Cisco Content Services Switch Advanced Configuration Guide

Software Version 7.10
December, 2002

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Customer Order Number: DOC-7813887=
Text Part Number: 78-13887-03

Cisco Content Services Switch Advanced Configuration Guide

FIGURES

About This Guide

This guide provides instructions for configuring the advanced features of the Cisco 11500 and 11000 series content services switch (hereinafter referred to as the CSS). Information in this guide applies to all CSS models except where noted. For information on CSS administration, refer to the *Cisco Content Services Switch Administration Guide*. For information on basic CSS configuration, refer to the *Cisco Content Services Switch Basic Configuration Guide.*

The CSS software is available in a Standard or 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 SSH are optional features.

Note You must enter a software license key when you boot the CSS for the first time. After you boot the CSS, you can activate a CSS software option (for example, SSH) that you purchased using the **license** command. For more information, refer to the *Cisco Content Services Switch Hardware Installation Guide*, Chapter 3, Booting and Configuring the CSS.

Note If you configure your 11000 series CSS for Proximity Database, you cannot use the CSS for load balancing. For details on Proximity Database, refer to Chapter 5, Configuring Network Proximity.

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 section describes the chapters and contents in this guide.

Chapter	Description
Chapter 1, Configuring the CSS Domain Name Service	Configure the Domain Name Service (DNS) on a CSS to translate domain names into IP addresses.
Chapter 2, Configuring DNS Sticky	Configure DNS Sticky on a CSS to maintain persistence on a server for e-commerce clients.
Chapter 3, Configuring a CSS as a Content Routing Agent	Configure a CSS as a content routing agent (CRA) to enhance a user's browser experience.
Chapter 4, Configuring a Client Side Accelerator	Configure a CSS as a Client Side Accelerator (CSA) to accelerate the retrieval of domain content.
Chapter 5, Configuring Network Proximity	Configure Network Proximity on a CSS to improve network performance.
Chapter 6, Configuring VIP and Virtual IP Interface Redundancy	Configure VIP and virtual IP interface redundancy on a CSS to maintain network integrity.
Chapter 7, Configuring Redundant Content Services Switches	Configure box-to-box redundancy between two mirrored CSSs.

Chapter	Description
Chapter 8, Configuring Content Replication	Configure demand-based content replication and content synchronization using publisher and subscriber services on a CSS.
Chapter 9, Configuring SSL Termination on the CSS	Configure the CSS and the SSL Acceleration Module to perform Secure Sockets Layer (SSL) termination between the client and the Web servers.
Chapter 10, Configuring Firewall Load Balancing	Configure firewall load balancing between CSSs for enhanced security.
Chapter 11, Using the CSS Scripting Language	Use the CSS scripting language to automate configuration tasks and create script keepalives. Includes example scripts.

Related Documentation

In addition to the *Cisco Content Services Switch Advanced Configuration Guide*, the Content Services Switch documentation includes the following publications.

Document Title	Description
Release Note for the Cisco 11500 Series Content Services Switch	Provides information on operating considerations, caveats, and CLI commands for the Cisco 11500 series CSS.
Release Note for the Cisco 11000 Series Content Services Switch	Provides information on operating considerations, caveats, and CLI commands for the Cisco 11000 series CSS.
Cisco 11500 Series Content Services Switch Hardware Installation Guide	Provides information for installing, cabling, and booting the 11500 series CSS. In addition, this guide provides information about CSS specifications, cable pinouts, and troubleshooting.

Document Title	Description
Cisco Content Services Switch Getting Started Guide	Provides information for installing, cabling, and booting the 11000 series CSS. In addition, this guide provides information about CSS specifications, cable pinouts, and troubleshooting.
Cisco Content Services Switch Administration Guide	Describes how to perform administrative tasks on the CSS including logging into the CSS, upgrading your CSS software, and configuring the following: • Management ports, interfaces, and circuits • DNS, ARP, RIP, IP, and bridging features • OSPF • Logging, including displaying log messages and interpreting sys.log messages • User profile and CSS parameters • SNMP • RMON • Offline Diagnostic Monitor (Offline DM) menu

Document Title	Description
Cisco Content Services Switch Basic Configuration Guide	Describes how to perform basic CSS configuration tasks, including: • Services • Owners • Content rules • Sticky parameters • HTTP header load balancing • Source groups, Access Control Lists (ACLs), Extension Qualifier Lists (EQLs), Uniform Resource Locator Qualifier Lists (URQLs), Network Qualifier Lists (NQLs), and Domain Qualifier Lists (DQLs) • Caching
Cisco Content Services Switch Command Reference	Provides an alphabetical list of all CSS Command Line Interface commands by mode including syntax, options, and related commands.
Cisco Content Services Switch Device Management User's Guide	Provides an overview on using the WebNS Device Management user interface, an HTML-based Web application that you use to configure and manage a CSS.

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.

Italics text indicates the first occurrence of a new term, book title, and emphasized text.

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.

- A bulleted list indicates that the order of the list topics is unimportant.
  - An indented list indicates that the order of the list subtopics is unimportant.

Obtaining Documentation

These sections explain how to obtain documentation from Cisco Systems.

World Wide Web

You can access the most current Cisco documentation on the World Wide Web at this URL:

http://www.cisco.com

Translated documentation is available at this URL:

http://www.cisco.com/public/countries_languages.shtml

Documentation CD-ROM

Cisco documentation and additional literature are available in a Cisco Documentation CD-ROM package, which is shipped with your product. The Documentation CD-ROM is updated monthly and may be more current than printed documentation. The CD-ROM package is available as a single unit or through an annual subscription.

Ordering Documentation

You can order Cisco documentation in these ways:

- Registered Cisco.com users (Cisco direct customers) can order Cisco product documentation from the Networking Products MarketPlace:

  http://www.cisco.com/cgi-bin/order/order_root.pl

- Registered Cisco.com users can order the Documentation CD-ROM through the online Subscription Store:

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

- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco Systems Corporate Headquarters (California, U.S.A.) at 408 526-7208 or, elsewhere in North America, by calling 800 553-NETS (6387).

Documentation Feedback

You can submit comments electronically on Cisco.com. In the Cisco Documentation home page, click the **Fax** or **Email** option in the "Leave Feedback" section at the bottom of the page.

You can e-mail your comments to bug-doc@cisco.com.

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

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

We appreciate your comments.

Obtaining Technical Assistance

Cisco provides Cisco.com as a starting point for all technical assistance. Customers and partners can obtain online documentation, troubleshooting tips, and sample configurations from online tools by using the Cisco Technical Assistance Center (TAC) Web Site. Cisco.com registered users have complete access to the technical support resources on the Cisco TAC Web Site.

Cisco.com

Cisco.com is the foundation of a suite of interactive, networked services that provides immediate, open access to Cisco information, networking solutions, services, programs, and resources at any time, from anywhere in the world.

Cisco.com is a highly integrated Internet application and a powerful, easy-to-use tool that provides a broad range of features and services to help you with these tasks:

- Streamline business processes and improve productivity
- Resolve technical issues with online support
- Download and test software packages

- Order Cisco learning materials and merchandise
- Register for online skill assessment, training, and certification programs

If you want to obtain customized information and service, you can self-register on Cisco.com. To access Cisco.com, go to this URL:

http://www.cisco.com

Technical Assistance Center

The Cisco Technical Assistance Center (TAC) is available to all customers who need technical assistance with a Cisco product, technology, or solution. Two levels of support are available: the Cisco TAC Web Site and the Cisco TAC Escalation Center.

Cisco TAC inquiries are categorized according to the urgency of the issue:

- Priority level 4 (P4)—You need information or assistance concerning Cisco product capabilities, product installation, or basic product configuration.
- Priority level 3 (P3)—Your network performance is degraded. Network functionality is noticeably impaired, but most business operations continue.
- Priority level 2 (P2)—Your production network is severely degraded, affecting significant aspects of business operations. No workaround is available.
- Priority level 1 (P1)—Your production network is down, and a critical impact to business operations will occur if service is not restored quickly. No workaround is available.

The Cisco TAC resource that you choose is based on the priority of the problem and the conditions of service contracts, when applicable.

Cisco TAC Web Site

You can use the Cisco TAC Web Site to resolve P3 and P4 issues yourself, saving both cost and time. The site provides around-the-clock access to online tools, knowledge bases, and software. To access the Cisco TAC Web Site, go to this URL:

http://www.cisco.com/tac

All customers, partners, and resellers who have a valid Cisco service contract have complete access to the technical support resources on the Cisco TAC Web Site. The Cisco TAC Web Site requires a Cisco.com login ID and password. If you have a valid service contract but do not have a login ID or password, go to this URL to register:

http://www.cisco.com/register/

If you are a Cisco.com registered user, and you cannot resolve your technical issues by using the Cisco TAC Web Site, you can open a case online by using the TAC Case Open tool at this URL:

http://www.cisco.com/tac/caseopen

If you have Internet access, we recommend that you open P3 and P4 cases through the Cisco TAC Web Site.

Cisco TAC Escalation Center

The Cisco TAC Escalation Center addresses priority level 1 or priority level 2 issues. These classifications are assigned when severe network degradation significantly impacts business operations. When you contact the TAC Escalation Center with a P1 or P2 problem, a Cisco TAC engineer automatically opens a case.

To obtain a directory of toll-free Cisco TAC telephone numbers for your country, go to this URL:

http://www.cisco.com/warp/public/687/Directory/DirTAC.shtml

Before calling, please check with your network operations center to determine the level of Cisco support services to which your company is entitled: for example, SMARTnet, SMARTnet Onsite, or Network Supported Accounts (NSA). When you call the center, please have available your service agreement number and your product serial number.

Configuring the CSS Domain Name Service

This chapter provides an overview of the CSS Domain Name Service (DNS) feature and describes how to configure it for operation. Information in this chapter applies to all CSS models, except where noted.

Note The CSS Domain Name Service feature is part of the CSS Enhanced feature set.

This chapter provides the following sections:

CSS Domain Name Service Overview

The CSS Domain Name Service (DNS) feature enables you to configure one or more CSSs together to construct highly available, distributed, and load-sensitive Web sites. Groups of CSSs may host many distributed Web sites concurrently. These groups make decisions that can be configured independently for each distributed Web site using local and remote load-balancing information.

CSSs that are configured together for DNS form a *content domain*. Within the content domain, CSSs are known as peers. You can configure peers to exchange content rules, load-balancing information, and service availability.

Each CSS becomes aware of all the locations for the content associated with a domain name and the operational state and load of the location. The CSS can then intelligently direct clients to a site where they can best obtain the desired content. In addition, a CSS never sends a client to a location that is overburdened or out of service.

You can use DNS to configure a CSS as a DNS authoritative server. A CSS defined as an authoritative DNS server resolves DNS names when requested by a client.

For example, when a user clicks on a URL on a Web page:

1. The client asks the locally configured DNS server for a translation of a domain name to an IP address. The DNS server contains the CSS virtual IP address (VIP) and DNS names.
2. The DNS server requests address resolution from the CSS DNS authoritative server.
3. The CSS DNS authoritative server returns the VIP address of the best location (that is, server availability and load) where the client can retrieve the content.
4. The DNS server responds to the client with the VIP.
5. The client uses the VIP to access the content.

Note The CSS implementation of DNS server functionality is a streamlined, endnode-only approach. The CSS does not support zone transfer among other DNS servers. However, each CSS configured in a content domain can act as the authoritative DNS server.

CSS Application Peering Protocol Overview

CSSs configured within the same content domain initiate communication over Application Peering Protocol (APP) sessions with their peers upon system bootup or when peers first become connected through an APP session. Thereafter, changes in local configurations are relayed to the peers automatically as they occur. When the APP session is up, the peers exchange owner names according to the DNS exchange policies configured for each owner.

For each owner that a CSS is configured to share with its peers, the CSS sends the locally configured content rules and DNS name information. Upon receiving a peer's content rule information, the CSS compares each DNS name and content rule to its local configuration.

Content rules that:

- Match a locally configured content rule cause a *dynamic service* to be added automatically to the local content rule. The local content rule points to the peer for an alternate location for the content.
- Do not have a corresponding local entry cause the CSS to automatically create a *dynamic content rule* containing a dynamic service that points to the peer that has the content rule configured.

The determination of whether or not a content rule matches is based strictly on content rule name. Peers having matching content rule names must have exact copies of rule definitions with the exception of VIP addresses. DNS names do not need to be identical.

Note CSSs do not include dynamic services or dynamic content rules in their running- or startup-config files. Dynamic services and dynamic content rules are temporary and are removed when the peer connection terminates.

For example, when a client requests *www.arrowpoint.com*:

1. The client browser asks the locally configured DNS server for a translation to an IP address.
2. The DNS server round-robins an address resolution request to one of the CSSs.

3. The selected CSS DNS authoritative server determines server availability based on the DNS balance type.

   If the CSS is configured as DNS balance type **dnsbalance preferlocal** and is:

   - Able to locally handle the request for this DNS name, it returns the local VIP to the DNS server.
   - Not able to handle the request for this DNS name (the server has reached a defined load threshold or is unavailable), the CSS returns the dynamic content rule VIP to the DNS server.

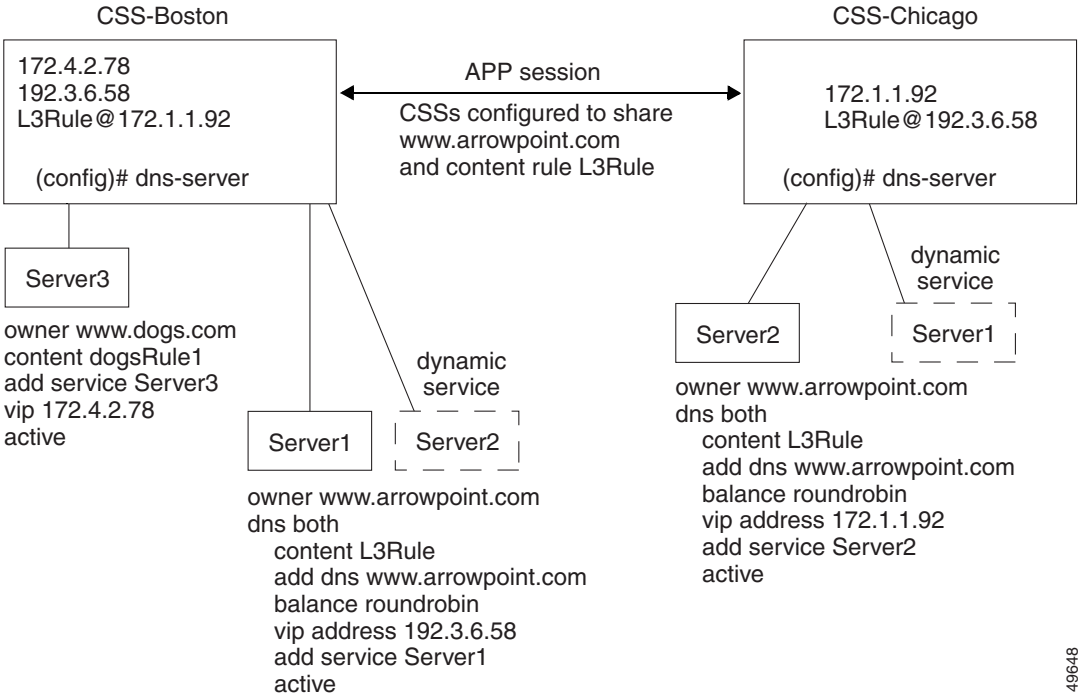
   If the CSS is configured as DNS balance type **dnsbalance roundrobin** the CSS resolves requests by evenly distributing the load to resolve domain names among local and remote content domain sites.

   For information on configuring DNS balance types, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 3, Configuring Content Rules.

4. The DNS server forwards the resolved VIP to the client.

Figure 1-1 illustrates two peer CSSs configured as authoritative DNS servers. Each CSS knows its local content rule VIPs and dynamic content rule VIPs. The @ sign within a content rule VIP indicates a dynamic content rule. Owner *www.arrowpoint.com* is configured for **dns both** (push and accept owner *www.arrowpoint.com* and its content rule *L3Rule*). Even though CSS-Boston contains owners *www.arrowpoint.com* and *www.dogs.com*, only owner *www.arrowpoint.com* and content rule *L3Rule* are shared between the CSSs.

Figure 1-1 CSS Configured as Authoritative DNS



Configuring Application Peering Protocol

When two CSSs communicate, they use an Application Peering Protocol (APP) session. An APP session allows the exchange of content information between a pair of configured CSSs. APP provides a guaranteed and private communications channel for this exchange. Two or more CSSs that are configured to exchange content rules over APP sessions form a content domain and are considered peers.

Note The Application Peering Protocol feature is part of the CSS Enhanced feature set.

To configure APP sessions, use the **app** command. The options for this global configuration mode command are:

- **app** - Enables all APP sessions
- **app framesz** - Sets the maximum frame size allowed by the APP
- **app port** - Sets the TCP port that listens for APP connections
- **app session** - Creates an APP session

For example:

```
(config)# app
```

To disable all APP sessions, enter:

```
(config)# no app
```

Configuring APP Frame Size

To set the maximum size allowed by the APP, use the **app framesz** command. Enter the maximum APP frame size from 10240 to 65535. The default is 10240. Upon session establishment, peers select the smallest configured frame size to use for session communication. For example, CSS-A is configured for frame size 5000 and CSS-B is configured for frame size 6000. Once the session is established, CSS-B will use frame size 5000.

For example:

```
(config)# app framesz 5096
```

To restore the default frame size to 10240, enter:

```
(config)# no app framesz
```

Configuring APP Port

To set the TCP port number, use the **app port** command. This port listens for APP connections. Enter a port number from 1025 to 65535. The default TCP port is 5001.

For example:

```
(config)# app port 21
```

To restore the default port number to 5001, enter:

```
(config)# no app port
```

Configuring an APP Session

To create an APP session between two CSSs, use the **app session** command. The CSSs use APP sessions to create a content domain that shares the same content rules, load, and DNS information with each other.

The syntax and options for this global configuration mode command are:

app session *ip_address* {*keepalive frequency* {**authChallenge**|**authNone** *session_secret* {**encryptMd5hash**|**encryptNone** {**rcmdEnable**|**rcmdDisable**}}}}

Note The **authChallenge**|**authNone** and **encryptMd5hash**|**encryptNone** APP command options must be identical for both CSSs in an APP session or the session will not come up.

The **keepalive** and **rcmd** command options do not have to be identical between CSS peers.

The variables and options are:

- *ip_address* - IP address for the peer CSS.
- *keepalive frequency* - Optional time in seconds between sending keepalive messages to this peer CSS. Enter an integer from 14 to 255. The default is 14.
- **authChallenge**|**authNone** - Optional authentication method for the session. Enter either **authChallenge** for Challenge Handshake Authentication Protocol (CHAP) method or **authNone** for no authentication method. The default is no authentication.

- *session_secret* - Secret used with AuthChallenge to authenticate a peer or used with encryptMd5hash to provide an MD5hash encryption scheme for the session. Enter an unquoted text string with a maximum of 32 characters.
- **encryptMd5hash**|**encryptNone** - Optional encryption method for the packets. Enter either **encryptMd5hash** for MD5 base hashing method or **encryptNone** for no encryption method. The default is no encryption.
- **rcmdEnable**|**rcmdDisable** - Optional setting for sending remote CLI commands to the peer through the **rcmd** command. Enter either **rcmdEnable** to allow the sending of CLI commands or **rcmdDisable** to disallow the sending of CLI commands. The default setting is enabled.

To terminate an APP session, enter the **no app session** command and an IP address:

```
(config)# no app session 192.2.2.2
```

For example, to configure a CSS in Boston (IP address 172.1.1.1) to be a peer of a CSS in Chicago (IP address 192.2.2.2), use the **app** command to configure:

```
CSS-Boston(config)# app session 192.2.2.2
CSS-Chicago(config)# app session 172.1.1.1
```

Using the RCMD Command

To issue remote CLI commands to a CSS peer, use the **rcmd** command. Before you can use this command, use the **(config) app session** command to configure an APP session. The **rcmd** command is available in SuperUser mode.

The syntax for this command is:

rcmd *ip_address* or *host* **"***CLI command* {;*CLI command...*}**"** {*timeout_response*}

The variables are:

- *ip_address* or *hostname* - The IP address or host name for the peer.
- *CLI command* - One or more CLI commands you want to issue to the peer. Enter the command, its options, and variables exactly. Enclose the command text string in quotes (“”). When entering multiple CLI commands, insert a semicolon (;) character to separate each command.

Note You cannot issue **grep**, **grep** within a script command, or **redirect** commands.

- *timeout_response* - The optional amount of time, in seconds, to wait for the output command response from the peer. Enter an integer from 3 to 300 (5 minutes). The default is 3 seconds.

Note By default, the APP session is configured to allow sending remote commands to a CSS peer. If you disable this function using the **no app session** command, use the **(config) app session** command to enable it.

For example:

```
# rcmd 192.2.2.2 "show domain" 10
```

Displaying APP Configurations

To display the APP configuration or session information, use the **show app** command. APP is the method in which private communications links are configured between CSSs in the same content domain. A content domain consists of group of CSSs configured to exchange content information.

The syntax and options for this command are:

- **show app** - Displays whether APP is enabled, its port number, and frame size setting. For example:

  ```
  (config)# show app
  ```

- **show app session** - Displays all IP session information including the session ID, IP address, and state. For example:

  ```
  (config)# show app session
  ```

- **show app session** *ip_address* - Displays the IP session information including the session ID, IP address, and state. For example:

  ```
  (config)# show app session 192.168.10.10
  ```

- **show app session verbose** - In addition to displaying the IP session information, the verbose keyword displays detailed information about the IP configuration parameters for the session including the local address, keepalive frequency, authorization and encryption type, frame size, and packet activity. For example:

  ```
  (config)# show app session verbose
  ```

- **show app session** *ip_address* **verbose** - Displays the same information as the **show app session verbose** command except that it displays information only for the specified IP address. For example:

  ```
  (config)# show app session 192.168.10.10 verbose
  ```

  To display a list of IP addresses, enter **show app ?** or **show app session verbose ?**.

Table 1-1 describes the fields in the **show app** output.

Table 1-1 Field Descriptions for the show app Command

Field	Description
Enabled or Disabled	Whether all APP sessions are enabled or disabled.
PortNumber	The TCP port number that listens for APP connections. The port can be a number from 1 to 65535. The default is 5001.
MaxFrameSize	The maximum frame size allowed on an APP channel between CSSs. The maximum frame size is a number from 10240 to 65535. The default is 10240.

Table 1-2 describes the fields in the **show app session** output.

Table 1-2 Field Descriptions for the show app session Command

Field	Description
App Session Information	DNS-resolved hostname as defined through the **host** command.
Session ID	The unique identifier for the session.
IP Address	The IP address for the peer CSS.

Table 1-2 Field Descriptions for the show app session Command (continued)

Field	Description
State	The current state of the session. The possible states include: • **APP_SESSION_STOP** - Indicates that the session is about to be deleted • **APP_SESSION_INIT** - Indicates that the session is initializing • **APP_SESSION_OPEN** - Indicates that the connection to the peer has been made • **APP_SESSION_AUTH** - Indicates that the authentication is occurring • **APP_SESSION_UP** - Indicates that the session is up • **APP_SESSION_DOWN** - Indicates that the session is down
Local Address	The local interface address. If the session is down, no address is displayed.
rcmdEnable	The setting for the sending of remote CLI commands to the peer through the **rcmd** command. The Enabled setting allows the sending of CLI commands. The Disabled setting disallows the sending of CLI commands. The default setting is enabled.
KalFreq	The time in seconds between sending keepalive messages to this peer CSS. The time can be from 14 to 255 seconds (15 minutes). The default is 14.
Auth Type	The authentication method for the session. The method is either authChallenge for Challenge Handshake Authentication Protocol (CHAP) method or none for no authentication method. The default is no authentication.
Encrypt Type	The encryption method for the packets. The method is either encryptMd5hash for MD5 base hashing method or none for no encryption method. The default is no encryption.

Table 1-2 Field Descriptions for the show app session Command (continued)

Field	Description
MaxFrameSz	The maximum frame size allowed on an APP channel between CSSs. The frame size is a number from 10240 to 65535. The default is 10240.
Pkts Tx	The number of packets sent during the session.
Pkts Rx	The number of packets received during the session.
Pkts Rej	The number of packets rejected during the session.
Last UP event	The day and time of the most recent UP event.
Last DOWN event	The day and time of the most recent DOWN event.
FSM Events	Finite State Machine events as related to the state field.
STOP	The number of APP_SESSION_STOP events. This field will always be at 0.
INIT	The number of APP_SESSION_INIT events.
OPEN	The number of APP_SESSION_OPEN events.
AUTH	The number of APP_SESSION_AUTH events.
UP	The number of APP_SESSION_UP events.
DOWN	The number of APP_SESSION_DOWN events.
Attached Apl	The application identifier.

Configuring a CSS as an Authoritative DNS Server

Use the **dns-server** command and its options to enable DNS server functionality on a CSS. The options for this global configuration mode command are:

- **dns-server** - Enables the DNS server functionality on a CSS.
- **dns-server zone** - Enables a Proximity Domain Name Server (PDNS) in a Network Proximity configuration, or enables zone-based DNS in a non-proximity configuration.
- **dns-server zone load** - Enables the processing and sharing of local DNS server load information.
- **dns-server bufferCount** - Modifies the DNS response buffer count.
- **dns-server respTasks** - Modifies the DNS responder task count.
- **dns-server forwarder** - Enables a DNS server forwarder (a CSS or a fully-functional BIND server), which resolves DNS requests that a CSS cannot resolve.

Enabling a DNS Server

Use the **dns-server** command to enable the DNS server functionality on a CSS. The syntax of this global configuration mode command is:

dns-server

For example:

```
(config)# dns-server
```

To disable DNS server functionality on a CSS, enter:

```
(config)# no dns-server
```

Configuring DNS Server Zones

Use the **dns-server zone** command to enable zone-based DNS on a CSS in a global server load-balancing (GSLB) environment. In a Network Proximity configuration, use this command to enable a PDNS. For more information on Network Proximity, refer to Chapter 5, Configuring Network Proximity.

Note Before you enable a Proximity Domain Name Server (PDNS), you must configure APP-UDP and APP. Zone-based DNS also requires APP. For details on configuring APP-UDP, refer to Chapter 5, Configuring Network Proximity in the section "Configuring APP-UDP and APP". For details on configuring APP, see "Configuring Application Peering Protocol" earlier in this chapter.

The syntax for this global configuration mode command is:

dns-server zone *zoneIndex* {**tier1**|**tier2** {**"***description***"**
{**roundrobin**|**preferlocal**|**leastloaded**|**srcip**|**weightedrr**|*ip_address*
{**roundrobin**|**preferlocal**|**leastloaded**|**srcip**|**weightedrr**}}}}

The **dns-server zone** command supports the following variables and options:

- *zone_index* - The numerical identifier of the DNS server zone. The *zone_index* value must be a unique zone number on the network. In a Network Proximity configuration, this number must match the zone index configured on the Proximity Database (PDB). Enter an integer from 0 to 15. Valid entries are 0 to 5 for tier 1 and 0 to 15 for tier 2. There is no default.
- **tier1**|**tier2** - The optional maximum number of zones (peers) that may participate in the CSS peer mesh. The tier you select must be the same as the tier for the other CSSs participating in the peer mesh. Enter **tier1** for a maximum of 6 zones. Enter **tier2** for a maximum of 16 zones. The default is tier1.
- *description* - Optional quoted text description of the DNS server zone. Enter a quoted text string with a maximum of 20 characters.
- **roundrobin**|**preferlocal**|**leastloaded**|**srcip -** The optional load-balancing method that the DNS server uses to select returned records when a PDB is unavailable or not configured.
  - **roundrobin** - The CSS cycles among records available at the different zones. This load-balancing method is the default.

  - **preferlocal** - The CSS returns a record from the local zone whenever possible, using roundrobin when it is not possible.
  - **leastloaded** - The CSS reports loads and selects a record from the zone that has the least traffic.
  - **srcip** - The CSS uses a source IP address hash to select the zone index to return to the client.
  - **weightedrr** - This option is available only on an 11500 series CSS. The CSS gives a zone priority over other zones in a peer mesh according to the assigned domain weights. Each CSS maintains an internal list of services ordered from highest to lowest according to weight. The heaviest server (the server with the highest weight number) receives DNS requests until it reaches its maximum number of requests, then the next heaviest server receives DNS requests until it reaches its maximum, and so on. When all the servers have reached their maximum number of requests, the CSS resets the counters and the cycle starts over again.

    When you add a new DNS zone, each CSS adds the new servers to its list by weight. In this case, the CSSs do not reset their hit counters. This process prevents flooding of the heaviest zone every time you add or remove a zone.

    For example, a domain with a weight of 10 in the local zone will receive twice as many hits as the same domain with a weight of 5 in another zone. Use the **dns-record** command to assign domain weights. See the "Configuring Domain Records" section later in this chapter.

- *ip_address* - The IP address of the PDB. In a proximity configuration, enter the address in dotted-decimal notation (for example: 172.16.2.2). If you choose the zone capabilities (peer mesh) of a DNS server in a non-proximity environment, do not use this variable.

For example:

```
(config)# dns-server zone 0 tier1 "pdns-usa" weightedrr
```

To disable the local DNS zone, enter:

```
(config)# no dns-server zone
```

Note If you need to modify a **dns-server zone** value, you must first disable the DNS server using the **no dns-server** command and then remove the zone using the **no dns-server zone** command. Restore the DNS server zone with the value change, then reenable the DNS server.

Configuring DNS Server Zone Load

Use the **dns-server zone load** command on an 11000 series CSS to configure how the CSS handles the leastloaded balance method.

The syntax for this global configuration mode command is:

dns-server zone load [**reporting**|**frequency** *seconds*|**variance** *number*]

- **reporting** - Enables the processing of local DNS server zone load information and sharing it with peers. The default is enabled.
- **frequency** - Specifies the period of time between the processing of local DNS server load information and the subsequent delivery of load information to peers.
- *seconds* - The frequency time (in seconds). Enter an integer between 5 and 300 seconds (5 minutes). The default is 30 seconds.
- **variance** - Specifies the range of load numbers between zones that will be considered similar. If the load numbers of all zones are within the specified range, the CSS uses minimum response times to identify the leastloaded site.
- *number* - The variance value. Enter an integer between 1 and 254. The default is 50.

For example, to set the DNS server zone load frequency to 120, enter:

```
(config)# dns-server zone load frequency 120
```

To disable DNS server zone load reporting, enter:

```
(config)# no dns-server zone load reporting
```

Note To enable or disable **dns-server zone load reporting**, you must first disable the DNS server using the **no dns-server** command. Then issue the **dns-server zone load reporting** or the **no dns-server zone load reporting** command.

Configuring DNS-Server BufferCount

To change the DNS response buffer count on the CSS, use the **dns-server bufferCount** command. Enter the number of buffers allocated for query response from 2 to 1000. The default is 50.

Use this command with the **show dns-server** command to tune the CSS only if the CSS experiences buffer depletion during normal operation. If the number of available name server buffers (NS Buffers) displayed by the **show dns-server** command drops below 2, use the **dns-server bufferCount** to increase the buffer count. You can also use the reclaimed buffer count as an indication of buffer depletion. When the supply of available buffers is depleted, the CSS reclaims used buffers.

For example:

```
(config)# dns-server bufferCount 100
```

To set the DNS response buffer count to its default value of 50, enter:

```
(config)# no dns-server bufferCount
```

Configuring DNS-Server RespTasks

To change the DNS responder task count, use the **dns-server respTasks** command. Enter the number of tasks to handle DNS responses as an integer from 1 to 250. The default is 2.

For example:

```
(config)# dns-server respTasks 3
```

To set the DNS responder task count to its default value of 2, enter:

```
(config)# no dns-server respTasks
```

Configuring a DNS Server Forwarder

Use the **dns-server forwarder** command to configure a DNS server forwarder on a CSS. If the CSS cannot resolve a DNS request, it sends the request to another DNS server to obtain a suitable response. This server, called a DNS server forwarder, can be a fully functional Berkeley Internet Name Domain (BIND) DNS server or a CSS configured for DNS. The CSS sends to the forwarder DNS requests that:

- Are not resolvable by the CSS
- Contain an unsupported request or record type

Note For Client Side Accelerator (CSA) configurations, the forwarder must be a BIND DNS server. For details on CSA, refer to Chapter 4, Configuring a Client Side Accelerator.

The forwarder resolves the DNS requests and sends DNS responses to the client transparently through the CSS. To monitor forwarder health, a keepalive mechanism (internal to the CSS) sends queries periodically to the forwarder to validate its state.

Note You must configure at least one local DNS server zone before configuring a DNS server forwarder. For details on DNS server zones, see "Configuring DNS Server Zones" earlier in this chapter.

The syntax for this global configuration mode command is:

dns-server forwarder [**primary** *ip_address*|**secondary** *ip_address*|**zero**]

The variables and options are:

- **primary** - Specifies a DNS server as the first choice forwarder. The CSS sends unresolvable requests to the primary forwarder unless it is unavailable, in which case, it uses the secondary forwarder. When the primary forwarder is available again, the CSS resumes sending requests to the primary forwarder.
- **secondary** - Specifies a DNS server as the second choice forwarder.

- *ip_address* - Specifies the IP address of the forwarder. Enter the address in dotted-decimal notation (for example, 192.168.11.1).
- **zero** - Resets the statistics of both forwarders on a CSS.

For example:

```
(config)# dns-server forwarder primary 192.168.11.1 secondary
192.168.11.2
```

To delete the primary forwarder on a CSS, enter:

```
(config)# no dns-server forwarder primary
```

Displaying DNS Server and Zone Information

To display DNS server configuration and database information, use the **show dns-server** command and the **show zone** command. These commands provide the following options and information:

- **show dns-server** - Displays DNS server configuration information
- **show dns-server dbase** - Displays the DNS database information
- **show dns-server stats** - Displays the DNS database statistics
- **show dns-server forwarder** - Displays DNS server forwarder statistics
- **show zone** - Displays information about a specified DNS server zone or all zones in a peer mesh

Displaying DNS Server Configuration Information

Use the **show dns-server** command to display information about your DNS server configuration. The syntax for this global configuration mode command is:

show dns-server

Table 1-3 describes the fields in the **show dns-server** output.

Table 1-3 Field Descriptions for the show dns-server Command

Field	Description
DNS Server Configuration	The enable or disable state of the DNS server function on the CSS. When enabled, the CSS acts as the authoritative name server for the content domain.
ACL Index	The ACL index number applied to the DNS server. If this field is 0, no ACL has been applied.
Responder Task Count	The configured DNS server responder task count. These tasks handle responses to incoming DNS query requests. The default is 2. The range is from 1 to 250.
Name Server Buffers	
Total Count	The configured DNS server buffer count. The responder tasks share the buffers to handle incoming queries. The default is 50.
Current Free Count	The number of buffers available (not queried).
Minimum Free Count	The smallest number of buffers that will be available.
Reclaimed Count	The number of buffers forcibly reclaimed by the DNS server software.
Requests Accepted	The number of DNS queries accepted.
Responses Sent	The number of DNS responses sent.
No Error	The number of queries that the DNS server successfully answered.
Format Error	The number of queries received that had a packet format error.
Server Failure	The number of times that a referenced name server did not reply to a query.
Name Error	The number of queries received that the DNS server was not able to answer.
Not Implemented	The number of queries received requesting an operation that has not been implemented in the DNS server.

Table 1-3 Field Descriptions for the show dns-server Command (continued)

Field	Description
Operation Refused	The number of queries the DNS server received that it refused to answer.
Internal Resolver	
Requests Sent	The number of queries sent to another name server for resolution.
Responses Accepted	The number of replies received from another name server.
Proximity Lookups	
Requests Sent	The number of proximity lookups sent to the PDB.
Responses Accepted	The number of proximity lookups received from the PDB.

Note Proximity lookup information is displayed only when you configure a PDB IP address. For information on configuring a PDB, refer to Chapter 5, Configuring Network Proximity, in the section "Configuring a Proximity Database".

Displaying DNS Server Database Statistics

Use the **dns-server dbase** command to display DNS server database statistics. The DNS server database contains DNS names that are configured locally or learned from peers and Time to Live (TTL) information for each DNS name. The syntax for this global configuration mode command is:

show dns-server dbase

Table 1-4 describes the fields in the **show dns-server dbase** output.

Table 1-4 Field Descriptions for the show dns-server dbase Command

Field	Description
DN	The domain name of the entry.
DNSCB	The address of the DNS control block structure to return a DNS query response for the entry. This address is the location best suited to handle the request.
PROX	The address for the proximity record.

Note When DNSCB and PROX have null values (0x0), these values indicate a host table mapping.

Displaying DNS Server Domain Statistics

Use the **show dns-server stats** to display DNS server domain statistics. The syntax for this global configuration mode command is:

show dns-server stats

Table 1-5 describes the fields in the **show dns-server stats** output.

Table 1-5 Field Descriptions for the show dns-server stats Command

Field	Description
DNS Name	The domain name entry
Content Name	Where the domain entry is mapped (A-record, NS-record, or host table), or a content rule name
Location	The IP address associated with the entry
Resolve Local	The number of local resolutions performed for the entry
Remote	The number of remote resolutions performed for the entry

Displaying DNS Server Forwarder Statistics

Use the **show dns-server forwarder** command to display statistics on the CSS for the DNS server forwarders. The syntax for this global configuration mode command is:

show dns-server forwarder

Table 1-6 describes the fields in the **show dns-server forwarder** output.

Table 1-6 Field Descriptions for the show dns-server forwarder Command

Field	Description
DNS Server Forwarder Primary	The state of the primary forwarder. The states are: • Not Configured • Up • Down
DNS Server Forwarder Secondary	The state of the secondary forwarder. The states are: • Not Configured • Up • Down
State Changes	The number of times that the forwarder's state changed.
Requests Sent	The total number of requests sent to a particular forwarder.
Responses Accepted	The total number of responses received from a particular forwarder.
Totals:	
Request Sent	The total number of requests sent to forwarders (primary and secondary).
Responses Accepted	The total number of responses received from forwarders (primary and secondary).

Displaying DNS Server Zones

Use the **show zone** command to display information about communication and the state of the specified DNS server zone or proximity zone, or all zones in a peer mesh.

The syntax for this global configuration command is:

show zone {*zone* {**verbose**} | **local** | **verbose**}

The variable and options for this command are:

- *zone* - The zone index of a peer. If you omit this variable, this command displays the states of all proximity zones.
- **local** - Display local zone information. This information includes a count of transmitted and received client packet types, the count of client packets, and a count of transmit errors.
- **verbose** - Display extra information per APP negotiation. This information includes a count of transmitted and received client packet types, the count of client packets, and a count of APP transmit errors.

For example:

```
(config)# show zone
```

To display proximity zones, including a count of transmitted and received client packet types, the count of client packets, and a count of APP transmit errors, enter:

```
(config)# show zone 1 verbose
```

Table 1-7 describes the fields in the **show zone** output.

Table 1-7 Field Descriptions for the show zone Command

Field	Description
Index	The zone index of the peer. The initial value is 255. Once peer communications are established using APP, the value changes to the zone index of the peer. If peer communications cannot be negotiated, the value remains at 255.
Description	Zone description as supplied by the peer from the **dns-server zone** command.

Table 1-7 Field Descriptions for the show zone Command (continued)

Field	Description
IP Address	The IP address of the peer. It corresponds to a locally configured APP session.
State	The state of the peer negotiation, which includes: • **INIT** - Initializing. Waiting for local configuration to complete. • **SREQ** - A connection request message has been sent to the peer. • **RACK** - An acknowledgment message has been received from the peer. • **SACK** - An acknowledgment request has been sent to the peer. • **OPEN** - Negotiations with the peer have completed successfully and the connection is open. • **CLOSED** - Negotiations with the peer have failed and the connection is closed.
State Chgs	The number of times the state has transitioned to OPEN and CLOSED.
UpTime	The amount of time that APP has been in the OPEN state.

Configuring Domain Records

Peer CSS DNS servers that participate in a zone mesh share domain record information using APP (see "Configuring Application Peering Protocol" earlier in this chapter). The DNS servers use the resulting database of domain names and their zone index information to make zone-based DNS decisions.

Use the **dns-record** command and its options to create domain records on a CSS configured as a DNS server. This command is not available on a CSS configured as a PDB. The CSS uses the following types of domain records to map a domain name to an IP address or to another DNS server, or to accelerate a domain:

- **a** - A domain record mapped to an IP address
- **ns** - A domain record mapped to a DNS server IP address
- **accel** - An accelerated domain associated with a Client Side Accelerator

Configuring A-Records

Use the **dns-record a** command to create an address record (A-record) on a CSS that maps the domain name to an IP address. Use the **no** form of this command to delete an A-record.

The syntax for this global configuration mode command is:

dns-record a *dns_name ip_address* {*ttl_value* {**single|multiple** {**kal-ap|kal-icmp|kal-none** {*ip_address2* {*threshold* {**sticky-enabled|sticky-disabled** {**usedefault|weightedrr|srcip|leastloaded|preferlocal|roundrobin|proximity** {*weight*}}}}}}}}

The **dns-record a** command supports the following variables and options:

- *dns_name* - The domain name mapped to the address record. Enter the name as a lower case unquoted text string with no spaces and a maximum length of 63 characters.
- *ip_address* - IP address bound to the domain name within the DNS server zone. Enter the address in dotted-decimal notation (for example 172.16.6.7).
- *ttl_value* - The optional Time to Live (TTL) value in seconds. This value determines how long the DNS client remembers the IP address response to the query. Enter a value between 0 to 65535. The default is 0.

- **single**|**multiple** - The optional number of records to return in a DNS response message. By default, the DNS server returns a single A-record. Specifying **single** returns one A-record. Specifying **multiple** returns two A-records.
- **kal-ap** - The optional keepalive message type keyword that specifies the CSS keepalive message. This is the recommended keepalive message type to obtain load information from remote as well as local services.

Note To use kal-ap proximity keepalive messages, lower-level CSSs acting as either data centers or DNS servers must be running the Enhanced feature set. When the Proximity Domain Name Server (PDNS) is directly attached to a server farm, an internal keepalive is used.

- **kal-icmp** (default keepalive) - The optional keepalive message type keyword that specifies ICMP echo (PING). To obtain load information from local services only, use the **add dns** *record_name* command in the associated content rule. See "Adding a DNS Service to a Content Rule" later in this chapter.
- **kal-none** - The optional keepalive message type keyword that specifies no keepalive messaging.

  For example:

  ```
  (config)# dns-record a www.home.com 172.16.6.7 15 single kal-icmp
  ```

- *ip_address2* - IP address of the local interface receiving CSS keepalive messages. If you omit this address while the keepalive type is specified, the CSS uses the DNS IP address to complete keepalive messaging.
- *threshold* - The load threshold is used only with the kal-ap CSS keepalive. Typically, the CSS keepalive reports 255 when a service is unavailable. This threshold allows the CSS to interpret lower reported numbers as unavailable. For example, if this parameter has a value of 100, all received load numbers greater than or equal to 100 cause the domain record to become unavailable for DNS decisions. Enter a value from 2 to 254. The default is 254.

  For example:

  ```
  (config)# dns-record a www.home.com 172.16.6.7 15 single kal-ap
  123.45.6.12 100
  ```

- **sticky-enabled** - Causes an 11000 series CSS DNS server to attempt to send a sticky response to the client for the specified domain. The CSS makes a decision based on one of the following three scenarios:
  - In a global server load-balancing (GSLB) environment without a global sticky database (GSDB), the CSS selects a server based on the srcip hash (regardless of the default load-balancing method) and the availability of the domain in the zone mesh. The use of the srcip hash ensures that the CSS selects a consistent zone for a given source IP address.
  - In a GSLB environment with a GSDB, the CSS sends a lookup request to the Global Sticky Database for the requesting client's local DNS server. If the GSDB has an entry in its sticky database for the client's local DNS server IP address, it returns the appropriate zone index to the CSS. The CSS then returns the associated IP address to the client. Otherwise, the CSS selects a zone based on the default load-balancing method and informs the GSDB about the selected zone.
  - In a Network Proximity environment, the CSS configured as a Proximity Domain Name Server (PDNS) first consults the GSDB. If a sticky database entry exists for the client's local DNS server IP address, the PDNS sends the appropriate IP address to the client based on the zone index returned by the GSDB. If the GSDB does not contain an entry for the client's local DNS server IP address, the PDNS consults the Proximity Database (PDB).

    If the PDB contains an entry for the client's local DNS server IP address, the PDNS formulates a response to the client based on the ordered zone index returned by the PDB and keepalive information. The PDNS informs the GSDB about the selected zone (performs a "set" function). If the PDB does not have an entry for the client's local DNS server IP address or the sticky zone is unavailable, the CSS selects a zone based on its default load-balancing method and informs the GSDB about the selected zone.

Caution If you configure any sticky domains in a particular zone, you must configure all sticky domains participating in the peer mesh in that same zone. Otherwise, the thrashing of the sticky zone index will cause DNS Sticky to fail.

- **sticky-disabled** - Disables DNS Sticky for the specified domain on an 11000 series CSS. This is the default. For details on configuring DNS Sticky, refer to Chapter 2, Configuring DNS Sticky.

For example:

```
(config)# dns-record a www.home.com 123.45.6.7 15 single kal-ap
172.16.6.12 100 sticky-enabled
```

- **usedefault** - Returns domain records using the default DNS load-balancing method configured for the zone. See "Configuring DNS Server Zones" earlier in this chapter.
- **weightedrr** - For an 11500 series CSS, returns domain records based on the weighted roundrobin load-balancing method. This method uses the *weight* value to determine the zone from which the record should be requested.
- **srcip** - Returns domain records using a source IP address hash. For sticky-enabled domains without a GSDB, the CSS uses the srcip method regardless of the configured balance method.
- **leastloaded** - Returns domain records from the zone with the smallest load.
- **preferlocal** - Returns local domain records whenever possible. If no local record exists, the CSS uses the balance method configured for the zone with the lowest zone index.
- **roundrobin** - Returns domain records by cycling among records available at the different zones to evenly distribute the load.
- **proximity** (the default) - Returns domain records based on proximity information. If a PDB is not configured or is unavailable in a zone, the CSS applies the default balance method for the selected zone for DNS resolution.

  For example:

  ```
  (config)# dns-record a www.home.com 172.16.6.7 15 single kal-ap
  172.16.6.12 100 sticky-enabled leastloaded
  ```

Note For sticky-enabled domains without a GSDB, a CSS uses the srcip method regardless of the configured balance method. For sticky-enabled domains with a GSDB, a CSS uses the configured balance method when the GSDB does not contain an entry for the requested domain.

- *weight* - For an 11500 series CSS, a value assigned to a domain in the local zone to determine how many requests the local zone receives for the specified domain compared with other zones in a peer mesh. A domain with a weight of 10 in the local zone will receive twice as many requests as the same domain in another zone with a weight of 5.

Use this parameter on an 11500 series CSS with the weighted roundrobin DNS load-balancing method. (See "Configuring DNS Server Zones" earlier in this chapter.) CSSs configured as authoritative DNS servers in a peer mesh share domain weights, hit counts, maximum hit counts, and a zone pointer with each other. Enter an integer from 1 to 10. The default is 1.

Note If your configuration includes 11000 series CSSs, the *weight* value defaults to 1 and is not configurable for those CSSs.

The CSS uses the following guidelines when selecting a DNS load-balancing method on a domain basis:

- If a local record exists, the CSS uses the configured domain balance method to determine local DNS resolutions. This rule applies regardless of the keepalive state of the local record.
- If no local record exists, the CSS uses the balance method configured for the zone with the lowest zone index.

For example, consider the following configuration.

Zone	Domain Record	Balance Method
0	www.test.com	leastloaded
1	www.test.com	roundrobin
2	no local record configured for www.test.com	none configured

With this configuration, you can expect the following behavior:

- DNS resolutions occurring on the Zone 0 and Zone 2 DNS servers will use the leastloaded balance method.
- DNS resolutions occurring on the Zone 1 DNS server will use the roundrobin balance method.

Note If you need to modify an existing A-record parameter, you must first remove the record using the **no dns-record a** *domain_name* command. Then recreate the A-record with the parameter change using the **dns-record a** command.

Configuring NS-Records

Use the **dns-record ns** command to create a name server record (NS-record) on a CSS that maps the domain name to the IP address of a lower-level DNS server. Use the **no** form of this command to delete an NS-record.

