This chapter contains the following major sections:

- Configuring Demand-Based Content Replication
- Configuring Content Staging and Replication
Configuring Demand-Based Content Replication

One of the biggest challenges for a Web site includes managing unpredictable traffic and flash crowds caused by sudden hot content. Using demand-based content replication, the CSS can track content requests and identify and replicate hot content to overflow Web servers or caches dynamically.

**Note**

The Demand-Based Content Replication feature requires the CSS Enhanced feature set license.

Demand-based content replication is traffic-based. Increases in the flow of traffic make content available automatically at replication services. When you configure demand-based content replication, the CSS automatically:

1. Uses hot lists to detect hot content when the URL hits exceed the configured hot list threshold.
2. Modifies the content rules dynamically to provide additional services from which the hot content may be served.

The following sections describe how to configure service replication:

- Demand-Based Content Replication Quick Start
- Configuring Hot Lists
- Specifying Service Type for Replication
- Configuring Max Age
- Configuring Max Content
- Configuring Max Usage
- Configuring FTP Access for Content Replication
- Creating an FTP Record
Demand-Based Content Replication Quick Start

Table 13-1 provides a quick overview of the steps required to configure demand-based content replication. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI command, see the sections following Table 13-1.

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter global configuration mode.</td>
</tr>
<tr>
<td># config</td>
</tr>
<tr>
<td>(config)#</td>
</tr>
<tr>
<td>2. If necessary, create an FTP record. See the “Creating an FTP Record” section.</td>
</tr>
<tr>
<td>(config)# ftp-record myftprecord 172.16.6.58 bobo “secret” /</td>
</tr>
<tr>
<td>3. Enter owner mode.</td>
</tr>
<tr>
<td>(config)# owner arrowpoint</td>
</tr>
<tr>
<td>(config-owner[arrowpoint])#</td>
</tr>
<tr>
<td>4. Enter config-owner-content mode.</td>
</tr>
<tr>
<td>(config-owner[arrowpoint])# content rule1</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint-rule1])#</td>
</tr>
<tr>
<td>5. Use the hotlist command to configure a list that captures the names of the most requested content (hot content). For details, see the “Configuring Hot Lists” section. You can set the following hotlist command options:</td>
</tr>
<tr>
<td>• interval</td>
</tr>
<tr>
<td>• size</td>
</tr>
<tr>
<td>• threshold</td>
</tr>
<tr>
<td>• type</td>
</tr>
<tr>
<td>(config-owner-content[arrowpoint-rule1])# hotlist interval 10</td>
</tr>
</tbody>
</table>
The following running-configuration example shows the results of entering the commands in Table 13-1.

```
!************************** SERVICE **************************
service local_serv
service rep_serv
    type rep-cache-redir
    access ftp myftprecord

!*************************** OWNER ***************************
owner arrowpoint
    address "200 Beaver Brook Road, Boxborough, MA 01719"
```
content rule1
  vip address 192.1.1.100
  protocol tcp
  port 80
  hotlist interval 10

Configuring Hot Lists

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. The CSS enables you to configure hot-list attributes for content rules. Use the `hotlist` command to define a list that captures the names of the most requested content (hot content).

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

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

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

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

  To disable a hot list, enter:

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

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

  ```
  (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** - Sets 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:

```
(config-owner-content[arrowpoint-rule1])# hotlist size 20
```

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

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

• **hotlist threshold** - Sets 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:

```
(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 type hitCount** - Sets the hot-list type to hit count, which is how many times the content was accessed. For example:

```
(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
```

### Specifying Service Type for Replication

Within a replication configuration, you must configure at least two servers: one local and one replication type. The CSS provides the following service types specific to replication:

- **type rep-cache-redir** - Specifies the service is a replication cache with redirect.

- **type rep-store** - Specifies the service is a replication store, which is a local overflow service used to load balance content requests.

- **type rep-store-redir** - Specifies the service is a replication store with redirect.

When you specify a service as **type rep-cache-redir**, the CSS uses the service as a cache server, caching hot content and sending requests to it. Once content is cached on the replication server, the CSS creates a dynamic content rule for the hot content and a dynamic service.
Chapter 13 Configuring Content Replication

Configuring Demand-Based Content Replication

The CSS deletes the hot content when the max-age time has elapsed. See the section, “Configuring Max Age” later in this chapter.

For example:

```
(config)# service serv1
(config-service[serv1])# type rep-cache-redir
```

When you specify a service as type rep-store, the CSS replicates hot content on the service. Once content is replicated on the replication server, the CSS creates a dynamic content rule for the hot content automatically. The dynamic content rule inherits all the attributes of the existing rule with the following changes:

- Specifically identifies the hot content with the content rule `uri` command
- Changes the server type from replication-store to type local

The CSS deletes the dynamic content rule after the maximum age time elapses. See the following section, “Configuring Max Age”. The CSS lists the dynamic content rule in the `show rule` display. It is not displayed in the running- or startup-config files.

**Note**

A replication service type is not included in the load-balancing algorithm until content is replicated on the service.

For example:

```
(config)# service serv1
(config-service[serv1])# type rep-store
```

**Configuring Max Age**

To define the maximum age for replicated objects on services defined as type `rep-cache-redir`, `rep-store`, or `rep-store-redir`, use the `max age` command. Enter the maximum age in minutes from 1 to 1440. The default is 120.

For example:

```
(config-service[serv1])# max age 10
```

To set the maximum age for replicated objects to its default value of 120, enter:

```
(config-service[serv1])# no max age
```
Configuring Max Content

To define the maximum pieces of content for replication on services defined as type rep-cache-redir, rep-store, or rep-store-redir, use the max content command. Enter the maximum pieces of content from 1 to 65535. The default is 100.

For example:

```
(config-service[serv1])# max content 50
```

To set the maximum content to its default value of 100, enter:

```
(config-service[serv1])# no max content
```

Configuring Max Usage

To define the maximum disk space allowed for replication on services defined as type rep-cache-redir, rep-store, or rep-store-redir, use the max usage command. Enter the disk space for a service from 1 to 1000 MB. The default is 1.

For example:

```
(config-service[serv1])# max usage 100
```

To set the maximum disk space to its default value of 1, enter:

```
(config-service[serv1])# no max usage
```

Configuring FTP Access for Content Replication

You must associate an FTP access mechanism for each service that offers publishing services. Use the access ftp command to associate an FTP access mechanism with a service for demand-based replication activities.