The syntax for this global configuration mode command is:

dns-record ns *dns_name ip_address* {*ttl_value* {**single|multiple** {**kal-ap|kal-icmp|kal-none** {*ip_address2* {*threshold* {**default|forwarder** {**sticky-enabled|sticky-disabled** {**usedefault|weightedrr|srcip|leastloaded|preferlocal|roundrobin|proximity** {*weight*}}}}}}}}}

The **dns-record ns** command supports the following options and variables:

- *dns_name* - The domain name mapped to the address record. Enter the name as a lowercase unquoted text string with no spaces and a maximum length of 63 characters.
- *ip_address* - IP address of the DNS server bound to the domain name within the DNS server zone. Enter the address in dotted-decimal notation (for example 123.45.6.8).
- *ttl_value* - The optional Time to Live (TTL) value in seconds. This value determines how long the DNS client remembers the IP address response to the query. Enter a value between 0 and 65535. The default is 0.
- **single|multiple** - The optional number of records to return in a DNS response message. By default, the DNS server returns a single NS-record. Specifying **single** returns one NS-record. Specifying **multiple** returns two NS-records.
- **kal-ap** - The optional keepalive message type keyword that specifies the CSS keepalive message.

Note To use the kal-ap keepalive messages, lower level CSSs acting as either data centers or DNS servers must be running the Enhanced feature set. When the DNS server is directly attached to a server farm, an internal keepalive is used.

- **kal-icmp** (default keepalive) - The optional keepalive message type keyword that specifies ICMP echo (PING).

- **kal-none** - The optional keepalive message type keyword that specifies no keepalive messaging.

  For example:

  ```
  (config)# dns-record ns www.work.com 172.16.6.8 15 single kal-icmp
  ```

- *ip_address2* - IP address of the local interface receiving CSS keepalive messages.
- *threshold* - The load threshold is used only with the kal-ap CSS keepalive. Typically, the CSS keepalive reports 255 when a service is unavailable. This threshold allows the CSS to interpret lower reported numbers as unavailable. For example, if this parameter has a value of 100, all received load numbers greater than or equal to 100 cause the domain record to become unavailable for DNS decisions. Enter a value from 2 to 254. The default is 254.

  For example:

  ```
  (config)# dns-record ns www.home.com 172.16.6.7 15 single kal-ap
  123.45.6.12 100
  ```

- **default** - In a proximity configuration, the CSS uses PDB information to return the next most proximate location. When a PDB is not available or not configured, the CSS uses the roundrobin load-balancing method. There is no failover scenario.
- **forwarder** - Use this option to eliminate a potential single point of failure by providing up to two alternative DNS servers called forwarders. A forwarder can be a CSS configured as a DNS server or a fully-functional BIND DNS server. If an optimal miss occurs (the lower-level DNS server indicated in the NS-record is Down), the PDNS sends the DNS request to the primary or secondary forwarder, depending on forwarder health and configuration. An optimal miss occurs when the PDNS cannot return the NS-record for the zone that the PDB indicated was most proximate. For this failover to occur, the local NS-record must be in the Down state, and the PDB has indicated the local zone to be the zone most proximate to the client. For information on configuring a DNS server forwarder, see "Configuring a DNS Server Forwarder" earlier in this chapter.

- **sticky-enabled** - Causes the CSS DNS server to attempt to send a sticky response to the client for the specified domain. The CSS makes a decision based on one of the following three scenarios:
  - In a global server load-balancing (GSLB) environment without a global sticky database (GSDB), the CSS selects a server based on the srcip hash (regardless of the default load-balancing method) and the availability of the domain in the zone mesh. The use of the srcip hash ensures that the CSS selects a consistent zone for a given source IP address.
  - In a GSLB environment with a GSDB, the CSS sends a lookup request to the Global Sticky Database for the domain requested by the client. If the GSD has an entry in its sticky database for the client's local DNS server IP address, it returns the appropriate zone index to the CSS. The CSS then returns the associated IP address to the client. Otherwise, the CSS selects a zone based on the default load-balancing method and informs the GSDB about the selected zone.
  - In a Network Proximity environment, the CSS configured as a Proximity Domain Name Server (PDNS) first consults the GSDB. If a sticky database entry exists for the client's local DNS server IP address, the PDNS sends the appropriate IP address to the client based on the zone index returned by the GSDB. If the GSDB does not contain an entry for the client's local DNS server IP address, the PDNS consults the Proximity Database (PDB).

    If the PDB contains an entry for the client's local DNS server IP address, the PDNS formulates a response to the client based on the ordered zone index returned by the PDB and keepalive information. The PDNS informs the GSDB about the selected zone. If the PDB does not have an entry for the client's local DNS server IP address or the sticky zone is unavailable, the CSS selects a zone based on its default load-balancing method and informs the GSDB about the selected zone.

Note If you configure any sticky domains in a particular zone, you must configure all sticky domains participating in the peer mesh in that same zone. Otherwise, the thrashing of the sticky zone index will cause DNS Sticky to fail.

- **sticky-disabled** - Disables DNS Sticky for the specified domain. This is the default.

  For example:

  ```
  (config)# dns-record ns www.home.com 172.16.6.7 15 single kal-icmp
  123.45.6.12 100 sticky-enabled
  ```

Note For details on configuring DNS Sticky, refer to Chapter 2, Configuring DNS Sticky.

- **usedefault** - The CSS returns domain records using the default DNS load-balancing method configured for the zone. See "Configuring DNS Server Zones" earlier in this chapter.
- **weightedrr** - For an 11500 series CSS, returns domain records based on the weighted roundrobin load-balancing method. This method uses the *weight* value to determine the zone from which the record should be requested.
- **srcip** - The CSS returns domain records using a source IP address hash. For sticky-enabled domains without a GSDB, the CSS uses the srcip method regardless of the configured balance method.
- **leastloaded** - The CSS returns domain records from the zone with the smallest load.
- **preferlocal** - The CSS returns local domain records whenever possible. If no local record exists, the CSS uses the balance method configured for the zone with the lowest zone index.
- **roundrobin** - The CSS returns domain records by cycling among records available at the different zones to evenly distribute the load.
- **proximity** (the default) - Returns domain records based on proximity information. If a Proximity Database (PDB) is not configured or is unavailable in a zone, the CSS applies the default balance method for the selected zone for DNS resolution.

  For example:

  ```
  (config)# dns-record ns www.home.com 172.16.6.7 15 single kal-icmp
  172.16.6.12 100 sticky-enabled leastloaded
  ```

Note For sticky-enabled domains without a GSDB, a CSS uses the srcip method regardless of the configured balance method. For sticky-enabled domains with a GSDB, a CSS uses the configured balance method when the GSDB does not contain an entry for the requested domain.

- *weight* - For an 11500 series CSS, a value assigned to a domain in the local zone to determine how many requests the local zone receives for the specified domain compared with other zones in a peer mesh. A domain with a weight of 10 in the local zone will receive twice as many requests as the same domain in another zone with a weight of 5.

  Use this parameter on an 11500 series CSS with the weighted roundrobin DNS load-balancing method. (See "Configuring DNS Server Zones" earlier in this chapter.) CSSs configured as authoritative DNS servers in a peer mesh share domain weights, hit counts, maximum hit counts, and a zone pointer with each other. Enter an integer from 1 to 10. The default is 1.

Note If your configuration includes 11000 series CSSs, the *weight* value defaults to 1 and is not configurable for those CSSs.

The CSS uses the following guidelines when selecting a DNS load-balancing method on a domain basis:

- If a local record exists, the CSS uses the configured domain balance method to determine local DNS resolutions. This rule applies regardless of the keepalive state of the local record.
- If no local record exists, the CSS uses the balance method configured in the zone with the lowest zone index.

For example, consider the following configuration.

Zone	Domain Record	Balance Method
0	www.test.com	leastloaded
1	www.test.com	roundrobin
2	no local record configured for www.test.com	none configured

With this configuration, you can expect the following behavior:

- DNS resolutions occurring on the Zone 0 and Zone 2 DNS servers will use the leastloaded balance method.
- DNS resolutions occurring on the Zone 1 DNS server will use the roundrobin balance method.

Note If you need to modify an existing NS-record parameter, you must first remove the record using the **no dns-record ns** *domain_name* command. Then recreate the NS-record with the parameter change using the **dns-record ns** command.

Removing a Domain Record

Use the **no dns-record command** to remove a domain record.

The syntax for this global configuration mode command is:

no dns-record *dns_name*

The *dns_name* variable maps the DNS name to the address record. Enter the name as a case-sensitive unquoted text string with no spaces and a maximum length of 63 characters.

For example:

```
(config)# no dns-record www.home.com
```

Resetting the DNS Record Statistics

Use the **dns-record zero** command to reset the DNS record statistics displayed by the **show dns-record** command. The syntax for this global configuration mode command is:

dns-record zero [**a/ns** {*dns_name*}|**accel** {*dns_name*}]

The options and variables for this command are:

- **a/ns** - Resets the statistics to zero for the domain records that are displayed by the **show dns-record statistics** command (see "Displaying DNS-Record Statistics" later in this chapter) and the **show dns-record proximity** command (refer to Chapter 5, Configuring Network Proximity).
- *dns_name* - Resets the statistics for the specified domain name mapped to the DNS record. To view a list of domain names, enter:

```
dns-record zero [a/ns|accel] ?
```

- **accel** - Resets the counters to zero for the accelerated records that are displayed by the **show dns-record accel** command. Refer to Chapter 4, Configuring a Client Side Accelerator, in the section "Displaying Domain Acceleration Records Statistics".

Displaying DNS Record Information

Use the **show dns-record** command to display statistics about the DNS records that were manually configured or learned from peers.

Displaying DNS-Record Statistics

Use the **show dns-record statistics** command to display statistics associated with the address records (A-records), name server records (NS-records), or acclerated domain records (accel) configured locally and learned by the CSS from its peers.

The syntax for this global configuration mode command is:

show dns-record statistics {*dns_name*}

You may enter an optional domain name target to display content. If you omit the domain name, all domains appear.

For example:

```
(config)# show dns-record statistics
```

Table 1-8 describes the fields in the **show dns-record statistics** output.

Table 1-8 Field Descriptions for the show dns-record statistics Command

Field	Description
<*Domain name*>	Domain name for the record.
Local	State of the local entry for the record. Up indicates that the entry is configured. A "-" character indicates that the entry is learned and not configured. Down indicates that the keepalive failed.
Zone Count	Number of zones where this record is configured.
Zone	Index number for the zone. A "*" character prepending the zone number indicates that the zone is a local entry.
Description	Zone description.
Type	DNS record type: • **A** - Address record • **NS** - Name-server record • **Accel** - An accelerated domain associated with a Client Side Accelerator (CSA)
IP Address	Configured IP address for the zone.
TTL	Time to Live, which indicates how long the receiver of a DNS reply for the given domain should cache the address information. By default, the TTL value is 0, indicating that the name server receiving the response should not cache the information.
Hits	Total number of DNS hits.

Displaying DNS Record Keepalive Information

Use the **show dns-record keepalive** command to display DNS record keepalive information. The syntax for this global configuration mode command is:

show dns-record keepalive {*dns-name*}

The variable for this command is *dns-name*, the domain name associated with the DNS record. You can enter an optional domain name target to display content. If you omit this variable, all DNS records appear.

Table 1-9 describes the fields in the **show dns-record keepalive** output.

Table 1-9 Field Descriptions for the show dns-record keepalive Command

Field	Description
Name	Domain name for the record.
Type	Keepalive message type for the record: Accel, AP, ICMP, or none.
IP	Destination IP address of the keepalive message.
State	State of the record, either UP or DOWN.
Transitions	Number of state transitions.
Load	Load for the record, which applies only to an AP record type. All other types always have a load of "-", indicating an undetermined load (load reports are not being received). If the load value exceeds the threshold value, the DNS server removes the DNS record from eligibility.
Threshold	Configured load threshold for the record. This threshold applies only to an AP record type. Record types of ICMP and none do not use the threshold value.

Displaying DNS Record Weight

Use the **show dns-record weight** command to display the configured weight and the number of hits for all domains or the specified domain. The syntax for this global configuration command is:

show dns-record weight {*dns_name*}

The *dns-name* variable for this command is the domain name associated with the DNS record. You can enter an optional domain name target to display information for the specified domain record. If you omit this variable, all DNS records appear.

Table 1-10 describes the fields in the **show dns-record weight** output.

Table 1-10 Field Descriptions for the show dns-record weight Command

Field	Description
Name	Domain name for the record.
Total Hits	Total number of hits in all zones for the specified domain name.
Zone	Zone index for each zone where the domain record resides. An asterisk indicates the local zone.
Description	Text description of the zone.
IP Address	IP address of the DNS server bound to the domain name within the DNS server zone.
Weight	Configured weight value for the record.
Current Hits	Current number of hits for the domain record in the zone.
Total Hits	Total number of hits for the domain record in the zone.

Configuring Content Rule-Based DNS

The following sections describe how to configure DNS using content rules and asociated commands.

Note The recommended method for configuring DNS in a global server load balancing environment is zone-based DNS. For details on enabling DNS server functionality on a CSS, see "Configuring a CSS as an Authoritative DNS Server" earlier in this chapter.

Configuring CSS DNS Peering

Use the **dns-peer** command and its options to enable DNS peer functionality on a CSS. Peer functionality includes the sharing of content rules. The syntax and options for this global configuration mode command are:

- **dns-peer interval** - Sets the time between the load reports to the CSS DNS peers
- **dns-peer receive-slots** - Sets the maximum number of DNS names that the CSS can receive from each CSS DNS peer
- **dns-peer send-slots** - Sets the maximum number of DNS names that the CSS can send to each CSS DNS peer

Configuring DNS Peer-Interval

To set the time between generating load reports to the CSS DNS peers, use the **dns-peer interval** command. Enter the peer interval time from 5 to 120 seconds. The default is 5.

For example:

```
(config)# dns-peer interval 60
```

To reset the DNS peer interval to its default value of 5 seconds, enter:

```
(config)# no dns-peer interval
```

Configuring DNS Peer-Receive-Slots

To set the maximum number of DNS names that the CSS can *receive* from each CSS DNS peer, use the **dns-peer receive-slots** command. Enter a number from 128 to 1024. The default is 128. Use this command to tune a heavily-accessed CSS that is resolving more than 128 DNS names.

For example:

```
(config)# dns-peer receive-slots 200
```

To reset the DNS peer receive slots number to its default of 128, enter:

```
(config)# no dns-peer receive-slots
```

Configuring DNS Peer-Send-Slots

To set the maximum number of DNS names that the CSS can *send* to each CSS DNS peer, use the **dns-peer send-slots** command. Enter a number from 128 to 1024. The default is 128. Use this command to tune a CSS that is reporting over 128 DNS names to the peer.

For example:

```
(config)# dns-peer send-slots 200
```

To reset the DNS peer send slots number to its default of 128, enter:

```
(config)# no dns-peer send-slots
```

Show DNS Peer Information

To display the DNS peering configuration, use the **show dns-peer** command.

For example:

```
(config)# show dns-peer
```

Table 1-11 describes the fields in the **show dns-peer** output.

Table 1-11 Field Descriptions for the show dns-peer Command

Field	Description
CSD Peer Rcv Slots	The configured maximum number of DNS names that the CSS can receive from each CSS DNS peer over an APP connection. The default is 128. The range is from 128 to 1024.
CSD Peer Snd Slots	The configured maximum DNS names that the CSS can send to each CSS DNS peer. The default is 128. The range is from 128 to 1024.
Peer Report Interval	The configured time in seconds between sending load reports to CSS DNS peers over an APP connection. The default is 5. The range is from 5 to 120.

Configuring Owner DNS

To set the DNS exchange policy for an owner, use the **dns** command. The syntax and options for this owner mode command are:

- **no dns** - Sets no DNS exchange policy for this owner (default). This owner is hidden from the CSS peer.
- **dns accept** - Accepts all content rules for this owner proposed by the CSS peer.
- **dns push** - Advertises the owner and push all its content rules to the CSS peer.
- **dns both** - Advertises the owner and push all its content rules to the CSS peer, and accept all this owner's content rules proposed by the CSS peer.

For example:

```
(config-owner[arrowpoint])# dns both
```

Adding a DNS Service to a Content Rule

To specify a DNS name that maps to a content rule, use the **add dns** command. Enter the DNS name as a lowercase unquoted text string with no spaces and a length of 1 to 31 characters.

Note The **add dns** command is part of the CSS Standard feature set.

When you add the DNS name to the content rule, you may also enter an optional Time to Live (TTL) value in seconds. This value specifies 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 You must configure the TTL when you add the DNS name to the content rule. To add a TTL to an existing rule, use the **remove dns** command to remove the dns name. Then use the **add dns** command to reconfigure the DNS name with a TTL value.

For example:

```
(config-owner-content[arrowpoint-rule1])# add dns
www.arrowpoint.com 36
```

Removing DNS from Content Rule

To remove a DNS name from a content rule, use the **remove dns** command with the DNS name you wish to remove. Enter the DNS name as a case-sensitive unquoted text string with no spaces and a maximum of 31 characters.

Note The **remove dns** command is part of the CSS Standard feature set.

For example:

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

To display a list of DNS names, enter:

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

Configuring Source Groups to Allow Servers to Internet-Resolve Domain Names

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 (Virtual IP address) for the servers to allow them to resolve domain names.

To configure a server to resolve domain names:

1. Configure the server, if you have not already done so.

   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. This content rule is in addition to the content rules you created to process Web traffic. The content rule enables the CSS to perform Network Address Translation (NAT) to translate inbound DNS replies from the public VIP address (192.200.200.200) to the server's private IP address (10.0.3.251).

   The following example creates content rule *dns1* with a public Virtual IP address (VIP) 192.200.200.200 and adds server *Server1*.

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

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

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

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

The following example creates source group *dns1* with public VIP address 192.200.200.200 and adds server Server1.

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

Displaying Domain Summary Information

To display content domain summary information, use the **show domain** command. The syntax and options are listed below. For options that require an IP address, specify the IP address for the peer.

- **show domain** - Displays content domain summary information including the number of domain peers and information about each peer.
- **show domain** *ip_address* **send|receive** - Displays content domain summary information including the number of domain peers and information for the specified peer IP address. To see a list of addresses, enter **show domain ?**.
  - Include the **send** option to display only the send load reports and transmit message statistics.
  - Include the **receive** option to display only the receive load reports and receive message statistics.
- **show domain hotlist** - Displays configuration information about domain hotlists.
- **show domain owners** - Displays shared owner names.
- **show domain owners** *ip_address* - Displays shared owner names for the specified peer IP address.
- **show domain rules** - Displays locally created or negotiated names.

- **show domain rules** *ip_address* - Displays locally created or negotiated names for the specified peer IP address.

Table 1-12 describes the fields in the **show domain** output.

Table 1-12 Field Descriptions for the show domain Command

Field	Description
Content Domain Summary	The number of domain peers.
Peer	The address for the peer.
CCC State	The state of the master FSM (finite state machine) that negotiates the CAPP (CCC) link.
OWN State	The state of the owner policy negotiation FSM that determines the owners about whom the peers will share domain name and rule information.
Rule State	The state of the rule policy negotiation FSM that exchanges individual domain name and rule matching criteria and load report information.
SendSlots	The number of individual domain name rules on which the CSS will send load reports to the peer.
ReceiveSlots	The number of individual domain name rules on which the CSS will receive load reports from the peer.
Interval	The time interval in seconds that load reports are sent to the peer.
MinRespTime	The minimum local flow response time. This number is shared with the peer to be used in conjunction with load numbers to normalize the load numbers shared between peers.
MaxRespTime	The maximum local flow response time. This number is shared with the peer to be used in conjunction with load numbers to normalize the load numbers shared between peers.
Policy	The negotiated load report send and receive policies.

Table 1-12 Field Descriptions for the show domain Command (continued)

Field	Description
Sending Load Reports for	The list of domain names for which the CSS is sending load reports to the peer.
Receiving Load Reports for	The list of domain names for which the CSS is receiving load reports from the peer.
CCC Msg stats	The number of times each of the message types used in the CCC/OWN/Rule FSM negotiations with the peer has been sent or received.

Configuring DNS Sticky

This chapter provides an overview of the CSS Domain Name Service (DNS) Sticky feature and describes how to configure it for operation. Information in this chapter applies to all CSS models, except where noted.

Note The DNS Sticky feature is part of the CSS Enhanced feature set.

This chapter provides the following sections:

DNS Sticky Overview

Configure DNS Sticky on a CSS to ensure that e-commerce clients remain connected to a particular server for the duration of a transaction even when the client's browser refreshes the DNS mapping. While some browsers allow client connections to remain for the lifetime of the browser instance or for several hours, other browsers impose a connection limit of 30 minutes before requiring a DNS re-resolution. This may not be long enough for a client to complete an e-commerce transaction. A new DNS resolution could cause the client to connect to a server different from the original server and interrupt the transaction. DNS Sticky ensures that a client can complete a transaction if a DNS re-resolution occurs.

DNS Sticky extends the functionality of global server load balancing (GSLB) and Network Proximity by providing:

- **Stickiness on a per domain basis** - Allows you to configure DNS Sticky only on the domains you want.
- **Zone-based DNS** - Provides service for configured domains in a maximum of 256 zones using the roundrobin, preferlocal, leastloaded, or srcip (source IP address) load-balancing method.
- **Global Sticky Database (GSDB)** - Maintains a database of sticky mappings and provides appropriate responses to DNS Sticky queries from CSSs configured as authoritative DNS servers. The GSDB is a dedicated CSS 11150 with 256 MB of RAM configured as a sticky database. You configure a GSDB on a CSS configured as a Proximity Database (PDB) in each GSLB zone.

You can configure DNS Sticky in three different environments, depending on your current configuration and business needs as follows:

- DNS Sticky Without a GSDB
- DNS Sticky With a GSDB
- DNS Sticky in a Network Proximity Environment

DNS Sticky Without a GSDB

DNS Sticky without a GSDB in a GSLB environment provides a static, simple, and cost-effective solution to the DNS sticky problem. This solution:

- Allows you to configure DNS Sticky on the domains you want
- Uses the srcip load-balancing method to keep clients connected to a particular zone based on a srcip hash
- Provides services for domains in up to 256 zones (using two tier2 levels)
- Does not require the configuration of a dedicated GSDB

In a GSLB sticky configuration without a GSDB, the CSS configured as an authoritative DNS server selects a server for a sticky domain request based on the srcip hash (regardless of the default load-balancing method) and the availability of the domain in the zone mesh. The use of the srcip hash ensures that the CSS selects a consistent zone for a given source IP address.

DNS Sticky With a GSDB

DNS Sticky with a GSDB in a GSLB environment provides a more robust sticky load-balancing solution than one without a GSDB. This solution includes all of the benefits of DNS Sticky without a GSDB, plus:

- A GSDB to keep track of sticky mappings and provide responses to requests for sticky-enabled domains
- Configuration of up to two GSDB interfaces on the authoritative CSS DNS server for redundancy purposes
- More effective sticky load balancing across all domain sites using the leastloaded load-balancing method

Note If you configure a GSDB and any sticky domains in a particular zone, you must configure all sticky domains participating in the peer mesh in that same zone. Otherwise, the thrashing of the sticky zone index could cause DNS Sticky to fail. For details on configuring sticky domains, refer to Chapter 1, Configuring the CSS Domain Name Service,in the section "Configuring Domain Records".

In a GSLB sticky configuration with a GSDB, a CSS configured as an authoritative DNS server sends a lookup request to the GSDB for a sticky domain requested by a client. If the GSDB finds an entry for the client's local DNS server IP address in its sticky database, it returns the sticky zone index to the CSS. The CSS uses the sticky zone index and keepalive information to send the appropriate IP address to the client. If the GSDB does not have an entry for the client's local DNS server IP address or the zone in the sticky zone index returned by the GSDB is unavailable, the CSS selects a zone in the mesh based on the configured load-balancing method and informs the GSDB about the selected zone.

Note Configuring a GSDB requires the prior configuration of a Proximity Database (PDB) on the same CSS. For details on configuring a PDB, refer to Chapter 5, Configuring Network Proximity, in the section "Configuring a Proximity Database".

DNS Sticky in a Network Proximity Environment

Configure DNS Sticky in a Network Proximity environment to add stickiness to your network. This solution adds all the benefits of DNS Sticky with a GSDB to your existing proximity configuration. In this case, you can specify critical e-commerce sites as sticky domains and use proximity for your other domains.

In a Network Proximity environment, you configure a GSDB on the CSS configured as a Proximity Database (PDB) and at least one GSDB interface on the Proximity Domain Name Server (PDNS) in each zone. The IP address of the primary **GSDB interface** is typically the same as the PDB IP address. In addition, you configure sticky domain records using the **dns-record** command.

When a CSS configured as a PDNS receives a client request for a sticky domain, the PDNS first consults the GSDB. If a sticky database entry exists for the client's local DNS server IP address, the PDNS sends the appropriate IP address to the client based on the zone index returned by the GSDB. If a sticky database entry does not exist for the client's local DNS server IP address, the PDNS consults the PDB for a Proximity-based answer. The PDNS formulates a response to the client based on the ordered zone index returned by the PDB and keepalive information. The PDNS informs the GSDB about the selected zone.

If neither the GSDB nor the PDB returns a suitable response, the PDNS selects a zone based on its configured default load-balancing method to formulate an appropriate response to the client and informs the GSDB about the selected zone.

For details on configuring Network Proximity, refer to Chapter 5, Configuring Network Proximity.

DNS Sticky Quick Start Procedures

The following sections provide the procedures required to configure DNS Sticky on a CSS.

DNS Sticky Without a GSDB Configuration Quick Start

Table 2-1 provides a quick overview of the steps required to configure DNS Sticky on a CSS without a GSDB. 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, refer to the DNS Sticky commands later in this chapter.

Table 2-1 DNS Sticky Without a GSDB Configuration Quick Start

Task and Command Example
1. On a CSS that you want to configure DNS Sticky without a GSDB, enter config mode. **# config** (config)#
2. Enable Application Peering Protocol (APP). Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol". **(config)# app**
3. Configure the DNS server zone. Specify the zone, tier number, and an optional text description. Do not enter a Proximity Database (PDB) IP address. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring DNS Server Zones". (config)# **dns-server zone 0 tier1 "usa"**

Table 2-1 DNS Sticky Without a GSDB Configuration Quick Start (continued)

Task and Command Example
4. Configure the CSS to act as a DNS server. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring a CSS as an Authoritative DNS Server". `(config)# dns-server`
5. Configure APP sessions with other DNS servers (if any) that are participating in the peer mesh. The IP address you enter is a local interface address (circuit address) on the DNS server in another zone. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol". `(config)# app session 200.16.2.5`

6. Configure sticky domain records. Refer to "Configuring Sticky Domain Records" later in this chapter.
`(config)# dns-record a www.home.com 192.168.1.1 15 single kal-ap 172.68.25.1 50 sticky-enabled`
`(config)# dns-record ns www.work.com 192.172.12.1 15 single kal-ap 172.92.33.1 100 default sticky-enabled`

DNS Sticky with a GSDB Configuration Quick Start

The following sections describe the steps required to configure DNS Sticky with a GSDB. You can configure the GSDB and the DNS server in any order.

Global Sticky Database Configuration Quick Start

Table 2-2 provides a quick overview of the steps required to configure the GSDB on a CSS. 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, refer to the DNS Sticky commands later in this chapter.

Table 2-2 Global Sticky Database Configuration Quick Start

Task and Command Example

1. On a dedicated CSS 11150 with 256 MB of RAM that you want to configure as a Global Sticky Database (GSDB), enter config mode.
`# config`
`(config)#`

2. Enable the Application Peering Protocol-User Datagram Protocol (APP-UDP) to allow the GSDB to communicate with the CSS authoritative DNS server in the same zone. Refer to Chapter 5, Configuring Network Proximity, in the section "Configuring APP-UDP and APP". `(config)# app-udp`
3. Enable the Application Peering Protocol (APP) to allow the GSDB to communicate with other GSDBs. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol". `(config)# app`
4. Configure APP sessions with other GSDBs that are participating in the peer mesh with this GSDB. The IP address you enter is a local interface address on another GSDB. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol". `(config)# app session 200.16.2.3`
5. Configure a Proximity Database (PDB) if not already configured. (To configure a GSDB, you must configure a PDB first.) For details on configuring a PDB, refer to Chapter 5, Configuring Network Proximity, in the section "Configuring a Proximity Database". `(config)# proximity db 0 tier1 "usa"`
6. Enable the GSDB. `(config)# gsdb`
7. Optionally, configure the time-to-live (TTL) in seconds for the GSDB sticky entries. Enter an integer between 300 and 1000000. The default is 7200 seconds (2 hours). `(config)# gsdb ttl 14400`

DNS Server Configuration Quick Start

Table 2-3 provides a quick overview of the steps required to configure the DNS Sticky feature on a CSS acting as an authoritative DNS server and using a GSDB. 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, refer to the DNS Sticky commands later in this chapter.

Table 2-3 DNS Server Configuration Quick Start

Task and Command Example
1. On a CSS different from the GSDB, but in the same zone, enter config mode. **# config** (config)#
2. Enable APP-UDP to allow the CSS to communicate with the GSDB. Refer to Chapter 5, Configuring Network Proximity, in the section "Configuring APP-UDP and APP". (config)# **app-udp**
3. Enable Application Peering Protocol (APP). Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol". **(config)# app**
4. Configure up to two interfaces on the CSS to communicate with the GSDB. Refer to "Configuring the Global Sticky Database Interface" later in this chapter. (config)# **gsdb-interface primary 192.172.68.1** (config)# **gsdb-interface secondary 192.172.68.2**
5. Configure the DNS server zone for zone-based DNS. Specify the zone, tier number, and an optional text description. Do *not* enter a PDB IP address. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring DNS Server Zones". (config)# **dns-server zone 0 tier1 "usa"**
6. Configure the CSS to act as a DNS server. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring a CSS as an Authoritative DNS Server". (config)# **dns-server**

Table 2-3 DNS Server Configuration Quick Start (continued)

Task and Command Example
7. Configure APP sessions with other DNS servers (if any) that are participating in the peer mesh with this zone. The IP address you enter is a local interface address on the DNS server in another zone. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol". `(config)# app session 200.16.2.5`

8. Configure A-records and NS-records on the CSS. Use the **sticky-enabled** option for those domains that clients will use for e-commerce applications and any other applications requiring stickiness. See "Configuring Sticky Domain Records" later in this chapter.
`(config)# dns-record a www.home.com 192.168.1.1 15 single kal-ap 172.68.25.1 50 sticky-enabled`
`(config)# dns-record ns www.work.com 192.172.12.1 15 single kal-ap 172.92.33.1 100 default sticky-enabled`

DNS Sticky with Network Proximity Configuration Quick Start

The following sections describe the steps required to configure DNS Sticky in an existing Network Proximity configuration. For details on configuring Network Proximity, refer to Chapter 5, Configuring Network Proximity.

Global Sticky Database Configuration Quick Start

Table 2-4 provides a quick overview of the steps required to configure the GSDB on a PDB. 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 DNS Sticky commands later in this chapter.

Table 2-4 Global Sticky Database Configuration Quick Start

Task and Command Example

1. On a PDB (CSS 11150 with 256 MB of RAM configured as a Proximity Database) that you want to configure as a Global Sticky Database (GSDB), enter config mode.
`# config`
`(config)#`

2. Enable the GSDB. `(config)# gsdb`
3. Optionally, configure the time-to-live (TTL) in seconds for the GSDB sticky entries. Enter an integer between 300 and 1000000. The default is 7200 (2 hours). `(config)# gsdb ttl 14400`

DNS Server Configuration Quickstart

Table 2-5 provides a quick overview of the steps required to configure the DNS Sticky feature on a PDNS. 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 DNS Sticky commands later in this chapter.

Table 2-5 DNS Server Configuration Quick Start

Task and Command Example

1. On a CSS that you want to configure DNS Sticky, enter config mode.
`# config`
`(config)#`

2. Configure up to two interfaces on the CSS to communicate with the GSDB. See "Configuring the Global Sticky Database Interface" later in this chapter.
`(config)# gsdb-interface primary 192.172.68.1`
`(config)# gsdb-interface secondary 192.172.68.2`

Table 2-5 DNS Server Configuration Quick Start (continued)

Task and Command Example

3. Configure A-records and NS-records on the CSS. Use the **sticky-enabled** option for those domains that clients will use for e-commerce applications and any other applications requiring stickiness. See "Configuring Sticky Domain Records" later in this chapter.
`(config)# dns-record a www.home.com 192.168.1.1 15 single kal-ap 172.68.25.1 50 sticky-enabled`
`(config)# dns-record ns www.work.com 192.172.12.1 15 single kal-ap 172.92.33.1 100 default sticky-enabled`

Converting Content Rule-Based DNS to Zone-Based DNS

DNS Sticky requires a zone-based DNS configuration. If you currently have a content rule-based DNS configuration, use the following procedure to convert your DNS configuration to a zone-based DNS configuration.

1. Remove all rule-based DNS commands from the existing configuration by issuing the "no" form of the commands. For example:

```
(config)# no dns-peer interval
(config)# no dns-peer receive-slots
(config)# no dns-peer send-slots

(config-owner)# no dns

(config-owner-content)# remove dns
(config-owner-content)# no dns-balance
```

2. Use the **dns-server zone** command to create zone information for each network location. Refer to Chapter 1, Configuring DNS Sticky, in the section "Configuring DNS Server Zones".

   Note the following:

   - The *zone_index* value must be different for each zone.
   - You can select tier2 for up to 16 different zones (tier1 allows 6 zones).
   - Select the load-balancing method of your choice.

3. Create DNS records that point to VIPs that are currently associated with DNS names. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Domain Records".

   For example, suppose you have the following owner configuration.

```
!************************** OWNER ***************************
owner GLB

content rule1
    add service s1
    vip address 5.5.5.5
    add dns www.work.com
    active
```

   You would need to add a record similar to the following.

```
(config)# dns-record a www.work.com 5.5.5.5 0 single kal-ap
1.1.1.1
```

For details on configuring zone-based DNS, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring a CSS as an Authoritative DNS Server".

For details on configuring content rule-based DNS, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Content Rule-Based DNS".

Configuring DNS Sticky Parameters

The following sections describe the commands and their options and variables that you use to configure DNS Sticky.

Enabling the Global Sticky Database

Note Because the Global Sticky Database (GSDB) requires the configuration of a Proximity Database (PDB), you must configure a PDB before you enable the GSDB on the same CSS. For details on configuring a PDB, refer to Chapter 5, Configuring Network Proximity, in the section "Configuring a Proximity Database".

Use the **gsdb** command to enable a GSDB on a dedicated CSS 11150 with 256 MB of RAM when you are configuring GSLB with a GSDB (see "DNS Sticky With a GSDB" earlier in this chapter) or when you are using DNS Sticky in a Network Proximity configuration (see "DNS Sticky in a Network Proximity Environment" earlier in this chapter).

Note The **gsdb** command and its **show** and **no** versions are part of the PDB feature set and require the PDB license key.

Note You do not need to configure a GSDB to use the basic DNS Sticky feature in a GSLB environment. However, a GSDB provides a more robust DNS Sticky and load-balancing configuration. For details on the available types of DNS Sticky configurations, see "DNS Sticky Overview" earlier in this chapter.

The syntax for this global configuration mode command is:

gsdb

To disable a GSDB, enter:

```
(config)# no gsdb
```

Resetting the Global Sticky Database Statistics

Use the **gsdb zero** command to reset the Sticky Lookups and Sticky Sets statistics that are displayed by the **show gsdb** command. The syntax for this global configuration mode command is:

gsdb zero

Configuring the Global Sticky Database Interface

Use the **gsdb-interface** command on the CSS DNS server to create an interface for the CSS to communicate with a GSDB. A GSDB responds with a zone index to sticky queries from CSS DNS servers. All GSDBs participating in a peer mesh share sticky TTL and sticky zone information over APP.

Note The **gsdb-interface** command and its **no** version are part of the Enhanced feature set.

The syntax for this global configuration mode command is:

gsdb-interface [**primary**|**secondary**] *ip_address*

The variables and options are:

- **primary**|**secondary** - Specifies an interface for the primary or secondary GSDB. The CSS uses the primary GSDB for sticky requests unless it is unavailable, in which case it uses the secondary GSDB.
- *ip_address* - IP address of the GSDB. Enter the address in dotted-decimal notation (for example, 192.168.11.1).

Note In a Network Proximity configuration, the IP address of the primary GSDB interface is typically the same as the IP address of the PDB.

For example:

```
(config)# gsdb-interface primary 192.168.11.1
```

To delete a primary GSDB interface, enter:

```
(config)# no gsdb-interface primary
```

Resetting the Global Sticky Database Interface Statistics

Use the **gsdb-interface zero** command to reset the GSDB interface statistics that are displayed by the **show gsdb-interface** command. The syntax for this global configuration mode command is:

gsdb-interface zero

Configuring the Time to Live for Global Sticky Database Entries

Issue the **gsdb ttl** command on the GSDB to specify a time to live (TTL) for the GSDB sticky domain entries. The value you enter determines the length of time in seconds that GSDB entries are valid. Any new request from a D-proxy for a sticky domain that arrives before the timer expires, resets the timer.

The syntax for this global configuration mode command is:

gsdb ttl *ttl_value*

The variable is *ttl_value*, which specifies the length of time in seconds that GSDB entries are valid. Enter an integer between 300 and 1000000. The default is 7200 seconds (2 hours).

For example:

```
(config)# gsdb ttl 7200
```

Configuring Sticky Domain Records

Use the **dns-record** command to configure sticky domain records on the CSS configured as a DNS server. Domain records labeled as **sticky-enabled** indicate to the CSS that it should attempt to provide a sticky response before it answers the DNS query from the client.

Note If you configure a GSDB and any sticky domains in a particular zone, you must configure all sticky domains participating in the peer mesh in that same zone. Otherwise, the thrashing of the sticky zone index could cause DNS Sticky to fail.

For details on configuring domain records, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Domain Records".

Configuring Server Zones for DNS Sticky

Use the **dns-server zone** command to configure DNS server zones in the CSS. This feature allows the CSS to respond to DNS requests based upon different balance criteria and domain availability within zones or locations. For details on configuring zones, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring DNS Server Zones".

Displaying DNS Sticky Statistics

Use the following **show** commands to display DNS Sticky statistics for the GSDB, GSDB interface, domain records, and DNS server zones.

Displaying Global Sticky Database Statistics

Use the **show gsdb** command to display GSDB statistics. This command is part of the PDB feature set and is available in all modes. The syntax is:

show gsdb

Table 2-6 describes the fields in the **show gsdb** output.

Table 2-6 Field Descriptions for the show gsdb Command

Field	Description
Sticky Lookups	The number of sticky requests received from a CSS DNS server.
Sticky Sets	The number of times the DNS server selected the sticky zone and informed the GSDB because the GSDB did not have the zone information in its database.
Sticky BLKs Present	The number of sticky blocks that are currently in the GSDB. The sticky blocks contain the zone and TTL information for sticky-enabled domains.
Sticky TTL	The time to live (in seconds) of a sticky entry in the GSDB. Values range from 300 to 1000000 seconds. The default is 7200 seconds (2 hours).

Displaying GSDB Interface Statistics

Use the **show gsdb-interface** command to display statistics for the GSDB interface on the DNS server CSS. This command is part of the Enhanced feature set and is available in all modes.

Note This command is not available on a PDB or a GSDB.

The syntax is:

show gsdb-interface

Table 2-7 describes the fields in the **show gsdb-interface** output.

Table 2-7 Field Descriptions for the show gsdb-interface Command

Field	Description
Active GSDB	The GSDB that is currently being used: Primary or Secondary.
Primary Trans	The number of times the primary GSDB transitioned state between Up and Down.
Primary Req	The number of requests received by the primary GSDB from DNS servers.
Primary Rsp	The number of responses sent to DNS servers by the primary GSDB.
Secondary Trans	The number of times the secondary GSDB transitioned state between Up and Down.
Secondary Req	The number of requests received by the secondary GSDB from DNS servers.
Secondary Rsp	The number of responses sent to DNS servers by the secondary GSDB.
Total Req	The total number of requests sent by the DNS server to the GSDB.
Total Rsp	The total number of responses received by the DNS server from the GSDB.

Displaying DNS Sticky Domain Record Statistics

Use the **show dns-record sticky** command to view statistics associated with sticky domain records. This command is part of the Enhanced feature set and is available in all modes. The syntax is:

show dns-record sticky {*dns_name*}

The variable is *dns_name*, which is the DNS name mapped to a domain record for which you want to display sticky domain statistics. Enter the name as a lower case unquoted text string with no spaces and a maximum of 63 characters.

Table 2-8 describes the fields in the **show dns-record sticky** output.

Table 2-8 Field Descriptions for the show dns-record sticky Command

Field	Description
Name	The name of the sticky domain associated with the record.
Last Zone	The zone index of the last zone that was selected either by the GSDB or by the DNS server's load-balancing method.
Last IP Used	The last source (D-proxy) IP address used as a key to make a sticky decision.
Sets	The number of times the DNS server selected the sticky zone and informed the GSDB because the GSDB did not have the zone information in its database.
GSDB Lookups	The number of times a DNS server sent a sticky lookup request to the GSDB for the specified domain.
GSDB Responses	The number of times the GSDB responded to GSDB Lookup requests from a DNS server for the specified domain.

Displaying Domain Load Statistics

Use the **show dns-record load** command to display load information associated with domains. The syntax for this all configuration mode command is:

show dns-record load {*dns_name*}

The variable is *dns_name*, which is the DNS name mapped to a domain record for which you want to display load statistics. Enter the name as a lower case unquoted text string with no spaces and a maximum of 63 characters.

Table 2-9 describes the fields in the **show dns-record load** output.

Table 2-9 Field Descriptions for the show dns-record load Command

Field	Description
Name	The name of the domain associated with the record.
LeastLoaded	The zone index of the current least-loaded zone in the peer mesh.

Table 2-9 Field Descriptions for the show dns-record load Command (continued)

Field	Description
Zone	The zone index of the zone or zones in which the record exists. An asterisk (*) indicates the zone index of the local zone.
Description	A text description of the zone.
Type	The record type: • **A** - Address record • **NS** - Name server record
IP Address	The IP address associated with the record for the returned zone.
Load	The load number, an integer from 2 to 255 indicating the zone's current burden for the specified domain. A load of 255 indicates that the service is offline.A dash (-) indicates an undefined load, that is, load reports are not being received.
MinRespTime	The response time of the fastest server associated with the zone. This parameter value is used to break ties when load numbers are similar. A dash indicates an undefined MinRespTime.

Displaying DNS Record Statistics

Use the **show dns-record statistics** command to display statistics associated with the domain records configured locally and learned by the CSS from its peers. For details on displaying DNS record statistics, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Displaying DNS Record Information".

Displaying DNS Record Keepalives

Use the **show dns-record keepalive** command to display information about keepalives associated with DNS records. For details on displaying DNS record keepalives, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Displaying DNS Record Keepalive Information".

Displaying Proximity/GSDB Metrics

Use the **show proximity metric** command to display GSDB and/or PDB metrics (in milliseconds) associated with a client's local DNS server IP address. This command is available on a GSDB, a PDB, and a PDNS.

The syntax for this global configuration mode command is:

show proximity metric *ip_address* {*ip_prefix* {**aggregate**}}

The variables and options are:

- *ip_address* - IP address of the client's local DNS server for which you want to display proximity metrics. Enter the address in dotted-decimal notation (for example, 192.168.11.1).
- *ip_prefix* - This optional parameter is used to map an IP prefix to an IP address allowing you to view metrics over a range of IP addresses. Enter the prefix as either:
  - A prefix length in CIDR bitcount notation (for example, /24).
  - A subnet mask in dotted-decimal notation (for example, 255.255.255.0).
- **aggregate** - This optional keyword allows you to view aggregated metrics that are available in both /16 and /8 subnet masks.

Note Probed metrics are statistically aggregated at the /8 and /16 prefix levels.

In the GSDB, the metrics are sorted by sticky zone index. In the PDB, the round-trip time (RTT) metrics are sorted by proximity zone. In the PDNS, the metrics are sorted by RTT. An asterisk next to a zone indicates the local zone where the command was issued.

Note The maximum value of an RTT metric is 3968 ms. A value of 4095 ms indicates that a client's local name server was unreachable or had an RTT value of more than 4 seconds.

For example, to view the PDB and /or GSDB metrics associated with the client IP address of 172.23.5.7 and an IP prefix of 24, enter:

```
(config)# show proximity metric 172.23.5.7/24
```

Table 2-10 describes the fields in the **show proximity metric** output.

Table 2-10 Field Descriptions for the show proximity metric Command

Field	Description
IP Address	The IP address of the client's local DNS server for which you want to display metrics.
IP Prefix	The IP pefix length or subnet mask associated with the IP address.
Index	The zone index number associated with the DNS zone. An asterisk (*) indicates the local zone where you issued the command.
Description	A logical name or text description of the zone.
Metric	The round-trip time (RTT) between the PDB and a referral-DNS server. The DNS server uses the RTT as the proximity metric for load-balancing decisions.
Sticky Value	The sticky zone index stored in the GSDB and returned to the PDNS after a sticky lookup.
TTL	For DNS Sticky configurations, the remaining time-to-live (TTL) in seconds for this GSDB entry.

Displaying Server Zones for DNS Sticky

Use the **show zone command** to display information about DNS server zones communicating in a zone mesh. For details on displaying DNS server zones, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Displaying DNS Server Zones".

Configuring a CSS as a Content Routing Agent

This chapter provides an overview of the CSS Content Routing Agent (CRA) feature and describes how to configure it for operation. Information in this chapter applies to all CSS models, except where noted.

This chapter provides the following sections:

Overview of the Content Routing Agent Feature

To improve a client's overall browser experience by decreasing the response times for content requests, configure a CSS as a Content Routing Agent (CRA). A Cisco Content Router 4430-B (Content Router) running software version 1.1 redirects a client to the closest (best) replicated-content site represented by a CRA, based on network delay using a software process called boomerang. For details on the Cisco Content Router software and boomerang, refer to the *Cisco Content Routing Software Configuration Guide and Command Reference*, Release 1.1.

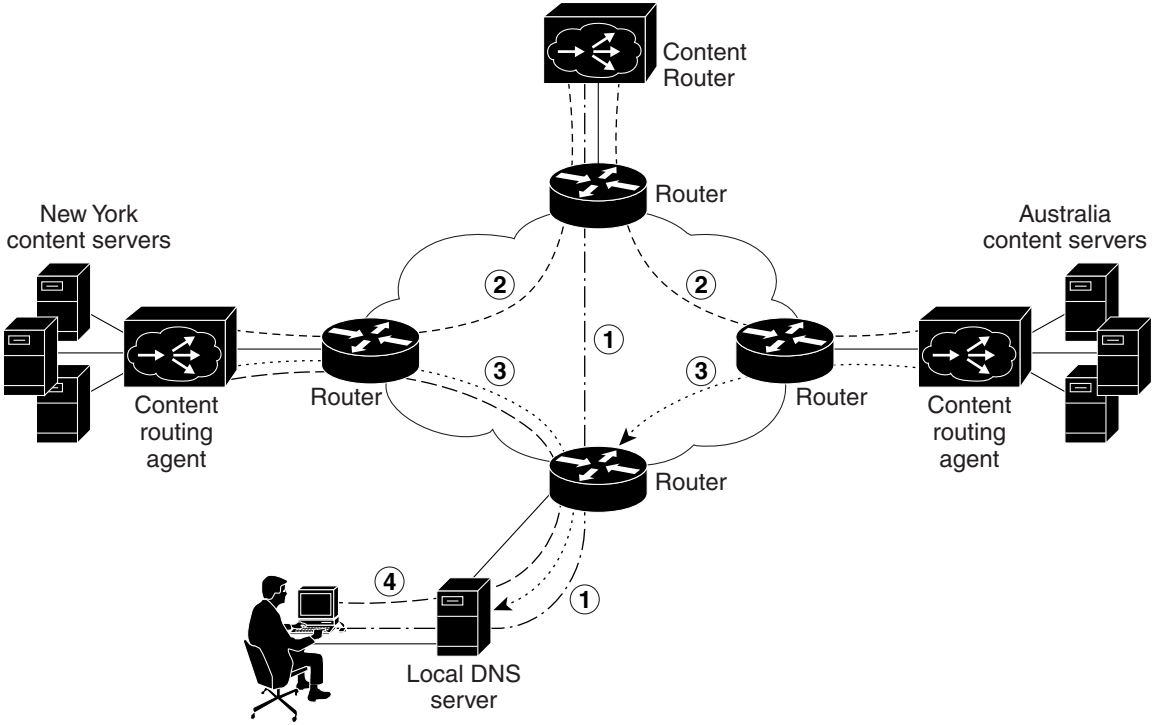
Configure a CRA at each content site within each domain that you want to support. This configuration also requires a Content Router.

The Content Router intercepts DNS requests from a client, then routes them to a CRA. For example, to route www.foo.com, the address record (A-record) in the primary DNS server for www.foo.com is changed to a name server record (NS-record) pointing to the Content Router. The Content Router and its CRAs handle all requests for the IP address of www.foo.com. When the CRAs receive a DNS request from the Content Router, the CRAs respond to the client's local name server at the same time. The first response through the network is used and the local name server discards all other responses. The CRA with the winning response is the site to which the client connects.

Figure 3-1 shows an example of the boomerang process in direct mode. A CSS configured as a CRA also works with a Content Router operating in (WCCP) mode. For more information on Content Router modes, refer to the *Cisco Content Routing Software Configuration Guide and Command Reference*, Release 1.1.

Note The Content Routing Agent feature is part of the CSS Standard feature set.

Figure 3-1 Example of Boomerang Content Routing Process - Direct Mode



1. DNS request is sent to Content Router.
2. Content Router forwards request to content routing agents.
3. Agents simultaneously send responses back to local DNS server. First response through the network contains the IP address of the best site.
4. User connects to best site.

46626

Content Routing Agent Quick Start

Table 3-1 provides a quick overview of the steps required to configure the Content Routing Agent feature on a CSS. 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, refer to the following sections.

Table 3-1 Content Routing Agent Configuration Quick Start

Task and Command Example
1. Configure a Cisco Content Router 4430-B. Configure Content Routing Agents (CRAs) and the domains associated with them on the Content Router. For details, refer to the *Cisco Content Routing Software Configuration Guide and Command Reference*, Release 1.1.
2. On a CSS that you want to configure as a CRA, enter config mode. **# config** (config)#
3. Enable the CRA feature on a CSS. **(config)# dns-boomerang client enable**
4. Create a domain record in the CRA for each domain with which you associated the CRA when you configured the Content Router. **(config)# dns-boomerang client domain www.sample.com 192.168.1.1**
5. Optionally, configure an alias for each configured domain to reduce administrative overhead. **(config)# dns-boomerang client domain www.sample.com alias gif.www.sample.com**
6. Display CRA statistics. (config)# **show dns-boomerang client**

Configuring Content Routing Agent Parameters

The following sections describe the CLI commands and their options and variables that you use to configure the CSS as a CRA.

Enabling the Content Routing Agent

Use the **dns-boomerang client enable** command to enable the CRA functionality on the CSS. There are no options for this global configuration mode command.

For example:

```
(config)# dns-boomerang client enable
```

To disable the CRA, enter:

```
(config)# no dns-boomerang client enable
```

Configuring the CPU Load Threshold

Use the **dns-boomerang client cpu-threshold** command to set the CSS CPU load threshold for domains configured to use or return a local virtual IP address (VIP). If the CPU load exceeds the configured threshold value, then the CSS drops subsequent incoming DNS requests from the Content Router until the load is lower than the threshold.

The syntax for this global configuration mode command is:

dns-boomerang client cpu-threshold *threshold*

The variable for this command is *threshold*. Enter an integer from 1 to 99. The default is 99.

For example:

```
(config)# dns-boomerang client cpu-threshold 50
```

To reset the CSS CPU threshold to the default value, enter:

```
(config)# no dns-boomerang client cpu-threshold
```

Note To display the CPU load, use the **show system-resources** command.

Configuring Content Routing Agent Domain Records

Use the **dns-boomerang client domain** command to create a domain record in the Content Routing Agent DNS server for each of the domains you associated the agent with when you configured domains on the Content Router. If the matching domain record keepalive messaging succeeds, the CSS uses this record for DNS resolutions. There is no Content Routing Agent configuration mode. Unlike other **dns-record** commands on the CSS, this command requires keywords for specifying options. (For details on configuring DNS domain records for other DNS applications, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Domain Records".)

The syntax for this global configuration mode command is:

dns-boomerang client domain *dns_name ip_or_host* {**"***uri***"**} {**key** [**"***secret***"**|**des-encrypted** *encrypted_key*|**"***encrypt_key***"**]} {**dns-ttl** *number1*} {**ip-ttl** *number2*} {**threshold** *number3*}]

The variables and options for this command are:

- *dns_name* - The domain name mapped to the client record. Enter the name as a case-sensitive, unquoted text string with no spaces and a maximum length of 72 characters. For example, www.sample.com.
- *ip_or_host* - The IP address or the host name of the content server or web cache bound to the domain name on the CSS. This IP address can be a local VIP. Enter the address in dotted-decimal notation (for example, 192.168.11.1).
- **"***uri***"** - The optional URI that the CSS uses for the keepalive probe to the Content Router for a domain. Enter a quoted text string with a maximum of 255 characters. If you do not prepend the URI with a slash (/) character, the CSS prepends it.
- **key** - The optional keyword that defines the clear-text shared RC4 secret or DES encryption key on the Content Router. See Table 3-2 for a comparison of how you configure a password on a CSS (configured as a CRA) and on a Content Router.

- **"***secret***"** - The optional clear-text Content Router secret for encrypting packets sent between a Content Router and a CRA. The secret you specify here must match the secret configured on the Content Router. Enter the secret as a case-sensitive quoted text string with no spaces and a maximum of 64 characters (not including the quotes). For example, if MySecret is the secret configured on the Content Router for this domain, then enter **"MySecret"**.
- **des-encrypted** - The optional keyword that specifies that a Data Encryption Standard (DES)-encrypted password follows.
- *encrypted_key* - The DES encryption key that the CSS had previously encrypted. The CSS does not re-encrypt this key and saves it in the running-config as you entered it. Enter an unquoted case-sensitive text string with no spaces and a maximum of 64 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 16 characters.

Table 3-2 Configuring a Password on a CSS (CRA) Versus a Content Router

CSS Password Command	Content Router Password Command
key "*secret***"**	no equivalent
key des-encrypt "*password***"**	**key** *word* or **key 0** *word*
key des-encrypt *password*	**key 7** *word*

Note The DES encryption algorithm on the CSS is different from the Cisco Type 7 encryption algorithm on the Content Router. Therefore, encrypted passwords are displayed differently on the CSS and on the CR.

- **dns-ttl** *number* - The optional DNS time-to-live keyword and value in seconds returned in the CRA's DNS responses. This option determines the length of time a DNS server caches the returned information for reuse. Enter an integer from 1 to 2147483647 seconds. The default is the dns-ttl value configured on the Content Router.

- **ip-ttl** *number* - The optional IP routing time-to-live keyword and value in router hops returned in the CRA's DNS responses. This option determines how many router hops a response packet traverses enroute to the D-Proxy before it is discarded. This helps to eliminate the CRA from longer response routes. Enter an integer from 1 to 255. The default is the ip-ttl value configured on the Content Router.
- **threshold** *number* - The optional load threshold for testing the keepalive state of a local VIP. If the load on the dns-record associated with the content rule is greater than the threshold value, then the CSS drops subsequent Content Router requests for that record until the load is lower than the threshold. Enter an integer from 2 to 254. The default value is 254.

Note You must also configure the **add dns** command in the VIP's content rule to add domain names. Refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 3, Configuring Content Rules, in the section "Adding a Domain Name System to a Content Rule".

For example:

```
(config)# dns-boomerang client domain www.foo.com 192.168.11.1 key
"MySecret" dns-ttl 240 ip-ttl 5 threshold 175
```

To remove a CRA domain, enter:

```
(config)# no dns-boomerang client domain www.foo.com
```

Configuring an Alias for an Existing Client Domain

Use the **dns-boomerang client domain alias** command to create an alias for an existing client domain. The alias behaves exactly the same as the configured domain name. This command reduces administrative overhead by allowing you to use the shorter alias name instead of the domain name, IP address, and all the other options and variables for the configured record. The syntax for this global configuration mode command is:

dns-boomerang client domain *dns_name* **alias** *alias_name*

The variables and options are:

- *dns_name* - The domain name of a configured client record. Enter the name as a case-sensitive, unquoted text string with no spaces and a maximum of 72 characters.
- **alias** - The keyword required to create an alias name.
- *alias_name* - The domain name that the CSS treats exactly the same as the associated *dns_name*. Enter the name as a case-sensitive, unquoted text string with no spaces and a maximum of 72 characters.

For example:

```
(config)# dns-boomerang client domain www.sample.com alias
gif.www.sample.com
```

To remove the alias, enter:

```
(config)# no dns-boomerang client domain alias www.sample.com
```

Clearing Domain Statistics

Use the **dns-boomerang client zero** command to clear the statistics for one or all configured domains. If you do not specify a domain name, the CSS clears the statistics for all configured domains. This command is available in SuperUser and all configuration modes.

The syntax for this global configuration mode command is:

dns-boomerang client zero *dns-name*

The variable for this command is *dns_name*, the domain name mapped to the client record statistics that you want to clear. Enter the name as a case-sensitive, unquoted text string with no spaces and a maximum of 72 characters.

For example:

```
(config)# dns-boomerang client zero www.sample.com
```

Displaying Content Routing Agent Statistics

Use the **show dns-boomerang client** command to display information for all configured domains. This command is available in SuperUser and all configuration modes.

The syntax for this global configuration mode command is:

show dns-boomerang client {**all**|**global**|**domain** {*domain_name*}}

The options and variable for this global configuration mode command are:

- **all** - Displays all information (global and domain-related) for all domains mapped to a client record. Same as the **show dns-boomerang client command**.
- **global** - Displays global information only for all domains mapped to a client record.
- **domain** - Displays domain-related information for all domains mapped to a client record.
- *domain_name* - Displays domain-related information for the specified domain.

For example:

```
(config)# show dns-boomerang client global
```

Table 3-3 describes the fields in the **show dns-boomerang client** output.

Table 3-3 Field Descriptions for the show dns-boomerang client Command

Field	Description
Total DNS A-record requests	The total number of address record requests from the Content Server.
Total packets dropped	
Unknown domain	The number of DNS packets with domains not configured on this CSS (for Content Routing).
Invalid source address	The number of packets with invalid source addresses.

Table 3-3 Field Descriptions for the show dns-boomerang client Command (continued)

Field	Description
Excess length	The number of packets that had lengths longer than what the CR could send.
CPU threshold exceeded	The number of packets dropped because the CPU threshold was exceeded.
Configured CPU threshold	The configured threshold value above which the CSS drops requests from the Content Router.
Rule load threshold exceeded	The number of requests from the Content Router that were dropped because the load on a local rule exceeded the configured threshold.
Keepalive state Down	The number of packets dropped because the keepalive failed.
Security failure	The number of requests for this domain that were dropped due to security errors (key/secret failure or mismatch).
Domain	The DNS name mapped to the client record.
Content server	The address of the content server bound to the domain.
Origin server	The address for the most recently used origin server that was passed from the Content Router.
Bad probes	The number of times (in percent) that the keepalive message failed to find the service Up.
DNS A-record requests	The number of DNS address record requests for this domain from the Content Router.
Dropped (server down)	The number of requests for this domain that were dropped because the server was down.
Dropped (CPU busy)	The number of requests for this domain that were dropped because the CPU threshold was exceeded.
Dropped (rule load exceeded)	The number of requests from the Content Router that were dropped because the load on a local rule exceeded the configured threshold.

Table 3-3 Field Descriptions for the show dns-boomerang client Command (continued)

Field	Description
Configured threshold	The load threshold value you configured with the **dns-boomerang client domain** command to test the keepalive state of a local VIP.
Security failures	The number of requests for this domain that were dropped due to security errors (key/secret failure or mismatch).
Alias	The alias that maps to the configured domain name.
DNS A-record requests	The number of DNS address record requests for this alias from the Content Router.

Configuring a Client Side Accelerator

This chapter provides an overview of the CSS Client Side Accelerator (CSA) feature and describes how to configure it for operation. Information in this chapter applies to all CSS models, except where noted.

Note The CSA feature is part of the CSS Enhanced feature set.

This chapter provides the following sections:

Overview of Client Side Accelerator

To accelerate the retrieval of domain content, configure a CSS as a Client Side Accelerator (CSA), using transparent caches (TCs) to store content locally. This feature improves a user's experience by reducing the time for content to arrive in a browser.

A CSA resides on the client side of the Internet and is the first DNS server to which clients send a DNS request. When a CSA receives a DNS request for content located in a domain configured for acceleration and the number of requests exceeds the configured threshold, the CSA returns an address record (A-record) of the local virtual IP address (VIP) to the client. The client uses the IP address in the A-record to connect to the service in a local TC farm.

For non-accelerated content or unresolvable DNS requests, the CSA sends the DNS request to a DNS server forwarder. The forwarder, which is not a CSS, is a fully-functional Berkeley Internet Name Domain (BIND) DNS server. The forwarder returns a DNS response to the client transparently through the CSA.

You can configure a peer mesh of multiple CSAs that belong to one service provider but are located at various points of presence (POPs). Using Application Peering Protocol (APP), the CSAs in a peer mesh share accelerated domain records. This allows you to leverage content available at a cache farm in one POP to provide content to clients located at another POP. Once the same candidate domain has been accelerated at two POPs, cache backup can occur if a cache at one of those POPs fails.

Service providers can use CSAs to bill back domain acceleration to content providers. You can configure certain domains for acceleration, then bill back content providers based on the number of hits on the accelerated domains.

Example CSA Configurations

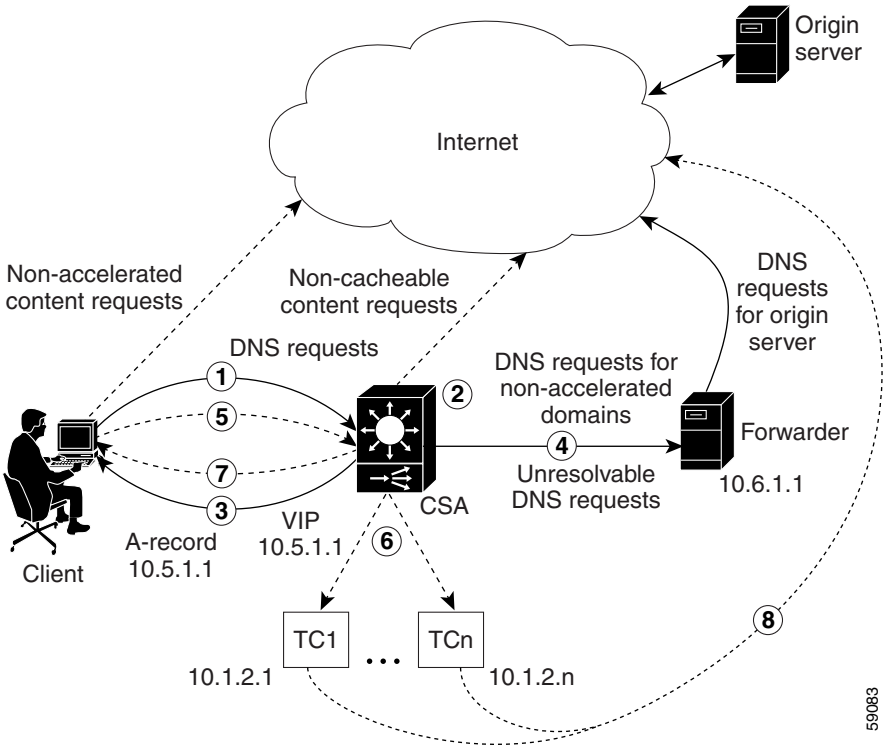
This section describes two possible CSA configurations: the first, a single POP (Figure 4-1) and the second, two POPs using an APP peer mesh (Figure 4-2).

Single POP CSA Configuration

The following sequence describes the steps depicted in Figure 4-1:

1. The Client population sends DNS requests to the CSA for DNS resolution of the domain name www.acme.com.
2. The CSA is configured to accelerate the domain www.acme.com.
3. When the number of requests for www.acme.com exceed the configured threshold (or the threshold has already been exceeded), the CSA returns the accelerated VIP in an A-record to the clients.
4. For all other requests, the CSS forwards the queries to the DNS server forwarder for resolution.
5. Clients initiate a connection with the CSA for www.acme.com using the VIP in the A-record.
6. The CSA matches the request on a layer 5 content rule that has transparent caches configured as the services. The CSA performs destination NATing based on the host tag and performs MAC forwarding.
7. If the cache contains the content, the CSA returns it to the clients.
8. If the cache does not contain the content, the cache fetches the content from the origin server.

Figure 4-1 Example of a Client Side Accelerator Configuration Example

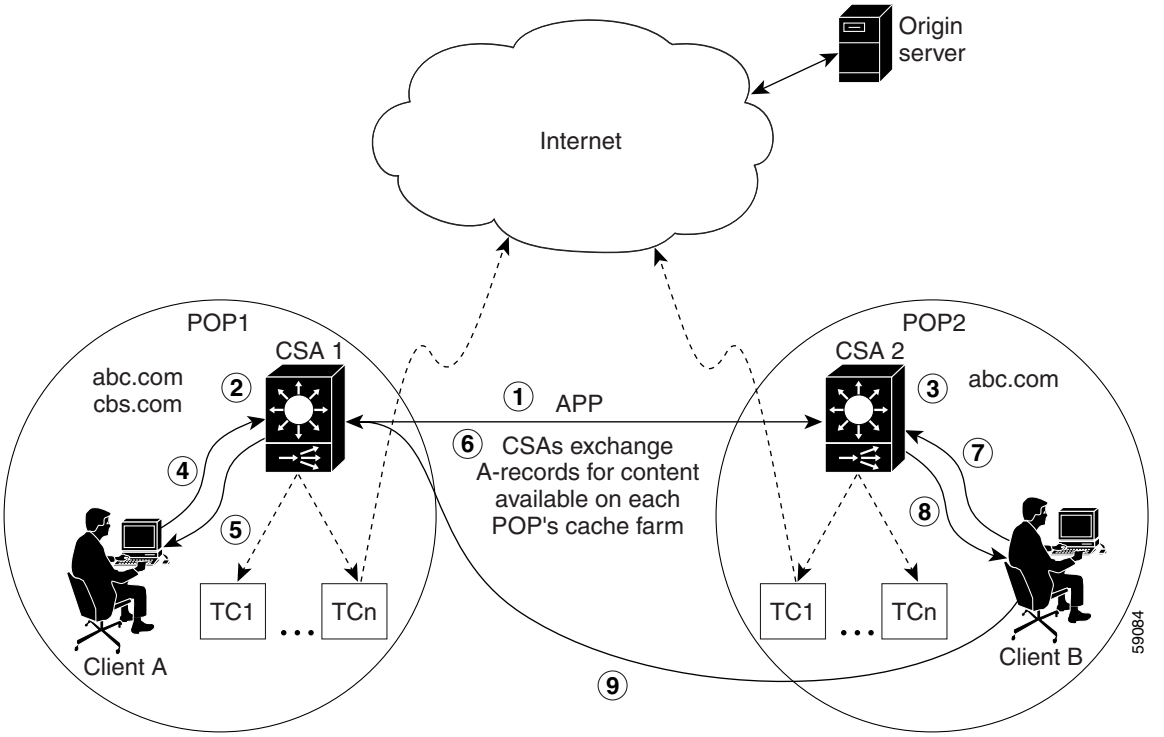


Multiple POP CSA Configuration

The following sequence describes the steps depicted in Figure 4-2.

1. An APP mesh is configured for the CSAs in POP1 and POP2.
2. CSA1 in POP1 is configured to accelerate the domains abc.com and cbs.com.
3. CSA2 in POP2 is configured to accelerate the domain abc.com.
4. Client A population sends DNS requests to CSA1 in POP1 for abc.com.
5. When the number of requests for abc.com exceeds the configured threshold, CSA1 creates an A-record for abc.com and returns it to the clients. Clients in POP1 initiate a connection with CSA1 using the VIP in the A-record.
6. CSA1 also sends the A-record for abc.com out on the APP mesh.
7. Client B population sends DNS requests for abc.com to CSA2 in POP2. If CSA2 is configured to accelerate abc.com in multiple locations and if the domain becomes hot (requests exceed configured threshold), repeat steps 5 and 6 for CSA2 in POP2.
8. If abc.com is not yet hot in POP2 or if CSA2 is configured to accelerate the domain in a single location, CSA2 sends the A-record in its domain database (learned in step 6) to the Client B population.
9. The clients in POP2 request content in abc.com from CSA1 in POP1.

Figure 4-2 Example of a Client Side Accelerator APP Mesh Configuration



Client Side Accelerator Quick Start

Table 4-1 provides a quick overview of the steps required to configure the Client Side Accelerator feature on a CSS. 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, refer to the following sections.

Table 4-1 Client Side Accelerator Configuration Quick Start

Task and Command Example
1. Enter config mode. **# config** (config)#
2. Enable Application Peering Protocol (APP). **(config)# app**
3. Configure an APP session with each remote CSA peer to create a peer mesh. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring an APP Session". **(config)# app session 192.160.1.2**
4. Configure back-end remapping as the preferred connection reset method. Refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 3, Configuring Content Rules. This step is recommended, but not required. **(config)# persistence reset remap** **(config)# bypass persistence disable**
5. Configure candidate domains for acceleration. See "Configuring Accelerated Domains" later in this chapter. **(config)# dns-record accel abc.com 192.168.1.1** **(config)# dns-record accel cbs.com 192.168.1.1**
6. Configure CSA and enable sharing of content between peer CSAs. See "Enabling the Client Side Accelerator" later in this chapter. **(config)# dns-server accelerate domains 50 30 256 single_location**
7. Configure the DNS server and the number of CSA peers. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring DNS Server Zones". **(config)# dns-server zone 1 tier2**

Table 4-1 Client Side Accelerator Configuration Quick Start (continued)

Task and Command Example
8. Configure the DNS server forwarder. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring a DNS Server Forwarder". `(config)# dns-server forwarder primary 192.168.1.10`
9. Enable the DNS server. `(config)# dns-server`

10. Configure the transparent caches as services. Refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 7, Configuring Caching.
`(config)# service transHosttag1`
`(config-service[transHosttag])# ip address 10.1.2.1`
`(config-service[transHosttag])# protocol tcp`
`(config-service[transHosttag])# port 80`
`(config-service[transHosttag])# type transparent`
`(config-service[transHosttag])# transparent-hosttag`
`(config-service[transHosttag])# active`
Repeat for each additional cache.

11. Configure a content rule with the same VIP address as that used for the accelerated records. Add the transparent caches as services. Refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 3, Configuring Content Rules.
`(config)# owner accelerate`
`(config-owner[accelerate])# content 15-accel`
`(config-owner-content[accelerate-15-accel])# VIP address 192.168.1.1`
`(config-owner-content[accelerate-15-accel])# protocol TCP`
`(config-owner-content[accelerate-15-accel])# port 80`
`(config-owner-content[accelerate-15-accel])# url "/*" eql cacheable`
`(config-owner-content[accelerate-15-accel])# add service transHosttag1`
`(config-owner-content[accelerate-15-accel])# add service transHosttagn`
`(config-owner-content[accelerate-15-accel])# no persistent`
`(config-owner-content[accelerate-15-accel])# balance url`
`(config-owner-content[accelerate-15-accel])# active`

Table 4-1 Client Side Accelerator Configuration Quick Start (continued)

Task and Command Example

12. Configure a service to bypass the cache farm for non-cacheable content. Refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 1, Configuring Services and Chapter 7, Configuring Caching.
`(config)# service bypassCache`
`(config-service[bypassCache])# ip address 0.0.0.0`
`(config-service[bypassCache])# protocol tcp`
`(config-service[bypassCache])# port 80`
`(config-service[bypassCache])# keepalive type none`
`(config-service[bypassCache])# bypass-hosttag`
`(config-service[bypassCache])# active`

13. Configure a content rule for non-cacheable content on domains that you want to accelerate. Add the bypass cache as the only service. Refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 3, Configuring Content Rules.
`(config)# owner accelerate`
`(config-owner[accelerate])# content nonCacheable`
`(config-owner-content[accelerate-nonCacheable])# VIP address 192.168.1.1`
`(config-owner-content[accelerate-nonCacheable])# protocol TCP`
`(config-owner-content[accelerate-nonCacheable])# port 80`
`(config-owner-content[accelerate-nonCacheable])# url "/*"`
`(config-owner-content[accelerate-nonCacheable])# add service bypassCache`
`(config-owner-content[accelerate-nonCacheable])# no persistent`
`(config-owner-content[accelerate-nonCacheable])# active`

Configuring Client Side Accelerator Parameters

The following sections describe the CLI commands and their options and variables that you use to configure Client Side Accelerator on a CSS.

Enabling the Client Side Accelerator

Use the **dns-server accelerate domains** command to enable the CSA functionality on a CSS. This command enables the acceleration of domains that have been or will be configured for acceleration using the **dns-record accel** command. The CSA uses the *threshold* variable to determine if it should accelerate a candidate domain. You can also configure whether or not the CSA shares content with peer CSAs.

The syntax for this global configuration mode command is:

dns-server accelerate domains *threshold interval max_number* [**single_location**|**multi_location**]

The variables and options for this command are:

- *threshold* - The number of hits per *interval* below which the CSA does not accelerate a domain. When the number of hits equals or exceeds the threshold during the configured *interval*, the CSA accelerates the domain. Enter an integer from 0 to 65535. The default is 0, indicating immediate acceleration.
- *interval* - The time period in minutes over which the CSA samples the hits on a domain and compares the number of hits with the configured *threshold* value to determine hot content and domain acceleration. Enter an integer from 1 to 60 minutes. The default is 5 minutes.
- *max_number* - The maximum number of domains that the CSA can accelerate. Enter an integer from 0 to 4096. The default is 1024.
- **single_location** - Allows the acceleration of a domain at one cache farm (POP) at a time. This is the default behavior.

- **multi_location** - Allows multiple CSAs to accelerate the same domain, possibly resulting in multiple cache farms maintaining the same content. This can occur when two or more CSAs (located in different POPs) are configured for multi_location and accelerate the same domain. Each cache farm will maintain the same content after:
  - The CSAs accelerate the same domain
  - A cache in each POP retrieves the same content from the origin server

The following command example:

- Accelerates domains that are accessed at least at the rate of 50 hits per minute.
- Accelerates a maximum of 100 candidate domains at any given time.
- Forces this CSA to allow acceleration of a given domain to occur in only one cache farm at a time.

```
(config)# dns-server accelerate domains 50 1 100 single_location
```

To disable CSA functionality on a CSS, enter:

```
(config)# no dns-server accelerate domains
```

Configuring the Domain Cache

Use the **dns-server domain-cache** command to enable the tracking of DNS request counts and to configure the domain cache on the CSA. As requests arrive at the CSA, entries are created or updated in the domain cache with the hit counts. You can use this command with the **show dns-server domain-cache** command to determine which domains you want to accelerate.

Note Do not use this command during normal CSA operation. It causes unnecessary overhead on the CSA.

The syntax for this global configuration command is:

dns-server domain-cache {*cache_size ageout*|**purge** {*dns_name*}|**zero** {*dns_name*}}

The variables and options for this command are:

- *cache_size* - Specifies the number of domains that the CSA can cache. Enter an integer from 1 to 4096. The default is 1024.
- *ageout* - The maximum number of seconds that the domain entry remains in the cache. Enter an integer from 0 to 60. The default is 10 seconds. A value of zero causes the domain entry to never age out.
- **purge** - Removes all entries or the specified entries from the domain cache.
- *dns_name* - The DNS entry that you want to remove from the domain cache. To see a list of DNS entries, enter:

```
(config)# dns-server domain-cache purge ?
```

- **zero** - Resets all counters for all entries or the specified entry in the domain cache to zero.
- *dns_name* - The DNS entry for which you want to reset counters.

For example:

```
(config)# dns-server domain-cache 512
```

The above command creates a domain cache that can hold up to 512 most recently requested domain entries. The entries will age out and be removed from the domain cache after 10 seconds (the default).

Note The operation of the domain cache can impact the DNS request/response rate performance. Use the domain cache only when you need to identify potential acceleration candidates.

Configuring a DNS Server Forwarder

Use the **dns-server forwarder** command to configure a DNS server forwarder on a CSS configured as a CSA. If the CSA cannot resolve a DNS request, it sends the request to the forwarder to obtain a suitable response. For details on configuring a DNS server forwarder, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring a DNS Server Forwarder".

Configuring Accelerated Domains

Use the **dns-record accel** command to specify the domains you want to accelerate. Map the domain name to a content rule using an IP address.

Note If the content rule bound to the acceleration candidate domain is suspended or cannot provide service for content requests, the CSA does not accelerate the domain.

The syntax for this global configuration mode command is:

dns-record accel *dns_name ip_address* {*ageout*}

Note You cannot configure a domain name as two different DNS record types on the same CSS. For example, if you have configured abc.com as **dns-record accel**, you cannot configure it as **dns-record a** or **dns-record ns** on the same CSS. For information on configuring other types of DNS records, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Domain Records".

The variables and options for this command are:

- *dns_name* - The domain name that you want to map to the acceleration record. Enter the name as a case-sensitive unquoted text string with no spaces and a maximum of 63 characters.
- *ip_address* - The IP address of the local content rule that will handle content requests for the DNS name during content acceleration.
- *ageout* - The optional number of minutes that the domain remains accelerated. Enter an integer from 0 to 525600. The default is 180 minutes. If you enter 0, the accelerated domain record never ages out.

The following command example creates an acceleration record for the domain abc.com. When the number of requests for the domain exceeds the threshold specified in the **dns-server accelerate domains** command, the CSA accelerates the domain for six minutes. Clients can access the domain's content based on the content rule with the IP address 192.168.11.1.

```
(config)# dns-record accel abc.com 192.168.11.1 6
```

To delete the domain acceleration record, enter:

```
(config)# no dns-record accel abc.com
```

Resetting the DNS Record Statistics

Use the **dns-record zero** command to reset the DNS record statistics displayed by the **show dns-record** command. For details on resetting DNS record statistics, refer to Chapter 1, Configuring the CSS Domain Name Service, in the sections "Resetting the DNS Record Statistics" and "Displaying DNS-Record Statistics".

Configuring the CSA DNS Server Zones

Use the **dns-server zone** command to configure the number of DNS server zones on a CSA. For details on DNS server zones, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring DNS Server Zones".

Displaying Client Side Accelerator Information

Use the **show** commands in this section to display information and statistics for your CSA configuration.

Displaying the Client Side Accelerator Configuration

Use the **show dns-server accelerate domains** command to display the CSA configuration on a CSS. The syntax for this global configuration mode command is:

show dns-server accelerate domains

Table 4-2 describes the fields in the **show dns-server accelerate domains** output.

Table 4-2 Field Descriptions for the show dns-server accelerate domains Command

Field	Description
Current CSA Config	The state of the CSA configuration: disabled or enabled.
Threshold	The configured hits threshold used to determine whether or not a domain is accelerated. When the hits on the domain are greater than or equal to the threshold, the CSA accelerates the domain. The range is from 0 to 65535. The default is 0, indicating that the candidate domains are always accelerated.
Interval	The configured time interval, in minutes, over which the CSS samples the hits on the domain and compares it with the threshold. The range is from 1 to 60 minutes. The default is 5 minutes.
Expirations	The number of times that the configured interval expired. You can use this value to determine whether domains are being accelerated or decelerated as expected. The CSA decelerates an accelerated domain only after the interval expires.
Max. to Accel	The maximum number of domains that can be accelerated. The range is 0 to 4096. The default is 1024.
Location	Indicates whether a single or multiple CSAs maintain the same content. • Single-location, the default setting, allows the acceleration of a domain at one cache farm (POP) at a time. • Multi-location allows multiple CSAs to accelerate the same domain resulting in multiple cache farms maintaining the same content.
Candidates Total	The total number of candidate domains that are configured on the CSA.
Candidates Accelerated	The total number of domains that are currently accelerated.

Displaying DNS Server Domain Cache Statistics

Us the **show dns-server domain-cache** command to display statistics for the DNS server domain cache. Use this command with the **dns-server domain-cache** command to help you determine which domains you want to accelerate. The syntax for this global configuration mode command is:

show dns-server domain-cache {**summary**}

Table 4-3 describes the fields in the **show dns-server domain-cache** output.

Table 4-3 Field Descriptions for the show dns-server domain-cache Command

Field	Description
Domain	The domain name of the entry
Counter	The number of DNS requests

Note If you include the **summary** option, the command output displays the domain cache configuration and state only.

Displaying DNS Server Zones

Use the **show zone command** to display information about communication with, and the state of, the specified DNS zone or all zones in a peer mesh. For details on displaying DNS server zones, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Displaying DNS Server Zones".

Displaying DNS Record Keepalive Information

Use the **show dns-record keepalive** command to display DNS record keepalive information. For details on displaying DNS record keepalives, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Displaying DNS Record Keepalive Information".

Displaying Domain Acceleration Records Statistics

Use the **show dns-record accel** command to view statistics associated with domain acceleration records. The syntax for this global configuration mode command is:

show dns-record accel *dns-name*

The variable for this command is *dns_name,* the domain name mapped to the acceleration record that you want to display. Enter the name as a case-sensitive unquoted text string with no spaces and a maximum of 63 characters.

Table 4-4 describes the fields in the **show dns-record accel** output.

Table 4-4 Field Descriptions for the show dns-record accel Command

Field	Description
<Name>	The domain name for the acceleration record.
State	The state of the acceleration record, either ACCEL or NOT ACCEL. • **ACCEL** - Indicates that the record is currently accelerated • **NOT ACCEL** - Indicates the record is currently not accelerated
Vip Address	The virtual IP address of the local content rule that handles the content requests for the domain name during content acceleration.
Secs til Ageout	The number of seconds remaining until the CSA decelerates the domain record. The range is from 0 to 525600. The default is 180 seconds.
Interval Hits	The number of content hits that occurred during the interval set with the **dns-server domain-cache** command.
Total Hits	The total number of DNS hits for this record.
AccelCount	The number of times that content was accelerated.

Configuring Network Proximity

Network Proximity provides a content delivery solution that significantly improves network performance. This optional software feature uses a CSS 11150 configured as a database that is populated by actively probing the network to determine the proximity of clients and services. Additional CSSs (any model) perform database lookup requests and domain name resolution to determine the most proximate service for a client.

This chapter describes the Network Proximity feature and provides related configuration information in the following sections:

Caution If you configure your CSS as a Proximity Database, you cannot use the CSS for load balancing.

Entering Your Proximity License Keys

All CSSs include the Standard feature set. If you are upgrading your CSS with the Enhanced feature set or you have purchased the optional Proximity Database software feature, enter the appropriate software license key at the command line using the **license** command for the optional feature. If you ordered your software option with your CSS, you will find the option license key on a card in an envelope included in your CSS Accessory kit. If you ordered your software option after receipt of your CSS, the license key was mailed to you.

Note If you cannot locate the software license key, call the Cisco Technical Assistance Center (TAC) toll free, 24 hours a day, 7 days a week at 1-800-553-2447 or 1-408-526-7209. You can also e-mail the TAC at tac@cisco.com.

Proximity Database License Key

Caution If you configure your CSS as a Proximity Database, you cannot use the CSS for load balancing.

Enter the license key for the Proximity Database (PDB) software option on each CSS 11150 with 256 MB of memory that you want to use exclusively as a PDB.

To install the PDB software license key:

1. Log into the CSS and enter the **license** command.

   ```
   # license
   ```

2. Enter the 12-digit Secure Management license key.

   ```
   Enter the Software License Key (q to quit): nnnnnnnnnnnn
   ```

The PDB license key is now properly installed.

Note After you enter the software license key for the PDB, you must reboot the CSS before you can start using the new feature.

Proximity Domain Name Server License Key

Enter the license key for the Enhanced feature set, which includes the Proximity Domain Name Server (PDNS) feature, on each CSS (any model) that you want to use exclusively as a PDNS. For example:

```
# license
```

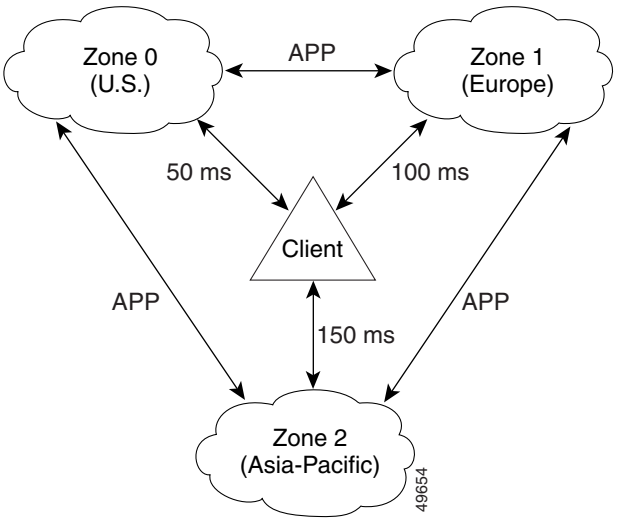
At the prompt, enter the 12-digit Enhanced feature set software license key. For example:

```
Enter the Software License Key (q to quit):
nnnnnnnnnnnn
```

Network Proximity Overview

Proximity represents a topological relationship between a client and content services. In a network topology perspective, as used in this chapter, proximity refers to connecting a client to the most proximate service based on a measurement of the round-trip time (RTT) between the client's local DNS server and a proximity zone (see Figure 5-1).

Figure 5-1 Simplified Example of Network Proximity



In Figure 5-1, the lowest RTT value is returned from Zone 0. Therefore, Network Proximity would link the client to a service located in Zone 0, regardless of the physical location of the client. The three zones communicate with each other using the Application Peering Protocol (APP). For details on APP, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol".

The major components and concepts in Network Proximity are:

- Proximity Database
- Proximity Domain Name Server
- Proximity Zones
- Peer Mesh

Proximity Database

A Proximity Database (PDB) is a dedicated CSS 11150 with 256 MB of memory *and* is configured as a PDB using the optional Proximity Database software feature. (For details on configuring a CSS 11150 as a PDB, see "Configuring a Proximity Database" later in this chapter.) One PDB and one or more Proximity Domain Name Servers (PDNSs) and data centers (server farms or lower-level DNS servers) define a subset of the Internet address space called a *proximity zone*. (For details on proximity zones, see "Proximity Zones" later in this chapter.)

Note A PDB can service up to four PDNSs generating their maximum request rates per zone. If the PDNSs are not fully loaded, you can configure additional PDNSs per zone.

Network Proximity, as implemented on a CSS, uses a topology-testing technique that actively probes clients to determine the relative location of clients and services. To accomplish this, a PDB uses ICMP and TCP requests to actively probe a client's local DNS server for proximity information. The PDB analyzes the probe responses, then stores the resulting network RTT metrics (in milliseconds) in its database.

When a PDNS sends the PDB a proximity lookup request for a client using APP-UDP, the PDB compares the RTT metrics for that client and responds immediately with an ordered *zone index*, a list of proximity zones in preferred order by RTT. The PDNS then uses the ordered zone index, along with domain name records and keepalive information, to determine the most proximate service for the client.

Note Probing conducted by the PDB is asynchronous with lookups conducted by the PDNS. Therefore, a PDB will never block a lookup request from a PDNS.

A PDB communicates with PDBs in other zones using a *peer mesh*, implemented with APP. (For details on APP, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol".) This enables PDBs to periodically "learn" the latest RTT metrics information from the other PDBs in the peer mesh to ensure that a client is connected to the most proximate service. Each PDB contains a metric for every source block of interest in each proximity zone. A source block of interest is a CIDR block that contains a client's local DNS server.

Proximity Domain Name Server

A Proximity Domain Name Server (PDNS) is any CSS that is running the Enhanced feature set *and* is configured as a PDNS using the Enhanced software feature set. (For details on configuring a CSS as a PDNS, see "Configuring a Proximity Domain Name Server" later in this chapter.) A PDNS performs PDB lookup requests, using Application Peering Protocol-User Datagram Protocol (APP-UDP), in response to DNS requests that the PDNS receives from a client's local DNS server. The PDB responds to these lookup requests immediately with the ordered zone index. The PDNS uses the ordered zone index along with domain name records and keepalive information to make an authoritative DNS response to the client's local DNS server.

The primary task of a PDNS is to respond to DNS requests based on proximity and domain availability. However, the CSS is not excluded from supporting local content rules and services, as well as non-Proximity-based DNS load balancing. These non-PDNS activities will affect the CSS's performance as a PDNS and the PDNS activities will affect the CSS's performance as a content services switch, depending on the PDNS's load.

Every proximity zone contains one or more PDNSs, up to a maximum of four generating their maximum request rates per zone. If the PDNSs are not fully loaded, you can configure additional PDNSs in a zone. Each PDNS within a zone acts as an authoritative DNS server for domains representing data centers. A data center can be a server farm attached directly to a CSS or can be a lower-level DNS server (which may or may not be a CSS) representing a server farm. You configure the domains statically on each PDNS.

Each PDNS maintains the following records for the domains configured on it:

- **Address record (A-record)** - Any domain that represents a data center, that is not front-ended by another DNS server, and that can be translated to an IP address.
- **Name server record (NS-record)** - Any domain that is front-ended by a lower-level DNS server (not necessarily a CSS).

A PDNS updates its domain records continually through keepalive messages (using ICMP or APP-UDP) that it sends to its locally configured virtual IP addresses (VIPs) and data centers. The PDNS uses the keepalive responses to track the load (kal-ap keepalive only, see below) and availability of locally configured domains. Each PDNS in a proximity zone shares its domain information with other PDNSs in each zone using an APP *peer mesh* (see "Peer Mesh" later in this chapter). There is no communication between PDNSs within the same zone, and each PDNS communicates with one PDNS per zone.

For the optional CSS keepalive type (kal-ap), the keepalive client resides on the PDNS, while the keepalive daemon resides on any CSS-based data center that is the configured recipient of A-records or NS-records as configured on the controlling PDNS. The keepalive daemon extracts the load information of the specified domain names and returns them to the PDNS. This load information originates from:

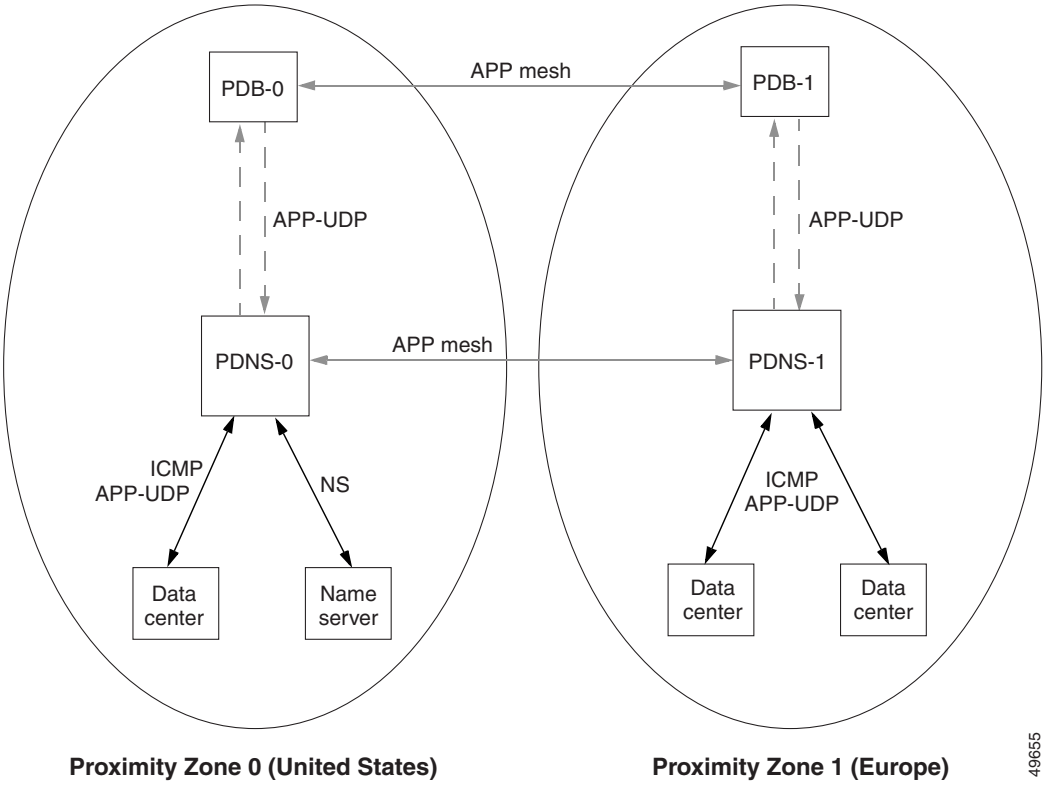
- The average load of the content rule to which the domain is attached
- The load of a locally configured A-record or NS-record

Proximity Zones

A proximity zone is a logical grouping of network devices that consists of one PDB, one or more PDNSs, and services. Although a zone is really a logical subset of the Ipv4 address space, a zone can also be geographically related to a continent, a country, or a major city. Refer to Figure 5-2.

For example, you can create proximity zones to group geographically distinct network devices. A proximity zone containing data centers in the United States logically groups nodes within a distinct geographical area. Another proximity zone may logically group nodes and data centers in Europe, for example. Zones are numbered beginning with zero.

Figure 5-2 Example of Network Proximity Zones



Peer Mesh

To communicate proximity information between proximity zones, Network Proximity uses APP to create a *peer mesh*. A peer mesh is an abstraction layer that uses APP to provide common functions (for example, zone configuration information) between Network Proximity devices. A *PDB mesh* allows PDBs to communicate with one another across proximity zones to share proximity metrics. A *PDNS mesh* allows a PDNS in one zone to communicate with one other PDNS in each proximity zone to share domain records and keepalive information. For details on APP, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol".

Note You can use the concept of zones with a peer mesh to share domain record information between CSSs acting as DNS servers without the use of a PDB. This configuration allows a scalable method of domain name sharing and the use of NS-records in a non-Proximity-based CSS DNS server environment. For more information, refer to Chapter 1, Configuring the CSS Domain Name Service.

Network Proximity Example

Note A PDB sends ICMP and TCP probes to a client's local DNS server the first time the PDB receives a lookup request for that client from a PDNS. If you configure refinement (refer to "Refining Proximity Metrics" later in this chapter), a PDB will continue to probe that client periodically. Based on the responses it receives from the probes and the information it receives through its peer mesh, a PDB builds and maintains a database of RTT metrics for clients throughout the network. This process is independent of, and asynchronous with respect to, client requests.

The following example illustrates a single point-in-time request from a client. Refer to Figure 5-3 for an illustration of the following steps.

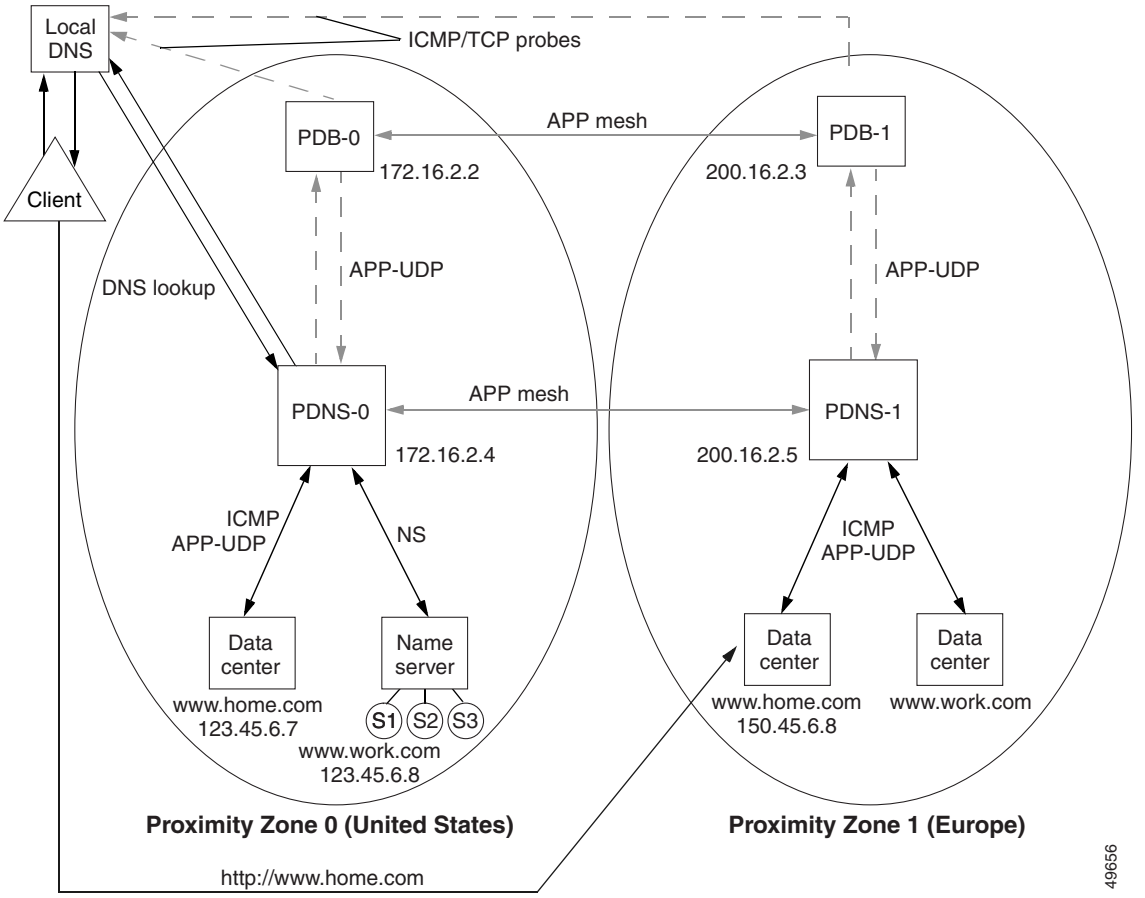
1. A client performs an HTTP request for the domain name *www.home.com*.
2. The local DNS server performs iterative DNS requests to the root server and to the .com server to resolve the domain name into an IP address or refers the client to an authoritative DNS server. (This step is not shown in Figure 5-3.)