When you use this command to associate an FTP access mechanism with a service, the base directory of an existing FTP record becomes the tree root. To maintain coherent mapping between WWW daemons and FTP daemons, make the FTP access base directory equivalent to the WWW daemon root directory as seen by clients.

Enter the access ftp_record as the name of an existing FTP record. Enter the FTP record name as an unquoted text string with no spaces.
Creating an FTP Record

To create a File Transfer Protocol (FTP) record file to use when accessing an FTP server from the CSS, use the `ftp-record` command. The syntax for this global configuration mode command is:

```
ftp-record ftp_record ip_address or hostname username
["password"] des-password des_password] base_directory
```

The variables are:

- `ftp_record` - The name for this FTP record file. Enter an unquoted text string with no spaces and a maximum length of 16 characters.
- `ip_address` or `hostname` - The IP address or host name of the FTP server you want to access.
- `username` - A valid login username on the FTP server. Enter a case-sensitive unquoted text string with no spaces and a maximum of 32 characters.
- `"password"` - The password for the login username on the FTP server. Enter a case-sensitive quoted text string with no spaces and a maximum of 16 characters.
- `des_password` - The Data Encryption Standard (DES) encrypted password for the valid login username on the FTP server. Enter a case-sensitive unquoted text string with no spaces and a maximum of 64 characters.
- `base_directory` - An optional base directory when using this record.
For example:

```
(config)# ftp-record myftprecord 172.16.6.58 bobo "secret" /
```

To delete an FTP record file from the CSS, use the `no ftp-record` command and the ftp record name. For example:

```
(config)# no ftp-record myftprecord
```

### Configuring Content Staging and Replication

The CSS supports content staging and replication using Publisher and Subscriber services. With this feature, the CSS takes content (for example, a file, multiple files, or complete directories) that you post to the staging publisher server and replicates the content dynamically to multiple subscriber servers based on one of the following triggers:

- CLI commands.
- Detected changes to specific content on the staging server. The CSS then replicates that content to the subscriber servers or caches dynamically.

The CSS detects changes to specific content by performing an FTP-based examination of filenames, sizes, and file dates. The CSS performs this examination based on the configured publisher interval or by the `replicate` command. The subscriber knows how to interface to the publisher by virtue of the 'access ftp' associated with the publisher-designated service.

- (Optional) Preconfigured month, day, and time using the command scheduler feature.

**Note**

The Content Staging and Replication feature requires the CSS Enhanced feature set license.

For new configurations, after software upgrades, or after adding a new subscriber, we recommend that you use the `replicate` command with the `force` option to ensure that the publisher and the subscriber information is synchronized. Thereafter, the CSS automatically updates the associated subscribers with any file or directory changes that occur on the publisher according to the configured publisher interval.
If a CSS detects any differences on the publisher when you reboot the CSS on the same software image with content replication configured, the publisher applies those differences to the associated subscribers to maintain synchronization. This behavior assumes that you have previously synchronized the publisher and the subscriber.

**Note**

If you manually manipulate the files or directories on a subscriber, you invalidate the synchronization between the publisher and that subscriber. To resynchronize the subscriber content with the publisher content, you must enter the `replicate` command with the `force` option.

The content staging and replication feature does not limit the size of files to be replicated. However, larger files take a longer time to be replicated. The only limitation for this feature is a maximum of 1,000,000 replicated files. Note that the CSS does not store replicated files on its disk. The CSS creates a virtual path to transfer data between the publisher and subscriber services. Buffering is limited to the default buffering associated with TCP.

You can configure the CSS to continually update content that has been replicated. For example, the CSS can replicate content associated with a breaking news story. You can post updates to the staging server and the updates will be replicated to all distributed locations automatically.

Publisher and subscriber services are usually defined as type `local`. There is no need to change the service type.

The following sections describe how to configure publisher and subscriber services:

- Content Staging and Replication Quick Start
- Configuring FTP Access for Publishing and Subscribing
- Configuring a Publishing Service
- Configuring a Subscriber Service
- Configuring a Content Rule for Content Staging and Replication
- Configuring Publisher Content Replication
- Displaying Content
Content Staging and Replication Quick Start

Table 13-2 provides a quick overview of the steps required to configure demand-based content replication. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI command, see the sections following Table 13-2.

Table 13-2 Content Staging and Replication Configuration Quick Start

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enter config mode.</td>
</tr>
<tr>
<td># config</td>
</tr>
<tr>
<td>(config)#</td>
</tr>
<tr>
<td>2. If necessary, create an FTP record. See the “Creating an FTP Record” section.</td>
</tr>
<tr>
<td>(config)# ftp-record myftprecord 172.16.6.58 bobo &quot;secret&quot; /</td>
</tr>
<tr>
<td>3. Create a service to be used as a publishing service.</td>
</tr>
<tr>
<td>(config)# service pubserver</td>
</tr>
<tr>
<td>(config-service[pubserver])#</td>
</tr>
<tr>
<td>4. Configure the service as a publishing service. See the “Configuring a Publishing Service” section.</td>
</tr>
<tr>
<td>(config-service[pubserver])# publisher</td>
</tr>
<tr>
<td>5. Configure a recurrent time interval in minutes to synchronize content among the subscribers. See the “Configuring a Publishing Service” section.</td>
</tr>
<tr>
<td>(config-service[pubserver])# publisher interval 120</td>
</tr>
<tr>
<td>6. Associate an FTP access mechanism with a service for demand-based replication activities. You must use this command for each service that offers publishing services. See the “Configuring FTP Access for Content Replication” section.</td>
</tr>
<tr>
<td>(config-service[pubserver])# access ftp myftprecord</td>
</tr>
<tr>
<td>7. Create a service to be used as a subscriber service.</td>
</tr>
<tr>
<td>(config)# service subserver</td>
</tr>
<tr>
<td>(config-service[subserver])#</td>
</tr>
</tbody>
</table>
The following running-configuration example shows the results of entering the commands in Table 13-2.

```
!************************** SERVICE **************************
service pubserver
  publisher interval 120
  access ftp myftprecord
```