3. The .com server, which is an authoritative DNS server for *home.com*, has the IP addresses of PDNS-0 and PDNS-1 in its configuration. Both PDNSs are authoritative DNS server for *www.work.com*. In this example, the .com server refers the local DNS server to PDNS-0 in Zone 0. (Typically, the .com server uses a roundrobin or other load-balancing method to refer local DNS servers to a PDNS. This step is not shown in Figure 5-3.)

Note Your configuration may include an enterprise DNS server that is positioned between the .com server and the PDNS. The enterprise DNS server would be an authoritative DNS server for *home.com*. In this case, the enterprise DNS server contains the IP addresses of the PDNSs in its configuration and refers the local DNS server to the appropriate PDNS. In either configuration, the PDNS is authoritative for *www.home.com*.

4. The local DNS server forwards the client's request for *www.home.com* to PDNS-0 in Zone 0.

Figure 5-3 Two-Zone Network Proximity Example



5. PDNS-0 determines the most proximate zone to send the client to using one of the following scenarios:

   a. PDNS-0 first searches its cache for a previously saved ordered zone index, a preferred order of zones closest to the client as determined by PDB-0 and based on information from probes and the PDB's peer mesh. If PDNS-0 finds the ordered zone index in its cache, it uses that data along with keepalive information and domain records (locally configured and learned through its peer mesh) to determine the most proximate zone to service the client.

b. If the ordered zone index is not cached, PDNS-0 sends to PDB-0 (using APP-UDP) a lookup request that contains the IP address of the client. PDB-0 calculates the preferred order of zones for the client and returns the ordered zone index to PDNS-0 immediately. PDNS-0 uses the zone order along with keepalive information and domain records to determine the most proximate zone to service the client.

c. If the ordered zone index is not cached and PDB-0 is not available, PDNS-0 uses its keepalive information, domain records, and a roundrobin method to select a service to handle the request.

6. If the PDNS determines that the best selection is a name server (NS) record, the PDNS begins a recursive query of the name server to determine an authoritative response. If the PDNS finds that the best selection is an address record (A-record), it formulates an authoritative response immediately. In this example, PDNS-0 decides that the best selection is an A-record (learned through the peer mesh with PDNS-1) for a data center in Zone 1.
7. The PDNS sends an authoritative response that contains the resolved IP address of *www.home.com* to the client's local DNS server.
8. The local DNS server notifies the client that sufficient domain name resolution information is available to establish a data connection to *www.home.com*.
9. Lastly, the client uses the local DNS server response information (IP address) to connect to a service in the most proximate zone and starts receiving content. In this example, the most proximate service is located in Proximity Zone 1 at IP address 150.45.6.8.

Note For details on advanced Network Proximity topics, including tiers and nested zones, refer to "Using Network Proximity Tiers" later in this chapter.

Network Proximity Configuration Quick Start

Table 5-1 and Table 5-2 provide a quick overview of the steps required to configure the PDB and PDNS, respectively. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the command options, refer to the sections following Table 5-1 and Table 5-2.

Proximity Database Configuration Quick Start

Table 5-1 provides an overview of the steps required to configure a PDB on a dedicated CSS 11150 with 256 MB of RAM. Follow these steps to configure PDB-0 located in Proximity Zone 0 in Figure 5-3. Use the CLI commands outlined in the table to configure basic PDB settings.

Table 5-1 PDB Configuration Quick Start

Task and Command Example
1. Enter config mode by typing **config**. `(config)#`
2. Enable the Application Peering Protocol-User Datagram Protocol (APP-UDP) to allow PDB-0 to communicate with PDNS-0. `(config)# app-udp`
3. Enable the Application Peering Protocol (APP) to allow PDB-0 to communicate with PDB-1. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol". `(config)# app`
4. Configure the **app session** with PDB-1, which is participating in the peer mesh with PDB-0. The IP address you enter is a local interface address on PDB-1. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol". `(config)# app session 200.16.2.3`
5. Configure PDB-0 in Proximity Zone 0. `(config)# proximity db 0 tier1 "pdb-usa"`

Proximity Domain Name Server Configuration Quick Start

Table 5-2 provides an overview of the steps required to configure PDNS-0 located in Proximity Zone 0 in Figure 5-3. Use the CLI commands outlined in the table to configure basic PDNS settings.

Table 5-2 PDNS Configuration Quick Start

Task and Command Example
1. Enter config mode by typing **config**. `(config)#`
2. Enable APP-UDP to allow PDNS-0 to communicate with PDB-0. `(config)# app-udp`
3. Enable APP to allow PDNS-0 to communicate with PDNS-1. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol". `(config)# app`
4. Configure PDNS-0. Specify the proximity zone and tier number, an optional text description, and the IP address associated with PDB-0. `(config)# dns-server zone 0 tier1 "usa" 172.16.2.2`
5. Configure the CSS to act as a DNS server. `(config)# dns-server`
6. Configure the app session with PDNS-1 that is participating in the mesh with PDNS-0. The IP address you enter is a local interface address on PDNS-1. Refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol". `(config)# app session 200.16.2.5`
7. Create A-records for domains in Zone 0. Specify the domain name mapped to the address record and the IP address bound to the domain name. Include an optional time to live (TTL) value, the number of records to return in a DNS response message, and the keepalive message type. `(config)# dns-record a www.home.com 123.45.6.7 0 single kal-icmp`

Table 5-2 PDNS Configuration Quick Start (continued)

Task and Command Example
8. Create NS-records for domains on other DNS servers within the proximity zone. Specify the domain name mapped to a domain IP address. Include an optional TTL value, the number of records to return in a DNS response message, and the keepalive message type. `(config)# dns-record ns www.work.com 123.45.6.8 0 single kal-icmp`
9. Optionally, create content rules for local A-records. In some configurations, there may not be any local content rules or services. For details on creating content rules, refer to the *Cisco Content Services Switch Basic Configuration Guide,* Chapter 3, Configuring Content Rules.

Configuring a Proximity Database

Caution If you configure your CSS as a Proximity Database, you cannot use the CSS for load balancing.

A PDB is a dedicated CSS 11150 with 256 MB of RAM and configured as a Proximity Database. Configure one PDB in each Network Proximity zone you want to create. Once configured, a PDB stores network topology information used to determine the relationship between proximity zones and a client that requests a service. The PDB populates its database through active probing of clients (local DNS servers) and sharing information with PDBs in other zones using an APP mesh. The PDB also responds to lookup requests from each PDNS configured in a zone using APP-UDP.

Note You must connect a PDB to a PDNS over a reliable link because of the requirements of the APP-UDP-based proximity lookup mechanism.

Configuring a PDB requires the following two tasks:

- Configuring APP-UDP and APP
- Enabling the Proximity Database

Optionally, you can configure additional PDB parameters as follows:

- Assigning Proximity Metrics
- Flushing Proximity Assignments
- Configuring Proximity Time to Live
- Storing the Proximity Database
- Retrieving the Proximity Database
- Refining Proximity Metrics
- Using Proximity Reprobe
- Clearing the Proximity Database
- Configuring the Proximity Probe Module

Configuring APP-UDP and APP

Network Proximity uses the Application Peering Protocol-User Datagram Protocol (APP-UDP) to exchange proximity information between a PDB and a PDNS, and between a PDNS and services. APP-UDP is a connectionless form of APP.

Note After you configure APP-UDP, you need to configure APP. APP enables a PDB to exchange zone index information with other PDBs in a peer mesh and a PDNS to exchange address records and keepalive information with other PDNSs in a peer mesh. For information on configuring APP, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol".

Configuring APP-UDP for proximity requires you to enable APP-UDP.

Optionally, you can configure additional APP-UDP parameters as follows:

- Securing APP-UDP Datagrams
- Specifying APP-UDP Options
- Removing an APP-UDP Options Record
- Specifying the APP-UDP Port
- Showing APP-UDP Configurations

Enabling APP-UDP

Use the **app-udp** command to configure APP-UDP datagram messaging on the PDB and all PDNSs in each zone. This command is available in global configuration mode.

The **app-udp** command supports the following options:

- **app-udp** - Enables APP-UDP datagram messaging
- **app-udp secure** - Specifies that all incoming APP-UDP datagrams must be encrypted

- **app-udp options** - Configures APP-UDP options used when communicating with a CSS peer
- **app-udp port** - Sets the UDP port that listens for APP-UDP datagrams

For example:

```
(config)# app-udp
```

To disable APP-UDP messaging, enter:

```
(config)# no app-udp
```

Securing APP-UDP Datagrams

Use the **app-udp secure** command to require that all incoming APP-UDP datagrams be encrypted. Encryption prevents unauthorized messages from entering the CSS. This command is used in conjunction with the **app-udp options** command that specifies secure messages that the CSS accepts.

Caution Using this command without the **app-udp options** command results in all incoming data being dropped.

The syntax for this global configuration mode command is:

app-udp secure

The following example illustrates the use of the **app-udp secure** command. In this example, this configuration allows only incoming traffic from IP address 200.16.2.3 encrypted with the password *mysecret*. The password is an unquoted text string with a maximum of 31 characters. There is no default.

For example:

```
(config)# app-udp
(config)# app-udp secure
(config)# app-udp options 200.16.2.3 encrypt-md5hash mysecret
```

To restore the default behavior of the CSS to accept all APP-UDP datagrams, enter:

```
(config)# no app-udp secure
```

Specifying APP-UDP Options

Use the **app-udp options** command to associate APP-UDP options with an IP address. The CSS applies the options to packets sent to the destination address or applies them when the CSS receives datagrams with a matching source IP address. You can configure the IP address to 0.0.0.0 to apply a set of security options to all inbound and outbound datagrams that are not more specifically configured. Using the IP address 0.0.0.0 allows you to set a global security configuration that may be applied to an arbitrary number of peers. This command is available in configuration mode.

The syntax for this global configuration mode command is:

app-udp options ***ip_address*** **encrypt-md5hash** *secret*

The **app-udp options** command contains optional fields that allow you to encrypt datagrams. This encryption method applies to datagrams sent and received over an IP address. Encryption options include:

- *ip_address* - The IP address associated with this group of options. Enter the address in dotted-decimal notation (for example, 200.16.2.3).
- **encrypt-md5hash** - The MD5 hashing method used for datagram encryption.
- *secret* - The string used in the encryption and decryption of the MD5 hashing method. Enter an unquoted text string with a maximum of 31 characters. There is no default.

The following example configures the IP address with the **encrypt-md5hash** global option. Datagrams sent to or received from 200.16.2.3 are encrypted with the password *mysecret*. All other datagrams received or transmitted, are subjected to the default encryption secret *anothersecret*.

For example:

```
(config)# app-udp
(config)# app-udp options 200.16.2.3 encrypt-md5hash mysecret
(config)# app-udp options 0.0.0.0 encrypt-md5hash anothersecret
```

Removing an APP-UDP Options Record

Use the **no app-udp options** command to remove an options record. This command includes an *ip_address* option to enable the CSS to associate an IP address with a group of options. Enter the address in dotted-decimal notation (for example, 200.16.2.3).

The syntax for this global configuration mode command is:

no app-udp options *ip_address*

For example:

```
(config)# no app-udp options 200.16.2.3
```

Specifying the APP-UDP Port

Use the **app-udp port** command to set the UDP port number that listens for APP-UDP datagrams. The **app-udp port** command includes the *port_number* variable, which specifies the UDP port number. Enter a value of 1025 to 65535. The default is 5002.

The syntax for this global configuration mode command is:

app-udp port *port_number*

For example:

```
(config)# app-udp port 2
```

To restore the UDP port number to its default value of 5002, enter:

```
(config)# no app-udp port
```

Note Now that you have configured APP-UDP, you must configure APP as described in Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol" to enable PDB and PDNS peer meshes.

Showing APP-UDP Configurations

Use the **show app-udp** command to display APP-UDP global statistical information and security configuration settings.

The options for the **show app-udp** command are:

- **show app-udp global** - Displays global statistical information about the operation of APP-UDP
- **show app-udp secure** - Displays the current security configuration settings for APP-UDP

For example, to display statistical information about the operation of APP-UDP, enter:

```
(config)# show app-udp global
```

Table 5-3 describes the fields in **show app-udp global** output.

Table 5-3 Field Descriptions for the show app-udp global Command

Field	Description
Transmit Frames	The number of frames transmitted through APP-UDP
Transmit Bytes	The number of bytes transmitted through APP-UDP
Transmit Errors	The number of frames dropped because of transmits resource errors
Receive Frames	The number of frames received through APP-UDP
Receive Bytes	The number of bytes received through APP-UDP
Receive Errors	The number of frames dropped because of security mismatches

For example, to display the current security configuration settings for APP-UDP, enter:

```
(config)# show app-udp secure
```

Table 5-4 describes the fields in the **show app-udp secure** output.

Table 5-4 Field Descriptions for the show app-up secure Command

Field	Description
Allow non-secure	The setting for whether or not encryption is a requirement for all inbound APP datagrams. Yes indicates that the CSS will accept all datagrams (default). No indicates that encryption is required.
IP Address	The IP address associated with this group of APP-UDP options.
Type	The encryption method. Currently, the only method is MD5 hashing.
Secret	The string used in encryption and decryption of the MD5 hashing method.

Enabling the Proximity Database

Note Before you enable the PDB, you must configure APP-UDP and APP. For details on configuring APP-UDP, refer to "Configuring APP-UDP and APP" earlier in this chapter. For details on configuring APP, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol".

Use the **proximity db** command to enable the PDB on a dedicated CSS 11150 with 256 MB of RAM.

Note Once you have enabled APP-UDP and APP, **proximity db** is the only command that is required to use the PDB. Other PDB commands are optional, but recommended, depending on your application. For details, refer to each command description in the next sections.

The syntax for this global configuration mode command is:

proximity db *zoneIndex* {**tier1**|**tier2**} {**"***description***"**}}

The **proximity db** command supports the following variables and options:

- *zone_index* - Numerical identifier of the proximity zone of a CSS. This number should match the zone index you configured on the PDNS. For tier1, enter an integer from 0 to 5. For tier2, enter an integer from 0 to 15. There is no default.
- **tier1** | **tier2** - Specification of the tier in which a CSS participates. The tier dictates the maximum number of proximity zones that may participate in the mesh. Enter **tier1** for a maximum of six proximity zones. Enter **tier2** for a maximum of 16 proximity zones. The default is **tier1**.
- **"***description***"** - Optional quoted text description of a CSS proximity zone. Enter a quoted text string with a maximum of 32 characters.

For example:

```
(config)# proximity db 1 tier1 "pdb-usa"
```

To disable the Proximity Database, enter:

```
(config)# no proximity db
```

Assigning Proximity Metrics

Use the **proximity assign** command to provide a local metric or to provide metrics (in milliseconds) for all proximity zones. The **proximity assign** command overrides the default metric determination processes. This command allows you to turn off probe traffic to Classless Inter-Domain Routing (CIDR) blocks.

When you use this command, Network Proximity does not perform active probing of the assigned block. Assigned information is shared with all PDBs in the PDB mesh. You can use this Network Proximity command only on a PDB.

Note The **proximity assign** command is not added to the running-config.

The syntax for this SuperUser configuration mode command is:

proximity assign *ip_address ip_prefix* [**"***local_metric***"**|**"***metric_list***"**]

The **proximity assign** command supports the following variables:

- *ip_address* - Assigns metric information to the IP address.
- *ip_prefix* - Assigns metric information to the IP prefix length. Enter the prefix as either:
  - A prefix length in CIDR bitcount notation (for example, /24).
  - A subnet mask in dotted-decimal notation (for example, 255.255.255.0).
- **"***local_metric***"** - A single quoted metric (in milliseconds) used to represent the proximity zone where the command is issued. Enter a value between 1 and 255.
- **"***metric_list***"** - A list of quoted metrics (in milliseconds), in ascending proximity zone order, for the zones where you want to apply the **proximity assign** command. Enter a value between 0 and 255. A value of zero indicates no assignment for a zone, and is only a placeholder in a list of assigned metrics.

For example, the following command uses the *local_metric* variable to assign a value of 200 to all client DNS addresses included in the range **172.23.5.7/24.**

```
# proximity assign 172.23.5.7/24 "200"
```

This command uses the *metric_list* variable to assign a value of 200 ms to proximity zone 0, does not configure zone 1 (specified by a value of zero), and assigns a value of 50 ms to zone 2.

```
# proximity assign 172.23.5.7/24 "200 0 50"
```

Flushing Proximity Assignments

Use the **proximity assign flush** command to remove existing proximity assignments configured with the **proximity assign** command. You can use this Network Proximity command only on a PDB.

Note Using the **proximity assign flush** command without additional syntax removes all proximity assignments.

The syntax for this SuperUser configuration mode command is:

proximity assign flush {*ip_address ip_prefix*}

The **proximity assign flush** command supports the following variables:

- *ip_address* - The IP address of previous proximity assignments you wish to remove.
- *ip_prefix* - IP prefix of previous proximity assignments you wish to remove. Enter the prefix as either:
  - A prefix length in CIDR bitcount notation (for example, /24).
  - A subnet mask in dotted-decimal notation (for example, 255.255.255.0).

For example:

```
# proximity assign flush 172.23.5.7/24
```

Configuring Proximity Time to Live

Use the **proximity ttl** command to set the TTL value, in minutes, for each PDB response. This value tells the PDNS how long to cache the PDB response. You can use this Network Proximity command only on a PDB.

The syntax for this global configuration mode command is:

proximity ttl assigned | probe *minutes*

The options for this global configuration mode command are:

- **proximity ttl assigned** *minutes* - Sets the TTL value for client addresses that are assigned at the PDB. Enter a value from 0 to 255. The default is 60.
- **proximity ttl probe** *minutes* - Sets the TTL value for client addresses that are being probed by the PDB. Enter a value from 0 to 255. The default is 0, which disables the caching of responses.

For example:

```
(config)# proximity ttl assigned 25
```

To reset the TTL value to its default, enter:

```
(config)# no proximity ttl probe
```

Note A TTL value of 255 never ages out the entries.

Storing the Proximity Database

Use the **proximity commit** command to write a portion or all of the proximity database to a file in the log directory on the CSS disk or to a file on an FTP server. This command is useful for exporting the database so that you can view, modify, or recover information in the PDB. The database output contains metrics for all proximity zones, the current advertisement state, and hit counts.

To retrieve the database log file, use the **proximity retrieve** command. You can use this Network Proximity command only on a PDB.

By default, when you enter this command without any of its options, it writes the entire database to an XML-formatted file named proximity.db in the log directory on the CSS disk. You can optionally specify that the database be encoded using compact binary encoding. You can also specify that the database be written to a file on an FTP server.

The syntax for this SuperUser command is:

proximity commit {*ip_address ip_prefix*|**entire-db** {**ftp** *ftp_record ftp_filename* {**bin**}|**log** *filename* {**bin**}}}

The **proximity commit** command supports the following variables and options:

- *ip_address* - The starting IP address in the database that you want to write to the CSS disk or FTP server. Enter the IP address in dotted-decimal notation (for example, 175.23.5.7).
- *ip_prefix* - The IP prefix length of the database that you want to write to the CSS disk or FTP server. Enter the prefix as either:
  - A prefix length in CIDR bitcount notation (for example, /24).
  - A subnet mask in dotted-decimal notation (for example, 255.255.255.0).
- **entire-db** - An optional keyword to commit the entire PDB. By default, the entire database is written to disk.
- **ftp** - An optional keyword to write a specified file to an FTP server.
- *ftp_record* - The name for the FTP record file. Enter an unquoted text string with no spaces and a maximum of 16 characters. You must create an FTP record using the global config **ftp-record** command. For information on configuring an FTP record, refer to the *Cisco Content Services Administration Guide,* Chapter 1, Logging in and Getting Started.

- *ftp_filename* - The filename to use when copying the database to an FTP server.
- **log** - An optional keyword to write a specified file to the log directory on the CSS disk.
- *filename* - The filename to use when storing the PDB in the log directory on the CSS disk. Enter a filename with a maximum of 32 characters. The default filename is *proximity.db*.
- **bin** - Specifies binary output for the PDB. A binary encoded database requires approximately 32 bytes per entry.

Note An XML database occupies approximately three times the space a binary-encoded database occupies. However, a binary encoded database cannot be viewed.

For example:

```
# proximity commit 172.23.5.7/24 xml
```

Retrieving the Proximity Database

Use the **proximity retrieve** command to load a PDB from disk or an FTP server. Proximity metrics loaded from the database file replace any overlapping entries existing in the database and supplement any non-overlapping database entries. You can use this Network Proximity command only on a PDB.

Note If you enter the **proximity retrieve** command without any of its options, the CSS loads the file *proximity.db* from disk by default.

The **proximity retrieve** command distinguishes between XML encoded and binary database formats automatically.

The syntax for this SuperUser command is:

proximity retrieve {**ftp** *ftp_recordname* **ftp** *ftp_filename*|**log** *filename*}

The **proximity retrieve** command supports the following variables:

- **ftp -** The optional keyword to retrieve a specified file from an FTP server.
- *ftp_recordname* - The name of an existing FTP record for an FTP server. The FTP record contains the FTP server IP address, username, and password. To create an FTP record, use the **(config) ftp-record** command.
- *ftp_filename* - The PDB filename located on the FTP server.
- **log** - The optional keyword to retrieve a specified file (other than the proximity.db file) from the log directory on the CSS disk.
- *filename* - The PDB filename located in the log directory on the CSS disk.

For example:

```
# proximity retrieve ftp proxconfig proxconfignew
```

Refining Proximity Metrics

Use the **proximity refine** and the **proximity refine once** commands to initiate the continuous or single refinement, respectively, of metric entries in the PDB. Refinement updates the metric entries for all clients in the database to reflect conditions that exist at a particular point in time. You can use these Network Proximity commands only on a PDB.

When you issue the **proximity refine** command, the PDB probes all existing clients in the database periodically based on the size of the database and the database hit counts for the clients. The PDB organizes clients into three groups by hit count: N1, N2, and N3. The PDB probes N1 more frequently than N2, and N2 more frequently than N3. The percentage of time spent probing N1, N2, and N3 is approximately 45%, 35%, and 20%, respectively.

When you issue the **proximity refine once** command, the PDB probes all existing clients in the database only once.

The syntax for these SuperUser configuration mode commands are:

proximity refine

proximity refine once

To stop a refinement in progress, enter:

```
# no proximity refine
```

Using Proximity Reprobe

Use the **proximity reprobe** command to send additional probes to existing IP addresses in the proximity database once. You can use this Network Proximity command only on a PDB. You can use the **proximity reprobe** command with the **proximity refine** commands.

Note IP addresses configured with the **proximity assign** command are not eligible for reprobing.

The syntax for this SuperUser configuration mode command is:

proximity reprobe *ip_address* {*ip_prefix*}

The **proximity reprobe** command supports the following variables:

- *ip_address* - The IP address to probe.
- *ip_prefix* - The optional IP prefix to associate with the IP address that performs probing for a source block of addresses. Enter the prefix as either:
  - A prefix length in CIDR bitcount notation (for example, /24).
  - A subnet mask in dotted-decimal notation (for example, 255.255.255.0).

For example:

```
# proximity reprobe 172.23.5.7/24
```

Clearing the Proximity Database

Use the **proximity clear** command to remove entries from the proximity database.

Caution Be sure you want to permanently delete entries in the PDB before you use this command. Using the **proximity clear** command without optional variables permanently removes all entries in the proximity database.

The syntax for this SuperUser command is:

proximity clear *ip_address ip_prefix*

The **proximity clear** command supports the following variables:

- *ip_address* - The IP address for the entries you want to remove. Enter the address in dotted-decimal format (for example, 172.23.5.7).
- *ip_prefix* - The IP prefix length used in conjunction with the IP address. Enter the prefix as either:
  - A prefix length in CIDR bitcount notation (for example, /24).
  - A subnet mask in dotted-decimal notation (for example, 255.255.255.0)

Configuring the Proximity Probe Module

The Proximity Probe Module is responsible for sending ICMP and TCP probes to clients based on PDNS lookup requests to the PDB and refinement settings. Refer to the following sections to configure the Proximity Probe Module:

- Configuring the Proximity Probe Module Method
- Specifying the Proximity Probe Module Samples
- Configuring the Proximity Probe Module Metric Weighting
- Configuring the Proximity Probe Module Interval
- Specifying Proximity Probe Module TCP-ports

Configuring the Proximity Probe Module Method

Use the **proximity probe rtt method** command to configure the primary and secondary methods used for proximity metric discovery. You can use this Network Proximity command only on a PDB.

The syntax for this global configuration mode command is:

proximity probe rtt method [**icmp** {**tcp**}|**tcp** {**icmp**}]

The **proximity probe rtt method** command supports the following options:

- **icmp** - Use ICMP Echo requests as the primary method. The default is **icmp**.
- **tcp** - Use a TCP SYN/SYN ACK approach to the configured TCP ports as the primary RTT discovery method.

For example:

```
(config)# proximity rtt method icmp
```

Specifying the Proximity Probe Module Samples

Use the **proximity probe rtt samples** command to configure the number of ICMP requests to send for each client probe. You can use this Network Proximity command only on a PDB.

Note Because only one TCP SYN request is sent, you cannot configure this command for TCP probes.

The syntax for this global configuration mode command is:

proximity probe rtt samples *number*

The *number* variable specifies the default number of ICMP echo requests used for averaging during an initial probe. Enter a number from 1 to 30. The default is 2.

For example:

```
(config)# proximity probe rtt samples 5
```

To reset the number of ICMP echo requests to its default value of 2, enter:

```
(config)# no proximity probe rtt samples
```

Configuring the Proximity Probe Module Metric Weighting

Use the **proximity probe rtt metric-weighting** command to configure the percentage of the previously stored metric value in the database that is used to determine the new metric value. This command allows the PDB to smooth network metric variation caused by network congestion, flash crowds, and so on. You can use this Network Proximity command only on a PDB.

The syntax for this global configuration mode command is:

proximity probe rtt metric-weighting *number*

The *number* variable specifies the percentage of the previous metric value used. Enter a number from 0 to 99. The default is 0.

For example:

```
(config)# proximity probe rtt metric-weighting 10
```

For this example, suppose the previously stored metric value for a client's local DNS server is 40 and the current metric value is 50. If you issue the command above, the PDB adds 10% of the previous metric value (0.10 x 40) to 90% of the current metric value (0.90 x 50) to determine the new metric value. So, the new metric value would be 49. A *number* value of 50 causes the PDB to average the previous and current metric values.

To reset this command to its default value of 0, enter:

```
(config)# no proximity probe rtt metric-weighting
```

Configuring the Proximity Probe Module Interval

Use the **proximity probe rtt interval** command to configure the delay in seconds between ICMP samples. You can use this Network Proximity command only on a PDB.

The syntax for this global configuration mode command is:

proximity probe rtt interval *seconds*

The *seconds* variable identifies the length of time (in seconds) to wait between ICMP samples. Use a range between 1 to 10. The default is 1.

For example:

```
(config)# proximity probe rtt interval 5
```

To reset the delay between samples to its default value of 1 second, enter:

```
(config)# no proximity probe rtt interval
```

Specifying Proximity Probe Module TCP-ports

Use the **proximity probe rtt tcp-ports** command to configure the probe ports for TCP proximity metric discovery. You can use this Network Proximity command only on a PDB.

The syntax for this global configuration mode command is:

proximity probe rtt tcp-ports *port_number1* {*port_number2* {*port_number3* {*port_number4*}}}

Define the port number to be tried, in order of precedence. Enter a number from 1 to 65535 to enable a port. The defaults for the various port numbers include:

- *port_number1* is 23, Telnet port
- *port_number2* is 21, FTP port
- *port_number3* is 80, HTTP port
- *port_number4* is 0, this port is not tried

Note Ports that you do not specify default to 0.

To reset the probe ports to their default values, enter:

```
(config)# no proximity probe rtt tcp-ports
```

Using Network Proximity Tiers

The following sections describe the advanced Network Proximity concept of *tiers*. Network Proximity uses tiers to further expand the proximity architecture by allowing you to create more distinct network zones and subzones.

Proximity Tiers

Sharing information among multiple PDBs may result in the management of a very large data set. As you add more proximity zones to the network, Network Proximity scales to provide more distinct network zones, allowing zones or subzones to exist within other zones. Network Proximity treats these zones as:

- Level 1 zones (first level)
- Level 2 zones (nested levels)

Note You can configure six Level 1 proximity zones and 16 Level 2 proximity zones. A Level 1 tier supports up to 2 million unique local DNS server addresses. A Level 2 tier supports slightly less than one million unique local DNS server addresses.

In a tiered Network Proximity model, the owner of the name server record is a nested PDNS that is communicating with a nested PDB located within the Level 2 proximity subzone.

Tiered Network Proximity Example

In Figure 5-4, a two-tiered configuration example illustrates how you can use tiers to group more specific network proximity zones. The proximity zone that encompasses all network devices within the United States is broken down further to include an additional tier that comprises the more specific geographical proximity zones, East Coast and West Coast.

By adding a tier to this configuration, the capacity of Network Proximity is extended by creating two subzones (Zones 0.1 and 0.2) that include additional PDBs, PDNSs, and data centers. In this way, you can scale Network Proximity to meet your users' needs with increased proximity specificity and thereby increase network performance.

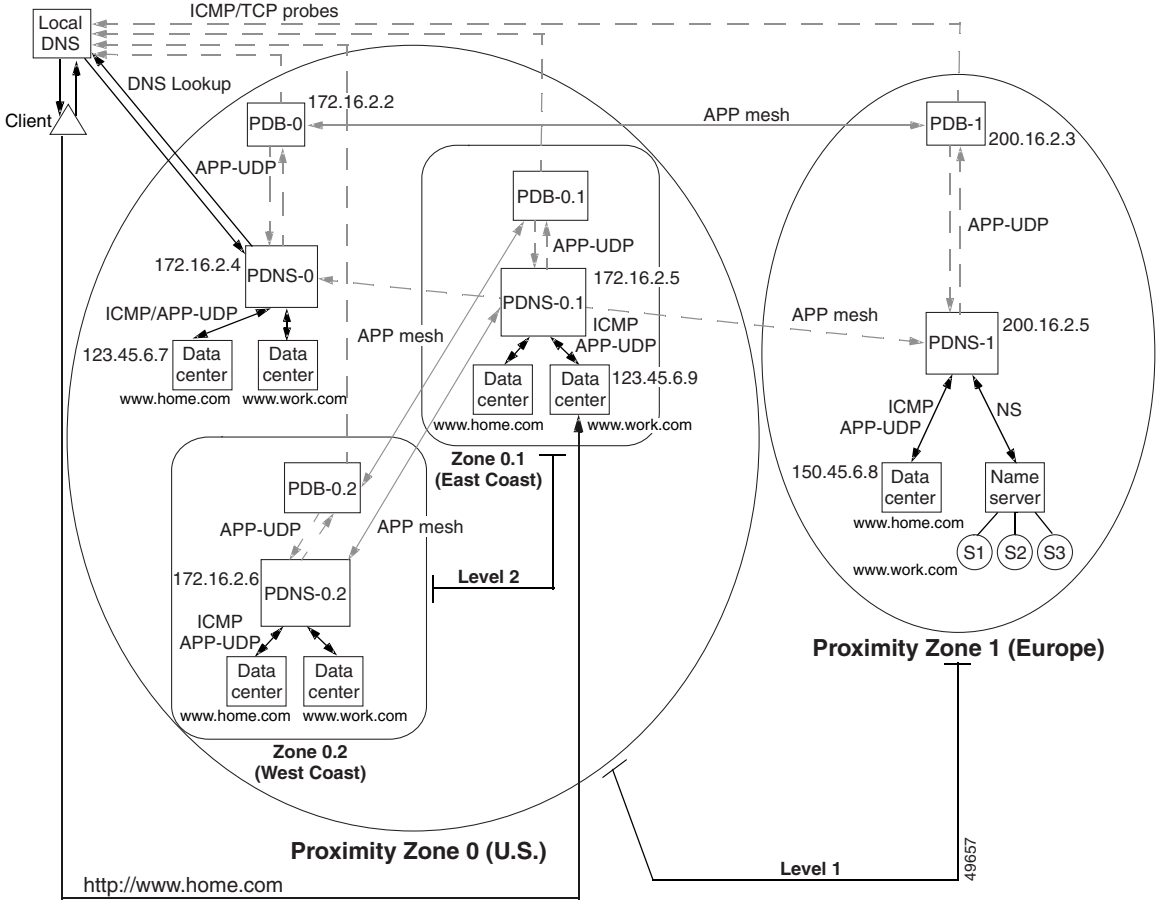
The following steps describe how Network Proximity determines the most proximate service for a client requesting the domain *www.work.com*. Refer to Figure 5-4.

1. The client performs an HTTP request for the domain name *www.work.com*.
2. The client's DNS server performs iterative DNS requests to the root server and to the .com server to start resolving the domain name into an IP address. (This step is not shown in Figure 5-4.)
3. The .com server, which is an authoritative DNS for *work.com*, has the IP addresses of PDNS-0 and PDNS-1 in its configuration. The Level 1 PDNSs are authoritative DNSs for *www.work.com*. In this example, the .com server refers the client's DNS server to PDNS-0 in Zone 0. (Typically, the .com server uses a roundrobin or other load-balancing method to refer local DNS servers to a PDNS. This step is not shown in Figure 5-4.)

Note Your configuration may include an enterprise DNS server that is positioned between the .com server and the PDNSs. The enterprise DNS server would be an authoritative DNS server for *work.com*. In this case, the enterprise DNS server contains the IP addresses of the PDNSs in its configuration and refers the local DNS server to the appropriate PDNS. In either configuration, the PDNS is authoritative for *www.work.com*.

4. The local DNS server forwards the client's request for *www.work.com* to PDNS-0 in Zone 0.
5. PDNS-0 determines the most proximate zone to send the client to using one of the following scenarios:
   a. PDNS-0 first searches its cache for a previously saved ordered zone index, a preferred order of zones closest to the client as determined by PDB-0 and based on information from probes and the PDB's peer mesh. If PDNS-0 finds the ordered zone index in its cache, it uses that data along with keepalive information and domain records (locally configured and learned through its peer mesh) to determine the most proximate zone to service the client.

Figure 5-4 Tiered Network Proximity Configuration



b. If the ordered zone index is not cached, PDNS-0 sends to PDB-0 (using APP-UDP) a lookup request that contains the IP address of the client. PDB-0 calculates the preferred order of zones for the client and returns the ordered zone index to PDNS-0 immediately. PDNS-0 uses the zone order along with keepalive information and domain records to determine the most proximate zone to service the client.

   c. If the ordered zone index is not cached and PDB-0 is not available, PDNS-0 uses its keepalive information, domain records, and a roundrobin method to select a service to handle the request.

6. If the PDNS determines that the best selection is a name server (NS) record, the PDNS begins a recursive query of the name server to determine an authoritative response. If the PDNS finds that the best selection is an address record (A-record), it formulates an authoritative response immediately. In this example, PDNS-0 decides that the best selection is a name server (NS) record for PDNS-0.1 in Zone 0.1.
7. PDNS-0.1 uses the same logic outlined in steps 5 and 6 above to determine the most proximate service for the client. In this example, PDNS-0.1 decides that the best selection is an address record (A-record) for one of its attached data centers. PDNS-0.1 then makes an authoritative response to PDNS-0 with the A-record for the data center that contains *www.work.com*.
8. PDNS-0 sends an authoritative response that contains the resolved IP address of *www.work.com* to the client's local DNS server.
9. The client's DNS server notifies the client that sufficient domain name resolution information is available to establish a data connection to *www.work.com*.
10. Lastly, the client uses the resolved IP address to connect to a service in the most proximate zone and starts receiving content. In this example, the most proximate service is located in Proximity Zone 0.1 (East Coast) at IP address 123.45.6.9.

Displaying Proximity Database Configurations

The CSS provides a comprehensive set of Network Proximity **show** commands that display information about the PDB. Use the **show proximity** command to display PDB configuration or session information. Refer to the following sections for information on using Proximity Database show commands:

Showing the Proximity Database

Use the **show proximity** command to display an activity summary of the proximity database. This command functions only on a PDB.

For example:

```
# show proximity
```

Table 5-5 describes the fields in the **show proximity** output.

Table 5-5 Field Descriptions for the show proximity Command

Field	Description
Lookups	The total number of resolved proximity requests
Lookup Rate	The number of resolved proximity requests per second
Probe TTL	The configured time-to-live value for client addresses that are probed

Table 5-5 Field Descriptions for the show proximity Command (continued)

Field	Description
Assigned TTL	The configured time-to-live value for client addresses that are assigned to the Proximity Database
Assigned Blocks	Blocks in the PDB that are assigned
Probed Blocks	Blocks in the PDB that are probed
Total Blocks	Total number of blocks in the PDB
Last Retrieve	The last time that a proximity retrieve was executed
Last Commit	The last time that a proximity commit was executed
DB Utilization	Percentage of the PDB used
Refinement	Whether or not refinement is activated
Total Peers	The total number of peers in the PDB mesh

Note All database values are cleared when you reboot the CSS or you issue the **no proximity db** command.

Showing Proximity Metrics

Use the **show proximity metric** command to display metrics (in milliseconds) associated with a client's local DNS server IP address. This command is available on a PDNS and a PDB.

The syntax and options for this global configuration mode command are:

show proximity metric *ip_address* {*ip_prefix* {**aggregate**}}

- *ip_address* - The IP address of a client's local DNS server for metric display. Enter the address in dotted-decimal notation (for example 172.23.5.7).
- *ip_prefix* - This optional parameter is used to map an IP prefix to an IP address allowing you to view metrics over a range of IP addresses. Enter the prefix as either:
  - A prefix length in CIDR bitcount notation (for example, /24).
  - A subnet mask in dotted-decimal notation (for example, 255.255.255.0).

- **aggregate** - This optional parameter allows you to view aggregated metrics that are available in both /16 and /8 subnet masks.

Note Probed metrics are statistically aggregated at the /8 and /16 prefix levels.

For example, to view the proximity metrics associated with the client IP address of 172.23.5.7 and an IP prefix of 24, enter:

```
(config)# show proximity metric 172.23.5.7/24
```

In the PDB, the RTT metrics are sorted by proximity zone. In the PDNS, the metrics are sorted by RTT. An asterisk next to a zone indicates the zone where the command was issued.

Note The maximum value of an RTT metric is 3968 ms. A value of 4095 ms indicates that a client's local name server was unreachable or had an RTT value of more than 4 seconds.

Table 5-6 describes fields in the **show proximity metric** output.

Table 5-6 Field Descriptions for the show proximity metric Command

Field	Description
Index	The zone index number associated with the PDNS zone. An asterisk (*) indicates the local zone where you issued this command.
Description	A logical name or description to the zone.
Metric	Round-Trip Time (RTT) between the PDB and a Referral-DNS as the proximity metric for load balancing decisions.

Showing Proximity Statistics

Use the **show proximity statistics** command to display statistics associated with client IP addresses. This Network Proximity command is only available on the PDB.

The syntax and options for this global configuration mode command are:

show proximity statistics *ip_address* {*ip_prefix* {**aggregate**}}

- *ip_address* - The IP address for statistics display. Enter the address in dotted-decimal notation (for example 172.23.5.7).
- *ip_prefix* - This optional parameter is used to map an IP prefix to an IP address. This allows you to view metrics over a range of IP addresses, indicated by the prefix. Enter the prefix as either:
  - A prefix length in CIDR bitcount notation (for example, /24).
  - A subnet mask in dotted-decimal notation (for example, 255.255.255.0).
- **aggregate** - This optional parameter allows you to view aggregated statistics that are available in both /16 and /8 subnet masks.

For example, to view the proximity statistics associated with the client IP address of 10.1.0.0 and an IP prefix of 16, enter:

```
(config)# show proximity statistics 10.1.0.0/16
```

Table 5-7 describes fields in the **show proximity statistics** output.

Table 5-7 Field Descriptions for the show proximity statistics Command

Field	Description
IP/Prefix	The IP address and prefix associated with the statistics display.
Lookup Count	The number of resolved proximity requests per second.

Showing Proximity Refinement

Use the **show proximity refine** command to display information pertaining to entries being refined in the PDB. This Network Proximity command is only available on a PDB. For an explanation of the N1, N2, and N3 groups mentioned below, see "Refining Proximity Metrics" earlier in this chapter.

For example:

```
(config)# show proximity refine
```

Table 5-8 describes fields in the **show proximity refine** output.

Table 5-8 Field Descriptions for the show proximity refine Command

Field	Description
N1 - N3 Count	The number of entries in each N class
N1 - N3 Percent	Of all entries, the percentage of entries in the N class
N1 - N3 Rate	The number of probes per second
N1 - N3 Probed	The total number of probes since the **proximity refine** command was invoked
N1 - N3 Cycle Time	The amount of time to cycle through the N count
Aggregate Count	The total count for N1 through N3
Aggregate Probed	The probed total for N1 through N3
Aggregate Rate	The rate total for N1 through N3

Showing Proximity Assignments

Use the **show proximity assign** command to display the user-assigned metric values (in milliseconds) of all proximity zones or for a configured IP address range.

The syntax and variables for this global configuration mode command are:

show proximity assign {*ip_address ip_prefix*}

- *ip_address* - The optional IP address to display metrics over a range of IP addresses. Enter the IP address in dotted-decimal format (for example, 172.23.5.7).
- *ip_prefix* - The optional IP prefix to associate with the IP address that performs probing for a source block of addresses. Enter the prefix as either:
  - A prefix length in CIDR bitcount notation (for example, /24).
  - A subnet mask in dotted-decimal notation (for example, 255.255.255.0).

For example, to view the metric assignments for all IP addresses within the range of 200.16.0.0 to 200.16.255.255, enter:

```
(config)# show proximity assign 172.23.5.7/16
```

Table 5-9 describes the fields in the **show proximity assign** output.

Table 5-9 Field Descriptions for the show proximity assign Command

Field	Description
IP/Prefix	The IP address to search for in the cache and the IP prefix associated with the IP address for cache searching
Hits	The total number of hits
Zone Metrics	The list of metrics in ascending order to represent all zones

Showing Proximity Zones

Use the **show proximity zone** command to view the state information for each proximity zone, excluding the local proximity zone. This command is similar to the **show zone** command for the PDNS; however, the **show proximity zone** command provides information from the perspective of the PDB. This Network Proximity command is available only on a PDB.

The syntax for this global configuration mode command is:

show proximity zone {*number*}

Use the *number* variable to display the state information for a specific proximity zone. Enter a zone number from 0 to 15.

For example, to display the state information for proximity zone 1, enter:

```
(config)# show proximity zone 1
```

Table 5-10 describes the fields in the **show proximity zone** output.

Table 5-10 Field Descriptions for the show proximity zone Command

Field	Description
Index	The local index number associated with the PDNS zone. The * indicates the local zone where you issued this command.
Description	A text description of the zone that associates a logical name description to the zone.
IP Address	The IP address used for PDB communication with the zone peer.
State	The state of the PDB connection with the peer, which includes: • **Initializing** - The PDB state connection is initializing • **Sync** - The PDB state connection is synchronizing with the peer • **Normal** - The PDB state connection is normal • **Illegal** - The PDB state is an illegal connection
UpTime	Elapsed time since the **proximity db** command was executed locally, or since the peer entered the PDB mesh.

Showing Proximity Zone Statistics

The **show proximity zone statistics** command displays information about the APP peer mesh blocks sent and received for a peer for all proximity zones.

The syntax for this global configuration mode command is:

show proximity zone statistics {*number*}

Use the *number* variable to display statistics for a specific proximity zone. Enter a zone number from 0 to 15.

For example, to display the peer block information for zone 1, enter:

```
(config)# show proximity zone statistics 1
```

Table 5-11 describes the fields in the **show proximity zone statistics** display.

Table 5-11 Show Proximity Zone Statistics Display Fields

Field	Description
Index	The local index number associated with the proximity zone.
Description	A text description of the proximity zone that associates a logical name description with the proximity zone as entered with the **proximity db** command.
Sent	The number of blocks sent to the peer.
Received	The number of blocks received from the peer.
Last Update	The last time information was exchanged between the local PDB and the peer in either direction.

Showing Proximity Probe Module Statistics

Use the **show proximity probe rtt statistics** command to view the Round Trip Time (RTT) probe module statistics.

The syntax for this global configuration mode command is:

show proximity probe rtt statistics

For example:

```
(config)# show proximity probe rtt statistics
```

Table 5-12 describes the fields in the **show proximity probe rtt statistics** output.

Table 5-12 Field Descriptions for the show proximity probe rtt statistics Command

Field	Description
Total Client Probes	The total number of times that the PDB has probed a client to measure the RTT value. This may be more than the total number of unique clients and may be less than the actual number of ICMP or TCP requests.
Average Probes/minute	The cumulative average number of probes per minute since the PDB was last reset.

Table 5-12 Field Descriptions for the show proximity probe rtt statistics Command (continued)

Field	Description
ICMP requests sent	Specifies the number of ICMP probe requests used to calculate the RTT value.
ICMP responses	The total number of ICMP responses that the PDB has received. Valid ICMP responses are used to measure the RTT.
ICMP failures	The total number of ICMP requests that were successfully sent but did not receive a reply. The ICMP requests that do not receive a response are not used to measure the RTT value.
Average ICMP requests/minute	Specifies the time delay in seconds between consecutive ICMP requests to an individual client.
ICMP send failures	The total number of ICMP requests that the PDB tried to send but failed internally due to a missing route or other problem.
TCP requests	The total number of TCP requests that have been successfully sent from the PDB in order to measure the RTT value.
TCP responses	The total number of TCP responses that the PDB has received. Valid TCP responses are used to measure the RTT value.
TCP failures	The total number of failed TCP requests destined for the port on the client's local name server.
Average TCP requests/minute	The cumulative average of TCP requests per minute that were successfully sent during the time period since the PDB was last reset.
TCP send failures	The total number of TCP requests that the PDB tried to send but failed internally due to a missing route or other problem.

Configuring a Proximity Domain Name Server

The Proximity Domain Name Server (PDNS) is an authoritative DNS server that uses information from the Proximity Database (PDB) to resolve DNS requests based on an ordered zone index. As an authoritative DNS server, the PDNS uses domain records to map a given domain to an IP address or to a lower-level DNS server. You can configure a total of 1024 unique domain names for all PDNSs in a proximity mesh per proximity level. The same domain names can appear in all zones and on multiple PDNSs within a zone.

Note You must connect a PDNS to a PDB over a reliable link because of the requirements of the APP-UDP-based proximity lookup mechanism.

Configuring a PDNS involves the following required tasks:

- Configuring APP-UDP and APP
- Enabling the Proximity Domain Name Server
- Configuring Domain Records

Optionally, you can perform the following PDNS-related tasks:

- Disabling the Proximity Domain Name Server
- Clearing the DNS Server Statistics
- Enabling the Proximity Lookup Cache
- Removing Entries from the Proximity Lookup Cache

Configuring APP-UDP and APP

Network Proximity uses the Application Peering Protocol-User Datagram Protocol (APP-UDP) to exchange proximity information between a PDB and a PDNS, and between a PDNS and services. APP-UDP is a connectionless form of the Application Peering Protocol (APP). For details, see "Configuring APP-UDP and APP" earlier in this chapter.