### Table 13-2 Content Staging and Replication Configuration Quick Start

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>8.</strong> Use the <strong>subscriber</strong> command to configure a service as a subscriber to a publishing service. See the “Configuring a Subscriber Service” section.</td>
</tr>
<tr>
<td>(config-service[subserver])# subscriber pubserver</td>
</tr>
<tr>
<td><strong>9.</strong> Configure a URL in a content rule to define which files you want replicated:</td>
</tr>
</tbody>
</table>
| (config-owner-content[arrowpoint-products.html])# url 
  
  
  “/announcements/*.html” |
| For a complete description of configuring URLs, refer to Chapter 9, Configuring Content Rules. |
| **10.** Use the **add service** command to add the subscriber services to the content rule. |
| (config-owner-content[arrowpoint-products.html])# add service subserver |
| **11.** Use the **replicate** command to start replicating content between a publisher and all associated subscribers immediately. |
| # replicate pubserver |
| **12.** (Optional) Alternatively, use the command scheduler to specify a month, day, and time when you want content replication to occur. For details about using the command scheduler, refer to the Cisco Content Services Switch Administration Guide. |
| # (config)# cmd-sched record content_replication 30 21 3 6 1 |
| **13.** (Recommended) Use the **show publisher** command to display the operational status of the publishing service and content information |
| # show publisher |
| **14.** (Recommended) Use the **show content** command to display content entries in the Content Service Database (CSD) of a CSS. |
| # show content |
service subserver
   subscriber pubserver

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

   content products.html
   protocol tcp
   port 80
   url "/announcements.html"
   add service subserver

### Configuring FTP Access for Publishing and Subscribing

You must associate an access mechanism for each service that offers publishing services and for each service that you configure as a subscriber. Use the `access ftp` command to associate an access mechanism with a service for use during publishing and subscribing activities.

Enter the FTP record as the name of an existing FTP record. Enter the FTP record name as an unquoted text string with no spaces.

Note

When you configure content staging and replication, you must create the FTP record prior to configuring any other content staging and replication command or the feature will not work properly. To create an FTP record, use the `(config) ftp-record` command. For more information see “Creating an FTP Record” earlier in this chapter.

The syntax for this service mode command is:

```
(config-service[pubserver])# access ftp myftprecord
```

To remove a service access mechanism, enter:

```
(config-service[pubserver])# no access ftp
```
Configuring a Publishing Service

A publishing service synchronizes content among associated subscriber services. To move the content during publishing activities, you must configure an access mechanism for the publisher service. Use the (config-service) access ftp command defined earlier in this chapter to configure a mechanism for the publisher service.

When you define the interval to synchronize the subscriber, the interval begins at the time you issue the command. Subscribers that are unavailable for synchronization are placed in an offline state and retried until the operation is completed.

There is no limit on the size of the files that a CSS can replicate between a publisher and a subscriber. When transferring data between a publisher and a subscriber, a CSS creates a virtual pipe so that the replicated files never use the CSS disk. The CSS uses the default buffering associated with the TCP communications stack.

Note

The publisher service does not become active until it has at least one configured subscriber. You do not need to configure the publisher before configuring the subscriber, but the publisher must be configured before the subscriber can receive any content synchronization updates.

Use the publisher command to configure a service as a publishing service. A publishing service can be any type of service that applies to your applications (for example, local or proxy-cache). For a complete description of service types, see Chapter 3, Configuring Services.

The syntax and options for this service mode command are:

- **publisher** - Configures the service as a publishing service.
- **publisher interval minutes** - Defines a recurrent interval in minutes to synchronize content among the subscribers. You can enter this command only after you configure this service as a publishing service. Enter the synchronization interval in minutes. Enter the number from 0 to 3600. The default is 0 which disables the interval.
Configuring Content Staging and Replication

13-16

Cisco Content Services Switch Content Load-Balancing Configuration Guide

- **publisher interval minutes trigger_filename** - Defines a recurrent interval in minutes to synchronize content among the subscribers only when the specified trigger file is modified. Specify the trigger_filename from 1 to 64 characters in length. You can enter this command only after you configure the service as a publishing service.

  To configure publishing on a service, enter:

  ```
  (config-service[pubserver])# publisher
  ```

  To remove publishing on a service, enter:

  ```
  (config-service[pubserver])# no publisher
  ```

  To configure a publisher resynchronization interval, enter:

  ```
  (config-service[pubserver])# publisher interval 120
  ```

  To disable the publisher resynchronization interval by setting it to its default of 0, enter:

  ```
  (config-service[pubserver])# no publisher interval
  ```

**Displaying Publisher Configurations**

Use the **show publisher** command to display the operational status of the publishing service and content information. The options and syntax are:

- **show publisher** - Displays information about all configured publishing services.

- **show publisher publisher_name** - Displays information about the specified publishing service.

- **show publisher publisher_name content {verbose}** - Displays information about the content for the specified publishing service. Include the verbose option to display more detailed content information.

  To display information about the publishing services, enter:

  ```
  (config-service)# show publisher
  ```
Table 13-3 describes the fields in the `show publisher` output.

**Table 13-3  Field Descriptions for the show publisher Command**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>The state of the publisher service.</td>
</tr>
<tr>
<td>Access Type</td>
<td>The associated access mechanism with a service for use during publishing activities. Currently, the FTP record is the only mechanism.</td>
</tr>
<tr>
<td>Access IP</td>
<td>The IP address for the FTP record.</td>
</tr>
<tr>
<td>Access Port</td>
<td>The port number for the FTP record associated with the access mechanism.</td>
</tr>
<tr>
<td>Access Username</td>
<td>The username for the FTP server as defined through the FTP record.</td>
</tr>
<tr>
<td>Access Base Dir</td>
<td>The base directory as defined through the FTP record.</td>
</tr>
<tr>
<td>Published Files</td>
<td>The number of files published from the publisher to the subscriber.</td>
</tr>
<tr>
<td>Published Bytes</td>
<td>The number of bytes published from the publisher to its subscribers.</td>
</tr>
<tr>
<td>Subscribers</td>
<td>The number of subscribers configured to use the publisher.</td>
</tr>
<tr>
<td>Trigger File</td>
<td>The file upon modification that causes the synchronization between the publisher and the subscriber.</td>
</tr>
<tr>
<td>Publish Interval</td>
<td>The interval in seconds when the publisher checks for subscriber synchronization.</td>
</tr>
<tr>
<td>Next Interval</td>
<td>The time when the next publisher synchronization check will occur.</td>
</tr>
<tr>
<td>Managed Files</td>
<td>The number of files that the publisher will replicate.</td>
</tr>
<tr>
<td>Subscribers Synced</td>
<td>The number of synchronized subscribers.</td>
</tr>
<tr>
<td>Managed Dirs</td>
<td>The number of files that the publisher will replicate.</td>
</tr>
<tr>
<td>Managed Bytes</td>
<td>The number of bytes that the publisher is tracking.</td>
</tr>
</tbody>
</table>
Table 13-3 Field Descriptions for the show publisher Command (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Method</td>
<td>The last method that caused the publisher to attempt synchronization with the subscriber. The synchronization methods are:</td>
</tr>
<tr>
<td></td>
<td>• cli - User initiated</td>
</tr>
<tr>
<td></td>
<td>• interval - The configured time interval</td>
</tr>
<tr>
<td></td>
<td>• signal - Trigger file change</td>
</tr>
<tr>
<td></td>
<td>• retry - Retry when a publisher failed to synchronize previously</td>
</tr>
<tr>
<td></td>
<td>• reboot - CSS reboot</td>
</tr>
<tr>
<td>Last Time</td>
<td>The last time when the publisher attempted to synchronize with the subscriber.</td>
</tr>
</tbody>
</table>