Note In addition to configuring APP-UDP, you need to configure APP. APP enables a PDB and a PDNS to exchange proximity information with their peers. For information on configuring APP, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol".

Enabling the Proximity Domain Name Server

Note Before you enable the PDNS, you must configure APP-UDP and APP. For details on configuring APP-UDP, see "Configuring APP-UDP and APP" earlier in this chapter. For details on configuring APP, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol".

Use the **dns-server zone** and **dns-server** commands to enable the PDNS. The syntax for this global configuration mode command is:

dns-server zone *zone_index* {**tier1**|**tier2** {**"***description***"** {*ip_address* {**round-robin**|**prefer-local**}}}}

The **dns-server zone** command supports the following variables and options:

- *zone_index* - Numerical identifier of the proximity zone of the CSS. This number should match the zone index configured on the PDB. Enter an integer from 0 to 15. Valid entries are 0 to 5 for tier 1 and 0 to 15 for tier 2. There is no default.
- **tier1** | **tier2** - Specify the tier in which the CSS participates. The tier dictates the maximum number of proximity zones that may participate in the mesh. If you choose **tier1**, a maximum of six proximity zones may participate in the mesh. If you choose **tier2**, a maximum of 16 proximity zones may participate in the mesh. The default is **tier1.**
- *description* - Optional quoted text description of the CSS proximity zone. Enter a quoted text string with a maximum of 20 characters.
- *ip_address* - The IP address of the PDB. Enter the address in dotted-decimal notation (for example: 172.16.2.2). If you choose the zone capabilities (peer mesh) of a PDNS in a non-proximity environment, this variable is optional.

- **roundrobin|preferlocal -** The optional load-balancing method that the DNS server uses to select returned records when a Proximity Database is not configured or is unavailable.
  - **roundrobin** - The server cycles among records available at the different zones. This is the default method.
  - **preferlocal** - The server returns a record from the local zone whenever possible, using round-robin when it is not possible.

For example:

```
(config)# dns-server zone 1 tier1 "pdns-usa" 172.16.2.2
```

Configuring Domain Records

Use the **dns-record** command and its options to create a domain record on the PDNS. The PDNS uses two types of domain records to map a domain name to an IP address or to another DNS server:

- **A-record** - A domain record mapped to an IP address
- **NS-record** - A domain record mapped to a DNS server IP address

For details on configuring the **dns-record** command, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Domain Records".

Disabling the Proximity Domain Name Server

Use the **no dns-server zone** command to disable the PDNS Proximity functions by removing the **dns-server zone** command configuration.

Disabling the PDNS:

- Prevents it from submitting proximity metric lookup requests to the PDB
- Stops the peer mesh communications and record keepalive processing

After issuing the **no dns-server zone** command, you can still use the PDNS as a DNS server.

For example:

```
(config)# no dns-server zone
```

Note Before you can issue this command, you must issue the **no dns-server** command. The **no dns-server** command also disables the Network Proximity functions and DNS server functions on the PDNS. Because this command does not delete the **dns-server zone** command configuration, you may want to use the **no dns-server** command to disable a PDNS temporarily.

Clearing the DNS Server Statistics

Use the **dns-server zero** command in global configuration mode to set the DNS server request and response statistics displayed by the **show dns-server** command to zero.

For example:

```
(config)# dns-server zero
```

Enabling the Proximity Lookup Cache

Use the **proximity cache-size** command to modify the size of the proximity lookup cache. The PDNS uses the proximity lookup cache to store PDB responses. The proximity lookup cache allows the PDNS to resolve proximity decisions faster by allowing a local lookup.

Note The proximity cache is limited to 48,000 entries.

The syntax for this global configuration mode command is:

proximity cache-size *cache_size*

The **proximity cache-size** command includes a *cache size* variable that specifies the size of the proximity lookup cache. Enter a value between 0 and 48,000. Entering a value of 0 disables the proximity lookup cache. Modifying the cache size results in flushing the existing entries. The default cache size is 16,000.

For example:

```
(config)# proximity cache-size 30000
```

To restore the default cache size (16,000 entries), enter:

```
(config)# no proximity cache-size
```

Removing Entries from the Proximity Lookup Cache

Use the **proximity cache-remove** command to remove entries from the proximity lookup cache. The prefix length parameter allows you to remove multiple entries in a single operation. This Network Proximity command can be used only on a PDNS.

The syntax for this SuperUser configuration mode command is:

proximity cache-remove *ip_address ip_prefix*|**all**

Note If you specify **all,** you cannot specify an *ip_address* or *ip_prefix* value.

The **proximity cache-remove** command supports the following variables and option:

- *ip_address* - The IP address to remove from the proximity cache.
- *ip_prefix* - The IP prefix length associated with the IP address removed from the proximity cache. Enter the prefix as either:
  - A prefix length in CIDR bitcount notation (for example, /24).
  - A subnet mask in dotted-decimal notation (for example, 255.255.255.0).
- **all** - This keyword removes all entries in the proximity lookup cache.

For example:

```
# proximity cache-remove 150.45.6.10 /24
```

Displaying Proximity Domain Name Server Configurations

The CSS CLI provides a comprehensive set of Network Proximity **show** commands that display proximity configurations. Refer to the following sections for information on using PDNS **show** commands:

- Displaying the Proximity Cache
- Displaying DNS-Record Statistics
- Displaying DNS Server Zones
- Displaying DNS-Record Keepalives
- Displaying DNS-Record Proximity
- Displaying DNS Server Information, page 5-54

Displaying the Proximity Cache

Use the **show proximity cache** command to display the current state of the proximity lookup cache. This display provides information about the current cache configuration, entries present, number of hits, and so on. This command is available only on the PDNS.

The syntax for this global configuration command is:

show proximity cache {**all**|*ip_address ip_prefix*}

The **show proximity cach** command supports the following variables and option:

- **all** - Display all addresses in the cache.
- *ip_address* - The IP address to search for in the cache.
- *ip_prefix* - The IP prefix to associate with the IP address for cache searching. Enter the prefix as either:
  - A prefix length in CIDR bitcount notation (for example, /24).
  - A subnet mask in dotted-decimal notation (for example, 255.255.255.0).

For example:

```
(config)# show proximity cache
```

Table 5-13 describes the fields in the show proximity cache screen.

Table 5-13 Show Proximity Cache Display Fields

Field	Description
Maximum Entries	The maximum number of entries the cache supports
Used Entries	The number of entries used by the cache
Free Entries	The number of free entries in the caches
Percent Available	The available percentage of unused cache
Hits	The number of cache lookup hits
Misses	The number of cache lookup misses
Percent Hits	The percentage of cache lookup hits

To display all information pertaining to the proximity cache, enter:

```
(config)# show proximity cache all
```

Table 5-14 describes the fields in the **show proximity cache all** screen.

Table 5-14 Show Proximity Cache All Display Fields

Field	Description
IP/Prefix	The IP address in the cache and the IP prefix associated with the IP address.
Hits	The total number of hits the cache received.
Descending Zone Proximity	Indices of desirable zones ordered by proximity to the client.
TTL	The TTL value associated with the cache entry. The “N” in the second row tells the PDNS to never age out the entries in the cache and is enabled by a TTL value of 255.

Displaying DNS-Record Statistics

Use the **show dns-record statistics** command to display statistics associated with the domain records configured locally and learned by the CSS from its peers. For details, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Displaying DNS-Record Statistics".

Displaying DNS-Record Keepalives

Use the **show dns-record keepalive** command to display information about keepalives associated with DNS records. For details, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Displaying DNS Record Information".

Displaying DNS Server Zones

Use the **show zone** command to display information about proximity zones communicating with a CSS Network Proximity service. For details, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Displaying DNS Server Zones". To display PDB-related zone information, see "Showing Proximity Zones" earlier in this chapter.

Displaying DNS-Record Proximity

Use the **show dns-record proximity** command to display dns-record proximity statistics.

The syntax for this global configuration mode command is:

 # show dns-record proximity

For example:

```
(config)# show dns-record proximity
```

Table 5-15 describes the fields in the **show dns-record proximity** output.

Table 5-15 Field Descriptions for the show dns-record proximity Command

Field	Description
<*Domain name*>	The domain name for the record.
Zone	The index number for the zone. A "*" character preceding the zone number indicates that the zone is a local entry. A value of 255 indicates that the record never came up.
Description	The proximity zone description.
Hits Optimal	This entry increments when the DNS server returns the index that the PDB indicated was most proximate.
Hits SubOptimal	This entry increments when the DNS server returns an index that is different from the first one that the PDB indicated was most proximate.
Misses Optimal	This field increments when the PDNS must send a client to a zone that is not indicated by the first zone index returned by the PDB.
Misses SubOptimal	This field increments when the PDNS must send a client to a zone that is not indicated by either the first or second zone index returned by the PDB.

Displaying DNS Server Information

Use the **show dns-server** command to display DNS server configuration and database information. Although this command is not specifically a PDNS command, it is nonetheless useful for displaying DNS server information. For details on using the **show dns-server** command, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Displaying DNS Server and Zone Information".

Configuring VIP and Virtual IP Interface Redundancy

This chapter describes how to plan for and configure Virtual IP (VIP) and Virtual IP Interface Redundancy on the CSS. Information in this chapter applies to all CSS models except where noted.

This chapter contains the following sections:

VIP and Virtual IP Interface Redundancy Overview

The CSS enables you to configure redundancy for:

- Virtual IP Address (VIP) Redundancy
- Virtual IP Interface Redundancy

Virtual IP Address (VIP) Redundancy

When you configure more than one CSS for processing or forwarding client requests to the same Virtual IP address (VIP), the VIP is considered redundant. A typical use of VIP redundancy would be in a configuration where a master CSS processes all client requests to a VIP using a directly-connected dedicated Web server farm. If the Web server farm becomes unavailable, the backup CSS takes over using its own dedicated Web servers.

Note The CSS does not support VIP redundancy and CSS-to-CSS redundancy (IP redundancy) simultaneously. For information on IP redundancy, refer to Chapter 7, Configuring Redundant Content Services Switches.

To setup CSSs for VIP redundancy, you must configure a virtual router on each CSS that will participate in the redundant configuration. A virtual router is an entity within a CSS to which you associate an existing VIP. A VIP becomes redundant when you associate it with a virtual router. You may configure a maximum of 255 virtual routers for each VLAN. You can associate a virtual router only with a single VLAN.

Virtual routers providing redundancy for an IP address are considered peers. Each virtual router peer has the same identifier and runs on the same VLAN, but runs on a different CSS. Once the virtual routers are configured, the CSSs negotiate for mastership using Virtual Router Redundancy Protocol (VRRP). A virtual router in a redundant VIP configuration that is designated as:

- Master will process all client requests directed to the VIP
- Backup may be either a:
  - Backup virtual router, which forwards all client requests directed to the VIP to the master CSS
  - Shared backup virtual router, which processes all client requests it receives

A CSS can serve simultaneously as a master to one virtual router and as a backup to a different virtual router. All redundant VIP addresses will share the state of the virtual router to which it is associated.

Figure 6-1 shows an example of a redundant VIP configuration with:

- CSS-Boston configured as:
  - VLAN1 IP address 192.168.8.1.
  - Master virtual router 1 for VIP 192.168.8.4.
  - Backup virtual router 2 for VIP 192.168.8.6. Because virtual router 2 is operating as a backup, it will forward all client requests it receives for VIP 192.168.8.6 to CSS-Cambridge master virtual router 2.
- CSS-Cambridge configured as:
  - VLAN1 IP address 192.168.8.2.
  - Backup virtual router 1 for VIP 192.168.8.4. Because virtual router 1 is operating as a backup, it will forward all client requests it receives for VIP 192.168.8.4 to CSS-Boston master virtual router 1.
  - Master virtual router 2 for VIP 192.168.8.6.

Figure 6-1 Master and Backup Virtual Router Redundant VIP Configuration Example

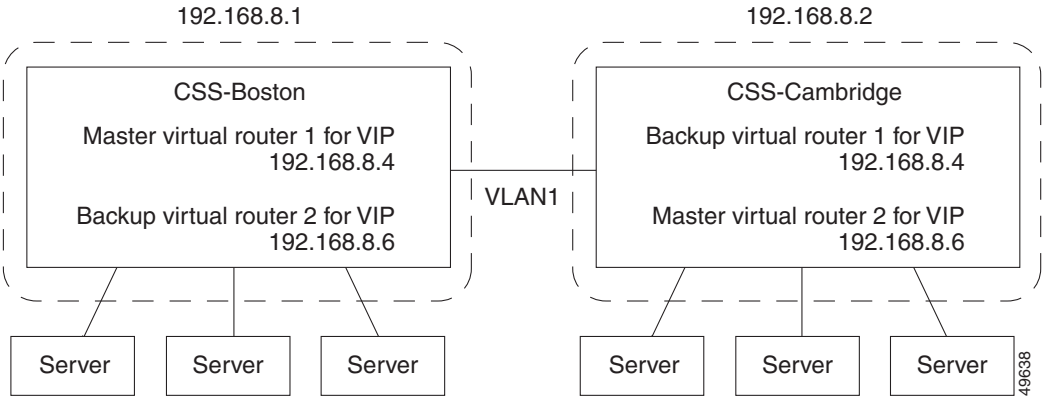
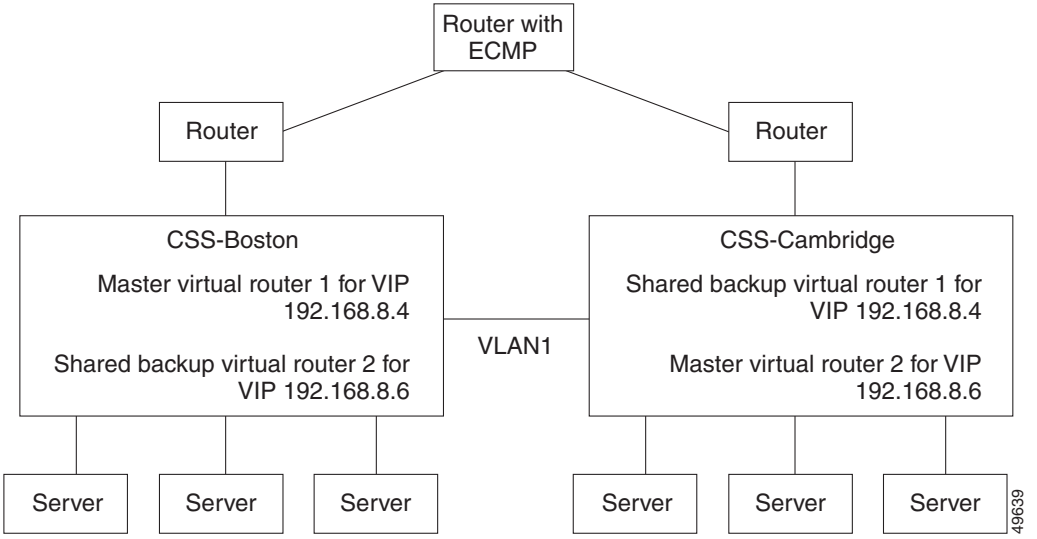


Figure 6-2 shows an example of a redundant VIP configuration with:

- CSS-Boston configured as:
  - Master virtual router 1 for VIP 192.168.8.4.
  - Shared backup virtual router 2 for VIP 192.168.8.6. Because virtual router 2 is operating as a shared backup, it will process all client requests it receives for VIP 192.168.8.6.
- CSS-Cambridge configured as:
  - Shared backup virtual router 1 for VIP 192.168.8.4. Because virtual router 1 is operating as a shared backup, it will process all client requests it receives for VIP 192.168.8.4.
  - Master virtual router 2 for VIP 192.168.8.6.

Figure 6-2 Master CSS and Shared Backup CSS Redundant VIP Configuration Example



Virtual IP Interface Redundancy

Virtual interface redundancy is a form of IP address redundancy that applies only to IP interfaces (not VIPs). A typical interface IP address on a CSS defines the interface in use on a particular VLAN. In this type of configuration, the CSS designated as master maintains control over the interface IP address.

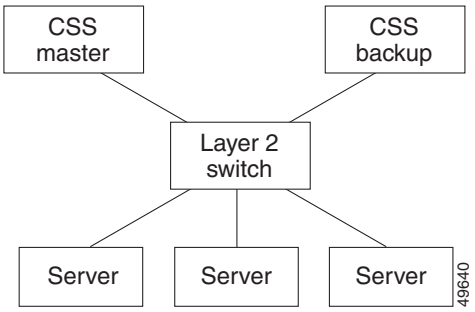
The typical use for virtual interface redundancy is in a configuration where servers are positioned behind a Layer 2 switch and CSSs with the redundant virtual interface are positioned in front of the Layer 2 switch. The servers would be configured with a default route pointing to the redundant virtual interface IP address.

A CSS designated as master of a virtual interface sends out gratuitous ARPs for the virtual interface's IP address. This enables the Layer 2 switch to learn where to forward packets that are directed to the virtual interface. This allows a server's default route to always point to the CSS designated as the master of the virtual interface.

Figure 6-3 shows an example of a virtual interface redundancy configuration with a master CSS and a backup CSS.

Note Interface redundancy does not support *shared* backup.

Figure 6-3 Virtual Interface Redundancy Configuration Example using a Master and a Backup CSS



VIP and Virtual IP Interface Redundancy Quick Start

Table 6-1 provides a quick overview of the steps required to configure VIP and virtual interface redundancy for *each* CSS in the redundant 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 command, refer to the sections following Table 6-1.

Table 6-1 VIP Redundancy and Virtual IP Interface Redundancy Configuration Quick Start

Task and Command Example

1. Enter into config mode.
`# config`
`(config)#`

2. Enter circuit mode for the desired circuit VLAN.
`(config)# circuit VLAN1`
`(config-circuit[VLAN1])#`

3. Configure a circuit IP address.
`(config-circuit[VLAN1])# ip address 192.168.8.1/24`
`(config-circuit-ip[VLAN1-192.168.8.1])#`

4. Configure the virtual router. Optionally, you may assign a priority different from the default of 100. Include the **preempt** keyword when configuring the master virtual router. The master virtual router must have the highest priority among its peers. `(config-circuit-ip[VLAN1-192.168.8.1])# ip virtual-router 1 priority 230 preempt`
5. Configure the redundant virtual IP interface on the virtual router. `(config-circuit-ip[VLAN1-192.168.8.1])# ip redundant-interface 1 192.168.8.6`

Table 6-1 VIP Redundancy and Virtual IP Interface Redundancy Configuration Quick Start (continued)

Task and Command Example

6. Configure the redundancy VIP on the virtual router. If you defined the content rule VIP using the range option, you must configure an identical range for the redundant VIP.
`(config-circuit-ip[VLAN1-192.168.8.1])# ip redundant-vip 1 192.168.8.10 range 10`
If you want the backup virtual router to process client requests, you must configure it as a shared backup.
`(config-circuit-ip[VLAN1-192.168.8.1])# ip redundant-vip 1 192.168.8.10 shared`

7. Configure the critical service for the virtual router. `(config-circuit-ip[VLAN1-192.168.8.1])# ip critical-service 1 serv1`
8. Display the configuration (optional). `(config)# show virtual-routers`

Because this chapter is dedicated to configuring VIP and virtual interface redundancy, it contains only those circuit IP commands that pertain to this feature. For a complete description of all circuit IP commands, refer to the *Cisco Content Services Switch Administration Guide*, Chapter 2, Configuring Interfaces and Circuits.

Configuring VIP and Virtual IP Interface Redundancy

The following sections describe how to configure VIP and virtual IP interface redundancy. You must configure each CSS in a redundant configuration.

Configuring a Circuit IP Interface

Before you can configure VIP and virtual interface redundancy, you must configure a circuit IP interface and assign it an IP address. To enter a specific circuit configuration mode, enter the **circuit** command and VLAN as shown in the following example:

```
(config)# circuit VLAN1
(config-circuit[VLAN1])#
```

Note When you use the **circuit** command, enter the word "VLAN" in uppercase letters and *do not* include a space between VLAN and the VLAN number (for example, VLAN1).

To assign an IP address to a circuit, use the **ip address** command from the specific circuit mode. Enter the IP address and a subnet mask in CIDR bitcount notation or a mask in dot-decimal notation. The subnet mask range is 8 to 32. For example, to configure an IP address and subnet mask for VLAN1, enter:

```
(config-circuit[VLAN1])# ip address 192.168.8.1/24
```

When you specify an IP address, the mode changes to the specific circuit-ip-VLAN-IP address as shown:

```
(config-circuit-ip[VLAN1-192.168.8.1])#
```

Configuring an IP Virtual Router

Use the **ip virtual-router** command to create a virtual router on a CSS and configure its identifier and priority used when negotiating control of associated VIPs. You must configure the virtual router before you can configure redundant VIPs.

A virtual router's role as a master or backup is determined during negotiations between all virtual routers with the same ID and on the same VLAN.

The syntax and options for the IP interface command are:

ip virtual-router *vrid* {**priority** *number*} {**preempt**}

The variables and options are:

- *vrid* - The virtual router identifier (VRID). Enter an integer between 1 and 255. You can configure 255 virtual routers per VLAN. Virtual routers are considered peers when they have the same VRID and are on the same VLAN.
- **priority** *number* - The optional priority of the virtual router with its peer. The default priority value is 100. Enter an integer between 1 and 255. A virtual router with the highest priority usually becomes master. However, a higher priority virtual router will not assume mastership from a lower priority master unless you include the **preempt** option. When a virtual router is the master, it handles the traffic directed to its associated VIPs. To set a virtual router so that it will always be master, set its priority to 255 and configure it with the **preempt** option.
- **preempt** - The optional keyword that allows a higher priority virtual router to assert mastership over a lower priority virtual router. By default, a virtual router does not become master if the current master has a lower priority. For example, if a CSS with a virtual router that has a low priority boots before other CSSs, that virtual router becomes the master. When another CSS with a virtual router that has a higher priority boots, it will not take the mastership from the first router unless you specify the **preempt** option on the higher priority virtual router. This option does not have an effect if the priority of the two virtual routers is identical. You can use this option with or without the **priority** option. You can configure only one virtual router as the master of a particular VIP.

Caution Never configure the **preempt** option on the same virtual router on both CSSs. Such a configuration may result in both CSSs becoming master, which will cause network problems.

Because a virtual router's priority is dependent on the state of the critical services, the priority field status in the **show virtual router** display may be different than the priority you configured. The priority may be different when you:

- Assign a priority of 255 to a virtual router and that virtual router gains mastership, the CSS automatically reconfigures that virtual router's priority to 254. This action ensures that you can assign a different virtual router a priority of 255.
- Configure critical services. The critical service types are:
  - **scripted -** the priority changes to 0 when one service in the scripted group goes down
  - **redundancy uplink** - the priority changes to 0 when all of the services in the uplink group go down
  - **local** - the priority changes to 0 when all of the services in the local group go down. Local services include all services other than scripted and uplink.

For information on configuring critical services, refer to "Configuring IP Critical Services" later in this chapter.

For example:

```
(config-circuit-ip[VLAN1-192.168.8.1])# ip virtual-router 1
priority 1 preempt
```

To remove the virtual router from the CSS, enter:

```
(config-circuit-ip[VLAN1-192.168.8.1])# no ip virtual-router 1
```

Configuring an IP Redundant Interface

Use the **ip redundant-interface** command to configure a redundant virtual interface address used for a backend server's default route. Servers use the IP address of the virtual interface as a default route to guarantee packets will be sent to the CSS containing the master virtual router. You may assign a redundant interface with the same virtual router of a VIP that has a rule that refers to the server. This ensures that the master for a VIP is also the CSS that is master for the redundant virtual interface.

The syntax for this IP mode command is:

ip redundant-interface *vrid ip_address*

The variables are:

- *vrid* - The ID for a previously configured virtual router.
- *ip_address* - The address for the redundant interface. Enter an IP address in dotted-decimal notation (for example, 192.168.8.6).

Note You cannot use an IP address that already exists for a VIP, redundant VIP, source group, service, log host, or IP interface address on a circuit. If you do, the following error message appears: `Address conflicts with local I/F, VIP, service, or sourcegroup.`

For example:

```
(config-circuit-ip[VLAN1-192.168.8.1])# ip redundant-interface 1
192.168.8.6
```

To remove an interface from a virtual router, enter:

```
(config-circuit-ip[VLAN1-192.168.8.1])# no ip redundant-interface
1 192.168.8.6
```

Note The CSS does not support a traceroute of a redundant IP interface.

Configuring an IP Redundant VIP

Use the **ip redundant-vip** command to associate an existing VIP to a virtual router and if required, configure the virtual router as a shared backup. A shared backup virtual router processes client requests. A redundant VIP configuration can consist of only two CSSs.

Note Before you use this command, the VIP must be configured in a minimum of one content rule. Additionally, if you defined the content rule VIP using the range option, you must configure an identical range for the redundant VIP.

The syntax for this IP mode command is:

ip redundant-vip *vrid vip_address* {**range** *number*} {**shared**}

The variables and options are:

- *vrid* - The ID for an existing virtual router.
- *vip_address* - The address for the redundant VIP. This address must be already configured in a content rule. Enter an IP address in dotted-decimal notation (for example, 192.168.8.10).
- **range** *number* - The optional keyword and variable if an IP address range is specified in the content rule. You cannot specify a range that differs from the content rule. Also, you cannot specify address ranges to overlap. Enter a number from 0 to 65535. The default is 1.
- **shared** - The optional keyword to enable shared VIP redundancy. When you use this option, the master and backup virtual routers share the processing of traffic directed to the VIP, so the backup does not forward packets to the master. Each VIP should be configured identically on both CSSs.

For example:

```
(config-circuit-ip[VLAN1-192.168.8.1])# ip redundant-vip 1
192.168.8.10 range 10 shared
```

To remove a VIP from a virtual router, enter:

```
(config-circuit-ip[VLAN1-192.168.8.1])# no ip redundant-vip 1
192.168.8.10
```

Configuring IP Critical Services

Use the **ip critical-service** command to associate a critical service with a virtual router. When a critical service goes down, the associated virtual router will also go down. There are three types of critical services that you can configure:

- A scripted critical service, as defined by the **(config-service) keepalive type script** command or the **(config-service) keepalive type named** command, that is constantly scanning for service and network availability. The keepalive sets the service to a down state whenever network or service availability is a problem. The virtual router goes down if *any* associated scripted service goes down.
- A redundancy uplink critical service, as defined by the **(config-service) type redundancy-up** command. The virtual router goes down when *all* associated redundancy uplink services go down regardless of any configured keepalive type. Refer to Chapter 7, Configuring Redundant Content Services Switches, in the section "Configuring Multiple Redundant Uplink Services".

Note You cannot add redundant uplink services to a content rule.

- Local critical services for any service other than scripted or redundancy uplink, such as a Web service. The virtual router goes down when *all* associated local critical services go down.

The syntax and options for the **ip critical-service** command are:

ip critical-service *vrid service_name*

The variables are:

- *vrid* - The ID for an existing virtual router.
- *service_name* - The name of the service. To see a list of services, enter **ip critical-service** *vrid* **?**.

For example:

```
(config-circuit-ip[VLAN1-192.168.8.1])# ip critical-service 1
serv1
```

To remove a critical service from a virtual router, enter:

```
(config-circuit-ip[VLAN1-192.168.8.1])# no ip critical-service 1
serv1
```

Note The **show service** command displays the current service type only. It does, however, display the keepalive type, so you can determine from it the behavior of a configured critical service. To display critical service-specific information, use the **show critical-services** command. See "Displaying IP Critical Services" later in this chapter.

Note SNMP values returned for services show the current service type only. To determine the critical service behavior of a particular service, you need to consult the service keepalive type. For more information on SNMP, refer to the *Cisco Content Services Switch Administration Guide*.

Synchronizing a VIP Redundancy Configuration

To ensure that your backup CSS can perform the same tasks as your master CSS in the event of a master CSS failure, the running-config on the backup must be identical (with some modifications) to the running-config on the master. To automate this configuration synchronization process, you can run a script (commit_VipRedundConfig) on the master CSS that copies the master CSS running-config to the backup CSS running-config.

There are two types of configuration synchronization:

- **Complete** - On CSSs that have an identical chassis (the same CSS model), produces a running-config on the backup CSS that exactly matches the running-config on the master CSS.

- **Partial** (default) - On CSSs with incompatible configurations, synchronizes all parameter values in the configuration except the interface and circuit configurations. For example, the master is a CSS 11506 and the backup is a CSS 11503. The script maintains the current backup interface and circuit configurations automatically.

Script Functions

The configuration synchronization script performs all the necessary steps to update the backup CSS with the master's running configuration. The script:

- Saves the master running-config to the startup-config
- Archives the startup-config
- Copies the startup-config to a temporary file (tmp.cfg)
- Calls a function that converts the master VRRP/APP IP addresses to the backup VRRP/APP IP addresses in tmp.cfg
- Changes all VRID priorities on the master to 254
- Changes all VRID priorities in the backup's new config to 1
- Removes all preempt configs from all VRIDs in the backup's new config

- Uses **rcmd** to:
  - Copy tmp.cfg to a temp file on the backup (newconfig)
  - Check newconfig and copy it to the startup-config
  - Clear the backup CSS's running-config and script play newconfig

The script performs some verifications before executing the above steps. It checks to see if the local switch is a backup for any VRIDs and asks you if you want to continue, thereby changing the state on the two CSSs. The script also checks the backup to see if it is the master for any VRIDs. If the state is Interface (IF) Down, the script asks you if you want to continue without synchronizing those VRIDs on interfaces that are Down.

Before You Begin

Before you run the configuration synchronization script, ensure that you have configured VIP/interface redundancy and the Application Peering Protocol (APP). For details on configuring VIP/interface redundancy, see "Configuring VIP and Virtual IP Interface Redundancy" earlier in this chapter. For details on configuring APP, refer to Chapter 1, Configuring the CSS Domain Name Service in the section "Configuring Application Peering Protocol".

The synchronization script does not support the following configurations:

- Active/active shared VIP
- Any configuration where some independent VIP addresses are a master while other VIP addresses are a backup

Running the Configuration Synchronization Script

To run the configuration synchronization script, use the **script play commit_vip_redundancy** command. The syntax is:

script play commit_vip_redundancy "*arguments***"**

You can also run the configuration synchronization script using the predefined alias that comes with all CSSs, as follows:

commit_VipRedundConfig "*arguments***"**

The arguments for the commit_redundancy script are:

- *ip address* - The IP addresses of the master and backup APP sessions. This is the only required argument for this script. Use the following syntax when entering the addresses:

  "local *master IP address* **remote** *backup IP address***"**

  For details on automating the entry of the IP address, see "Setting the BACKUP_VIPR_IP Variable" later in this section.

- **-a** (All) - Synchronizes the configuration completely. Use this argument only when the master and backup CSSs have identical chassis. This argument synchronizes the entire configuration and the interface mode.
- **-d** (Debug) - Debug switch for the commit_redundancy script, which displays the current task being performed as the script progresses. Debug messages display even when you specify the **-s** argument.

Caution Before you use the **-f** argument to remove a config sync lock file, ensure that no one else is running the config sync script on the CSS. Otherwise, if you remove the lock file and then run the script again while the script is in use, the resulting configurations may have some discrepancies.

- **-f** - After an abnormal script termination, removes the lock file so that you can run the script again. This argument overrides all other specified arguments and the script exits immediately after removing the lock file. For details on the lock file, see "Setting the BACKUP_VIPR_IP Variable" later in this section.
- **-nv** (No Verify) - Informs the script not to verify that the configuration synchronization was successful.

Note The script verifies the configuration synchronization by default.

- **-s** (Silent) - Suppresses script progress messages and displays only the result of running the script: Commit Successful or Commit Failed. The **-d** argument overrides the **-s** argument.

Note You can specify the script arguments in any order.

For example, on the master CSS, run the following script, which uses the defaults of verify on and partial synchronization, plus the IP addresses set as variables and the script alias name:

```
CSS11503# commit_VipRedundConfig
```

The following output appears:

```
CSS11503# commit_VipRedundConfig
Verifying app and redundancy configs ...
Checking vip redundancy state ...
Working \
Verifying running-config copy success ...
Commit successful!
```

In this example, the script:

- Performs a partial configuration synchronization (default)
- Verifies that the configuration synchronization was successful (default)

For more information on scripts, refer to Chapter 11, Using the CSS Scripting Language.

Config Sync Lock File

When you run the script, the software creates a lock file (vipr_config_sync_lock) in the script directory so that you cannot run the script from another session on the CSS. If the lock file exists and you run the script, the following message appears:

```
The script is in use by another session.
```

If the script terminates abnormally, the software does not remove the lock file. The next time you run the script, the above message appears. If you are certain that the script is not in use by another session, use the -f argument to remove the lock file. When you run the script with this argument, the following message appears and the script exits:

```
VIPR Config Sync lock file removed.
```

Now you can run the script again.

Setting the BACKUP_VIPR_IP Variable

To eliminate the need to specify IP addresses each time you run the configuration synchronization script, you can set the value of two variables (BACKUP _VIPR_IP and MASTER_VIPR_IP) to IP addresses and save them in your user profile. Once you set the variables and save them in your user profile, the variables will always be available after you log in to the CSS.

To set the variables, enter:

```
# set BACKUP_VIPR_IP "ip_address" {session}
# set MASTER_VIPR_IP "ip_address" {session}
```

where, *ip_address* is the IP address of the backup or master CSS.

To save the variable in your user profile, enter:

```
# copy profile user-profile
```

Now you can run the configuration synchronization script without typing an IP address.

Logging Configuration Synchronization Script Result Messages

You can specify that script result messages (script success or failure messages) be sent to the current logging device automatically each time you run the configuration synchronization script. To log the script result messages, enable logging on NETMAN with level info-6 or debug-7 by entering:

```
(config)# logging subsystem netman level info-6
```

Note Log messages are generated with or without the -s (silent) argument specified. See "Running the Configuration Synchronization Script" earlier in this chapter.

For example, if the APP session to the backup CSS is not running, the CSS generates the following log message:

```
vipr config sync: app session is DOWN
```

For ease of tracking, each log message contains the string "vipr config sync".

Displaying Redundant VIP and Virtual IP Interface Configurations

The CSS provides show commands to enable you to display redundant VIP and virtual interface configurations. The following sections describe the commands and provide examples of the screen displays and tables describing the screen fields.

Displaying IP Critical Services

Use the **show critical-services** command to display a list of all critical services configured on the CSS. You may provide an interface IP address option to display only the critical services present on a particular interface. You may also include a VRID to display only the critical service information for a particular virtual router.

The syntax for this command is:

show critical-services {*ip_address* {*vrid*}}

The optional variables are:

- *ip_address* - The address for the redundant interface. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1).
- *vrid* - The ID for an existing virtual router.

For example, to view all critical services on the CSS, enter:

 # show critical-services

Table 6-2 describes the fields.

Table 6-2 Field Descriptions for the Show Critical Services Command

Field	Description
Interface Address	The IP interface address associated with the virtual router.
VRID	The assigned identifier associated with the virtual router.
Service Name	The name of the critical service.
Service Type	The type of critical service. Possible critical service types are: • **Scripted** - A service whose state depends upon a running script or a named keepalive. • **Redundancy-up** - A service whose state depends upon the state of an ICMP keepalive on a router. • **Local** - Every type of service other than a scripted service or a redundancy uplink service. Typically, this is a Web server.

Displaying Redundant Interfaces

Use the **show redundant-interfaces** command to display a list of all redundant virtual IP interfaces configured on the CSS. You may provide an interface IP address option to display only the virtual interfaces present on a particular interface. You may also include a VRID to display only the virtual interface information for a particular virtual router.

The syntax for this command is:

show redundant-interfaces {*ip_address* {*vrid*}}

The optional variables are:

- *ip_address* - The address for the redundant interface. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1).
- *vrid* - The ID for an existing virtual router.

For example, to view all redundant interfaces on the CSS, enter:

```
(config) # show redundant-interfaces
```

Table 6-3 describes the fields.

Table 6-3 Field Descriptions for the Show Redundant Interface Command

Field	Description
Interface Address	The IP interface address associated with the redundant virtual interface.
VRID	The assigned identifier associated with the virtual router.
Redundant Address	The IP address of the redundant virtual interface.
Range	Not applicable. This field is always set to 1.
State	Current state of the virtual router. Possible states are: • **Master** - The virtual router is master. • **Backup** - The virtual router is backup. • **No Service** - One or more critical services associated with the virtual router is down. • **IF Down** - The IP interface associated with the virtual router is down.
Master IP	The IP address of the master virtual router.
State Changes	The number of times the redundant virtual interface state has changed.
Last Change	The date and time of the redundant virtual interface state last state change.

Displaying Redundant VIPs

Use the **show redundant-vips** command to display a list of all redundant VIPs configured on the CSS. You could provide an interface IP address option to display only the VIPs present on a particular interface. You can also include a VRID to display only the VIP information for a particular virtual router.

The syntax for this command is:

show redundant-vips {*ip_address* {*vrid*}}

The optional variables are:

- *ip_address* - The address for the redundant interface. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1).
- *vrid* - The ID for an existing virtual router.

For example, to view all redundant VIPs on the CSS, enter:

```
(config)# show redundant-vips
```

Table 6-4 describes the fields.

Table 6-4 Field Descriptions for show redundant-vips Command

Field	Description
Interface Address	The IP interface address associated with the redundant VIP.
VRID	The assigned identifier associated with the virtual router.
Redundant Address	The IP address of the VIP.
Range	The range associated with the VIP.

Table 6-4 Field Descriptions for show redundant-vips Command (continued)

Field	Description
State	Current state of the virtual router. Possible states are: • **Master** - The virtual router is master. • **Backup** - The virtual router is backup. • **No Service** - One or more critical services associated with the virtual router is down. • **IF Down** - The IP interface associated with the virtual router is down.
Master IP	The IP address of the master virtual router.
State Changes	The number of times the VIP state has changed.
Last Change	The data and time of the VIP last state change.

Displaying Virtual Router Configurations

Use the **show virtual-routers** command to display a list of all virtual routers configured on the CSS. You may provide an interface IP address option to display only the virtual routers present on a particular interface. You may also include a VRID to display only the information for a particular virtual router.

The syntax for this command is:

show virtual-routers {*ip_address* {*vrid*}}

The optional variables are:

- *ip_address* - The address for the redundant interface. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1).
- *vrid* - The ID for an existing virtual router.

For example, to view all virtual routers on the CSS, enter:

```
(config)# show virtual-routers
```

Table 6-5 describes the fields.

Table 6-5 Field Descriptions for the show virtual-routers Command

Field	Description
Interface Address	The IP interface address associated with the virtual router.
VRID	The assigned identifier associated with the virtual router.
Priority	The priority currently being advertised by the virtual router. Because the priority is dependent on the state of the critical services, the priority may be different than the one configured.
Config. Priority	The configured priority.
State	Current state of the virtual router. Possible states are: • **Master** - The virtual router is master. • **Backup** - The virtual router is backup. • **No Service** - One or more critical services associated with the virtual router is down. • **IF Down** - The IP interface associated with the virtual router is down. • **Idle** - The virtual router does not have any virtual interfaces or VIPs associated with it.
Master IP	The IP address of the master virtual router.
State Changes	The number of times the virtual router state has changed.
Last Change	The data and time of the virtual router last state change.
Preempt	True if preemption is enabled for the virtual router; false otherwise.
Critical-Services	The names of the critical services.

Table 6-5 Field Descriptions for the show virtual-routers Command (continued)

Field	Description
State	The current condition of the critical service. Possible states are: Alive, Down, or Suspended.
Type	The type of critical service. Possible types are: Scripted, RedundancyUp, or Local.

VIP and Virtual IP Interface Redundancy Running-Config Examples

The following running-config examples show VIP redundancy configured on two CSSs.

CSS-Boston Running-Config

```
!************************* GLOBAL **************************
!************************ INTERFACE ************************
interface e1
   bridge vlan 1
!************************ CIRCUIT **************************
circuit VLAN1
   ip address 192.168.8.1 255.255.255.0
   ip virtual-router 1 priority 230 preempt
   ip redundant-vip 1 192.168.8.10 shared
   ip critical-service 1 serv1
!************************* SERVICE *************************
service serv1
   ip address 20.1.1.1
   active
!************************** OWNER **************************
owner arrow
content L5_1
   protocol tcp
   vip address 192.168.8.10
   port 80
   url "/*"
   add service serv1
   active
```

CSS-Cambridge Running-Config

```
!************************** GLOBAL **************************
!************************ INTERFACE ************************
interface e1
   bridge vlan 1
!************************ CIRCUIT ************************
circuit VLAN1
   ip address 192.168.8.2 255.255.255.0
   ip virtual-router 1 priority 200
   ip redundant-vip 1 192.168.8.10 shared
   ip critical-service 1 serv2
!************************ SERVICE ************************
service serv2
   ip address 20.2.2.2
   active
!************************** OWNER **************************
owner arrow
content L5_1
   protocol tcp
   vip address 192.168.8.10
   port 80
   url "/*"
   add service serv2
   active
```

Configuring Adaptive Session Redundancy

Configure Adaptive Session Redundancy (ASR) on 11500 series CSS peers in an active-backup VIP redundancy and virtual IP interface redundancy environment to provide stateful failover of existing flows. ASR ensures that, if the master CSS fails, the backup CSS has the necessary flow state information to continue any active flows (including TCP and UDP) without interruption when the backup CSS assumes mastership. “Adaptive” means that you can configure ASR on a per content rule basis.

Use ASR for:

- Mission-critical enterprise applications.
- Long-lived flows such as FTP and HTTP file transfers.
- E-commerce applications, such as online stock trading or banking where users must remain connected to a service for the duration of a transaction even if the master CSS fails.

In an ASR configuration, CSSs replicate flows that are:

- Fully-resolved (the master CSS has received a SYN/ACK from a server)
- Set up using content rules, services, and source groups that you specify as *redundant*

Note For implicit or explicit Layer 5 rules, where there is delayed binding, binding is not complete until the CSS processes the SYN/ACK from the server. If a failover occurs in the middle of a spanned content request, the master CSS will not receive the SYN/ACK from the server and the flow will not be replicated on the backup CSS. No data is lost and users can simply refresh their browsers to restart the connection.

Note During an FTP failover, the control channel and/or the data channel need to share information with the backup CSS. If the current state information has not been fully transferred across the ISC link to the backup CSS, then the flow may be lost.

This section includes the following topics:

Stateful Failover

Active flows that match a redundant content rule, service, or source group on the master CSS are replicated as *dormant flows* on the backup CSS peer. A dormant flow contains all the flow state information necessary for the backup CSS to take over the flow if the master CSS fails, including the flow ID assigned by the session processor (SP) that created the flow. If the master CSS fails, the dormant flows on the backup CSS become active when the backup CSS assumes mastership of the VIP. In turn, the active flows on the former master CSS transition to a dormant state to fully back up the active flows on the new master CSS.

A master CSS maps a newly activated TCP flow after it receives the first packet for the flow. If it can resolve a single route back to the source address, a CSS attempts to map a UDP flow when it activates the flow. Otherwise, the CSS maps the UDP flow after it receives the first packet for the flow.

Inter-Switch Communications

In an ASR configuration, CSS peers share redundant flow state information over a maximum of two private Inter-Switch Communications (ISC) links after booting. ISC is a messaging service used by CSSs to exchange flow state information. Only one ISC link is active at a time. The other ISC link (if configured) remains in backup mode until needed.

To determine if an ISC link is up, a CSS uses a mechanism called LifeTick. LifeTick sends an asynchronous message that contains information about the selected path. If the CSS does not receive a LifeTick message within one second, the CSS considers the ISC link to be down. If a second link is configured, the CSS uses that link for ISC.

The ISC links use the Gigabit Ethernet ports or the Fast Ethernet ports on the CSS session processors (SPs) to send ISC messages containing the flow state information. Once you configure the ISC ports, you cannot use those same ports for non-ISC traffic.

Note You must connect the ISC ports directly to the two CSSs. You cannot use L2 devices on the ISC links between the two CSSs. Also, the ISC links must be dedicated to passing only ISC traffic.

For new flows, CSSs exchange flow states in real time over the ISC links. For existing flows, CSSs exchange flow states at boot-up time and at VIP redundancy failover.

Redundant Indexes

ASR uses unique global redundant indexes to keep track of content rules, services, and source groups configured on the redundant CSS peers. Set up the redundant indexes in rules, services, and groups using the **redundant-index** command. You must then configure identical redundant content rules, services, and source groups on CSS peers in the ASR configuration.

Each redundant index that you configure on a rule, service, or group must be unique among all rules, services, or groups configured on a redundant pair of CSSs. For example, if you configure a rule with a redundant index of 1 on a pair of CSSs, you cannot configure an index of 1 on another rule. However, you could configure an index of 1 on a group or service if that value has not already been used on a group or a service.

Note If you run traffic to a configuration that contains discrepancies between the redundant indexes on the two CSSs, the CPU utilization for each processor on the CSS may climb to an abnormal level (at 2000 flows/second, approximately 50 percent utilization for each processor). If you set the logging level to notice-5 or higher, the SCM utilization may peak at approximately 90 percent because each connection generates a redundant index mismatch log entry. For example: AUG 7 14:12:15 3/1 1124272 SLR-5: Rejected. Redundant global rule index (7) not found.

Configuration Requirements and Restrictions

The following requirements and restrictions apply to both CSS peers in an ASR configuration:

- Configure VIP/virtual IP interface redundancy on both CSS peers. For details, see "Configuring VIP and Virtual IP Interface Redundancy" earlier in this chapter.
- Configure a redundant VIP in a redundant content rule or source group. In order to activate a redundant content rule or source group, you must associate the rule or group with a redundant VIP.
- Ensure that VIP ranges specified in redundant content rules and source groups are the same as the VIPs associated with virtual routers for VIP redundancy. If the redundant content rule or source group VIPs are a superset, ASR is supported only for the VIPs that are associated with the virtual routers. For the remaining VIPs, the behavior is undefined when a failover occurs, because it is unclear whether those VIPs are mastered on the new master CSS or not.
- You cannot configure VIP wildcard or double-wildcard caching rules because they do not require a VIP. For information on wildcard cache rules, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 7, Configuring Caching.
- Configure ISC on both CSSs. This allows the CSSs to share flow state information.

- Configure a maximum of two ISC ports on a CSS. Multiple ports must reside on the same module in the CSS 11503 or CSS 11506 or on the same CSS 11501. Also, the ports must be of the same type (Gigabit Ethernet or Fast Ethernet) in both CSSs. Ensure that the ISC ports are not configured in any VLANs. If necessary, remove the designated ports from all VLANs before configuring ISC. For details on disabling an interface port from a VLAN, refer to the *Content Services Switch Administration Guide*.

  You must connect the ISC ports directly to the two CSSs. You cannot use L2 devices on the ISC links between the two CSSs. Also, the ISC links must be dedicated to passing only ISC traffic.
- If you configure any ISC ports on an SCM, you can have only one SCM installed in the CSS 11506.
- The CSS 11501 does not support redundant GE Inter-Switch Communications links for ASR because the switch includes only a single GBIC port.
- Ensure that any service configured with connection limits, marked as redundant, *and* used by at least one redundant content rule is used only by other content rules that are also redundant. If this is not true, there could be redundant and nonredundant flows connected to the service with connection limits. In case of a failover, no information is available for the nonredundant flows on the backup CSS. Until the server cleans up the nonredundant flow connections, they continue to contribute to the connection limit on the service without the backup CSS having any knowledge of how many such connections exist. Making all flows redundant by imposing the above restrictions eliminates this problem.
- When you configure critical services, be sure to change the default keepalive settings to the following recommended settings for ASR. For example, enter:

```
service CriticalService
    ip address 192.168.2.1
    keepalive frequency 2
    keepalive maxfailure 2
    keepalive retryperiod 2
    active
```

Note The above keepalive values are a recommended starting point. Some scripted keepalives may take longer than two seconds to run. You may need to adjust your keepalive values so that the CSS detects a failure before your application times out.

- Configure as redundant any source groups that you specify in ACL clauses. It is helpful to configure ACLs similarly on the master and backup CSSs. This ensures that the CSSs can share the portmap state during flow setup time, and, at failover time, a CSS finds the same ACL and source group configured on the peer. Otherwise, when a flow fails over to the backup, it is possible that the flow may match on a different ACL clause that has no source group configured or a different source group (possibly a nonredundant one).

  Source groups selected by ACL-checking always take precedence over other source group matches for a flow. Therefore, if the master and backup CSSs have different ACL definitions, when a flow fails over to the backup and the source group selected on the master is not found on the backup, the CSS rejects the flow. Also, if the flow matches on a different source group through an ACL, that source group takes precedence over the redundant source group that was sent from the master.
- Configure as redundant any preferred service that you configured in an ACL clause.
- Configure mutually exclusive portmap ranges on the redundant peers using the **global-portmap** command to avoid potential network port collisions. Keeping the portmap ranges mutually exclusive on the redundant peer also eliminates the need to dynamically update the global portmap database on the backup CSS. For more information on portmapping, refer to the *Cisco Content Services Switch Administration Guide*, Chapter 6, Configuring User Profiles and CSS Parameters.
- Do not configure ASR and stateless redundancy failover on the same CSS. Such a configuration is not supported. For details on stateless redundancy failover, refer to Chapter 7, Configuring Redundant Content Services Switches, in the section "Configuring Stateless Redundancy Failover".
- ASR does not support NAT Peering. For details on NAT Peering, refer to the *Content Services Switch Basic Configuration Guide*.

Adaptive Session Redundancy Quick Start

Table 6-6 provides a quick overview of the steps required to configure ASR for *each* CSS in the redundant configuration. Each step includes the CLI command or a reference to the procedure required to complete the task. For a complete description of each feature and all the options associated with the CLI command, refer to the sections following Table 6-6.

Table 6-6 Session Redundancy Configuration Quick Start

Task and Command Example

1. Enter config mode.
 	
`# config`
`(config)#`

2. Configure active/backup VIP/virtual IP interface redundancy. See "Configuring VIP and Virtual IP Interface Redundancy" earlier in this chapter.

3. Configure a maximum of two directly connected (no intervening L2 devices) ISC links on Gigabit Ethernet or Fast Ethernet ports between the two redundant CSSs. See "Configuring Inter-Switch Communications" later in this chapter.
 	
`(config)# interface 1/1`
`(config-if[ 1/1])# isc-port-one`
`(config)# interface 1/2`
`(config-if[ 1/2])# isc-port-two`

4. Configure services that are targets of redundant content rules. For more information on services, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 1, Configuring Services.
 	
`(config)# service server1`
`(config-service[server1])# ip address 192.168.100.100`
`(config-service[server1])# redundant-index 1`
`(config-service[server1])# active`

Table 6-6 Session Redundancy Configuration Quick Start (continued)

Task and Command Example

5. Configure redundant content rules and add the redundant services. For more information on content rules, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 3, Configuring Content Rules.
 	
`(config)# owner arrowpoint`
`(config-owner[arrowpoint])# content rule1`
`(config-owner-content[arrowpoint-rule1]# vip address 192.168.1.1`
`(config-owner-content[arrowpoint-rule1]# protocol tcp`
`(config-owner-content[arrowpoint-rule1]# port 80`
`(config-owner-content[arrowpoint-rule1]# url "/redundant.html"`
`(config-owner-content[arrowpoint-rule1]# add service server1`
`(config-owner-content[arrowpoint-rule1]# redundant-index 5`
`(config-owner-content[arrowpoint-rule1]# active`

6. Configure redundant source groups and add the redundant services. For more information on source groups, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 5, Configuring Source Groups, ACLs, EQLs, URQLs, NQLs, and DQLs.
 	
`(config)# group group1`
`(config-group[group1])# vip address 192.168.10.10`
`(config-group[group1])# add service server1`
`(config-group[group1])# redundant-index 4`
`(config-group[group1])# active`

7. Configure global portmapping (port translation) with mutually exclusive port ranges on the CSS peers to avoid potential port collisions. For more information on CSS portmapping, refer to the *Cisco Content Services Switch Administration Guide*, Chapter 6, Configuring User Profiles and CSS Parameters.
 	
For example, on one CSS peer, enter:
 	
`(config)# global-portmap base-port 3000 range 30000`
 	
On the other CSS peer, enter:
 	
`(config)# global-portmap base-port 33100 range 30000`

8. Configure the same redundant services, content rules, and source groups on the other CSS peer (synchronize the configurations).

Configuring Inter-Switch Communications

Inter-Switch Communications (ISC) is a messaging service that 11500 series CSS peers use to exchange flow state information in an ASR configuration. If the master CSS fails, the backup CSS already has the flow state information necessary to continue the current flows without interruption. Using ISC, CSSs exchange state information:

- For existing flows at boot-up time and at VIP redundancy failover
- For new flows in real time (after the CSS receives a SYN/ACK from the server)

Use the **isc-port-one** and **isc-port-two** commands in interface configuration mode to enable ISC between two 11500 series CSSs in an ASR configuration. You can configure a maximum of two ISC ports on each CSS. The two ports must be of the same type (Gigabit Ethernet or Fast Ethernet) and must be on the same module in the CSS 11503 or CSS 11506 or on the same CSS 11501.

The CSS 11501 does not support redundant GE Inter-Switch Communications links for ASR because the switch includes on a single GBIC port.

You must connect the ISC ports directly to the two CSSs. You cannot use L2 devices on the ISC links between the two CSSs. Also, the ISC links must be dedicated to passing only ISC traffic.

For example, to enable both ISC ports on a CSS 11506, enter:

```
(config)# interface 1/1
(config-IF[ 1/1])# isc-port-one
(config-IF[ 1/1])# interface 1/2
(config-IF[ 1/2])# isc-port-two
```

To disable both ISC ports on a CSS 11506, enter:

```
(config)# interface 1/1
(config-IF[ 1/1])# no isc-port-one
(config-IF[ 1/1])# interface 1/2
(config-IF[ 1/2])# no isc-port-two
```

Configuring Redundant Services

Use the **redundant-index** command to configure the global service index for a redundant service. A CSS uses the global service index to keep track of redundant services and associated flow state information.

The syntax for this service configuration mode command is:

redundant-index *index*

The variable *index* is a unique number you assign to a redundant service. Enter a unique integer from 0 to 32767, where a value of 0 disables ASR for a service. The default is 0, but it does not appear in the running-config even if you configure it explicitly.

For example:

```
(config-service[server1])# redundant-index 5
```

To disable ASR for a service, enter:

```
(config-service[server1])# no redundant-index
```

Note If you issue the **no redundant-index** command on an active redundant service for live redundancy peers, the command automatically suspends the service. Flows already mapped by a CSS are not affected. However, if a failover occurs during the life of an active flow that matches on such a suspended service, the backup CSS cannot map the flow because it cannot find the service with the same global index as that on the original master.

For more information on configuring services, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 1, Configuring Services.

Configuring Redundant Content Rules

Use the **redundant-index** command to configure the global content index for a redundant content rule. A CSS uses the global content index to keep track of redundant content rules and associated flow state information.

The syntax for this content configuration mode command is:

redundant-index *index*

The variable *index* is a unique number you assign to a redundant content rule. Enter a unique integer from 0 to 32767, where a value of 0 disables ASR on a content rule. The default is 0, but it does not appear in the running-config even if you configure it explicitly.

For example:

```
(config-owner-content[arrowpoint-rule1]# redundant-index 1
```

To disable ASR on a content rule, enter:

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

Note If you issue the **no redundant-index** command on an active redundant content rule for live redundancy peers, the command automatically suspends the content rule. Flows already mapped by a CSS are not affected. However, if a failover occurs during the life of an active flow that matches on such a suspended content rule, the backup CSS cannot map the flow because it cannot find the content rule with the same global index as that on the original master.

For more information on configuring content rules, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 3, Configuring Content Rules.

Configuring Redundant Source Groups

Use the **redundant-index** command to configure the global source group index for a redundant source group. A CSS uses the global source group index to keep track of redundant content rules and associated flow state information.

The syntax for this group configuration mode command is:

redundant-index *index*

The variable *index* is a unique number you assign to a redundant source group. Enter a unique integer from 0 to 32767, where a value of 0 disables ASR for a source group. The default is 0, but it does not appear in the running-config even if you configure it explicitly.

For example, to enable ASR for a source group:

```
(config-group[group1])# redundant-index 4
```

To disable ASR for a source group, enter:

```
(config-group[group1])# no redundant-index
```

Note If you issue the **no redundant-index** command on an active redundant source group on live redundancy peers, the command automatically suspends the source group. Flows already mapped by a CSS are not affected. However, if a failover occurs during the life of an active flow that matches on such a suspended source group, the backup CSS cannot map the flow because it cannot find the source group with the same global index as that on the original master.

For more information on configuring source groups, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 5, Configuring Source Groups, ACLs, EQLs, URQLs, NQLs, and DQLs.

Synchronizing Adaptive Session Redundancy Configurations

You must synchronize configurations on both CSS peers to ensure that the ASR-specific configurations on the master CSS and the backup CSS are the same. This is critical to the proper functioning of ASR.

For ASR, you must manually configure on each peer:

- ISC
- Redundant content rules
- Redundant services
- Redundant source groups

Displaying Adaptive Session Redundancy Information

Use the following commands to display information for:

- Inter-Switch Communications
- Dormant flows used for ASR
- ASR status and global redundant indexes

Displaying Inter-Switch Communications Ports

Use the **show isc-ports** command to display the ports configured for ISC on an 11500 series CSS.

Table 6-7 describes the fields in the **show isc-ports** output.

Table 6-7 Field Descriptions for the show isc-ports Command

Field	Description

Inter-Switch Communications Configuration	Lists the CSS ports (in slot/port format) configured for ISC port one and ISC port two. If ISC is not configured, the command displays the following messages:
	`Inter-Switch Port One is not configured.`
	`Inter-Switch Port Two is not configured.`

Inter-Switch Communications Status	Indicates whether ISC is Up or Down and, if Up, on which CSS port ISC is currently active.

Displaying Dormant Flow Information

Use the **show dormant flows** command to display information about the current dormant flows on the backup CSS in an ASR configuration. Dormant flows are flows on the backup CSS that become active if the master CSS fails and the backup CSS assumes mastership.

The syntax for this command is:

show dormant flows {*source_address* {*destination_address*}}

The optional variables for this command are:

- *source_address* - Displays dormant flows for the specified source IP address. Enter the IP address in dotted-decimal notation (for example, 192.168.11.1).
- *destination_address* - Displays dormant flows for the specified destination IP address. Enter the IP address in dotted-decimal notation (for example, 192.168.11.1).

Table 6-8 describes the fields in the **show dormant flows** output.

Table 6-8 Field Descriptions for the show dormant flows Command

Field	Description
Src Address	The source address for the flow.
SPort	The source port for the flow.
Dst Address	The destination address for the flow.
DPort	The destination port for the flow.
NAT Dst Address	The network address translation (NAT) destination address.
Prt In	Not applicable. A dormant flow does not have a port associated with it.
OutPort	Not applicable. A dormant flow does not have a port associated with it.

Use the **flow statistics dormant** command to display summary information about redundant dormant flows.

Table 6-9 describes the field in the **flow statistics dormant** output.

Table 6-9 Field Descriptions for the flow statistics dormant Command

Field	Description
Total Dormant Flows	The total number of inactive redundant flows mapped on the backup CSS from active redundant flows on the master CSS. The dormant flows contain all the flow state information necessary for the backup CSS to master the flows if the master CSS fails. If the master CSS fails, the backup CSS becomes the master CSS and the dormant flows become active flows.

Displaying ASR Information for Content Rules, Services, and Source Groups

The following sections describe how to display:

- ASR status and global index values
- Summary ASR information

Displaying ASR Status and Global Index Values

Use the **show rule**, **show service**, and **show group** commands to display information about ASR status and global redundant indexes. The relevant fields in the output of these commands are:

- **Session Redundancy** - The state of ASR for the content rule, service, or source group. Possible values are: Enabled or Disabled
- **Redundancy Global Index** - The unique global index value for ASR configured for the content rule, service, or source group using the **redundant-index** command.

For full details on the **show rule**, **show service**, and **show group** commands, refer to the *Cisco Content Services Switch Basic Configuration Guide*.

Displaying Summary ASR Information

Use the **show session-redundant** command to display summary ASR information about redundant content rules, services, and source groups.

The syntax for this global configuration mode command is:

show session-redundant [**rule**|**service**|**group**|**all**]

The optional keywords are:

- **rule** - Displays summary ASR information for redundant content rules.
- **service** - Displays summary ASR information for redundant services.
- **group** - Displays summary ASR information for redundant source groups.
- **all** - Displays summary ASR information for content rules, services, and source groups.

For example, to view summary ASR information for redundant content rules, enter:

```
(config)# show session-redundant rule
```

Table 6-10 describes the fields.

Table 6-10 Field Descriptions for the show session-redundant Command

Field	Description
Session Redundant Content Rules	
Content Rule	The redundant content rule name.
Content Rule State	The current state of the redundant content rule. Possible states are: Active or Suspend.
VIP Address	The virtual IP address of the redundant content rule in dotted decimal notation.
Redundancy Global Index	The ASR global index configured for the redundant content rule.
Redundancy State	The state of the CSS peer: Master, Backup, or Suspend.
Rule Redundant Services 1	The name of the redundant service and its global index value configured on the rule.
Session Redundant Services	

Table 6-10 Field Descriptions for the show session-redundant Command (continued)

Field	Description
Service	The name of the redundant service.
Service State	The current state of the redundant service. Possible states are: Alive, Dying, or Down.
IP Address	The virtual IP address of the redundant service in dotted-decimal notation.
Redundancy Global Index	The ASR global index configured for the redundant service.
Session Redundant Source Groups	
Source Group	The name of the redundant source group.
Source Group State	The current state of the redundant source group. Possible states are: Active or Suspend.
VIP Address	The virtual IP address of the redundant source group.
Redundancy Global Index	The ASR global index configured for the redundant source group.
Group Redundant Services	
Source Services	The redundant source services configured in this redundant source group, their keepalive state, and global index. If no source services are configured in this source group, the value is NONE.
Destination Services	The redundant destination services configured in this redundant source group and their keepalive state. If no destination services are configured in this source group, the value is NONE.

Configuring Redundant Content Services Switches

This chapter describes how to configure redundancy between two mirrored Content Services Switches (CSS). Information in this chapter applies to all CSS models, except where noted.

This chapter contains the following sections:

Redundancy Protocol Overview

The CSS redundancy protocol provides chassis-level redundancy between two CSSs. This feature protects the network and ensures that users have continuous access to servers and content.

Using the redundancy protocol CLI commands, you can configure two CSSs in a master and backup redundancy configuration. In a redundant configuration, the master CSS sends a redundancy protocol message (heartbeat) every second to inform the backup CSS that it is alive.

If the master CSS fails and does not send a message within 3 seconds, the backup CSS:

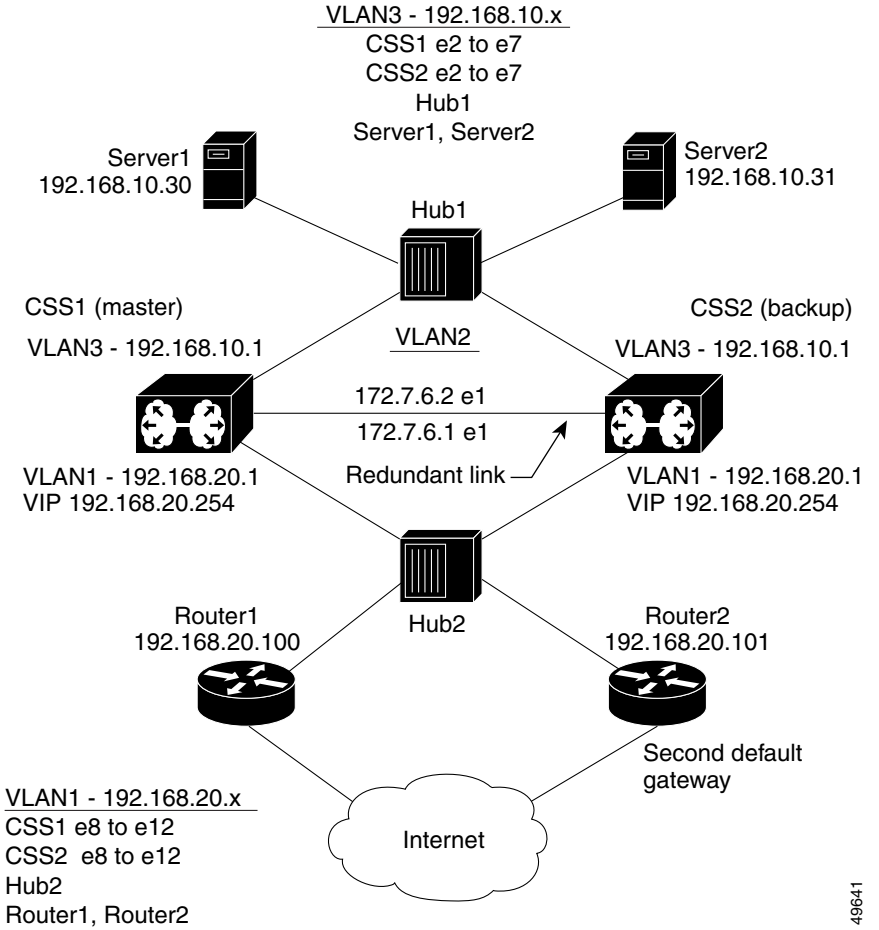
- Becomes the master CSS
- Begins sending out redundancy protocol messages
- Sends out gratuitous Address Resolution Protocol (ARP) messages in order to update the ARP tables on neighboring nodes and the forwarding tables of attached bridging devices (for example, Layer 2 switches)

If a former master comes back online, it becomes a backup CSS automatically when it receives the master CSS messages, unless you explicitly designated the CSS to be the master when you configured it. For details on IP redundancy, see "Configuring IP Redundancy" later in this chapter.

Caution When you use Access Control Lists (ACLs) in a redundant configuration, ensure that you permit all traffic on the redundant circuit between the master and backup CSSs. For information on ACLs, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 5, Configuring Source Groups, ACLs, EQLs, URQLs, NQLs, and DQLs.

Figure 7-1 shows an example of a redundant configuration with multiple VLANs. Table 7-1 uses this figure to define the command examples necessary to configure the redundancy protocol.

Figure 7-1 Redundancy Configuration Example



Redundancy Configuration Quick Start

Table 7-1 provides the steps required to configure the redundant configuration shown in Figure 7-1. Each step includes the CLI command required to complete the task. For a complete description of each feature, refer to the sections following Table 7-1.

Listed below are the basic steps to configure redundancy:

1. Install the crossover cable on the master and redundant CSS before you power them on.

Caution If you power on the CSSs before you install the cable, both units boot up as the master CSS and cause network problems. Do not remove the crossover cable from a redundant configuration or each CSS will become master

Note You must connect the crossover cable directly to the Fast Ethernet (FE) ports on the redundant CSSs. Do not use L2 devices between the two CSSs on the redundant link. Do not install the crossover cable on gigabit Ethernet (GE) ports; this configuration is not supported.

2. Configure each server's default gateway as the CSS's circuit VLAN IP address.
3. Configure redundancy on the existing master CSS and save the running-config to startup-config.
4. FTP the startup-config to a PC. Edit the file by including the backup CSS circuit VLAN IP addresses.
5. FTP the startup-config to the backup CSS. Reboot the backup CSS with the new startup-config.

As an alternative method, you can use CLI commands to manually configure the backup CSS with all necessary configurations including the redundancy protocol.

Note If you make configuration changes to the master CSS startup-config, you must make the same changes to the backup CSS startup-config. To learn how to synchronize the running-configs of the master CSS and the backup CSS, see "Synchronizing a Redundant Configuration" later in this chapter.

Table 7-1 Redundancy Configuration Quick Start

Task and Command Example
1. Install the crossover cable before you power up the CSSs. Make the CSS-to-CSS connection using a Category 5 crossover cable. This table uses port e1 (ethernet-1) on the master CSS and port e1 on the backup CSS.
2. Configure each server's default gateway as the CSS circuit VLAN IP address.
3. Enter the **ip redundancy** command on the master CSS to enable CSS-to-CSS redundancy. `(config)# ip redundancy`
4. Configure an interface on the master CSS for a redundant connection to the backup CSS. For information on configuring interfaces, refer to the *Cisco Content Services Switch Administration Guide*. `(config)# interface e1`
5. Assign the interface to the redundant connection VLAN. For information on bridging the interface to a VLAN, refer to the *Cisco Content Services Switch Administration Guide*. `(config-if[e1])# bridge vlan 2`

6. Enter circuit mode for the redundant VLAN. For information on configuring circuits, refer to the *Cisco Content Services Switch Administration Guide*.
`(config-if[e1])# circuit VLAN2`
`(config-circuit[VLAN2])#`

7. Assign an IP address and subnet mask to circuit VLAN2. For information on configuring a circuit IP address, refer to the *Cisco Content Services Switch Administration Guide*. `(config-circuit[VLAN2])# ip address 172.7.6.1/24`

8. Enable the redundancy protocol on the redundant IP interface.
`(config-circuit-ip[VLAN2-172.7.6.1])# redundancy-protocol`
`(config-circuit-ip[VLAN2-172.7.6.1])# exit`

9. Define the other circuits as redundant circuits.
`(config-if[e1])# circuit VLAN1`
`(config-circuit[VLAN1])# redundancy`
`(config-if[e1])# circuit VLAN3`
`(config-circuit[VLAN3])# redundancy`

Table 7-1 Redundancy Configuration Quick Start (continued)

Task and Command Example
10. From SuperUser mode, save the master CSS running-config to the startup-config. `# copy running-config startup-config`
11. FTP the startup-config to a PC for editing.
12. Using a text editor, edit the startup-config by including the backup CSS circuit VLAN IP address for the redundant connection (in Figure 7-1, circuit VLAN2, IP address 172.7.6.2/24). Do not change the backup CSS VIP. The master and backup CSS must have the same VIP. If you have multiple VIPs, you must configure them on both the master and backup CSSs.
13. FTP the new file to the backup CSS and, if necessary, rename it as startup-config.
14. Reboot the backup CSS.
15. Enter the **show redundancy** command to display the redundancy configuration and ensure that the backup CSS is configured properly.
16. Cable all hubs (or switches) to the backup CSS.

Cabling Redundant Content Services Switches

When you set up a redundant configuration, install a Category 5 crossover cable directly to the CSSs to connect the master and backup interfaces.

Caution If you power on the CSSs before you install the cable, both CSSs boot up as the master CSS and cause network problems. Do not remove the crossover cable from a redundant configuration or each CSS will become master.

Note You must connect the crossover cable directly to the Fast Ethernet (FE) ports before you power on the redundant CSSs. Do not use L2 devices between the two CSSs on the redundant link. Do not install the crossover cable on gigabit Ethernet (GE) ports; this configuration is not supported.

Table 7-2 lists the pinouts for the CSS Fast Ethernet connectors and the crossover pinouts.

Table 7-2 RJ-45 Fast Ethernet Connector Pinouts

Signal Name	Pin Number	Crossover Cable Pinouts
RX +	1	3
RX -	2	6
TX +	3	1
Unconnected	4	4
Unconnected	5	5
TX -	6	2
Unconnected	7	7
Unconnected	8	8

Configuring Redundancy

Configuring the redundancy protocol requires you to:

- Configure the master and backup CSSs for redundancy using the **ip redundancy** command.
- Enable the redundancy protocol on the master and backup circuit VLAN using the **redundancy-protocol** command.
- Enable the circuit VLAN for redundancy using the **redundancy** command.

Note The CSS does not support IP redundancy and VIP redundancy simultaneously. For information on VIP redundancy, refer to Chapter 6, Configuring VIP and Virtual IP Interface Redundancy.

Configuring IP Redundancy

Use the **ip redundancy** command to enable CSS-to-CSS redundancy on two CSSs interfaced with a crossover cable. By default, redundancy is disabled on CSSs until you issue this command on both CSSs.

When you include the **master** option with this command, you can designate which CSS is the master CSS. Initially, booting two CSSs interfaced with a crossover cable determines which is the master and which is the backup. The CSS that boots first is the master CSS. If the CSSs boot at the same time, the CSS with the numerically higher IP address becomes the master.

Note You cannot use the **ip redundancy master** command with either the **type redundancy-up** command (redundancy uplink service) or the **redundancy-phy** command (physical link redundancy). If necessary, disable the appropriate command using **no type redundancy-up** or **no redundancy-phy** before using the **ip redundancy master** command.

When you issue the **ip redundancy master** command on a CSS, it becomes the master CSS. You can issue this command on either the current master or backup. If you issue this option on the backup CSS, it becomes the master and the other CSS becomes the backup automatically.

If you designate a master CSS, it regains its master status after it goes down and then comes up again. For example, when the master CSS goes down, the backup CSS becomes master. However, when the former designated master CSS comes up again, it becomes the master again.

If you have no requirement to designate a CSS as the master when both CSSs are up, do not include the **master** option when enabling redundancy on the master CSS. In this configuration, if the master CSS goes down, the backup CSS becomes master. When the former master CSS comes up again, it becomes the backup CSS.

The syntax for this global configuration mode command is:

- **ip redundancy** - Enables CSS-to-CSS redundancy on the backup CSS. If you have no requirement to define a CSS as the master CSS, use this command on both CSSs in the redundant configuration.

- **ip redundancy master** - Enables CSS-to-CSS redundancy on the CSS that you want to designate as the master CSS. (Be sure to issue **ip redundancy** on the backup CSS.) You can issue **ip redundancy master** on a CSS:
  - Whether or not it was initially booted as the master or the backup. If you issue this command on the backup CSS, it becomes the master and the other CSS becomes the backup CSS automatically.
  - When CSS-to-CSS redundancy is currently enabled.

For example:

```
(config)# ip redundancy
```

Caution You cannot issue the **ip redundancy master** command on both the master and backup CSSs. This can cause severe network problems. Before you disable redundancy, ensure that you disconnect or disable all redundant circuits to prevent duplicate IP addresses from occurring on the network.

To disable CSS-to-CSS redundancy, enter:

```
(config)# no ip redundancy
```

Note The **no ip redundancy** command deletes the **redundancy** and **redundancy-protocol** commands from the running-config of the CSS.

Before the CSS disables redundancy, it displays the following message:

```
WARNING: Disabling redundancy may result in duplicate
IP addresses on the network.
Be sure you disconnect or disable all redundant circuits before
you disable redundancy.
Do you want to disable redundancy? [y/n]:
```

Type **y** to disable redundancy. Type **n** to cancel disabling redundancy.

To unassign the CSS as the master CSS, enter:

```
(config)# no ip redundancy master
```

Note The **no ip redundancy master** command does not disable CSS-to-CSS redundancy.

Configuring Redundant Circuits

To configure a circuit as a redundant circuit, use the **redundancy** command. The **redundancy** command is available in circuit configuration mode.

Note The redundancy command causes the specified VLAN to become silent when in backup mode.

When you configure redundancy, configure it on circuits (VLANs) that contain network IP addresses shared by the redundant CSSs (in Figure 7-1, these are VLAN1 and VLAN3). Do not configure redundancy on the circuit (VLAN) configured specifically for redundancy communications (in Figure 7-1, this is VLAN2).

The example below configures VLAN1 as a redundant circuit, which contains a shared network IP address (in Figure 7-1, this is 192.168.20.1).

For example:

```
(config-circuit[VLAN1])# redundancy
```

To remove a circuit from the redundancy configuration, enter:

```
(config-circuit[VLAN1])# no redundancy
```

Configuring the Redundancy Protocol

To configure the redundancy protocol on the circuit (VLAN) connecting the two CSSs, enter the **redundancy-protocol** command in IP interface configuration mode. Enter the command for the circuit you configured specifically for redundancy communications (in Figure 7-1, this is VLAN2).

Note Do not configure the redundancy protocol on CSS gigabit Ethernet (GE) ports; this configuration is not supported.

For example:

```
(config-circuit-ip[VLAN2-172.7.6.1])# redundancy-protocol
```

To stop running a redundancy protocol on an interface, enter:

```
(config-circuit-ip[VLAN2-172.7.6.1])# no redundancy-protocol
```

Configuring the VRRP Backup Timer

Configure the **vrrp-backup-timer** command on both redundant CSSs to specify the time interval in seconds that the backup CSS waits to assume mastership when the master CSS goes down. Because timer values greater than the 3-second default cause longer failover times, use this command only in environments where the CPU utilization on the CSS is close to 100 percent. After you set the timer value, you need to reissue the **redundancy-protocol** command on the redundant circuit between the CSSs for the new timer value to take effect. For details on configuring the redundancy protocol, see "Configuring the Redundancy Protocol" earlier in this chapter.

Note If you intend to use the commit_redundancy script to synchronize your redundant configuration, be sure to specify the -a argument in the **script play** command to ensure that the script copies the timer configuration setting from the master CSS to the backup CSS. For details on synchronizing your redundant configuration, see "Synchronizing a Redundant Configuration" later in this chapter.

The syntax for this global configuration mode command is:

vrrp-backup-timer *wait_time*

The variable for this command is *wait_time*. Enter an integer from 3 to 120 seconds. The default is 3 seconds.

For example:

```
(config)# vrrp-backup-timer 15
```

To reset the timer to the default value of 3 seconds, enter:

```
(config)# no vrrp-backup-timer
```

Synchronizing a Redundant Configuration

To ensure that your backup CSS can perform the same tasks as your master CSS in the event of a master CSS failure, the running-config on the backup must be identical to the running-config on the master. To automate this configuration synchronization process, you can run a script (commit_redundancy) on the master CSS that copies the master CSS running-config to the backup CSS running-config.

There are two types of configuration synchronization:

- **Complete** - On CSSs that have an identical chassis (the same CSS model), produces a running-config on the backup CSS that exactly matches the running-config on the master CSS.

- **Partial** (default) - On CSSs with an incompatible configuration syntax, synchronizes all parameter values in the configuration except the interface and circuit configurations. For example, the master is a CSS 11506 and the backup is a CSS 11150. The script maintains the current backup interface and circuit configurations automatically.

Before You Begin

Before you run the configuration synchronization script, ensure that you have set up the redundancy circuit between the two CSSs and that the Application Peering Protocol (APP) is running on that circuit. For details on configuring the redundancy circuit, see "Configuring Redundancy" earlier in this chapter. For details on configuring APP, refer to Chapter 1, Configuring the CSS Domain Name Service, in the section "Configuring Application Peering Protocol".

Running the Configuration Synchronization Script

To run the configuration synchronization script, use the **script play commit_redundancy** command. The syntax is:

script play commit_redundancy "*arguments***"**

You can also run the configuration synchronization script using the predefined alias that comes with all CSSs, as follows:

commit_RedundConfig "*arguments***"**

The arguments for the commit_redundancy script are:

- *ip address* - The IP address of the backup CSS. This is the only required argument for this script. For details on automating the entry of the IP address, see "Setting the BACKUP_IP Variable" later in this section.
- **-a** (All) - Performs a complete configuration synchronization. Use this option only when the master CSS and the backup CSS have identical chassis (see Table 7-3).
- **-d** (Debug) - Debug switch for the commit_redundancy script, which displays the current task being performed as the script progresses. Debug messages display even when you specify the **-s** argument.

Caution Before you use the **-f** argument to remove a config sync lock file, ensure that no one else is running the config sync script on the CSS. Otherwise, if you remove the lock file and then run the script again while the script is in use, the resulting configurations may have some discrepancies.

- **-f** - After an abnormal script termination, removes the lock file so that you can run the script again. This argument overrides all other specified arguments and the script exits immediately after removing the lock file. For details on the lock file, see "Setting the BACKUP_IP Variable" later in this section.
- **-int** (Interface) - Does not clear the interfaces on the backup CSS so that the link does not go down. Do *not* use this argument with the **-a** argument. If you do and the interface settings are different on the master and the backup CSSs, the configurations will not match and the script will not finish successfully.
- **-nv** (No Verify) - Informs the script not to verify that the configuration synchronization was successful.

Note The script verifies the configuration synchronization by default.

- **-s** (Silent) - Suppresses script progress messages and displays only the result of running the script: Config Sync Successful or Config Sync Failed.

Note You can specify the script arguments in any order.

For example:

```
CSS11503# script play commit_redundancy "10.7.100.93 -d -s"
```

The following output appears:

```
Verifying that IP redundancy is activated on Master switch.

Verifying that app session is up with backup switch.

Making sure app session is up.

Checking Master redundancy-config for redundancy-protocol set and
if so storing it in variable MASTER_IP.

Verify that the IP Address specified is the Backup IP Address.

Making sure app session is up.

Saving Master running-config to startup-config and archiving
startup-config.

Copying running-config to startup-config.

Archiving startup-config.

Copying startup-config to a temp file tmp.cfg.
Swapping Master and Backup ip addresses in tmp.cfg for app

Removing CIRCUIT and INTERFACE modes from tmp.cfg.

Checking if IP redundancy master is set.

Using rcmd to copy tmp.cfg to a file on Backup switch.

Retrieving circuit info for redundancy-protocol link.

Archiving copy to Backup startup-config.

Archiving Backup current startup-config.

Restoring startup-config (new copy) to startup-config.
```

```
Clearing running-config.

Script playing the copy script of the Master running-config.

Making sure app session is down.

Copy success being verified by comparing byte sizes of archived
running-configs of the Master switch and the Backup switch.

Making sure app session is up.

Comparing the byte count now.

Config Sync Successful.
```

In this example, the script:

- Performs a partial configuration synchronization (default)
- Displays the current task being performed as the script progresses (-d)
- Suppresses progress messages (-s)
- Verifies that the configuration synchronization was successful

For more information on scripts, refer to Chapter 11, Configuring Redundant Content Services Switches.

Config Sync Lock File

When you run the script, the software creates a lock file (config_sync_lock) in the script directory so that you cannot run the script from another session to the CSS. If the lock file exists and you run the script, the following message appears:

```
The script is in use by another session.
```

If the script terminates abnormally, the software does not remove the lock file. The next time you run the script, the above message appears. If you are certain that the script is not in use by another session, then you can use the -f argument to remove the lock file. When you run the script with this argument, the following message appears and the script exits:

```
Config Sync lock file removed.
```

Now you can run the script again.

Setting the BACKUP_IP Variable

To eliminate the need to specify an IP address each time you run the configuration synchronization script, you can set the value of a variable (BACKUP_IP) to an IP address and save it in your user profile. Once you set the variable and save it in your user profile, the variable will always be available after you log in to the CSS.

To set the BACKUP_IP variable, enter:

```
# set BACKUP_IP "ip_address" {session}
```

where, *ip_address* is the IP address of the backup CSS.

To save the variable in your user profile, enter:

```
# copy profile user-profile
```

Now you can run the configuration synchronization script without typing an IP address.

Logging Configuration Synchronization Script Result Messages

You can specify that script result messages (script success or failure messages) be sent to the current logging device automatically each time you run the configuration synchronization script. To log the script result messages, enable logging on NETMAN with level info-6 or debug-7 by entering:

```
(config)# logging subsystem netman level info-6
```

Note Log messages are generated with or without the -s (silent) argument specified. See "Running the Configuration Synchronization Script" earlier in this chapter.

For example, if the APP session to the backup CSS is not running, the following log message will be generated:

```
config sync: app session is DOWN
```

For ease of tracking, each log message contains the string "config sync".

Using the Redundancy Force-Master Command

Use the **redundancy force-master** command to configure a backup CSS as a master *temporarily*. This is a temporary setting because it is not copied to the running-config. This command is useful in a redundant configuration when you need to take the master CSS offline for maintenance or an upgrade.

By issuing the **redundancy force-master** command on the backup CSS in global configuration mode, you set that CSS to master and ensure that users have continuous access to servers and content. The forced master CSS remains the master:

- Until it goes down and comes back up as the backup, or
- You manually make the other CSS the master, using either the **redundancy force-master** command or the **ip redundancy master** command.

Note If you explicitly designated the master CSS using the **ip redundancy master** command, you cannot use the **redundancy force-master** command on the backup CSS. In this case, you must unassign the master CSS by issuing the **no ip redundancy master** command before you can use the **redundancy force-master** command on the backup CSS.

Configuring Multiple Redundant Uplink Services

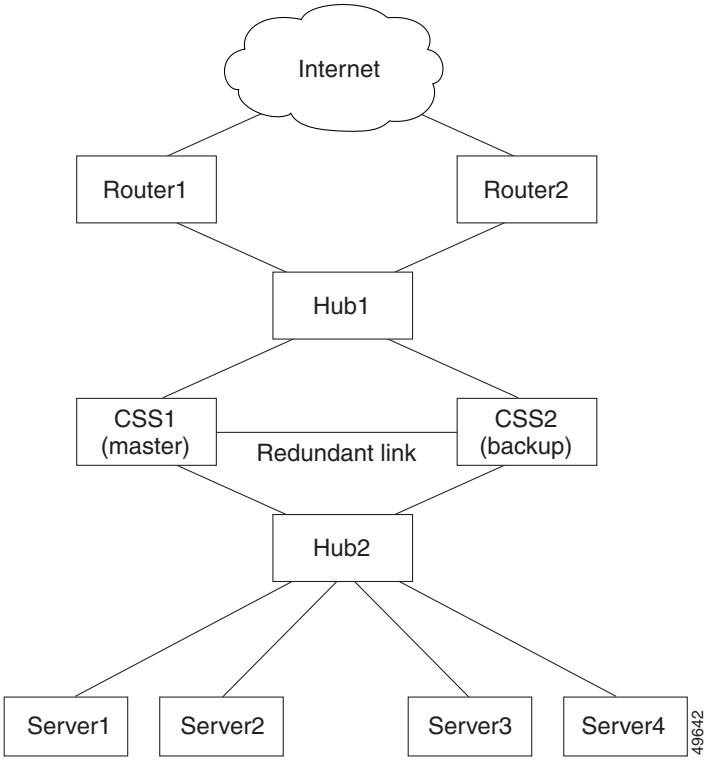
Within a redundant configuration, the CSS allows you to configure multiple redundancy uplink critical services (up to a maximum of 512). Use the **type redundancy-up** command to designate one or more routers as type redundancy-up critical services. (A typical configuration contains 10 or fewer routers.) This critical service type enables the master CSS to ping a router service using the default keepalive Internet Control Message Protocol (ICMP). If the master CSS fails or it detects that all router uplink critical services have failed, the backup CSS becomes the master.

In a redundant configuration that does not configure the routers as type redundancy-up critical services, a backup CSS becomes master only when the current master CSS fails. In this configuration, a switchover *does not* occur when the router services fail.

Note You cannot add redundancy uplink critical services to a content rule.

Figure 7-2 shows a typical redundant configuration. When CSS1 fails or CSS1 cannot communicate with both the Router1 critical service and the Router2 critical service, CSS2 becomes the master CSS automatically.

Figure 7-2 Multiple Redundant Uplink Services Configuration Example



Note If you explicitly designated the master CSS using the **ip redundancy master** command, you cannot use the **type redundancy-up** command on the CSS. In this case, you must unassign the master CSS by issuing the **no ip redundancy master** command before you can use the **type redundancy-up** command.

Use the **type redundancy-up** command to configure each router service as a redundancy uplink critical service. For example:

```
(config-service[router1])# type redundancy-up
(config-service[router1])# ip address 192.168.1.1
(config-service[router1])# active
```

Use the **show critical services** command to display critical services. For example:

```
CSS1(config)# show critical services
```

Using the redundancy-phy Command

Use the **redundancy-phy** command in interface mode to add an interface to the physical link configuration list. If any physical link in the configuration list goes down, the master CSS fails over to the backup CSS. For an 11500 series CSS, you can configure a maximum of 32 interfaces. For an 11000 series CSS, you can configure a maximum of five interfaces. The CSS saves this configuration information to the running-config.

Note You cannot use the **redundancy-phy** command if you used the **ip redundancy master** command to configure the master CSS. In this case, you must issue the **no ip redundancy master** command before you can use the **redundancy-phy** command.

To disable a configured interface and delete it from the physical link list, enter:

```
(config-if)# no redundancy-phy
```

Note When you use the redundancy-phy command and both CSSs are connected to a Layer 2 switch, be sure to monitor physical link failure only on the critical physical links and not on the redundant link between the two CSSs. This will avoid the detection of a physical link down and possible thrashing when one of the CSSs is rebooting or transitioning between master and backup states.

Configuring Stateless Redundancy Failover

Use the **redundancy-l4-stateless** command in content or group configuration mode to enable stateless redundancy failover in an IP redundancy or VIP/interface redundancy configuration. Stateless redundancy failover allows critical TCP/IP traffic to continue in case of a failure at the load-balancing CSS by allowing the backup CSS to set up a mid-stream TCP flow. This feature is disabled by default.

The default behavior of a CSS is to set up load-balanced TCP flows only when it receives a TCP frame that begins with SYN. When stateless redundancy failover is enabled and a failover occurs, the backup CSS establishes a mid-stream flow for any existing TCP sessions. The CSS still exhibits the default behavior for all new flows. To restore the default behavior of the CSS for all flows after issuing the **redundancy-l4-stateless** command, use the **no redundancy-l4-stateless** command.

Note This feature affects only TCP/IP sessions. UDP behaves normally because UDP is not a a session-oriented protocol.

Before you begin

Before you attempt to implement stateless redundancy failover for the first time, read this section in its entirety. You should already be familiar with the following concepts:

- Redundancy
- Layer 3 and layer 4 content rules
- Virtual IP addresses (VIPs)
- Load balancing
- Source groups
- Services
- Keepalives
- Convergence

Environment Considerations

Stateless redundancy failover requires a very specific redundant CSS configuration, where the state of the CSS can be determined after a failure. This feature supports redundant routes in the high availability topology surrounding the CSSs. However, the topology must *not* balance packets in a TCP/IP socket connection across more than one Ethernet port on the CSS.

Routed paths to the load-balanced VIP should be weighted to ensure that a single path is preferred for the lifetime of a TCP/IP connection.

Note Stateless redundancy failover does not support network address translation (NAT) where maintaining state is required nor does it support Layer 5 content rules.

General Configuration Requirements

The following sections describe the stateless failover requirements that apply to both IP redundancy and VIP/interface redundancy configurations.

Configuration Restrictions

The following configuration restrictions apply to all CSSs, except where noted.

- Stateless redundancy failover is incompatible with service remapping (the **persistence reset remap** command). Stateless redundancy failover requires that the CSS not NAT client source ports. Backend remapping enables CSS portmapping, which NATs source ports for all flows. For more information on service remapping, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 3, Configuring Content Rules.
- Configuring session-level redundancy and stateless redundancy failover on the same CSS is not supported.
- For an 11500 series CSS, do not configure a service that changes the destination port on a content rule. This causes the CSS to NAT (portmap) the destination port. If the CSS fails over, the backup CSS has no knowledge of the original destination port.

Configuring CSS Parameters

The following parameters must match exactly on both redundant CSSs:

- **Stateless redundancy failover command -** Include the **redundancy-l4-stateless** command in both the content rules and the source groups associated with the redundant VIP.
- **Content rules** - Create identical content rules on both CSSs with the following parameters. Refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 3, Configuring Content Rules.
  - **VIP** - Assign a virtual IP address to each content rule. No wildcard addresses are allowed and no VIP ranges on the content rule are allowed.
  - **Load-balancing method** - Configure the load-balancing method as source IP address (srcip), the only load-balancing method that is supported by stateless redundancy failover.
  - **Failover method** - Configure either 'linear' (default) or 'next' type as the service failover method. 'Bypass' is not supported.
- **Services** - For each load-balanced server farm, configure the following service-related parameters to be the same on both CSSs for each content rule. Refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 1, Configuring Services.
  - **Service name** - Use identical service names on the master and the backup CSS. Service names are case-sensitive.
  - **IP address** - Use identical IP addresses on the master and the backup CSS.

Note Configured services may not change the CSS destination port. In a stateless environment, there is no way to determine the original destination port when the packet returns from the server.

  - **Service number and order** - The CSS orders services internally in alphabetical order regardless of the order in which you enter them in the configuration.

  - **Keepalives** - Create keepalives using the global **keepalive** command, then associate the services with the keepalives using the **keepalive type named** command. Both CSSs must be able to send and receive keepalive messages with the same servers. This helps to ensure that a redistribution of the balance method does not occur in a failover event.
  - **Weight** - Routed paths to the load-balanced VIP should be weighted to ensure that a single path is preferred for the lifetime of a TCP/IP connection.

- **Source groups** - Create a source group with the same VIP as the content rule VIP on each CSS to NAT source addresses for packets returning from the server. In case of a failover, the source group will handle the connection setup for TCP/IP transmissions that arrive at the CSS from the servers. All servers in the farm must be members of the configured source group. Refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 5, Configuring Source Groups, ACLs, EQLs, URQLs, NQLs, and DQLs.

Note Do not configure source groups for outbound traffic from the servers, because the backup CSS does not know which ports were NATed by the source group on the master CSS if a failure occurs at the master CSS.

Synchronizing the CSS Configurations

You can manually synchronize the CSS configurations by ensuring that the configurations are exactly the same. In an IP redundancy configuration, you can run the commit_redundancy configuration synchronization script. The script automatically synchronizes the master and backup CSS configurations. See "Synchronizing a Redundant Configuration" earlier in this chapter.

IP Redundancy Configuration

For details on IP Redundancy, refer to the earlier sections in this chapter.

In case of a failure on the master CSS, the backup CSS becomes the master CSS. The following actions occur:

1. All VLANS become active.
2. Topology protocols (for example, Spanning Tree) initialize and converge.
3. All configured interface (circuit) and VIP addresses are acquired by gratuitous ARP.
4. The master CSS acquires servers through keepalives.

Note If your configuration is large or the servers respond slowly, the completion of step 4 may take several seconds.

Complex topologies surrounding the CSS converge after the CSS has determined a root bridge and has begun transmission of Address Resolution Protocol (ARP) and keepalive traffic. If a TCP/IP retransmission from a server arrives at the CSS before the CSS acquires the server, the CSS sets up the connection properly through the configured source group path. If a retransmission from a client arrives at the CSS before all servers have been acquired and the source IP address of the client indicates a server that is not yet alive, the CSS sets up the connection according to the failover method configured in the content rule (next or linear).

L2 and L3 Configuration and Convergence

Because IP Redundancy disables the forwarding of traffic through VLANs on the backup CSS, configure the CSS to provide either a bridged or routed path between servers and uplink routers. In either case, the CSS must be the default gateway for load-balanced servers. Refer to the *Cisco Content Services Switch Administration Guide*, Chapter 2, Configuring Interfaces and Circuits.

If you configure the CSSs with servers and a balance VIP on the same VLANs as the uplink router (bridged mode), then configure the CSSs to not send ICMP redirects to the servers, using the **no redirect** command. Refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 3, Configuring Content Rules.

Example Configuration

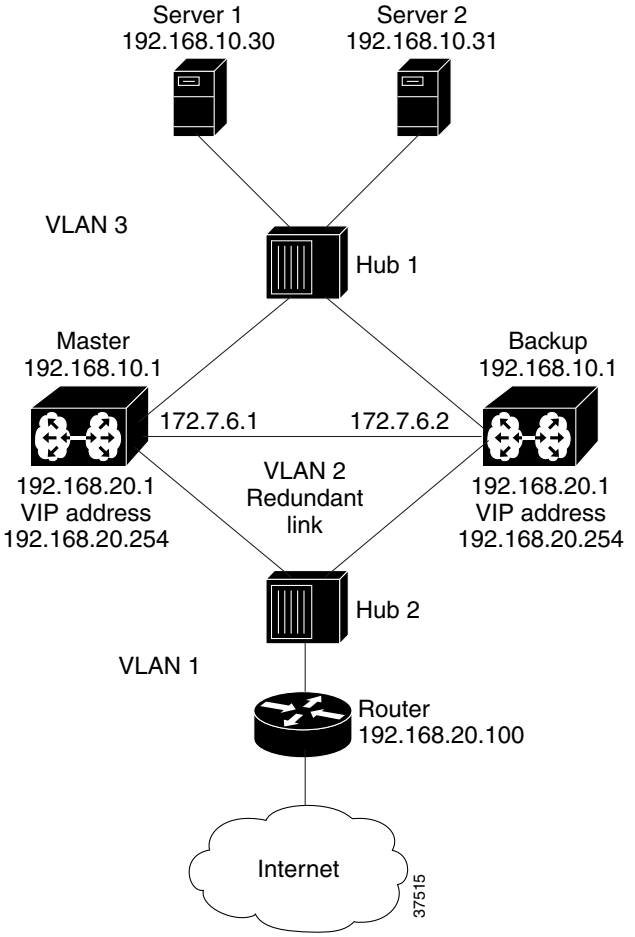
The following example configuration (see Figure 7-3) assumes that the:

- CSSs are acting as routers between two external VLANs in an IP Redundancy configuration.
- External VLANS exist on L2 and L3 devices.
- L2 and L3 devices are not the point of failure.

```
ip route 0.0.0.0.0.0.0.192.168.20.100
ip redundancy

interface e2
    bridge vlan 1
    description "uplink VLAN"

interface e5
    bridge vlan 1
    description "uplink VLAN"

interface e9
    bridge vlan 3
    description "server VLAN"

interface e12
    bridge vlan 3
    description "server VLAN"

interface e1
    bridge vlan 2
    description "Redundancy Protocol Heartbeat"
circuit VLAN1
    redundancy
    ip address 192.168.20.1 255.255.255.0

circuit VLAN3
    redundancy
    ip address 192.168.10.1 255.0.0.0
```

```
circuit VLAN2
    redundancy-protocol
    ip address 172.7.6.1 255.255.255.253

service s1
    ip address 192.168.10.30
    active

service s2
    ip address 192.168.10.31
    active

owner Redundant-Pool
    content web
        vip address 192.168.20.254
        protocol tcp
        port 80
        redundancy-l4-stateless
        add s1
        add s2
        balance srcip
        active

group Redundant-Pool
    vip address 192.168.20.254
    redundancy-l4-stateless
    add service s1
    add service s2
    active
```

Figure 7-3 Example IP Redundancy Configuration for Stateless Redundancy Failover



VIP/Interface Redundancy Configuration

For details on VIP/Interface Redundancy, refer to Chapter 6, Configuring VIP and Virtual IP Interface Redundancy.