Configuring a Subscriber Service

To configure a service as a subscriber to a publishing service, use the subscriber command. You can define a maximum of 31 subscribers per publisher.

You must configure an access mechanism for each subscriber. Use the (config-service) access ftp command defined earlier in this chapter to configure an access mechanism for each subscriber.

To configure a service as a subscriber to a publishing service, enter:

(config-service[subserver])# subscriber pubserver

To unsubscribe the service from a publishing service, enter:

(config-service[subserver])# no subscriber

Note A subscriber’s state will not be ready or will be in access failure until the publisher’s state is ready.
Displaying Subscriber Configurations

Use the `show subscriber` command to display the operational status of the subscriber services. The syntax is:

- `show subscriber` - Displays information about all configured subscriber services
- `show subscriber publisher_name` - Displays information about all subscriber services for the specified publishing service
- `show publisher publisher_name subscriber_name` - Displays information about the specified subscriber service for the specified publishing service

To display information about the subscriber services, enter:

```
(config)# show subscriber
```

Table 13-4 describes the fields in the `show subscriber` output.

**Table 13-4  Field Descriptions for the show subscriber Command**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>The state of the subscriber.</td>
</tr>
<tr>
<td>Access Type</td>
<td>The FTP access mechanism with a service for use during subscribing activities.</td>
</tr>
<tr>
<td>Access IP</td>
<td>The IP address for the FTP record associated with the access mechanism.</td>
</tr>
<tr>
<td>Access Port</td>
<td>The port number for the FTP record associated with the access mechanism.</td>
</tr>
<tr>
<td>Access Username</td>
<td>The username for the FTP record associated with the access mechanism.</td>
</tr>
<tr>
<td>Access Base Dir</td>
<td>The base directory for the FTP record associated with the access mechanism.</td>
</tr>
<tr>
<td>Subscribed Files</td>
<td>The number of files replicated on the subscriber.</td>
</tr>
<tr>
<td>Subscribed Bytes</td>
<td>The number of bytes replicated on the subscriber.</td>
</tr>
</tbody>
</table>
### Configuring a Content Rule for Content Staging and Replication

When you configure content staging and replication, you must configure a URL in a content rule to define which files you want replicated. Then add the subscriber services to the content rule.

**Note**

If you want all files in all directories replicated, you do not need to create a content rule. Create a content rule to specify only those files you want replicated.

**Note**

You cannot configure a URQL with subscriber services in a content rule.

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"
```
For a complete description of configuring URLs, see Chapter 9, Configuring Content Rules.

To add the subscriber services to the content rule, use the `add service` command. For example:

```
(config-owner-content[arrowpoint-products.html])# add service subserver
```

### Configuring Publisher Content Replication

To start replicating content between a publisher and all associated subscribers immediately, use the `replicate` command. You can use this command to replicate to subscribers changes in content on the publisher or to force resynchronization of all content to new subscribers.

Enter the `publisher_name` as the name of the existing publisher. Enter the `subscriber_name` as the name of the subscriber associated with the publisher service.

The syntax and options are:

- `replicate publisher_name` - Resynchronizes any changes to content between the specified publisher and its subscriber services. If the content has not changed, no resynchronization occurs.

- `replicate publisher_name subscriber_name` - Resynchronizes any changes to content between the specified publisher and the specified subscriber service. If the content has not changed, no resynchronization occurs.

- `replicate publisher_name subscriber_name force` - Resynchronizes all content between the specified publisher and the specified subscriber service whether or not content changes have occurred. Use this option only for:
  - New configurations
  - Software upgrades
  - New subscribers
  - A server that has had disk problems
We recommend that you do not use the `force` option for routine content replication. The `force` option does not scan the publisher directory structure and, therefore, does not account for files added to or deleted from the publisher directory structure since the last scan. If you want to use the `force` option in a situation other than the ones listed above, manually replicate the content first to allow the CSS to scan the publisher tree, then apply the `force` option.

For example:
```
# replicate pubserver
```

### Configuring File-Error Handling for Content Replication

Under certain rare circumstances, it is possible for the CSS to encounter a file error during content replication. A file error can occur when an application or a user deletes a file from the publisher tree during a replication operation. If such an event occurs, the scan does not detect the deleted file and during replication the CSS may keep retrying the file until another scan occurs or the file becomes available.

To specify how the CSS handles file errors during content replication, use the `replication file-error` command. The syntax of this global configuration mode command is:

```
replication file-error retry|skip
```

The command options are:

- **retry** - (Default) Replication pauses while the CSS periodically attempts to replicate a missing file
- **skip** - The CSS skips the missing file and continues the replication process

For example:
```
(config)# replication file-error skip
```
Displaying Content

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

The syntax is:

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

The variables and option are:

- **slot slot_number** - Display content from the module located in a specific slot in the CSS 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 chassis.

- **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 64 KB 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.

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

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

Table 13-5 describes the fields in the **show content** output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pieces of Content for Slot</td>
<td>The chassis slot number in which the module resides.</td>
</tr>
<tr>
<td>Subslot</td>
<td>The module slot number in which the Session Processor resides.</td>
</tr>
<tr>
<td>Total Content</td>
<td>The total number of content entries.</td>
</tr>
<tr>
<td>Index</td>
<td>Unique index for a known piece of content in the CSD.</td>
</tr>
</tbody>
</table>
Table 13-5  Field Descriptions for the show content Command (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;address&gt;</td>
<td>The IP address of the piece of content.</td>
</tr>
<tr>
<td>Protocol</td>
<td>The IP Protocol of the piece of content.</td>
</tr>
<tr>
<td>Port</td>
<td>Protocol port of the piece of content.</td>
</tr>
<tr>
<td>Best Effort</td>
<td>The QoS class of the piece of content. This field is not used by</td>
</tr>
<tr>
<td></td>
<td>the CSS at this time.</td>
</tr>
<tr>
<td>Streamed</td>
<td>Identifies if the piece of content is streaming media (video or audio). This field is not used by the CSS at this time.</td>
</tr>
<tr>
<td>URL</td>
<td>The Universal Resource Locator of the piece of content.</td>
</tr>
<tr>
<td>Domain</td>
<td>The domain name of the piece of content.</td>
</tr>
</tbody>
</table>
absolute load
configuration requirements and restrictions 6-13
configuring load variance 6-17, 6-18
optimizing the load number scale 6-17
overview 6-11
running-config example 6-14
sensitivity 6-16
ACA
load balancing 9-34, 12-14
using with server weight and load 6-26
access FTP
demand-based replication 13-8
publishing and subscribing 13-14
ACLS, global bypass counter 8-11
activating
content rule 9-32
global keepalive 4-7
service 3-18
URQL 9-49
Adaptive Session Redundancy 3-2, 5-5, 9-8, 9-64
adding
domain name service to content rule 9-31
service to content rule 9-27
sorry server to content rule 9-29
advanced balance string, configuring for service 3-9
advanced load balancing method
cookies 10-4
specifying for sticky content 10-9
agent, DFP 7-2, 7-5
application type, specifying in a content rule 9-67
ArrowPoint Content Awareness. See ACA
arrowpoint cookie
configuring 10-26
configuring a cookie path 10-29
expiration time, configuring 10-29
ASR. See Adaptive Session Redundancy
assigning
content rule to owner 9-7
IP address to a service 3-4
VIP to owner content 9-8
audience xxiv
Index

B

balance type
  for DNS 9-37
  load balancing 9-34
billing information, specifying for owner 8-4
bypass
  caches 9-64, 12-12
  for failover 9-64, 12-12
  parameter bypass 9-70
  persistence 9-58, 9-60
  transparent caches 9-70
bypassing
  content rule for cache server 3-15

C

cache
  bypass, configuring for a service 3-15, 3-16, 12-19
  bypassing transparent cache 9-70
  clustering 12-7
  hit 12-3
  miss 12-3
cache server
  bypassing content rules 3-15
  caching
    configuration quick start 12-8
    configuring 12-10
    content caching overview 12-2
  overview 12-1
  proxy 12-3
  reverse proxy 12-4
  running-config example 12-9
  specifying service type 12-10
  transparent 12-5
case-sensitivity, specifying for content requests 8-5
cautions
  keepalive type maximum 4-3, 4-12
  symbol overview xxix
  VIP addresses, configuring 9-8
  checksum, calculated for Web page 4-18
Class A keepalives 4-2
Class B keepalives 4-2
CLI conventions xxix
clustering cache servers 12-7
configuration example
  header field group 11-12
  NAT peering 12-21
configuration quick start
  caching 12-8
  content rule 9-5, 11-3
  HTTP header load balancing 11-3
  owner 8-2
  service 3-2, 4-4
  source groups 5-5
  URQL 9-44
  virtual web hosting 9-22
configuring
  caching 12-10
  content rule port information 9-34
  domain name in a content rule 9-11
  flow parameters 2-1
  global keepalive 4-6
  hot-list attributes for content rules 9-38
  load balancing 9-34, 12-14
  protocol for a content rule 9-33
  service 3-1
  service keepalive 3-17, 4-1
  source groups 5-1
  sticky mask 10-16
  sticky parameters 10-5
  string start and end range 10-18
  virtual IP address 9-8

content
  case-sensitivity 8-5
  displaying 9-70, 13-23
  EQL in a URL, specifying 9-55
  removing from owner 9-7, 9-33
  showing 9-70
  specifying an EQL in a URL 9-42
  staging and replication 13-10
  staging and replication file limitations 13-11
  staging and replication quick start 13-12
  sticky with SSL 10-17
  URL, specifying 9-52
  content requests
activating a service 3-18
  case-sensitivity 8-5
  domain name and VIP specific 9-13
  enabling to bypass transparent caches 9-70
  failover 9-64
  global bypass counters 8-11
  multiple domain names 9-12
  primary sorry server redirects 9-29
  redirecting to a service 3-12
content rule
  activating 9-32
  adding a DQL 9-20
  advanced load balancing method for sticky content 10-9
  assigning to owner 9-7
  bypassing for cache server 3-15
  configuration quick start 9-5
  counters, clearing 9-87
  defining failover 9-64
  displaying sticky configurations 10-26, 10-54, 10-72
  domain name, configuring 9-11
  domain name and VIP, using 9-13
  domain name service, adding 9-31
  domain name wildcards, specifying 9-16