L2 and L3 Configuration and Convergence

A CSS that is running Virtual Router Redundancy Protocol (VRRP) does not shut down any VLANs. Therefore, VRRP configurations may not be configured with content rule VIP, uplink, and server addresses in the same VLAN (bridged mode). Instead, the CSSs must be configured so that both CSSs in a redundant pair act as routers between the uplink VLAN and the server VLAN. The CSS uses a virtual router address for the default gateway on the servers.

Because both CSSs are active and participating in topology protocols, convergence time may be reduced in the event of a failure. Additionally, both CSSs acquire servers with keepalive traffic at all times, so that both CSSs agree on server availability.

VRRP provides for a redundant routed path out of a VLAN, but does not address synchronization of more than one VLAN in the decision. Due to this limitation, ensure that one CSS does not become master for the connection to the uplink VLAN, while the other CSS is master for the connection to the server VLAN. To avoid this split state, a CSS can monitor critical external IP addresses as part of the extended VRRP implementation.

Typically, you configure a single CSS with the highest priority and the **preempt** option for each VRID pair (uplink VLAN side/server VLAN side). This ensures that if the designated CSS is available, both VRIDs will converge there, avoiding a split state.

To address more complex failure scenarios, use a script keepalive. For details on script keepalives, refer to the *Cisco Content Services Switch Administration Guide*, Chapter 1, Configuring Services.

In the example that follows, the master CSS relinquishes control of both virtual interfaces upon loss of contact with either the uplink router or all web servers.

Example Configuration

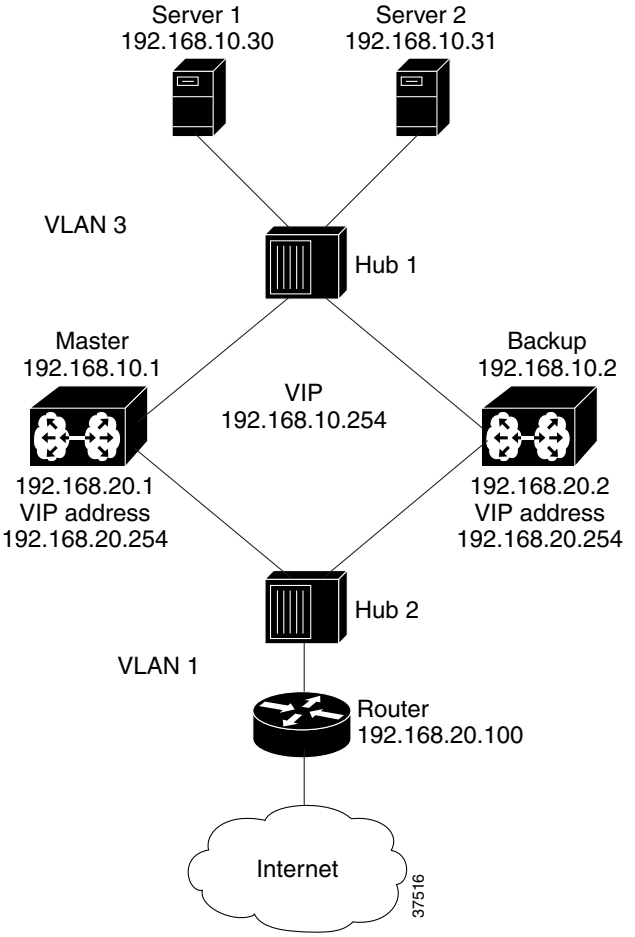
The following example configuration (see Figure 7-4) assumes that the:

- CSSs are acting as routers between two external VLANs in a VIP/interface Redundancy configuration.
- External VLANS exist on L2 and L3 devices.
- L2 and L3 devices are not the point of failure.

```
ip route 0.0.0.0 0.0.0.0 192.168.20.100

interface e2
    bridge vlan 1
    description "uplink VLAN"

interface e5
    bridge vlan 1
    description "uplink VLAN"

interface e9
    bridge vlan 3
    description "server VLAN"

interface e12
    bridge vlan 3
    description "server VLAN"

circuit VLAN1
    ip address 192.168.20.1 255.255.255.0
        ip virtual-router 1 priority 110 preempt
        ip redundant-vip 1 192.168.20.254
        ip redundant-interface 1 192.168.20.2
        ip critical-service 1 uplink
        ip critical-service 1 s1
        ip critical-service 1 s2

circuit VLAN3
    ip address 192.168.10.1.0.0.0
        ip virtual-router 1 priority 110 preempt
        ip redundant-vip 1 192.168.10.254
        ip redundant-interface 1 192.168.10.2
        ip critical-service 1 uplink
        ip critical-service 1 s1
        ip critical-service 1 s2
```

```
service uplink
    ip address 192.168.20.100
    type redundancy-up
    active

service s1
    ip address 192.168.10.30
    active
service s2
    ip address 192.168.10.31
    active

owner Redundant-Pool
    content web
        vip address 192.168.20.254
        protocol tcp
        port 80
        redundancy-l4-stateless
        add s1
        add s2
        balance srcip
        active

group Redundant-Pool
    vip address 192.168.20.254
    redundancy-l4-stateless
    add service s1
    add service s2
    active
```

Figure 7-4 Example VIP/Interface Redundancy Configuration for Stateless Redundancy Failover



Alternative Configurations

Stateless redundancy failover allows other possible configurations and topologies. To use this feature in other high-availability environments, see the other sections in this chapter and to Chapter 6, Configuring VIP and Virtual IP Interface Redundancy, for details and examples of CSS redundancy configurations. Refer to RFC-2338 *Virtual Router Redundancy Protocol* for additional information.

Managing Your Configuration

If you need to take a server offline for maintenance, you should also take the corresponding server offline at the redundant CSS. Failing to synchronize the state of the server farms results in mismapped connections if a failover occurs during the maintenance period. You can synchronize the service states in an IP redundancy configuration automatically by running the configuration synchronization script (commit_redundancy). For a VIP/interface redundancy configuration, manually synchronize the service states.

Other Considerations

The following conditions apply to stateless redundancy failover:

- After a failover, passive mode FTP will not continue because the NAT state of the data channel cannot be preserved. However, port mode FTP will continue to function.
- Because the portmap function of source groups is disabled, connections originated by the servers in the redundantly balanced farm do not have the benefit of source port translation. This may affect functions such as DNS.
- Service records may not be configured to change the destination port of traffic that is balanced.

- At any given time, some TCP/IP connections may be either in a state where the client is sending data to the server, or the server is sending data to the client. Packets that arrive while the topology is converging or before some services are acquired by keepalive traffic may be forwarded incorrectly. For example, a service may stop responding to keepalive traffic, but continue to service a long-lived TCP connection. In this case, the backup CSS would not have knowledge of the state of the long-lived connection, and would guess incorrectly when attempting to resume the connection.
- In a highly critical environment, set goals for connection loss ratio and convergence time. Then, test various topologies and topology protocol combinations to verify that the target connection loss ratios and convergence time goals are reached. This testing should account for all reasonable failure modes that the high-availability network is designed to withstand. If warranted by the critical nature of the traffic, you may want to construct a permanent testbed to validate the system of network components prior to the deployment of new configurations.

Displaying Redundant Configurations

To display CSS-to-CSS redundancy, use the **show redundancy** command.

For example:

```
(config)# show redundancy
```

When redundancy is not configured, the CSS displays the following status:

```
(config)# show redundancy
Redundancy: Disabled Redundancy Protocol: Not Running
```

The output of the show redundancy command varies depending on whether you issue the command on the master or the backup CSS.

Table 7-3 describes the fields in the **show redundancy** output.

Table 7-3 Field Descriptions for the show redundancy Command

Field	Description
Redundancy	Indicates whether or not redundancy is enabled on the CSS.
Redundancy Protocol	Indicates whether or not the redundancy protocol is running on the CSS.
Redundancy State	The current redundancy state of the CSS (Master or Backup).
MasterMode	Indicates whether or not the CSS is configured as master. Yes indicates that the CSS is designated as master through the **ip redundancy master** command. No indicates that the CSS is not the designated master through the **ip redundancy** command.
Number of times redundancy state changed to Master/Backup	The number of times that the CSS has changed to master or backup.
Redundancy interface	The address for the redundancy interface.
Current State Duration	How long the CSS has been in its current redundancy state (master or backup).
Last Fail Reason	A description of the last CSS redundancy failure.
VRID	The virtual router identifier (VRID).
Priority	The priority for the virtual router with its peer. The default priority value is 100. Enter an integer between 1 and 255.
Physical Link Failure Monitor on	
Interface/State	The list of interfaces configured through the **redundancy-phy** command and their states.
Uplink Enabled	The number of enabled service uplinks.
Number Alive	The number of alive (Up State) service uplinks.
Service Name/State	The list of uplink services and their states.

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 are part of the CSS Enhanced feature set.

The information in this chapter applies to all CSS models, except where noted.

This chapter contains the following sections:

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.

Demand-based content replication is traffic-based. Increases in the flow of traffic launch replication services automatically. When you configure demand-based content replication, the CSS automatically:

1. Uses hotlists to detect hot content when the URL hits or bandwidth exceeds the configured hotlist 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:

- Configuring Hotlists
- Specifying Service Type for Replication
- Configuring Max Age
- Configuring Max Content
- Configuring Max Usage
- Configuring FTP Access for Content Replication
- Creating an FTP Record

Configuring Hotlists

Use the **hotlist** command to define a hotlist that lists the content most requested (hot content) during a user-defined period of time. The CSS enables you to configure hotlist attributes for content rules. Defining hotlist 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.

Note You must configure and enable a hotlist for service types replication-store and replication-cache to work.

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

- **hotlist** - Enables the hotlist. To enable a hotlist 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 hotlist, enter:

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

- **hotlist interval** - Sets the hotlist 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 hotlist interval to the default of 1, enter:

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

- **hotlist size** - Sets the size of the hotlist. 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 hotlist size to the default of 10, enter:

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

- **hotlist threshold** - Sets the hotlist 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 hotlist threshold default of 0, enter:

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

- **hotlist type hitCount -** Sets the hotlist 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 hotlist 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 to which content requests are redirected. No content rules are applied to requests from this service type.

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.

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
- 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

Use the **max age** command to define the maximum age for replicated objects on services defined as type **rep-cache-redir**, **rep-store**, or **rep-store-redir**. 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

Use the **max content** command to define the maximum pieces of content for replication on services defined as type **rep-cache-redir**, **rep-store**, or **rep-store-redir**. 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

Use the **max usage** command to define the maximum disk space allowed for replication on services defined as type **rep-cache-redir**, **rep-store**, or **rep-store-redir**. Enter the disk space for a service from 1 to 1000 megabytes. 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

Use the **access ftp** command to 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.

When you use this command to associate an FTP access mechanism to 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.

Note To create an FTP record, use the **(config) ftp-record** command. For more information on creating an FTP record, see "Creating an FTP Record" later in this chapter.

For example:

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

To remove a service access mechanism, enter:

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

Note Content replication does not support the WSFTP FTP application.

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***"**|**encrypted-password** *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.
- **encrypted-password** *encrypted_password* - The 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 16 characters.
- *base_directory* - An optional base directory when using this record.

For example:

```
(config)# ftp-record ftp1 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 ftp1
```

Configuring Content Staging and Replication

The CSS supports content staging and replication using Publisher and Subscriber services. Content staging and replication is a timer-based replication service. At a pre-configured day and time, 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:

- CLI commands.
- Time of day using the Command Scheduler feature.
- 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 file names, 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.

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:

Configuring FTP Access for Publishing and Subscribing

Use the **access ftp** command to associate an access mechanism with a service for use during publishing and subscribing activities. You must use this command for each service that offers publishing services and for each service that you configure as a subscriber.

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

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, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 1, Configuring Services.

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.

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.

- **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 8-1 describes the fields in the **show publisher** output.

Table 8-1 Field Descriptions for the show publisher Command

Field	Description
State	The state of the publisher service.
Access Type	The associated access mechanism with a service for use during publishing activities. Currently, the FTP record is the only mechanism.
Access IP	The IP address for the FTP record.
Access Port	The port number for the FTP record associated with the access mechanism.
Access Username	The username for the FTP server as defined through the FTP record.
Access Base Dir	The base directory as defined through the FTP record.
Published Files	The number of files published from the publisher to the subscriber.
Published Bytes	The number of bytes published from the publisher to its subscribers.
Subscribers	The number of subscribers configured to use the publisher.
Trigger File	The file upon modification that causes the synchronization between the publisher and the subscriber.
Publish Interval	The interval in seconds when the publisher checks for subscriber synchronization.
Next Interval	The time when the next publisher synchronization check will occur.
Managed Files	The number of files that the publisher will replicate.
Subscribers Synced	The number of synchronized subscribers.
Managed Dirs	The number of files that the publisher will replicate.
Managed Bytes	The number of bytes that the publisher is tracking.

Table 8-1 Field Descriptions for the show publisher Command (continued)

Field	Description
Last Method	The last method that caused the publisher to attempt synchronization with the subscriber. The synchronization methods are: • **cli** - User initiated • **interval** - The configured time interval • **signal** - Trigger file change • **retry** - Retry when a publisher failed to synchronize previously • **reboot** - CSS reboot
Last Time	The last time when the publisher attempted to synchronize with the subscriber.

Configuring a Subscriber Service

Use the **subscriber** command to configure a service as a subscriber to a publishing service. 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 8-2 describes the fields in the **show subscriber** output.

Table 8-2 Field Descriptions for the show subscriber Command

Field	Description
State	The state of the subscriber.
Access Type	The FTP access mechanism with a service for use during subscribing activities.
Access IP	The IP address for the FTP record associated with the access mechanism.
Access Port	The port number for the FTP record associated with the access mechanism.
Access Username	The username for the FTP record associated with the access mechanism.
Access Base Dir	The base directory for the FTP record associated with the access mechanism.
Subscribed Files	The number of files replicated on the subscriber.
Subscribed Bytes	The number of bytes replicated on the subscriber.

Table 8-2 Field Descriptions for the show subscriber Command (continued)

Field	Description
Last Method	The last method that caused the publisher to attempt synchronization with the subscriber. The synchronization methods are: • **cli** - user initiated • **interval** - the configured time interval • **signal** - trigger file change • **retry** - retry when a publisher failed to synchronize previously • **reboot** - CSS reboot
Last Time	The last time when the publisher attempted to synchronize with the subscriber.
Synchronized	Indicates whether or not the subscriber is currently synchronized with the publisher.

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 on configuring URLs, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 5, Configuring Source Groups, ACLs, EQLs, URQLs, NQLs, and DQLs in the section, "Configuring Uniform Resource Locator Qualifier Lists".

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

Use the **replicate** command to start replicating content between a publisher and all associated subscribers. You can use this command to replicate content to new subscribers or force resynchronization immediately.

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.

For example:

```
# replicate pubserver
```

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 8-3 describes the fields in the **show content** output.

Table 8-3 Field Descriptions for the show content Command

Field	Description
Pieces of Content for Slot	The chassis slot number in which the module resides.
Subslot	The module slot number in which the Session Processor resides.
Total Content	The total number of content entries.
Index	Unique index for a known piece of content in the CSD.
<address>	The IP address of the piece of content.
Protocol	The IP Protocol of the piece of content.
Port	Protocol port of the piece of content.
Best Effort	The QoS class of the piece of content. This field is not used by the CSS at this time.
Streamed	Identifies if the piece of content is streaming media (video or audio). This field is not used by the CSS at this time.
URL	The Universal Resource Locator of the piece of content.
Domain	The domain name of the piece of content.

Configuring SSL Termination on the CSS

This chapter describes how to configure the CSS and the SSL Acceleration Module to perform Secure Sockets Layer (SSL) termination between a client and servers. It also provides an overview on SSL as implemented by the SSL Acceleration Module in the CSS. Information in this chapter applies only to the Cisco CSS 11503 and CSS 11506 containing an SSL Acceleration Module (hereinafter referred to as the SSL module).

Note The SSL feature is intended for use with the optional SSL module.

This chapter contains the following sections:

SSL Cryptography Overview

Secure Sockets Layer (SSL) is an applications-level protocol that provides encryption technology for the Internet ensuring secure transactions, such as the transmission of credit card numbers for e-commerce websites. SSL provides the secure transaction of data between a client and a server through a combination of privacy, authentication, and data integrity. SSL relies upon certificates, private-public key exchange pairs, and Diffie-Hellman key agreement parameters for this level of security.

The CSS uses the SSL module and a special set of SSL commands to perform the SSL cryptographic functions between the client and the HTTP servers. The SSL functions include user authentication, private-key and public-key generation, certificate management, and data packet encryption and decryption.

A typical SSL session with the SSL module requires encryption ciphers to establish and maintain the secure connection. Cipher suites provide the cryptographic algorithms required by the SSL module to perform key exchange, authentication, and Message Authentication Code (MAC). See "Specifying Cipher Suites" in this chapter for details about the supported cipher suites.

This section provides an overview on SSL cryptography as implemented through the SSL module in the CSS. It covers:

- SSL Public Key Infrastructure Overview
- SSL Module Cryptography Capabilities

Note The SSL module supports SSL version 3.0 and Transport Layer Security (TLS) version 1.0. The module understands and accepts an SSL version 2.0 ClientHello message to allow dual version clients to communicate with the CSS through the SSL module. In this case, the client indicates an SSL version of 3.0 in the version 2.0 ClientHello, which informs the SSL module that the client can support SSL version 3.0. The SSL module returns a version 3.0 ServerHello message.

Although there are very few clients on the market today that support only SSL version 2.0, if the client supports only version 2.0 the SSL module will be unable to pass network traffic.

SSL Public Key Infrastructure Overview

SSL provides authentication, encryption, and data integrity in a Public Key Infrastructure (PKI). PKI is a set of policies and procedures to establish a secure information exchange between devices. Three fundamental elements characterize the PKIs used in asymmetric cryptography. These three elements provide a secure system for deploying e-commerce and a reliable environment for building virtually any type of electronic transactions, from corporate intranets to Internet-based e-business applications.

These elements include:

- Confidentiality
- Authentication
- Message Integrity

Confidentiality

Confidentiality means that unintended users cannot view the data. In PKIs, confidentiality is achieved by encrypting the data through a variety of methods. In SSL, specifically, large amounts of data are encrypted using one or more symmetric keys that are known only by the two endpoints. Because the symmetric key is usually generated by one of the endpoints, it must be transmitted securely to the other endpoint. Secure transmittal of a symmetric key is generally achieved by two mechanisms, *key exchange* or *key agreement*.

Key exchange is the most common of these two secure transmittal mechanisms. In key exchange, one device generates the symmetric key and then encrypts it using an asymmetric encryption scheme before transmitting it to the other side. Asymmetric encryption requires both devices have a public key and a private key. The two keys are mathematically related; data that can be encrypted by the public key can be decrypted by the private key, and vice versa. The most commonly used key exchange algorithm is the Rivest Shamir Adelman (RSA) algorithm.

For SSL, the sender encrypts the symmetric keys with the public key of the receiver. This ensures that the private key of the receiver is the only key that can decrypt the transmission. The security of asymmetric encryption depends entirely on the fact that the private key is only known by the owner and not by any other party. If this key were compromised for any reason, a fraudulent Web user (or website) could decrypt the stream containing the symmetric key and could decrypt the entire data transfer.

In *key agreement*, the two sides involved in a data exchange cooperate to generate a symmetric (shared) key. The most common key agreement algorithm is the Diffie-Hellman algorithm Diffie-Hellman depends on certain parameters to generate the shared key that is calculated and exchanged between the client and the server.

Authentication

Authentication is necessary for one or more devices in the exchange to verify that the party to whom they are talking to is really who they claim to be. For example, assume a client is connecting to an e-commerce website. Before sending sensitive information such as a credit card number, the client verifies that the server is an e-commerce website. In certain instances, it may be necessary for both the client and the server to authenticate themselves to each other before beginning the transaction. In a financial transaction between two banks, both the client and the server need to be confident that the other is who they say they are. SSL facilitates this authentication through the use of digital certificates.

Digital certificates are a form of digital identification to prove the identity of the client to the server. A Certificate Authority (CA) issues digital certificates in the context of a PKI, which uses public-key and private-key encryption to ensure security. CAs are trusted authorities who "sign" certificates to verify their authenticity. The client or servers connected to the CSS must have trusted certificates from the same CA, or from different CAs in a hierarchy of trusted relationships (for example, "A" trusts "B," and "B" trusts "C," therefore "A" trusts "C").

A certificate ensures that the identification information is correct, and that the public key actually belongs to that client or server. Digital certificates contain information such as details about the owner, details about the certificate issuer, the owner's public key, validity and expiration dates, and associated privileges.

Upon receiving a certificate, a client can connect to the certificate issuer and verify the validity of the certificate using the issuer's public key. This ensures that the certificate is actually issued and signed by an authorized entity. A certificate remains valid until it expires or is terminated.

Message Integrity

Message integrity is a means of assuring the recipient of a message that the contents of the message have not been tampered with during transit. SSL achieves this by applying a message digest to the data before transmitting it. A message digest is a function that takes an arbitrary length message and outputs a fixed-length string that is characteristic of the message.

An important property of the message digest is that it is extremely difficult to reverse. Simply appending a digest of the message to itself before sending it is not enough to guarantee integrity. An attacker can change the message and then change the digest accordingly. Encoding the message digest with the sender's private key creates a Message Authentication Code (MAC), the message integrity algorithm, which the recipient can then decode using the sender's public key. SSL supports two different algorithms for a MAC: Message Digest 5 (MD5) and Secure Hash Algorithm (SHA).

This integrity scheme, however, does not work if the sender's private key is compromised. The attacker can now forge the sender's MACs. Message integrity also depends heavily on the protection of private keys. This process is known as digital signing.

RSA key pairs are effective for signing the MAC. However, it may be advantageous to separate the functions of key exchange and signing. The Digital Signature Algorithm (DSA) is an SSL algorithm that is used strictly for digital signatures but not for key exchange.

DSA was standardized as FIPS-186, which is the Digital Signature Standard (DSS). DSA and DSS can be used interchangeably. DSS uses the same crypto-math as Diffie-Hellman and requires parameters similar to Diffie-Hellman to generate keys. Additionally, DSS is restricted for use only with the Secure Hash Algorithm 1 (SHA-1) message digest.

SSL Module Cryptography Capabilities

Table 9-1 provides information on the SSL cryptography capabilities of the SSL module.

Table 9-1 SSL Module SSL Cryptography Capabilities

SSL Cryptography Function	Functions Supported by the SSL Module
SSL versions	SSL version 3.0 and Transport Layer Security (TLS) version 1.0
Public key exchange and key agreement algorithms	• **RSA** - 512-bit, 768-bit, 1024-bit, and 2048-bit (key exchange and key agreement algorithm) • **DSA** - 512-bit, 768-bit, 1024-bit, and 2048-bit[1] (certificate signing algorithm) • **Diffie-Hellman** - 512-bit, 768-bit, 1024-bit, and 2048-bit (key agreement algorithm)
Encryption types	• Data Encryption Standard (DES) • Triple-Strength Data Encryption Standard (3DES) • RC4 Refer to Table 9-12 for a list of supported cipher suites and key encryption types.
Hash types	• SSL MAC-MD5 • SSL MAC-SHA1 Refer to Table 9-12 for a list of supported cipher suites and hash types.

Table 9-1 SSL Module SSL Cryptography Capabilities (continued)

SSL Cryptography Function	Functions Supported by the SSL Module
Digital certificates	The SSL module supports all major digital certificates from Certificate Authorities (CAs), including those listed below: • VeriSign® • Entrust® • Netscape® iPlanet™ • Windows® 2000 Certificate Server • Thawte® • Equifax™ • Genuity™

1. The SSL module supports the importing of 2048-bit DSA key pairs. The SSL module does not support the generation of 2048-bit DSA key pairs.

SSL Module Termination Overview

The SSL module installs in the CSS chassis and operates as a virtual SSL server by adding security services between the Web browsers (the client) and HTTP servers. The 11500 series CSS supports multiple SSL modules, up to two in a CSS 11503 and up to four in a CSS 11506. Each SSL module terminates SSL connections received from a client.

To terminate SSL flows, the SSL module functions as a proxy server, which means that it is the TCP endpoint for inbound SSL traffic. The SSL module maintains a separate TCP connection for each side of the communications, the client side and the server side. The proxy server can perform both TCP and SSL handshakes.

Once the connection is terminated, the SSL module decrypts the data and transmits the data as clear text to the server. On the outbound flow from the CSS, the SSL module encrypts the data received from the server and sends the data back to the client.

SSL termination from the client and initiation of HTTP connections to servers requires the use of an SSL proxy list to configure the flow of information to and from an SSL module. You add an SSL proxy list to an SSL service to define how an SSL module processes SSL requests for content.

The SSL module is responsible for all user authentication, public/private key generation, certificate management, and packet encryption and decryption functions between the client and the server. It is dependent on the Switch Module to provide the interface for processing network traffic and the Switch Control Module (SCM) to send and receive configuration information.

The CSS stores all certificates and keys on the SCM disk. The CSS supports up to 256 certificates and 256 key-pairs, which equals approximately 3 MB of storage space on the disk. The CSS stores all certificate- and key-related files in a secure location on the disk. When processing connections, the CSS loads the certificates and keys into volatile memory on the SSL module for faster access.

No network traffic is sent to an SSL module from the SCM until an SSL content rule is activated to:

- Define where the content physically resides
- Where to direct the request for content (which service)
- Which load-balancing method to use

SSL Configuration Quick Starts

This section provides a quick overview on how to manage SSL certificates in the CSS, create an SSL proxy list, and add the SSL proxy list to an SSL service. Each step includes the CLI command required to complete the task. RSA has been chosen for the quick start procedures in this section because it is a popular public-key algorithm for encryption and authentication.

This section includes the following quick start procedures:

- RSA Certificate and Key Generation Quick Start
- RSA Certificate and Key Import Quick Start
- SSL Proxy List Quick Start
- SSL Service and Content Rule Quick Start

RSA Certificate and Key Generation Quick Start

Table 9-2 provides an overview of the steps required to generate and associate an RSA key pair and certificate in the CSS. Key and certificate generation may be necessary in instances when you do not have pre-existing keys or certificates for the CSS. You may want to initially generate RSA keys and temporary certificates on the CSS for internal SSL testing until the Certificate Authority responds to the certificate request and returns the authentic certificate. A generated certificate is temporary and expires in 30 days.

Table 9-2 RSA Certificate and Key Generation Quick Start

Task and Command Example

1. Enter configuration mode.
`# config`
`(config) #`

2. Generate the RSA key pair used in the exchange.
`(config) # ssl genrsa CSSrsakey1 1024 "passwd123"`
`Please be patient this could take a few minutes`

3. Associate the generated RSA key pair with a file. `(config) # ssl associate rsakey myrsakey1 CSSrsakey1`

Table 9-2 RSA Certificate and Key Generation Quick Start (continued)

Task and Command Example

4. After generating the RSA key pair, generate the Certificate Signing Request (CSR) file for the RSA key pair file and transfer the certificate request to the Certificate Authority (CA).
`(config) # ssl gencsr myrsakey1`
`You are about to be asked to enter information.`

5. While awaiting the return of the site certificate from the CA, testing can be done by generating a self-signed (temporary) certificate.
`(config) # ssl gencert certkey myrsakey1 signkey myrsakey1 CSScertfile1 "passwd123"`
`You are about to be asked to enter information`

6. Associate the generated certificate with a file. `(config) # ssl associate cert myrsacert1 CSScertfile1`

7. Compare the public key in the associated certificate with the public key stored with the associated private key and verify that they are the same.
`(config) # ssl verify myrsacert1 myrsakey1`
`Certificate mycert1 matches key mykey1`

RSA Certificate and Key Import Quick Start

Table 9-3 provides an overview of the steps required to import and associate an RSA certificate and key pair to the CSS from a remote server.

Table 9-3 RSA Certificate and Key Import Quick Start

Task and Command Example
1. Define a secure File Transfer Protocol (FTP) record file to import certificates and private keys into the CSS from an SFTP server. `# ftp-record ssl_record 192.168.19.21 johndoe "abc123" /home/johndoe`

2. Use secure FTP to transfer the imported certificates and private keys to the CSS.
`# copy ssl sftp ssl_record import rsacert.pem PEM "passwd123"`
`Connecting`
`Completed successfully`
 	
`# copy ssl sftp ssl_record import rsakey.pem PEM "passwd123"`
`Connecting`
`Completed successfully`

3. Enter configuration mode.
`# config`
`(config) #`

4. To use RSA public key exchange and authentication:
a. Associate the imported RSA certificate with a file.
`(config) # ssl associate cert myrsacert1 rsacert.pem`
b. Associate the imported RSA key pair with a file.
`(config) # ssl associate rsakey myrsakey1 rsakey.pem`

5. Compare the public key in the associated certificate with the public key stored with the associated private key and verify that they are the same.
`(config) # ssl verify myrsacert1 myrsakey1`
`Certificate mycert1 matches key mykey1`

SSL Proxy List Quick Start

Table 9-4 provides an overview of the steps required to create an SSL proxy list (for an RSA certificate and key pair).

Table 9-4 CSS SSL Proxy List Quick Start

Task and Command Example

1. Create the SSL proxy list.
`(config)# ssl-proxy-list ssl_list1`
`Create ssl-list <ssl_list1>, [y/n]: y`
Once you create an SSL proxy list, the CLI enters into ssl-proxy-list configuration mode for the newly created SSL proxy list.
`(ssl-proxy-list[ssl_list1])#`

2. Specify a number to identify a virtual SSL server in the SSL proxy list. `(ssl-proxy-list[ssl_list1])# ssl-server 20`
3. Specify a virtual IP address (VIP). Enter a VIP address that corresponds to an SSL content rule. `(ssl-proxy-list[ssl_list1])# ssl-server 20 vip address 192.168.3.6`
4. The CSS defaults to virtual TCP port number 443. If you need to change the port TCP port number to a different selection to correspond with the content rule, specify the virtual TCP port number. `(ssl-proxy-list[ssl_list1])# ssl-server 20 port 444`

5. For the RSA certificate and key pair, specify the name of an existing RSA certificate association and RSA key pair association for the SSL proxy list virtual SSL server.
`(ssl-proxy-list[ssl_list1])# ssl-server 20 rsacert myrsacert1`
`(ssl-proxy-list[ssl_list1])# ssl-server 20 rsakey myrsakey1`

Table 9-4 CSS SSL Proxy List Quick Start (continued)

Task and Command Example
6. Assign the appropriate cipher suite for the RSA certificates and keys in use, the IP address of the backend content rule/server used for the cipher suite, and the TCP port of the backend content rule/server. `(ssl-proxy-list[ssl_list1])# ssl-server 20 cipher rsa-export-with-rc4-40-md5 192.168.11.1 80 5`
7. Activate the completed SSL proxy list. `(ssl-proxy-list[ssl_list1])# active`

SSL Service and Content Rule Quick Start

Table 9-5 provides an overview of the steps required to create an SSL service, including adding the SSL proxy list to the service and creating an SSL content rule.

Table 9-5 CSS SSL Service and Content Rule Quick Start

Task and Command Example

1. Create an SSL service.
`(config)# service ssl_serv1`
`Create service <ssl_serv1>, [y/n]: y`

2. Specify **ssl-accel** as the service type. `(config-service[ssl_serv1])# type ssl-accel`
3. Specify the slot of the SSL module in the CSS chassis. `(config-service[ssl_serv1])# slot 3`
4. Disable the CSS from sending keepalive messages to the service. `(config-service[ssl_serv1])# keepalive type none`
5. Add the SSL proxy list to the SSL service. `(config-service[ssl_serv1])# add ssl-proxy-list ssl_list1`
6. Activate the SSL service. `(config-service[ssl_serv1])# active`

Table 9-5 CSS SSL Service and Content Rule Quick Start (continued)

Task and Command Example

7. Create an SSL content rule.
`(config)# owner ssl_owner`
`Create owner <ssl_owner>, [y/n]: y`
`(config-owner[ssl_owner])# content ssl_rule1`
`Create content <ssl_rule1>, [y/n]: y`

8. Configure a Virtual IP address (VIP) or domain name for the content rule. Ensure the VIP address is the same as the address specified in the SSL proxy list. `(config-owner-content[ssl-rule1]# vip address 192.168.3.6`
9. Specify a TCP port number for the content rule. Ensure the port number is the same as the port specified in the SSL proxy list. `(config-owner-content[ssl-rule1]# port 444`

10. If you are using two or more SSL modules, to use stickiness based on SSL version 3 session ID for a Layer 5 content rule, specify the following in the content rule to take advantage of the SSL session ID reuse:
• Enter the **application ssl** command to specify the SSL application type.
`(config-owner-content[ssl-rule1])# application ssl`
• Enter the **advanced-balance ssl** command to enable the content rule to be sticky based on SSL.
`(config-owner-content[ssl-rule1])# advanced-balance ssl`

11. Add the SSL service to the content rule. `(config-owner-content[ssl_rule1])# add service ssl_serv1`
12. Activate the content rule. `(config-owner-content[ssl_rule1])# active`
13. Save your configuration changes to the running-configuration. `# copy running-config startup-config`

Configuring SSL Certificates and Keys

Before creating an SSL proxy list and adding the SSL proxy list to an SSL service, you must load a set of digital certificates and public/private key pairs on the CSS disk (flash disk or hard disk). The digital certificates and key pairs are a form of digital identification for user authentication. Certificates are issued by Certificate Authorities (CAs) such as VeriSign® and Thawte®. Each certificate includes the name of the issuing authority, the name of the entity that the certificate was issued, the entity's public key, and timestamps that indicate the certificate's expiration date. You can use files received from a CA, import the certificate and keys from an existing secure server, or generate your own certificate and keys on the CSS.

The CSS supports the generation of certificates and keys directly within the CSS for purposes of testing. Your requirement to use generated certificates and keys instead of certificates and keys from a trusted authority depends on your environment. For example, the use of the CSS and SSL for a company's internal Web site may not require the use of certificates from a trusted CA. A certificate and key pair generated within the CSS may be sufficient to satisfy the intranet SSL requirement.

The CSS stores digital certificates and key pairs in encrypted files in a secure area on the CSS.

Caution When importing or exporting certificates and keys with the CSS, ensure that the CSS is not configured to perform a network boot from a network-mounted file system on a remote system (operating the CSS in a diskless environment). The network-mounted method of CSS booting is not supported with SSL termination; the certificates and keys must be local to the CSS and SSL module.

This section covers:

- Importing or Exporting Certificates and Private Keys
- Generating Certificates and Private Keys in the CSS
- Associating Certificates and Private Keys in the CSS
- Showing Certificate and Key Pair Information
- Removing Certificates and Private Keys from the CSS

Note To implement good security policies when importing or generating SSL certificates and key pairs, administrators should understand the various user modes of the CSS and have good password policies to protect those user modes. For more information, see the *Cisco Content Services Switch Command Reference*, Chapter 2 CLI Commands, the "(config) username-technician" section.

Importing or Exporting Certificates and Private Keys

You can import preexisting or new certificates and private keys to the CSS disk from a file, or a series of files, that are stored on a remote secure server. To transfer these files, Cisco Systems recommends that you use a secure encrypted transport mechanism between the CSS and the remote server. The CSS supports the Secure Shell protocol (SSHv2), which provides secure encryption communications between two hosts over an insecure network. The CSS supports file transport between network devices using the Secure File Transfer Protocol (SFTP) and the File Transfer Protocol (FTP). Of the two file transport protocols, Cisco Systems recommends SFTP as the transport mechanism of choice. It is similar to FTP except that it uses a secure and encrypted connection.

Before you import certificates or keys to the CSS:

- On the CSS, ensure that SSH access to the CSS is enabled to accept connections from SSH clients and that the Secure Management license key is installed prior to transferring certificates and keys. By default, SSH access is enabled through the **no restrict ssh** global command. If SSH access is restricted, or if the license key is not installed, SSH will not accept connections from SSH clients and the **copy ssl sftp** command will fail, resulting in generation of an error message.

Note For details about configuring Secure Shell Daemon on the CSS, refer to the *Cisco Content Services Switch Administration Guide*, Chapter 3, Configuring CSS Network Protocols.

- On the SFTP server, verify that the server is properly configured so that the user directory points to the directory where the certificates and keys reside. This path is required to ensure certificates and keys are properly copied from or to the SFTP server.

Caution When using SSH, ensure that the CSS is not configured to perform a network boot from a network-mounted file system on a remote system (a diskless environment). If SSH is enabled and the CSS has been booted using a network boot from a network-mounted file system, the CSS logs an error message by SSH as the protocol attempts to initialize and then exits from operation, which impacts importing and exporting of certificates and keys.

Configuring the Default SFTP or FTP Server to Import Certificates and Private Keys

Before you begin, use the **ftp-record** command to define the SFTP or FTP server that you intend to use to download imported certificates and private keys to the CSS disk. For details about using the **ftp-record** command to create an SFTP or FTP record file to use when accessing the server from the CSS, refer to the *Cisco Content Services Switch Administration Guid*e, Chapter 1, Logging in and Getting Started.

Note When defining the FTP record for the **copy ssl** command ensure that the base directory, if used, is relative to the SSH directory where the SSH server resides. For example, if the username is *sshlogin* and the SSH server is installed in d:\Program Files\Network, the default directory for the files would be d:\Program Files\Network\ssh. This path is required to ensure certificates and keys are properly copied to or from the SFTP server.

For example, to define the *ssl_record*, enter:

```
# ftp-record ssl_record 192.168.19.21 johndoe "abc123"
/home/johndoe
```

Transferring Certificates and Private Keys to the CSS

Use the **copy ssl** command to facilitate the import or export of certificates and private keys from or to the CSS. The CSS stores all imported files in a secure location on the CSS. This command is available only in SuperUser mode.

The syntax for this command is:

copy ssl [*protocol*] *ftp_record* [**import** *filename* [*format*] **"***password***"** {**"***passphrase***"**}|**export** *filename2* **"***password***"**]

The variables are:

- *protocol* - The type of protocol used to transfer the certificate and private key file. The valid entries are **sftp** or **ftp**. Cisco Systems recommends the SFTP protocol for the transport mechanism because it provides the most security.
- *ftp_record* - The name of the previously-created FTP record containing the remote host information.
- **import** - Import the file from the remote server.
- *filename* - The name of the file you want to import from the server. Include the full path to the file. You can enter up to 128 characters.

- *format* - The file format of the certificate to be imported. Once the certificate file is converted to PEM format and DES encoded, it is stored on the CSS SCM in a special (and secure) directory. The valid import file formats are:
  - **DER** - Binary format encoding of the certificate file in ASN.1 using the Distinguished Encoding Rules (DER-encoded X509 certificate). For example, an imported certificate from a Microsoft® Windows NT IIS 4.0 server.
  - **PEM** - Privacy Enhanced Mail, a base64 encoding of the certificate file (PEM-encoded X509 certificate). For example, an imported certificate from an Apache/SSL UNIX server.
  - **PKCS12** - Standard from RSA Data Security, Inc. for storing certificates and private keys. For example, an imported certificate from a Microsoft Windows 2000 IIS 5.0 server.
- *"password"* - The password used to DES (Data Encryption Standard) encode the imported certificate or private key. Encoding the imported file prevents unauthorized access to the certificate or private key on the CSS. Enter the password as a quoted string. The password appears in the CSS running configuration as a DES-encoded string.
- *"passphrase"* - (Optional for PEM files) The passphrase used to encrypt the certificate or key being imported into the CSS. Enter the passphrase as a quoted text string.

Note You must enter a passphrase for a PKCS12 file (.pfx). The CSS uses the passphrase to decrypt the file.

- **export** - Export the file to the remote server.
- *filename2* - The name you want to assign to the file on the server. Include the full path to the file. Enter an unquoted text string with no spaces and a maximum length of 32 characters.

Note An imported file can contain certificates, RSA or DSA key pairs, or Diffie-Hellman parameters. You must distinguish whether the files contain certificates, privates keys, or Diffie-Hellman parameters by associating the specific contents to a filename. See "Associating Certificates and Private Keys in the CSS" in this chapter for details.

For example, to import the *rsacert.pem* certificate from a remote server to the CSS, enter:

```
# copy ssl sftp ssl_record import rsacert.pem PEM "passwd123"
Connecting
Completed successfully
```

For example, to import the *rsakey.pem* certificate from a remote server to the CSS, enter:

```
# copy ssl sftp ssl_record import rsakey.pem PEM "passwd123"
Connecting
Completed successfully
```

To export the *rsacert.pem* certificate from the CSS to a remote server, enter:

```
# copy ssl sftp ssl_record export rsacert.pem "passwd123"
```

If the **copy ssl** command fails to import certificates or keys verify the following areas:

- The user account and password in the ftp record are correct
- The base directory is ssh or ssh/path
- The SSH server is reachable
- The SSH server IP address is correct in the ftp-record

Generating Certificates and Private Keys in the CSS

If you do not have preexisting keys, Diffie-Hellman parameters, and certificates for the CSS, you may want to generate them on the CSS disk for purposes of testing. The CSS includes a series of certificate and private key management utilities. These utilities simplify the process of generating an RSA private key, DSA private key, a Diffie-Hellman parameter file, a certificate signing request (CSR), and a self-signed temporary certificate.

Note The **ssl genrsa**, **gencsr**, **gendsa**, and **gencert** commands all produce a valid certificate or key pair. Be aware, however, that most Web browsers will flag the certificate as signed by an unrecognized signing authority.
A generated certificate is temporary and expires in 30 days.

This section covers:

- Generating an RSA Key Pair
- Generating a DSA Key Pair
- Generating Diffie-Hellman Key Parameters
- Generating a Certificate Signing Request Using an RSA Key
- Generating a Self-Signed Certificate

Generating an RSA Key Pair

Use the **ssl genrsa** command to generate an RSA private/public key pair for asymmetric encryption. RSA key pairs are used to sign and encrypt packet data, and they are a requirement before another device (client or server) can exchange an SSL certificate with the CSS. The key pair refers to a public key and its corresponding private (secret) key. The CSS stores the generated RSA key pair as a file on the CSS.

The syntax for this command is:

ssl genrsa *filename numbits* **"***password***"**

The variables are:

- *filename* - The name of generated RSA key pair file. Enter an unquoted text string up to 31 characters. The key pair filename is used only for identification in the CSS.
- *numbits* - The key pair strength. The number of bits in the key pair file defines the size of the RSA key pair used to secure Web transactions. Longer keys produce a more secure implementation by increasing the strength of the RSA security policy. Available entries (in bits) are 512 (least security), 768 (normal security), 1024 (high security), and 2048 (highest security).
- *"password"* - The password used to encode the RSA private key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the RSA key pair *myrsakeyfile1*, enter:

```
(config) # ssl genrsa myrsakeyfile1 1024 "passwd123"
 Please be patient this could take a few minutes
```

After you generate an RSA key pair, you can generate a Certificate Signing Request (CSR) file for the RSA key pair file and transfer the certificate request to the Certificate Authority (CA). This provide an added layer of security because the RSA private key originates directly within the CSS and does not have to be transported externally. You can then create a temporary certificate for internal testing until the CA responds to the certificate request and returns the authentic certificate. Each generated key pair must be accompanied by a certificate to work.

You must also associate an RSA key pair name to the generated RSA key pair, as discussed in the "Associating Certificates and Private Keys in the CSS" section of this chapter.

Generating a DSA Key Pair

Use the **ssl gendsa** command to generate a DSA private/public key pair for asymmetric encryption. DSA is the public key exchange cryptographic system developed by the National Institutes of Science and Technology. DSA can only be used for digital signatures (signings) but not for key private/public exchange. The CSS stores the generated DSA key pair as a file on the CSS.

The syntax for this command is:

ssl gendsa *filename numbits "password"*

The variables are:

- *filename* - The name of the generated DSA key pair file. Enter an unquoted text string up to 31 characters. The key pair filename is used only for identification in the CSS.
- *numbits* - The key pair strength. The number of bits in the key pair file defines the size of the DSA key pair used to secure Web transactions. Longer keys produce a more secure implementation by increasing the strength of the DSA security policy. Available entries (in bits) are 512 (least security), 768 (normal security), and 1024 (highest security).
- *"password"* - The password used to encode the DSA private key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the DSA key pair *mydsakeyfile2*, enter:

```
(config) # ssl gendsa mydsakeyfile2 512 "passwd123"
Please be patient this could take a few minutes
```

You must also associate a DSA key pair name to the generated DSA key pair as discussed in the "Associating Certificates and Private Keys in the CSS" section of this chapter.

Generating Diffie-Hellman Key Parameters

Use the **ssl gendh** command to generate a Diffie-Hellman key agreement parameter file. Diffie-Hellman is a shared key agreement algorithm. Diffie-Hellman key exchange uses a complex algorithm and public/private keys to encrypt and then decrypt packet data. The CSS stores the generated Diffie-Hellman key parameter file.

Note Generation of a Diffie-Hellman key agreement parameter file can sometimes take a lengthy period of time (perhaps up to 20 minutes) and is a CPU-intensive utility. If you are running the **ssl gendh** utility, ensure that the CSS is not actively passing traffic at the same time to avoid impacting CSS performance.

The syntax for this command is:

ssl gendh *filename numbits "password"*

The variables are:

- *filename* - The name of the file to store the Diffie-Hellman key parameters. Enter an unquoted text string up to 31 characters. The filename is used only for identification in the CSS.
- *numbits* - The key strength. The number of bits in the file defines the size of the Diffie-Hellman key used to secure Web transactions. Longer keys produce a more secure implementation by increasing the strength of the Diffie-Hellman security policy. Available entries (in bits) are 512 (least security), 768 (normal security), 1024 (high security), and 2048 (highest security).
- *"password"* - The password used to encode the Diffie-Hellman key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the Diffie-Hellman key parameter list *dhparamfile2*, enter:

```
(config) # ssl gendh dhparamfile2 512 "passwd123"
Please be patient this could take a few minutes
```

You must also associate a Diffie-Hellman parameter file name to the generated Diffie-Hellman parameter file, as discussed in the "Associating Certificates and Private Keys in the CSS" section of this chapter.

Generating a Certificate Signing Request Using an RSA Key

Use the **ssl gencsr** *rsakey* command to generate a Certificate Signing Request (CSR) file for an RSA key pair file and to transfer the certificate request to the Certificate Authority (CA). You must generate a CSR file if you are requesting a new certificate or renewing a certificate. When the CA signs the CSR, using its RSA private key, the CSR becomes the certificate.

The *rsakey* variable specifies the key that the RSA certificate is built on. It is the public key that is embedded in the certificate.

The RSA key pair must already be loaded on the CSS and you must associate an RSA key pair name to the generated RSA key pair (see "Associating Certificates and Private Keys in the CSS" in this chapter for details). If the appropriate key pair does not exist, the CSS logs an error message.