  EQLs, configuring 9-40
  header field group 11-10
  header load balancing 11-2
  hot list 13-5
  hot list, configuring 9-38
<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 3, Layer 4, Layer 5</td>
</tr>
<tr>
<td>load balancing for FTP, configuring</td>
</tr>
<tr>
<td>overview</td>
</tr>
<tr>
<td>persistence</td>
</tr>
<tr>
<td>port information, configuring</td>
</tr>
<tr>
<td>primary sorry server, adding</td>
</tr>
<tr>
<td>protocol, configuring</td>
</tr>
<tr>
<td>purpose</td>
</tr>
<tr>
<td>redirecting requests</td>
</tr>
<tr>
<td>removing</td>
</tr>
<tr>
<td>removing a DQL</td>
</tr>
<tr>
<td>replication and staging</td>
</tr>
<tr>
<td>secondary sorry server, adding</td>
</tr>
<tr>
<td>service, adding</td>
</tr>
<tr>
<td>showing</td>
</tr>
<tr>
<td>showing header field configurations</td>
</tr>
<tr>
<td>specifying failover type</td>
</tr>
<tr>
<td>specifying load threshold</td>
</tr>
<tr>
<td>sticky parameters, configuring</td>
</tr>
<tr>
<td>suspending</td>
</tr>
<tr>
<td>wildcards in domain names</td>
</tr>
<tr>
<td>control ports, reclaiming</td>
</tr>
<tr>
<td>cookies</td>
</tr>
<tr>
<td>advanced-balance</td>
</tr>
<tr>
<td>client</td>
</tr>
<tr>
<td>domain, configuring</td>
</tr>
<tr>
<td>e-commerce applications</td>
</tr>
<tr>
<td>end of string characters</td>
</tr>
<tr>
<td>layer 5 content rule</td>
</tr>
<tr>
<td>location</td>
</tr>
<tr>
<td>sticky</td>
</tr>
<tr>
<td>string operation</td>
</tr>
<tr>
<td>string prefix</td>
</tr>
<tr>
<td>string range</td>
</tr>
<tr>
<td>strings, spanning multiple packets</td>
</tr>
<tr>
<td>url</td>
</tr>
<tr>
<td>counters</td>
</tr>
<tr>
<td>content rule, clearing for</td>
</tr>
<tr>
<td>service, clearing for</td>
</tr>
<tr>
<td>custom header field</td>
</tr>
</tbody>
</table>

**D**

default sticky subnet | 10-16 |
delayed binding | 1-7 |
demand-based replication |
configuration quick start | 13-3 |
FTP access | 13-8 |
FTP record | 13-9 |
max age | 13-7 |
max content | 13-8 |
max usage | 13-8 |
running-config example | 13-4 |
service type | 13-6 |
DFP |
agent | 7-2, 7-5 |
configuring | 7-5 |
displaying configuration | 7-9 |
manager 7-2
messages 7-3
overview 7-2
reported weight 7-2
strong encryption 7-5
system flow 7-4
vectors 7-3
weight scaling 7-7
disabling
  DNS in a content rule 9-31
  hot list 9-38
  portmap 5-15
  script keepalive on a service 4-27
  string ASCII conversion 10-22
displaying
  global port mapping statistics 2-27
  noflow port mapping 2-30
DNS
  dnsbalance, leastloaded 9-37
  dnsbalance, preferlocal 9-37
  dnsbalance, roundrobin 9-37
type, specifying for owner 8-5
documentation
  audience xxiv
  chapter contents xxiv
  set xxvi
  symbols and conventions xxix
domain
  adding to a DQL 9-19
  names, configuring for server resolution 5-19
domain hot list, configuring 9-40
domain names
  content rule, configuring in a 9-11
  service, adding to content rule 9-31
  specifying 3-7
  using in a content rule 9-13
  using wildcards in content rules 9-16
Domain Qualifier List. See DQL
DQL
  adding a domain 9-19
  adding to a content rule 9-20, 9-53
  configurations 9-21
  creating 9-18
  describing 9-19
  removing from a content rule 9-20
  showing configurations 9-21
Dynamic Feedback Protocol. See DFP

E
e-commerce
  applications, sticky requirements 10-3
  configuring sticky parameters 10-26
  configuring wireless users 10-49
  using stickiness 10-3
e-mail address, specifying for owner 8-6
EQL
  configuring 9-40
  displaying extensions and descriptions 9-43
displaying in a content rule 9-42
specifying in a URL 9-42, 9-55
example
running-config for location cookie 10-35, 10-38, 10-41
Extension Qualifier List. See EQL

failover
bypass 9-64, 12-12
defining for a content rule 9-64, 12-11
linear 9-64, 12-12
next 9-65, 12-12
file extensions, entering in an EQL 9-41
flow
cleanup 1-9
control block 1-5
dropping long-lived idle connections 1-10
inactivity timeout 2-6
Layer 5 1-6
maintaining long-lived idle connections 1-10
overview 1-4
parameters, configuring 2-1
permanent connections for TCP/UDP ports 2-2
reset reject 9-57
resource reclamation 1-9, 2-6
state table 2-19
statistics, showing 2-4
TCP 1-4
TCP maximum segment size, configuring 2-3
UDP 1-5
fragmentation
collection restrictions 2-11
configuring flow processing for TCP and UDP 2-9
displaying statistics 2-14
maximum assembled size 2-12
minimum fragment size 2-13
overview 2-10
resetting statistics 2-16
FTP
configuring load balancing 5-18, 9-69
connections, configuring a source group 5-18, 9-69
ftp-control, specifying application type 9-68
reclaiming reserved control ports 2-4
FTP access
demand-based content replication 13-8
publishing and subscribing 13-14
FTP record
associating with replication services 13-8, 13-14
demand-based content replication 13-9

garbage collection 1-9
global bypass counters
   descriptions 8-11
   in show summary command 8-10

global port mapping 2-25

graceful shutdown 3-11, 9-28

group
   configuration mode 5-8
   configuring for FTP 5-18
   displaying 5-21
   source 5-1
   domains, configuring for 9-40
   enabling 9-38, 13-5

HTTP 1-7, 10-32
   cookie, configuring for a service 3-9, 3-10
   redirection 9-58, 9-60
   service remapping 9-60
   specifying as application type in a content rule 9-68
   status code 302 9-62

HTTP header field, using in a content rule 11-2

HTTP header load balancing
   configuration quick start 11-3
   overview 11-2
   running-config example 11-4
   spanning multiple packets 9-27, 9-56, 10-9

hash
   balance domainhash 9-35
   balance urlhash 9-36
   keepalive, configuring for 4-19
   XOR hash 9-35, 9-36

header field entry, configuring 11-6

header field group
   associating with a content rule 11-10
   configuration examples 11-12
   configurations, showing in a content rule display 11-11
   creating 11-4
   describing 11-5
   showing 11-11

hot list
   content rules, configuring for 9-38
   disabling 9-38, 13-5

Internet Assigned Name Authority 9-8

internet service providers 9-8

IP fragmentation
   configuration restrictions 2-11
   configuring flow processing for TCP and UDP 2-9
   displaying statistics 2-14
   maximum assembled size 2-12
   minimum fragment size 2-13
   overview 2-10
   resetting statistics 2-16
K

keepalive
activating global 4-7
associating service to global keepalive 4-8
categories 4-2
configuring global 4-4
configuring service 4-4
description, configuring (global) 4-7
frequency, configuring 4-9
global keepalive, creating 4-6
hash, configuring 4-19
HTTP response code, configuring 4-17
IP address, configuring (global) 4-17
maximum keepalive types 4-24
method, configuring 4-15
overview 4-2
port, configuring 4-15
retry period, configuring 4-10
running-config example global 4-6
script 4-23, 4-27, 4-28, 4-29
service, configuring for 3-17, 4-1
show group field 5-22
showing configurations 4-21
suspend, configuring (global) 4-8
TCP graceful socket close (FIN), configuring 4-14
TCP RST, configuring 4-14
type, configuring 4-11
URI, configuring 4-18
virtual web hosting 9-23

L

Layer 3
content rule 10-8
content rule description 9-2
sticky 10-4
Layer 4
content rule 10-8
content rule description 9-2
SSL-Layer 4 fallback, configuring 10-13
sticky 10-4, 10-5
Layer 5
content rule 10-8
content rule, specifying application type 9-67
content rule description 9-2
spanning multiple packets 9-27, 9-56, 10-9
load
absolute, configuring 6-11
absolute sensitivity 6-16
ageout timer, configuring 6-8
calculation, absolute 6-15
configuring for FTP 5-18
configuring for services 6-4
displaying statistics 6-19
manually configuring 6-27
optimizing the absolute load number scale 6-17
relative 6-2
relative, configuring 6-4
reporting, configuring 6-6
showing for services 6-9
step, configuring 6-6
step, configuring for services 6-6
tear down timer, configuring 6-8
teardown timer, configuring 6-8
threshold, configuring for services 6-7
variance 6-18
load balancing
ACA 9-34, 12-14
configuring 9-34, 12-14
destip 9-34, 9-64, 12-14
domain 9-34, 9-64, 12-14
domainhash 9-35, 9-64, 12-15
least connection 9-35, 12-15
roundrobin 9-35, 12-15
srcip 9-35, 9-64, 12-15
url 9-35, 9-64, 12-15
urlhash 9-36, 9-64, 12-16
weighted roundrobin 9-36, 12-15
load threshold, specifying for content rule 9-56
location cookie
configuring 10-42
domain name, configuring 10-43
location service, configuring 10-43
overview 10-31, 10-32
pass-through method example 10-44
quick start 10-33
redirect method example 10-46
running-config example 10-35, 10-38, 10-41
location services, configuring 10-43
max
age, demand-based replication 13-7
connections, configuring for service 3-17
content, demand-based replication 13-8
usage, demand-based replication 13-8
MD5 7-5
MSISDN 10-12, 10-49, 11-8, 11-15
N
NAT 2-24, 5-2
client 5-2
server 2-19, 5-2
NAT peering
configuration example 12-21
configuring 12-22
functions 12-20
running-config example 12-26, 12-27
Network Address Translation Peering. See NAT peering
noflow port mapping 2-29
no-portmap errors 5-15
origin servers  9-64, 12-12
owner
  address, specifying  8-4
  assigning content rule  9-7
  configuration quick start  8-2
  creating  8-3
  DNS type, specifying  8-5
  email address, specifying  8-6
  overview  1-1, 1-3, 9-2
  owner billing information, specifying  8-4
  removing  8-6
  removing content  9-7, 9-33
  running-config example  8-2, 9-7
  showing global bypass counters  8-10
  showing information  8-7

packets, processing fragmented TCP and UDP  2-9
param-bypass  9-70
PAT  2-24, 5-2
permanent connections, configuring for TCP/UDP ports  2-2, 2-3
  persistence  1-8
    configuring in a content rule  9-58
  port
  permanent connections for TCP/UDP ports  2-2
    specifying for a service  3-6
    translation  2-24
  port address translation. See PAT
  port mapping
    global  2-24, 2-25, 2-27
    noflow  2-24, 2-29, 2-30
    source group  5-12
    VIP address range  5-13
  primary sorry server, adding to content rule  9-30
  protocol
    content rule  9-33
    for a service  3-6
    TCP  3-6
    UDP  3-6
  proxy-cache, specifying for service  12-11
  proxy caching  12-3
  publisher
    content replication  13-21
    displaying service configurations  13-16
    service  13-15
quick start
  caching  12-8
  configuring caching  12-8
  content rule  9-5
demand-based content replication 13-3
location cookie 10-33
owner 8-2
service 3-2, 4-4, 6-5, 6-13
SIP 10-52
source groups 5-5
URQLs 9-44
virtual web hosting 9-22

replication
content rule 13-20
content staging 13-10
content staging configuration quick start 13-12
demand-based 13-2
FTP access 13-8
FTP record, creating 13-9
hot lists 13-5
max age 13-7
max content 13-8
max usage 13-8
publisher 13-21
publishing and subscribing 13-14
service type 13-6
replication and staging, configuring a content rule 13-20
reset, TCP 2-17
resetting the back-end connection 1-9
resource reclamation, flow 2-6
reverse proxy caching 12-4
roundrobin
least connection 9-35, 12-15
load balancing 9-35, 12-15
running-config example
absolute load 6-14
caching 12-9
content staging and replication 13-13
demand-based replication 13-4
global keepalive 4-6

realaudio-control, specifying as application type 9-68
reclaiming flow resources 2-6
redirection 1-8, 10-32
HTTP 9-60
requests for content 9-62
relative load
configuring 6-1, 6-4
displaying statistics 6-19
overview 6-2
running-config example 6-6
remapping 1-9
configuring in a content rule 9-58
showing 9-63
remote service 3-12
removing
content rule 9-33
content rule from owner 9-7
owner 8-6

Cisco Content Services Switch Content Load-Balancing Configuration Guide
Index

HTTP header load balancing 11-4
location cookie 10-35, 10-38, 10-41
NAT peering 12-26, 12-27
owner 8-2, 9-7
relative load 6-6
service 3-3
source group 5-7
URQL 9-45
virtual web hosting 9-24

S
script keepalives 4-23
configuring 4-26
displaying 4-27
maximum keepalive types 4-24
overview 4-23
status codes 4-28
upgrading WebNS software 4-29
usage considerations 4-25
secondary sorry server, adding to a content rule 9-30
Secure Socket Layer. See SSL
server
load, configuring for ACA 6-27
order in which types are hit 3-14, 9-26
primary sorry 9-29
secondary sorry 9-30
types, how CSS handles 3-14
weight and load, using with ACA 6-26
serverdown failover, configuring for sticky applications 10-15
service
absolute load 6-11
access, configuring 3-14
activating 3-18
adding to a content rule 9-25, 9-27
advanced balanced string, configuring 3-9
assigning an IP address 3-4
bypassing content rules for cache server 3-15
cache bypass, configuring 3-15, 3-16
configuration quick start 3-2, 4-4, 6-5, 6-13
configuring 3-1
configuring cache bypass 12-19
configuring for NAT peering 12-22
counters, clearing 3-25, 9-87, 9-88
creating 3-4
global load reporting, configuring 6-6
global load threshold, configuring 6-7
graceful shutdown 3-11, 9-28
HTTP cookie, configuring an 3-9, 3-10
keepalive, configuring 3-17, 4-1
load, configuring 6-1, 6-27
load ageout timer, configuring 6-8
load step, configuring 6-6
load tear down timer, configuring 6-8
max connections, configuring 3-17
maximum TCP connections 3-17
order in which types are hit 3-14, 9-26
overview 1-1, 1-3, 9-2
port, specifying 3-6
primary sorry 9-29
protocol, specifying 3-6
publisher 13-15
remapping 9-58
remapping and HTTP redirection, configuring 9-60
removing from source group 5-10
replication 13-2
running-config example 3-3
secondary sorry 9-30
showing configuration 3-19
showing load 6-9
specifying a protocol 3-6
specifying type 3-12, 12-10
subscriber 13-18
suspending 3-18
weight, configuring 3-10
service type
local 12-10
nci-direct-type 3-12, 12-10
nci-info-type 3-12, 12-10
proxy-cache 3-12, 12-11
redirect 3-12
redundancy-up 3-12
replication cache 12-11
replication cache redirect 3-12, 12-11, 13-6
replication-store 3-12, 13-6
replication-store redirect 3-12, 13-6
specifying for replication 13-6
ssl-accel 3-13
ssl-accel-backend 3-13
ssl-init 3-13
transparent-cache 3-13, 12-11
showing
content 9-70
content rules 9-72
global bypass counters 8-10
global keepalives 4-21
header field groups 11-11
owner information 8-7
remapping 9-63
service configuration 3-19
SIP
overview 10-11
quick start 10-52
specifying as advanced-balance method in a content rule 10-11
SIP, specifying as application type in a content rule 9-68
sorry server
adding a primary to a content rule 9-29
adding a secondary to a content rule 9-30
source group
configuring 5-1
configuring for domain name resolution 5-19
configuring for FTP connections 5-18
displaying 5-21
port mapping 5-12
removing service 5-10
Index

running-config example 5-7
spanning multiple packets 9-27, 9-56, 10-9
spoofing 1-6, 12-20

SSL
configuring sticky content for 10-17
specifying as advanced-balance method in a content rule 10-11
specifying as application type in a content rule 9-68
SSL-Layer 4 fallback, configuring 10-13
staging and replication, configuring for content 13-10

statistics
flow 2-4
fragmentation 2-16
global port mapping 2-27
noflow port mapping 2-30
showing flows 2-4

sticky
configuring failover 10-15
configuring sticky no cookie found action 10-25
configuring string operation 10-19
configuring string start and end range 10-18
default subnet 10-16
displaying configuration 10-54, 10-64, 10-72
e-commerce application requirements 10-3
end of string characters 10-23
handling multiple string matches 10-22
inactive timeout 10-16
mask 10-16

overview 10-2
procedure for configuring on the CSS 10-5
purpose 10-3
serverdown failover, configuring for sticky applications 10-15
showing configurations 10-25
skip length 10-25
SSL-Layer 4 fallback, configuring 10-13
string operation, choosing a destination server 10-19
string prefix 10-24
string process length 10-24
string range, configuring for stickiness 10-18
using string ASCII conversion 10-22
WAP load balancing 10-49

sticky content
configuring for SSL 10-17
specifying an advanced load balancing method 10-9
specifying in a content rule 10-9
sticky parameters
configuring 10-5
configuring for e-commerce 10-26
sticky string operation, choosing a destination server 10-19

sticky table
displaying configurations 10-64, 10-72
overview 10-2
subscriber service
configuring 13-18
displaying configurations 13-19
suspending
  content rule 9-32
  service 3-18
symbol overview xxix

T

TCP 2-19
  3-way handshake 1-6
  flow 1-4
  flow reset reject 9-57
  IP packet fragments 2-9
  keepalive type tcp 4-14
  max connections, configuring for service 3-17
  port destination number, specifying 3-6
  port destination port number, specifying 9-34
  protocol, specifying for service 3-6
  protocol, specifying in content rule 9-33
  reset 2-17
TCP ports
  destination number, specifying 3-6
  permanent connections, configuring 3-6
Telnet, reclaiming reserved control ports 2-4
threshold
  global load threshold 6-7
  load threshold, specifying 9-56
timeout, flow inactivity 2-6
transparent-cache
  bypassing 9-70

specifying for service 12-11
transparent caching 12-5
type, specifying for service 3-12

U

UDP 2-19
  DNS port numbers, mapping 2-29
  flow 1-5
  IP packet fragments 2-9
  port destination port number, specifying 3-6, 9-34
  protocol, specifying for service 3-6
  protocol, specifying in content rule 9-33
Universal Resource Locator. See URL
upgrading WebNS software, script keepalives 4-29
URI, specifying for HTTP keepalive 4-16
URL
  configuring in a URQL 9-46
  content, specifying for 9-52
  defining in a URQL 9-47
  strings, spanning multiple packets 9-27, 9-56, 10-9
URQL
  activating 9-49
  adding to content rule 9-48
  configuring 9-44
  creating 9-46
  describing 9-49
Index

designating URL domain name 9-48
displaying configurations 9-50
quick start 9-44
running-config example 9-45
suspending 9-50

V

virtual IP address, configuring 9-8
virtual web hosting
  running-config example 9-24
virtual web hosting, configuring 9-21

W

WAP 10-12, 10-49, 11-8, 11-15
warning symbol overview xxix
web page, verifying checksum 4-16
weight
  configuring for a service 3-10
  graceful shutdown, specifying 3-11, 9-28
  reported by DFP 7-2
weighted roundrobin, load balancing 9-36, 12-15
wildcards
  domain names in content rules 9-14
  using in content rule domain names 9-16
Wireless Application Protocol. See WAP

X

XOR hash
  used in domainhash balance algorithm 9-35, 12-15
  used in urlhash balance algorithm 9-36, 12-16