For example, to generate a CSR based on the RSA key pair *myrsakey1*, enter:

```
CSS11503(config)# ssl gencsr myrsakey1
You are about to be asked to enter information
that will be incorporated into your certificate
request. What you are about to enter is what is
called a Distinguished Name or a DN.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [US]US
State or Province (full name) [SomeState]MA
Locality Name (city) [SomeCity]Boxborough
Organization Name (company name) [Acme Inc]Cisco Systems, Inc.
Organizational Unit Name (section) [Web Administration]Web Admin
Common Name (your domain name) [www.acme.com]www.cisco.com
Email address [webadmin@acme.com]webadmin@cisco.com
```

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBWDCCAQICAQAwgZwxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJNQTETMBEGA1UE
BxMKQm94Ym9yb3VnaDEcMBoGA1UEChMTQ2lzY28gU3lzdGVtcywgSW5jLjESMBAG
A1UECxMJV2ViIEFkbWluMRYwFAYDVQQDEw13d3cuY2lzY28uY29tMSEwHwYJKoZI
hvcNAQkBFhJra3JvZWJlckBjaXNjby5jb20wXDANBgkqhkiG9w0BAQEFAANLADBI
AkEAqHXjtQUVXvmo6tAWPiMpe6oYhZbJUDgTxbW4VMCygzGZn2wUJTgLrifDB6N3
v+1tKFndE686BhKqfyOidml3wQIDAQABoAAwDQYJKoZIhvcNAQEEBQADQQA94yC3
4SUJJ4UQEnO2OqRGLOZpAElc4+IV9aTWK6NmiZsM9Gt0vPhIkLx5jjhVRLlb27Ak
H6D5omXa0SPJan5x
-----END CERTIFICATE REQUEST-----

CSS11503(config)#
```

The **ssl gencsr** command generates the CSR and outputs it to the screen. Most major Certificate Authorities have Web-based applications that require you to cut and paste the certificate request to the screen. Note that the CSR is not saved in the CSS.

To test your CSR file, you can create a temporary certificate by generating a CSR and signing it with your own private key. While this produces a valid certificate, most browsers flag the certificate as signed by an unrecognized signing authority. See "Generating a Self-Signed Certificate" in this chapter for details.

Generating a Self-Signed Certificate

Use the **ssl gencert** command to generate and save a temporary certificate to a file on disk in the CSS. For purposes of SSL testing you can generate a temporary certificate by generating a CSR and signing it with your own private key. A generated certificate is temporary and expires in 30 days.

Note The **ssl gencert** command produces a valid certificate. However, most Web browsers flag this certificate as signed by an unrecognized signing authority.

You generate the certificate by considering:

- The key pair that the certificate is based on (RSA or DSA).
- The key used to sign the certificate.

The **ssl gencert** command can sign RSA or DSA certificates with either an RSA key pair or a DSA key pair.

Note Although the CSS allows signing an RSA certificate with a DSA key (and a DSA certificate with an RSA key) it is a more standard practice that an RSA certificate is signed with RSA keys (and DSA certificate is signed with a DSA key).

The syntax for this command is:

ssl gencert certkey *certkey* **signkey** *signkey certfile* **"***password***"**

The variables are:

- **certkey** *certkey* - The name of the RSA or DSA key pair that the certificate is based on. Enter an unquoted text string up to 31 characters.
- **signkey** *signkey* - The RSA or DSA key pair to be used to sign the certificate. Enter an unquoted text string up to 31 characters.
- *certfile* - The name of the file used to store the certificate as a file on the CSS. Enter an unquoted text string up to 31 characters.
- **"***password***"** - The password used to encode the certificate file using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string. The password appears in the CSS running configuration as a DES-encoded string.

For example, to interactively generate the *mycertfile2* certificate, enter:

```
CSS11503(config)# ssl gencert certkey myrsakey signkey
myrsasignkey myrsacertfile "passwd123"
You are about to be asked to enter information
that will be incorporated into your certificate
request. What you are about to enter is what is
called a Distinguished Name or a DN.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [US]US
State or Province (full name) [SomeState]MA
Locality Name (city) [SomeCity]Boxborough
Organization Name (company name) [Acme Inc]Cisco Systems, Inc.
Organizational Unit Name (section) [Web Administration]Web Admin
Common Name (your domain name) [www.acme.com]www.cisco.com
Email address [webadmin@acme.com ]webadm@cisco.com

CSS11503(config)#
```

You must also associate the contents of this temporary certificate to a filename, as discussed in the "Associating Certificates and Private Keys in the CSS" section of this chapter.

Associating Certificates and Private Keys in the CSS

After you import or generate certificate and key pair files, you must distinguish to the CSS whether these files contain certificates, private keys, or Diffie-Hellman parameters. You do this by associating certificate names, private/public key pair names, or Diffie-Hellman parameter names to the particular imported files.

When you associate the entries specified in the various certificate and private key commands to files, CSS stores the bindings in the running configuration. Before you log out or reboot the CSS, you must copy the contents of the running-config file to the startup-config file to save configuration changes and have the CSS use this configuration on subsequent reboots. When you reboot the CSS, the certificate and key associations are automatically loaded.

This section covers:

- Associating a Certificate to a File
- Associating an RSA Key Pair to a File
- Associating a DSA Key Pair to a File
- Associating Diffie-Hellman Parameters to a File
- Verifying a Certificate Against a Key Pair

Associating a Certificate to a File

Use the **ssl associate cert** command to associate a certificate name to an imported or generated certificate. Use the **no** form of the command to remove the association to the file.

The syntax for this command is:

ssl associate cert *certname filename*

The variables are:

- *certname* - The name of the certificate association. Enter an unquoted text string up to 31 characters.
- *filename* - The name of the file containing the certificate. Up to 128 characters can be used. To see a list of imported or generated certificates, use the **ssl associate cert** *certname* **?** command.

For example, to associate the certificate name *myrsacert*1 to the imported certificate file *rsacert.pem*, enter:

```
(config) # ssl associate cert myrsacert1 rsacert.pem
```

To remove the association to the file, enter:

```
(config) # no ssl associate ssl cert myrsacert1
```

Note The **no** form of the command will not function if the associated certificate is in use by an active SSL proxy list.

Associating an RSA Key Pair to a File

Use the **ssl associate rsakey** command to associate an RSA key pair name to an imported or generated RSA key pair. Use the **no** form of the command to remove the association to the file.

The syntax for this command is:

ssl associate rsakey *keyname filename*

The variables are:

- *keyname* - The name of the RSA key pair association. Enter an unquoted text string up to 31 characters.
- *filename* - The name of the file containing the RSA key pair. Up to 128 characters can be used. To see a list of imported or generated RSA keys, use the **ssl associate rsakey** *keyname* **?** command.

For example, to associate the RSA key name *myrsakey1* to the imported *rsakey.pem*, enter:

```
(config) # ssl associate rsakey myrsakey1 rsakey.pem
```

To remove the association to the file, enter:

```
(config) # no ssl associate rsakey myrsakey1
```

Note The **no** form of the command will not function if the associated RSA key pair is in use by an active SSL proxy list.

Associating a DSA Key Pair to a File

Use the **ssl associate dsakey** command to associate a DSA key pair name to an imported or generated DSA key pair. Use the **no** form of the command to remove the association to the file.

The syntax for this command is:

ssl associate dsakey *keyname filename*

The variables are:

- *keyname* - The name of the DSA key pair association. Enter an unquoted text string up to 31 characters.
- *filename* - The name of the file containing the DSA key pair. Up to 128 characters can be used. To see a list of imported or generated DSA keys, use the **ssl associate dsakey** *keyname* **?** command.

For example, to associate the DSA key name *mydsakey1* to the imported *dsakey.pem*, enter:

```
(config) # ssl associate dsakey mydsakey1 dsakey.pem
```

To remove the association to the file, enter:

```
(config) # no ssl associate dsakey mydsakey1
```

Note The **no** form of the command will not function if the associated DSA key pair is in use by an active SSL proxy list.

Associating Diffie-Hellman Parameters to a File

Use the **ssl associate dhparam** command to associate a Diffie-Hellman name to an imported or generated Diffie-Hellman parameter file. Use the **no** form of the command to remove the association to the file.

The syntax for this command is:

ssl associate dhparam *paramname filename*

The variables are:

- *paramname* - The name of the Diffie-Hellman parameter association. Enter an unquoted text string up to 31 characters.
- *filename* - The name of the file containing the Diffie-Hellman parameters. Up to 128 characters can be used. To see a list of imported or generated Diffie-Hellman files, use the **ssl associate dhparam** *filename* **?** command.

For example, to associate the Diffie-Hellman file name *mydhparam1* to the imported *dhparams.pem*, enter:

```
(config) # ssl associate dhparam mydhparam1 dhparams.pem
```

To remove the association to the file, enter:

```
(config) # no ssl associate dhparam mydhparam1
```

Note The **no** form of the command will not function if the associated Diffie-Hellman parameter list is in use by an active SSL proxy list.

Verifying a Certificate Against a Key Pair

A digital certificate is built around a public key, and it can only be used with one key pair. Use the **ssl verify** command to compare the public key in the associated certificate with the public key stored with the associated private key, and verify that they are both the same. To see a list of certificate and key pair associations, use the **ssl verify ?**command.

Note If the certificate does not match the public/private key pair, the CSS logs an error message.

The syntax for this command is:

ssl verify *certname keyname*

The variables are:

- *certname* - The association name of the certificate used to verify against the specified key pair.
- *keyname* - The association name of the key pair used to verify against the specified certificate.

For example, to verify the *myrsacert1* digital certificate against the *myrsakey1* key pair, enter:

```
(config)# ssl verify myrsacert1 myrsakey1
Certificate and key match
```

Showing Certificate and Key Pair Information

A number of **show** commands in the CSS enable you to display information about SSL certificates and key pairs stored on the CSS. Enter the following **show** commands from any mode:

- **show ssl associate cert** - Displays certificate associations
- **show ssl associate rsakey** - Displays RSA key pair associations
- **show ssl associate dsakey** - Displays DSA key pair associations
- **show ssl associate dhparam** - Displays information about Diffie-Hellman parameter associations
- **show ssl associate** - Displays all file associations for the CSS
- **show ssl files** - Displays all certificate, key pair, and Diffie-Hellman parameter files loaded on the CSS

Showing SSL Certificates

Use the **show ssl associate cert** *certname* command to display summary data for certificate associations in the CSS. You can optionally specify a certificate name to view detailed information about the certificate, corresponding to the certificate association. If you do not specify a certificate name, all certificate associations appear in the **show ssl associate cert** output.

To display information about all certificate associations:

```
show ssl associate cert
```

Table 9-6 describes the fields in the **show ssl associate cert** output.

Table 9-6 Field Descriptions for the show ssl associate cert Command

Field	Description
Certificate Name	The name of the certificate association
File Name	The name of the file containing the certificate
Used By List	Indicates if the certificate association is used by the SSL proxy list containing the VIP address of the virtual server

To display information about a specific certificate association, enter:

```
show ssl associate cert myrsacert1
```

Table 9-7 describes the fields in the **show ssl associate cert** *certname* output.

Table 9-7 Field Descriptions for the show ssl associate cert certname Command

Field	Description
Certificate	The name of the Certificate Association (CA) that issued the certificate.
Version	The version of the certificate.
Serial Number	The serial number associated with the certificate.
Signature Algorithm	The digital signature algorithm (such as RSA) used for the encryption of information with a public/private key pair.
Issuer	The organization that generated the certificate and will vouch for it. An issuer is also the Certificate Authority (CA).

Table 9-7 Field Descriptions for the show ssl associate cert certname Command (continued)

Field	Description
Validity	
Not Before	The starting time period, before which the certificate is not considered valid.
Not After	The ending time period, after which the certificate is not considered valid.
Subject	The certified party that possesses the private key.
Subject Public Key Info	
Public Key Algorithm	The name of the key exchange algorithm used to generate the public key (for example, RSA).
RSA Public Key	The number of bits in the key to define the size of the RSA key pair used to secure Web transactions.
Modulus	The actual public key that the certificate was built upon.
Exponent	One of the base numbers used to generate the key.
X509v3 Extensions	An array of X509v3 extensions added to the certificate.
X509v3 Basic Constraints	Indicates if the subject may act as a CA, with the certified public key being used to verify certificate signatures. If so, a certification path length constraint may also be specified.

Table 9-7 Field Descriptions for the show ssl associate cert certname Command (continued)

Field	Description
Netscape Comment	A comment that may be displayed when the certificate is viewed.
X509v3 Subject Key Identifier	Identifies the public key being certified. It enables distinct keys used by the same subject to be differentiated (for example, as key updating occurs).
X509v3 Authority Key Identifier	Identifies the public key to be used to verify the signature on this certificate or CRL. It enables distinct keys used by the same CA to be distinguished (for example, as key updating occurs).
Signature Algorithm	The name of the algorithm used for digital signatures (but not for key exchanges).
Hex Numbers	The actual signature of the certificate. The client can regenerate this signature using the specified algorithm to make sure that the certificate data has not been changed.

Showing SSL RSA Private Keys

Use the **show ssl associate rsakey** *keyname* command to obtain information about RSA private key associations in the CSS. You can optionally specify an RSA key name to view information about a specific RSA key association (key size and type). If you do not specify an RSA keyname, you see a list of all RSA key associations.

Note When you view the contents of a specific key only specifics on the key size and key type appears. This restriction occurs because the key contents are secure and should not be viewed.

To display information about all RSA private key associations:

```
(config) # show ssl associate rsakey
```

Table 9-8 describes the fields in the **show ssl associate rsakey** output.

Table 9-8 Field Descriptions for the show ssl associate rsakey Command

Field	Description
Key Name	The name of the RSA key association
File Name	The name of the file containing the RSA key pair
Used By List	Indicates if the RSA key association is used by the SSL proxy list containing the VIP address of the virtual server

To display information about a specific RSA key pair association, enter:

```
(config) # show ssl associate rsakey myrsakey1
1024-bit RSA keypair
```

Showing SSL DSA Private Keys

Use the **show ssl associate dsakey** *keyname* command to obtain information about DSA private key associations in the CSS. You can optionally specify a DSA key name to view information about a specific DSA key association (key size and type). If you do not specify a DSA keyname, you see a list of all DSA key associations.

Note When you view the contents of a specific key only specifics on the key size and key type appears. This restriction occurs because the key contents are secure and should not be viewed.

To display information about all DSA key associations:

```
(config) # show ssl associate dsakey
```

Table 9-9 describes the fields in the **show ssl associate dsakey** output.

Table 9-9 Field Descriptions for the show ssl associate dsakey Command

Field	Description
Key Name	The name of the DSA key association
File Name	The name of the file containing the DSA key pair
Used By List	Indicates if the DSA key association is used by the SSL proxy list containing the VIP address of the virtual server

To display information about a specific DSA key pair association, enter:

```
(config) # show ssl associate dsakey mydsakey1
1024-bit DSA keypair
```

Showing SSL Diffie-Hellman Parameters

Use the **show ssl associate dhparam** *paramname* to obtain information about Diffie-Hellman parameters. You can optionally specify a parameter file name to view information about a specific Diffie-Hellman parameter file association. If you do not specify a Diffie-Hellman parameter file name, you see a list of all Diffie-Hellman parameter file associations.

To display information about all Diffie-Hellman associations:

```
(config) # show ssl associate dhparam
```

Table 9-10 describes the fields in the **show ssl associate dhparam** output.

Table 9-10 Field Descriptions for the show ssl associate dhparam Command

Field	Description
Parameter Name	The name of the Diffie-Hellman parameter association

Table 9-10 Field Descriptions for the show ssl associate dhparam Command (continued)

Field	Description
File Name	The name of the file containing the Diffie-Hellman parameters
Used By List	Indicates if the Diffie-Hellman file association is used by the SSL proxy list containing the VIP address of the virtual server

To display information about a specific Diffie-Hellman parameter file association, enter:

```
(config) # show ssl associate dhparam mydhparam1
512-bit DH parameters
```

Showing SSL Associations

Use the **show ssl associate** to display a summary of all certificate and key associations stored on the CSS.

To display a summary of SSL associations for the CSS, enter:

```
CSS11506(config)# show ssl associate
 Certificate Name        File Name            Used by List
 ----------------        ---------            ------------
 rsacert                 rsacert.pem          yes

 RSA Key Name            File Name            Used by List
 ------------            ---------            ------------
 rsakey                  rsakey.pem           yes

 DH Param Name           File Name            Used by List
 -------------           ---------            ------------
 dhparams                dhparams.pem         no

 DSA Key Name            File Name            Used by List
 ------------            ---------            ------------
 dsakey                  dsakey.pem           no
```

Showing SSL Certificates, Key Pairs, and Diffie-Hellman Parameter Files

Use the **show ssl files** to display a list of certificates, key pairs, and Diffie-Hellman parameter files loaded on the CSS.

For example, enter:

```
(config) # show ssl files
```

Table 9-11 describes the fields in the **show ssl files** output.

Table 9-11 Field Descriptions for the show ssl files Command

Field	Description
File Name	The name of the imported or manually generated certificate, RSA key pair, DSA key pair, or Diffie-Hellman parameter file.
File Type	The format of the imported or manually generated certificate, RSA key pair, DSA key pair, or Diffie-Hellman parameter file. File types can include DES-encoded, PEM-encoded, or PKCS#12-encoded.
File Size	The total size (in Kbytes) of the certificate, RSA key pair, DSA key pair, or Diffie-Hellman parameter file.

Removing Certificates and Private Keys from the CSS

Use the **clear ssl file** command to remove certificates and private keys from the CSS that are no longer valid. Note that the **clear ssl file** command does not function if the file currently has an association with it. First remove the association to the file by specifying the **no ssl associate** command (see "Associating Certificates and Private Keys in the CSS" in this chapter for details).

The syntax for this global command is:

clear ssl file *filename password*

The variables are:

- *filename* - The name of the certificate, key pair, or Diffie-Hellman parameter file that you want to remove from the CSS.

- *password* - The password used to encode the file using DES when it was originally imported or generated by the CSS. This password must be an exact match or the file cannot be cleared.

For example, to remove *dsacert.pem* from the CSS, enter:

```
# clear ssl file dsacert.pem "passwd123"
```

Creating an SSL Proxy List

SSL termination with the CSS and initiation of HTTP connections to servers requires the use of an SSL proxy list to configure the flow of SSL information among the CSS SSL module, the client, and the server. An SSL proxy list comprises one or more virtual SSL servers (related by index entry). You can define up to 256 virtual SSL servers for a single SSL proxy list.

You add the active SSL proxy list to an ssl-accel type service to initiate the transfer of SSL configuration data for the SSL module. You add an SSL service to an SSL content rule.

For example, if the e-commerce vendor Brand New Products, Inc. wants to set up the CSS to perform SSL termination, they need to divert all traffic intended for https://www.brandnewproducts.com to the SSL module in the CSS. To do this, they must identify a VIP address for a virtual SSL server in the SSL proxy list and link the list to the same VIP of a content rule. The VIP address requires the following additional SSL configuration parameters:

- Identification of a virtual TCP port number that corresponds with a content rule
- An existing RSA or DSA certificate for identification purposes
- An appropriate SSL key pair to perform encryption and signing (assuming you are using an RSA key pair)
- Diffie-Hellman parameters if your CSS SSL security requires the Diffie-Hellman key exchange algorithm
- Assignment of a cipher suite

This section describes the steps to configure an SSL proxy list, add the completed SSL proxy list to an SSL service, and add the SSL service to a content rule. It covers:

- Configuring an SSL Proxy List
- Configuring Virtual SSL Servers for an SSL Proxy List

Configuring an SSL Proxy List

An SSL proxy list is a group of related virtual SSL servers that are associated with an SSL service. An SSL module in the CSS uses the virtual SSL servers to properly process and terminate SSL communications between the client and the server.

The following sections describe how to create an SSL proxy list:

- Creating the SSL Proxy List
- Adding a Description to an SSL Proxy List

Creating the SSL Proxy List

Use the **ssl-proxy-list** command to create an SSL proxy list. You can access the ssl-proxy-list configuration mode from most configuration modes except for the ACL, boot, group, rmon, or owner configuration modes. You can also use this command from the ssl-proxy-list configuration mode to access another SSL proxy list.

Enter the SSL proxy list name as an unquoted text string from 1 to 31 characters in length.

For example, to create the ssl-proxy-list *ssl_list1*, enter:

```
(config)# ssl-proxy-list ssl_list1
Create ssl-list <ssl_list1>, [y/n]: y
```

Once you create an SSL proxy list, the CLI enters into the ssl-proxy-list configuration mode.

```
(ssl-proxy-list[ssl_list1])#
```

To delete an existing ssl-proxy-list, enter:

```
(config)# no ssl-proxy-list ssl_list1
Delete ssl-list <ssl_list1>, [y/n]: y
```

Note You cannot delete an SSL proxy list if an SSL service is in use and contains the active SSL proxy list. You must first suspend the SSL service to delete a specific SSL proxy list.

Adding a Description to an SSL Proxy List

Use the **description** command to specify a description for an SSL proxy list. Enter the description as a quoted text string with a maximum of 64 characters, including spaces.

For example, to add a description to the *ssl_list1* SSL proxy list, enter:

```
(ssl-proxy-list[ssl_list1])# description "This is the SSL list for
www.brandnewproducts.com"
```

To remove the description from a specific SSL proxy list, enter:

```
(ssl-proxy-list[ssl_list1])# no description
```

Configuring Virtual SSL Servers for an SSL Proxy List

This section discusses creating one or more virtual SSL servers for a SSL proxy list. Use the **ssl-server** command to define an index entry in the SSL proxy list that you then use to configure specific SSL parameters associated with the SSL proxy list. An SSL module in the CSS uses the virtual SSL servers to properly process and terminate SSL communications between the client and the server. You must define an ssl-server index number before configuring ssl-proxy-list parameters. You can define up to 256 virtual SSL servers for a single SSL proxy list.

Note You cannot modify the virtual SSL servers in an active SSL proxy list. You must first suspend the SSL proxy list to make modifications to any of ssl-servers in a specific SSL proxy list. Once you have modified the SSL proxy list, suspend the SSL service, activate the SSL proxy list, and then activate the SSL service.

The following sections describe how to create a virtual SSL server for an SSL proxy list:

- Creating an SSL Proxy Configuration ssl-server Index
- Specifying a Virtual IP Address
- Specifying a Virtual Port
- Specifying the RSA Certificate Name
- Specifying the RSA Key Pair Name
- Specifying the DSA Certificate Name

- Specifying the DSA Key Pair Name
- Specifying the Diffie-Hellman Parameter File Name
- Specifying Cipher Suites
- Specifying SSL/TLS Version
- Specifying SSL Session Cache Timeout
- Specifying SSL Session Handshake Renegotiation
- Specifying SSL TCP Client-Side Connection Timeout Values
- Specifying SSL TCP Server-Side Connection Timeout Values

Creating an SSL Proxy Configuration ssl-server Index

Use the **ssl-server** *number* command to identify SSL-specific parameters for the SSL proxy list. This command creates a number (index entry) in the SSL proxy list that you use to configure specific SSL parameters associated with the virtual SSL server (VIP, certificate name, key pair, and so on). You must create a virtual SSL server before you can configure SSL proxy list parameters.

Enter a value between 1 and 256 that corresponds to the SSL proxy list virtual SSL server number.

For example, to specify virtual SSL server 20, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20
```

To remove the virtual SSL server from the SSL proxy list, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20
```

Specifying a Virtual IP Address

Use the **ssl-server** *number* **vip address** *ip_or_host* command to specify a virtual IP (VIP) address. Enter a VIP address for the virtual SSL server that corresponds to an SSL content rule. The SSL module uses this VIP address as the means to know which traffic it should accept. Ensure that the VIP address matches a VIP address configured in a content rule. See "Configuring an SSL Content Rule" in this chapter for details.

Enter a valid VIP address in either dotted-decimal IP notation (for example, 192.168.11.1) or mnemonic host-name format (for example, myhost.mydomain.com).

Note When you use the mnemonic host name format for the VIP, the CSS includes a Domain Name Service (DNS) facility that translates host names such as myhost.mydomain.com to IP addresses such as 192.168.11.1. If the host name cannot be resolved, the VIP address setting is not accepted and an error message appears indicating host resolution failure. For details on configuring a Domain Name Service, refer to the *Cisco Content Services Switch Administration Guide*, Chapter 3, Configuring CSS Network Protocols.

If the VIP address has not been defined for the virtual SSL sever when you activate the ssl-proxy-list (see "Activating an SSL Proxy List" in this chapter for details), the CSS logs an error message and does not activate the SSL proxy list. When you activate a content rule with a configured SSL service, the CSS verifies that each VIP address configured in the content rule matches at least one VIP address configured in the SSL proxy list in each of the added services. If a match is not found, the CSS logs an error message and does not activate the content rule.

For example, to specify a VIP address for the virtual SSL server that corresponds to a VIP address configured in a content rule, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 vip address 192.168.3.6
```

To remove a VIP from a specific virtual SSL server, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 vip address
```

Specifying a Virtual Port

Use the **ssl-server** *number* **port** *number* command to specify a virtual TCP port number for the virtual SSL server. Enter a TCP port number that corresponds with an SSL content rule, which uses the specified TCP port number. The SSL module uses this virtual port to know which traffic it should accept.

Specify a port number ranging from 1 to 65535. The default port is 443. Ensure that the specified port number matches the port configured in a content rule (see "Configuring an SSL Content Rule" in this chapter for details).

If the virtual port has not been defined for the virtual SSL server when you activate the ssl-proxy-list (see "Activating an SSL Proxy List" in this chapter for details), the CSS logs an error message and does not activate the SSL proxy list. When you activate a content rule with a configured SSL service, the CSS verifies that each virtual port configured in the content rule matches at least one port configured in the SSL proxy list in each of the added services. If a match is not found, the CSS logs an error message and does not activate the content rule.

For example, to specify a virtual port of 444, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 port 444
```

To reset the virtual port to the default of 443, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 port
```

Specifying the RSA Certificate Name

Use the **ssl-server** *number* **rsacert** *name* command to identify the name of an RSA certificate association to be used in the exchange of a public/private key pair for authentication and packet encryption. To see a list of existing RSA certificate associations, use the **ssl-server** *number* **rsacert ?** command.

The specified RSA certificate must already be loaded on the CSS and an association made (see "Configuring SSL Certificates and Keys" in this chapter for details). If there is not a proper RSA certificate association, upon activation of the SSL proxy list the CSS logs an error message and does not activate the list.

For example, to specify a previously defined RSA certificate association named *rsacert*, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 rsacert myrsacert1
```

To remove an RSA certificate association from a specific virtual SSL server, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 rsacert
```

Specifying the RSA Key Pair Name

Use the **ssl-server** *number* **rsakey** *name* command to identify the name of an RSA key pair association. RSA key pairs are a requirement before another device (client or server) can exchange an SSL certificate with the CSS. To see a list of existing RSA key pair associations, use the **ssl-server** *number* **rsakey ?** command.

The RSA key pair must already be loaded on the CSS and an association made (see "Configuring SSL Certificates and Keys" in this chapter for details). If there is not a proper RSA key pair association, upon activation of the SSL proxy list the CSS logs an error message and does not activate the list.

For example, to specify a previously defined RSA key pair association named *rsakey*, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 rsakey myrsakey1
```

To remove an RSA key pair association from a specific virtual SSL server, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 rsakey
```

Specifying the DSA Certificate Name

Use the **ssl-server** *number* **dsacert** *name* command to identify the name of a DSA certificate association that is to be used in the exchange of digital signatures. To see a list of existing DSA certificate associations, use the **ssl-server** *number* **dsacert ?** command

The specified DSA certificate must already be loaded on the CSS and an association made (see "Configuring SSL Certificates and Keys" in this chapter for details). If there is not a proper RSA certificate association, upon activation of the SSL proxy list the CSS logs an error message and does not activate the list.

For example, to specify a previously defined DSA certificate association named *dsacert*, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 dsacert mydsacert1
```

To remove a DSA certificate association from a specific virtual SSL server, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 dsacert
```

Specifying the DSA Key Pair Name

Use the **ssl-server** *number* **dsakey** *name* command to identify the name of a DSA key pair association. DSA key pairs are used to sign packet data, and they are a requirement before another device (client or server) can exchange an SSL certificate with the CSS. To see a list of existing DSA key pair associations, use the **ssl-server** *number* **dsakey ?** command.

The DSA key pair must already be loaded on the CSS and an association made (see "Configuring SSL Certificates and Keys" in this chapter for details). If there is not a proper DSA key pair association, upon activation of the SSL proxy list the CSS logs an error message and does not activate the list.

For example, to specify a previously defined DSA key pair association named *dsakey*, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 dsakey mydsakey1
```

To remove a DSA key pair association from a specific virtual SSL server, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 dsakey
```

Specifying the Diffie-Hellman Parameter File Name

Use the **ssl-server** *number* **dhparam** *name* command to identify the name of a Diffie-Hellman key exchange parameter file association. The Diffie-Hellman key exchange parameter file ensures that the two devices in a data exchange cooperate to generate a shared key for packet encryption and authentication. To see a list of existing Diffie-Hellman key exchange parameter files, use the **ssl-server** *number* **dhparam ?** command.

The Diffie-Hellman parameter file must already be loaded on the CSS and an association made (see "Configuring SSL Certificates and Keys" in this chapter for details). If there is not a proper Diffie-Hellman parameter file association, upon activation of the SSL proxy list the CSS logs an error message and does not activate the list.

To specify a previously defined Diffie-Hellman parameter file association, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 dhparam mydhparams1
```

To remove a Diffie-Hellman parameter file association from a specific virtual SSL server, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 dhparam
```

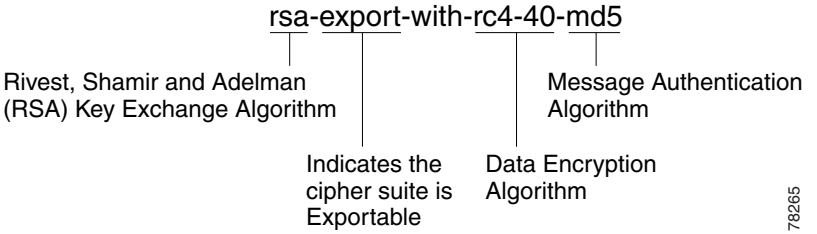
Specifying Cipher Suites

The SSL protocol supports a variety of different cryptographic algorithms, or ciphers, for use in operations such as authenticating the server and client to each other, transmitting certificates, and establishing session keys. Clients and servers may support different cipher suites, or sets of ciphers, depending on various factors such as the version of SSL they support, company policies regarding acceptable encryption strength, and government restrictions on export of SSL-enabled software. Among its other functions, the SSL handshake protocol determines how the server and client negotiate which cipher suites they will use to authenticate each other to transmit certificates and to establish session keys.

Note Exportable cipher suites are those cipher suites that are considered not to be as strong as some of the other cipher suites (for example, 3DES or RC4 with 128-bit encryption) as defined by U.S. export restrictions on software products. Exportable cipher suites may be exported to most countries from the United States, and provide the strongest encryption available for exportable products.

Each cipher suite specifies a set of key exchange algorithms. Figure 9-1 summarizes the algorithms associated with the rsa-export-with-rc4-40-md5 cipher suite.

Figure 9-1 Cipher Suite Algorithms



Use the **ssl-server** *number* **cipher** command to assign a cipher suite for the SSL proxy list. The cipher suite that you choose must correlate to the certificates and keys that you have either imported to or generated on the CSS. For example, if you choose **all-cipher-suites** you must have an RSA certificate and key, a DSA certificate and key, and a Diffie-Hellman parameter file prior to activating the SSL proxy list.

For each available SSL version, there is a distinct list of supported cipher suites representing a selection of cryptographic algorithms and parameters. Your choice depends on your environment, certificates and keys in use, and security requirements. By default, no supported cipher suites are enabled.

The syntax for this command is:

ssl-server *number* **cipher** *name ip_address* or *hostname port* {**weight** *number*}

The options and variables are:

- **ssl-server** *number* - The number used to identify the virtual SSL server in the SSL proxy list.
- **cipher** *name* - The name of a specific cipher suite (as listed in Table 9-12).
- *ip_address* or *hostname* - The IP address to assign to the backend content rule/server used with the cipher suite. Specify the IP address in either dotted-decimal IP notation (for example, 192.168.11.1) or mnemonic host-name format (for example, myhost.mydomain.com).
- *port* - The TCP port of the backend content rule/server through which the backend HTTP connections are sent.
- **weight** *number* - Optional parameter. Assigns a priority to the cipher suite, with 10 being the highest weight. By default, all configured cipher suites have a weight of 1. When negotiating which cipher suite to use, the SSL module selects from the client list based on the cipher suite configured with the highest weight. A higher weight will bias towards the specified cipher suite. To set the weight for a cipher suite, enter a number from 1 to 10. The default is 1.

For example, to select the *dhe-rsa-with-3des-ede-cbc-sha* cipher suite with an assigned weight of 5, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 cipher
dhe-rsa-with-3des-ede-cbc-sha 192.168.11.1 80 weight 5
```

To remove a specific cipher suite from a specific virtual SSL server, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 cipher
dhe-rsa-with-3des-ede-cbc-sha
```

Table 9-12 lists all supported cipher suites and values for the specific virtual SSL server (and corresponding SSL proxy list). Table 9-12 also lists whether those cipher suites are exportable from the CSS, along with the authentication certificate and encryption key required by the cipher suite.

Caution The dh-anon series of cipher suites are intended for completely anonymous Diffie-Hellman communications in which neither party is authenticated. Note that this cipher suite is vulnerable to attacks.

Cipher suites with "export" in the title indicate that they are intended for use outside of the domestic United States and that they have encryption algorithms with limited key sizes.

Table 9-12 SSL Cipher Suites Supported by the CSS

Cipher Suite	Exportable	Authentication Certificate Used	Key Exchange Algorithm Used
all-cipher-suites	No	RSA certificate, DSA certificate	RSA key exchange, Diffie-Hellman
dhe-dss-export1024-with-rc4-56-sha	Yes	DSA (DSS) certificate	Ephemeral Diffie-Hellman
rsa-export1024-with-rc4-56-sha	Yes	RSA certificate	RSA key exchange
dhe-dss-export1024-with-des-cbc-sha	Yes	DSA (DSS) certificate	Ephemeral Diffie-Hellman
rsa-export1024-with-des-cbc-sha	Yes	RSA certificate	RSA key exchange
dh-anon-export-with-des40-cbc-sha	Yes	Neither party is authenticated	Diffie-Hellman
dh-anon-export-with-rc4-40-md5	Yes	Neither party is authenticated	Diffie-Hellman

Table 9-12 SSL Cipher Suites Supported by the CSS (continued)

Cipher Suite	Exportable	Authentication Certificate Used	Key Exchange Algorithm Used
dhe-rsa-export-with-des40-cbc-sha	Yes	RSA certificate	Ephemeral Diffie-Hellman
dhe-dss-export-with-des40-cbc-sha	Yes	DSA (DSS) certificate	Ephemeral Diffie-Hellman and DSA key exchange
rsa-export-with-des40-cbc-sha	Yes	RSA certificate	RSA key exchange
rsa-export-with-rc4-40-md5	Yes	RSA certificate	RSA key exchange
dhe-dss-with-rc4-128-sha	No	DSA (DSS) certificate	Ephemeral Diffie-Hellman
dh-anon-with-3des-ede-cbc-sha	No	Neither party is authenticated	Diffie-Hellman
dh-anon-with-des-cbc-sha	No	Neither party is authenticated	Diffie-Hellman
dh-anon-with-rc4-128-md5	No	Neither party is authenticated	Diffie-Hellman
dhe-rsa-with-3des-ede-cbc-sha	No	RSA certificate	RSA key exchange
dhe-rsa-with-des-cbc-sha	No	Ephemeral Diffie-Hellman with RSA certificates	Ephemeral Diffie-Hellman and RSA key exchange
dhe-dss-with-3des-ede-cbc-sha	No	DSA (DSS) certificate	Ephemeral Diffie-Hellman
dhe-dss-with-des-cbc-sha	No	DSA (DSS) certificate	Ephemeral Diffie-Hellman
rsa-with-3des-ede-cbc-sha	No	RSA certificate	RSA key exchange
rsa-with-des-cbc-sha	No	RSA certificate	RSA key exchange
rsa-with-rc4-128-sha	No	RSA certificate	RSA key exchange
rsa-with-rc4-128-md5	No	RSA certificate	RSA key exchange

Specifying SSL/TLS Version

Use the **ssl-server** *number* **version** *protocol* command to specify the SSL or Transport Layer Security (TLS) protocol version. The default value is for the support of both SSL protocol version 3.0 and TLS protocol version 1.0.

The options include:

- **ssl-tls** - SSL protocol version 3.0 and TLS protocol version 1.0 (default)
- **ssl** - SSL protocol version 3.0
- **tls** - TLS protocol version 1.0

For example, to specify SSL version 3.0, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 version ssl
```

To reset the SSL version to the default of SSL version 3.0 and TLS version 1.0, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 version
```

Specifying SSL Session Cache Timeout

Use the **ssl-server** *number* **session-cache** *seconds* command to configure the SSL module to resume connection with a client using a previously established secret key. In SSL, a new session ID is created every time the client and the CSS SSL module go through a full key exchange and establish a new master secret key. Specifying an SSL session cache timeout allows the SSL module to reuse the master key on subsequent connections with the client, which can speed up the SSL negotiation process. You can specify a timeout value to set the total amount of time an SSL session ID remains valid before the SSL module requires the full SSL handshake to establish a new SSL connection.

The selection of an SSL session cache timeout value is important when using the **advanced-balance ssl** load balancing method for a Layer 5 content rule to help fine-tune the SSL session ID that is used to stick the client to the server.

Enter an SSL session cache timeout value in seconds, from 0 (SSL session ID reuse disabled) to 72000 (20 hours). The default is 300 seconds (5 minutes). By disabling this option (entering a value of 0), the full SSL handshake occurs for each new connection between the client and the SSL module.

Note Cisco Systems does not recommend specifying a zero value for the **ssl-server** *number* **session-cache** *seconds* command. A non-zero value ensures that the SSL session ID is reused to improve CSS performance.

For example, to configure the reuse of an SSL session ID with a client using a timeout value of 10 hours, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 session-cache 36000
```

To reset the SSL session reuse timeout back to 300 seconds, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 session-cache
```

Specifying SSL Session Handshake Renegotiation

The SSL session handshake commands send the SSL HelloRequest message to a client to restart SSL handshake negotiation. SSL rehandshake is useful when a connection has been established for a lengthy period of time and you want to ensure security by reestablishing the SSL session.

Use the **ssl-server** *number* **handshake data** *kbytes* command to specify the maximum amount of data to be exchanged between the CSS and the client, after which the CSS transmits the SSL handshake message and reestablishes the SSL session. By setting the data value, you force the SSL session to renegotiate a new session key after a session has transferred the specified amount of data. Specify an SSL handshake data value in Kbytes, from 0 (handshake disabled) to 512000. The default is 0.

For example, to configure an SSL rehandshake message for the SSL proxy list after a data exchange of 125000 Kbytes is reached with the client, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 handshake data 125000
```

To disable the rehandshake data option, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 handshake data
```

Use the **ssl-server** *number* **handshake timeout** *seconds* command to specify a maximum timeout value, after which the CSS transmits the SSL handshake message and reestablishes the SSL session. By setting a timeout value you can force the SSL session to renegotiate a new session key after a session has lasted the defined number of seconds. The selection of an SSL rehandshake timeout value is important when using the **advanced-balance ssl** load balancing method for a Layer 5 content rule to help fine-tune the SSL session ID used to stick the client to the server. Specify an SSL handshake timeout value in seconds, from 0 (handshake disabled) to 72000 (20 hours). The default is 0.

For example, to configure an SSL rehandshake message after a timeout value of 10 hours is reached, enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 handshake timeout 36000
```

To disable the rehandshake timeout option, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 handshake timeout
```

Note If a connection is stuck using SSL sticky, be aware that the connection loses SSL sticky persistence each time the CSS performs handshake renegotiation because the SSL session ID regenerates within an existing TCP flow. Because of this, the CSS is not aware of the new SSL session ID. When the next TCP connection comes in for this SSL flow, the CSS treats it like a new SSL connection and load balances the connections to a SSL service. If there is more than one service and multiple SSL modules, the CSS may send the connection to a different SSL module.The connection will be a new SSL connection to that SSL module, which causes the connection to be renegotiated for a second time. After the second renegotiation the CSS is aware of the SSL session ID and the SSL connection sticks to the other SSL module.

In this case, turning on SSL re-handshaking can cause SSL connections to require additional resources to perform handshake renegotiate. If you are operating in a high traffic environment, this could impact overall SSL performance.

Specifying SSL TCP Client-Side Connection Timeout Values

The TCP connection between the CSS and a client is terminated when the specified time interval expires. The TCP timeout functions enable you to have more control over the TCP connection between the CSS SSL module and a client.

To configure an SSL proxy list virtual SSL server for termination of a TCP connection with the client, refer to the following sections:

- Specifying a TCP SYN Timeout Value (Client-Side Connection)
- Specifying a TCP Inactivity Timeout Value (Client-Side Connection)

Specifying a TCP SYN Timeout Value (Client-Side Connection)

Use the **ssl-server** *number* **tcp virtual syn-timeout** *seconds* command to specify a timeout value that the CSS uses to terminate a TCP connection with a client that has not successfully completed the TCP three-way handshake prior to transferring data. The CSS SYN timer counts the delta between the CSS sending the SYN/ACK and the client replying with an ACK as the means to terminate the TCP three-way handshake.

Enter a TCP SYN inactivity timeout value in seconds, from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.

Note The connection timer should always be shorter than the retransmit termination time for new SSL and TCP connections.

For example, to configure a TCP SYN timeout of 30 minutes (1800 seconds), enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual syn-timeout
1800
```

To reset the TCP SYN timeout back to 30 seconds, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp virtual
syn-timeout
```

Specifying a TCP Inactivity Timeout Value (Client-Side Connection)

Use the **ssl-server** *number* **tcp virtual inactivity-timeout** *seconds* command to specify a timeout value that the CSS uses to terminate a TCP connection with the client when there is little or no activity occurring on the connection. The timeout value begins once the CSS receives an ACK from the client to terminate the TCP three-way handshake. The inactivity timer resumes immediately following where the SYN timer stops, in regards to traffic flow.

Enter a TCP inactivity timeout value in seconds, from 0 (TCP inactivity timeout disabled) to 3600 (1 hour). The default is 240 seconds.

For example, to configure a TCP inactivity time of 30 minutes (1800 seconds), enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
inactivity-timeout 1800
```

To reset the TCP inactivity timer back to 240 seconds, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp virtual
inactivity-timeout
```

Specifying SSL TCP Server-Side Connection Timeout Values

The TCP connection between the CSS and a server is terminated when the specified time interval expires. The TCP timeout functions enable you to have more control over TCP connections between the CSS SSL module and a server.

To configure an SSL proxy list virtual SSL server for termination of a TCP connection with the server, refer to the following sections:

- Specifying a TCP SYN Timeout Value (Server-Side Connection)
- Specifying a TCP Inactivity Timeout Value (Server-Side Connection)

Specifying a TCP SYN Timeout Value (Server-Side Connection)

Use the **ssl-server** *number* **tcp server syn-timeout** *seconds* command to specify a timeout value that the CSS uses to abort a TCP connection with a server that has not successfully completed the TCP three-way handshake prior to transferring data. The SYN timer counts the delta between the CSS initiating the backend TCP connection by transmitting a SYN and the server replying with a SYN/ACK.

Enter a TCP SYN timeout value in seconds, from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.

Note The connection timer should always be shorter than the retransmit termination time for new SSL and TCP connections.

For example, to configure a TCP SYN timeout of 30 minutes (1800 seconds), enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server syn-timeout
1800
```

To reset the TCP SYN timeout back to 30 seconds, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp server
syn-timeout
```

Specifying a TCP Inactivity Timeout Value (Server-Side Connection)

Use the **ssl-server** *number* **tcp server inactivity-timeout** *seconds* command to specify a timeout value that the CSS uses to terminate a TCP connection with a server when there is little or no activity occurring on the connection. The timeout value begins once the CSS receives a SYN/ACK from the server. The inactivity timer resumes immediately following where the SYN timer stops, in regards to traffic flow.

Enter a TCP inactivity timeout value in seconds, from 0 (TCP inactivity timeout disabled) to 3600 (1 hour). The default is 240 seconds.

For example, to configure a TCP inactivity time of 30 minutes (1800 seconds), enter:

```
(ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server
inactivity-timeout 1800
```

To reset the TCP inactivity timer back to 240 seconds, enter:

```
(ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp server
inactivity-timeout
```

Activating an SSL Proxy List

Use the **active** command to activate the new or modified SSL proxy list. Before you can activate an SSL proxy list, ensure that you have created at least one virtual SSL server in the list (see "Configuring Virtual SSL Servers for an SSL Proxy List" in this chapter for details).

The CSS checks the SSL proxy list to verify that all of the necessary components are configured, including verification of the certificate and key pair against each other. If the verification fails, the certificate name is not accepted and the CSS logs the error message `Certificate and key pair do not match` and does not activate the SSL proxy list. You must either remove the configured key pair or configure an appropriate certificate.

No modifications to an SSL proxy list are permitted on an active list. Suspend the list prior to making changes, and then reactivate the SSL proxy list once the changes are complete. Once you have modified the SSL proxy list, suspend the SSL service, reactivate the SSL proxy list, then reactivate the SSL service.

To activate an SSL proxy list, enter:

```
(ssl-proxy-list[ssl_list1])# active
```

Note Use the **show ssl-proxy-list** to review the virtual SSL servers in a list (see "Showing SSL Proxy Configuration Information" in this chapter for details). Make modifications as necessary.

Suspending the SSL Proxy List

Use the **suspend** command to suspend an active SSL proxy list.

To suspend an active SSL proxy list, enter:

```
(ssl-proxy-list[ssl_list1])# suspend
```

Adding SSL Proxy Lists to Services

After you configure an SSL proxy list for an SSL module, add the active list to an SSL service to define how the CSS processes SSL requests for content through the specific SSL module. An SSL proxy list may belong to multiple SSL services (one SSL proxy list per service), and an SSL service may belong to multiple content rules. You can apply the services to content rules that allow the CSS to direct SSL requests for content.

Note The CSS supports one active SSL service for each SSL module in the CSS, 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.

This section covers:

- Creating an SSL Service for an SSL Module
- Specifying the SSL Acceleration Service Type
- Specifying SSL Module Slot
- Disabling Keepalive Messages for the SSL Module
- Specifying the SSL Session ID Cache Size
- Activating the SSL Service
- Configuring an SSL Content Rule

Creating an SSL Service for an SSL Module

When creating a service for use with an SSL module, you must identify it as an SSL service for the CSS to recognize it as such. For additional details on creating a service, refer to the *Cisco Content Services Switch Basic Configuration Guide,* Chapter 1, Configuring Services.

Enter the SSL service name, from 1 to 31 characters.

To create service *ssl_serv1*, enter:

```
(config)# service ssl_serv1
Create service <ssl_serv1>, [y/n]: y
```

The CSS transitions into the newly created service mode.

Specifying the SSL Acceleration Service Type

After you create the SSL service, and the CSS enters into service mode, you must specify **ssl-accel** as the service type to:

- Configure the service as an SSL acceleration service.
- Add the SSL proxy list to an SSL service.

Use the **type** command to specify the SSL acceleration service type. For details on specifying an SSL service type, refer to the *Cisco Content Services Switch Basic Configuration Guide,* Chapter 1, Configuring Services.

To specify the SSL acceleration service type, enter:

```
(config-service[ssl_serv1])# type ssl-accel
```

Specifying SSL Module Slot

Use the **slot** command to specify the slot in the CSS chassis where the SSL module is located. The 11500 series CSS supports multiple SSL modules, up to two in a CSS 11503 and up to four in a CSS 11506. The SSL service requires the SSL module slot number to correlate the SSL proxy list and virtual SSL server(s) to a specific SSL module. The valid entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506). Slot 1 is reserved for the SCM.

Note The CSS supports one active SSL service for each SSL module in the CSS (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.

For example, to identify an SSL module in slot 3 of the 11500 series CSS chassis, enter:

```
(config-service[ssl_serv1])# slot 3
```

Disabling Keepalive Messages for the SSL Module

Use the **keepalive type none** command to instruct the CSS not to send keepalive messages to a service. The SSL module is an integrated device within the CSS chassis and, therefore, does not require the use of keepalive messages for the service. For details on specifying a keepalive type, refer to the *Cisco Content Services Switch Basic Configuration Guide,* Chapter 1, Configuring Services.

For disable sending keepalive messages for an SSL service, enter:

```
(config-service[ssl_serv1])# keepalive type none
```

Adding SSL Proxy Lists to a Service

Use the **add ssl-proxy-list** command in service mode to include an SSL proxy list as part of an SSL service. An SSL proxy list may only be added to an SSL service with **ssl-accel** specified as the service type. Enter the name of the previously created SSL proxy list (see "Configuring an SSL Proxy List" in this chapter for details) that you want to add to the service.

To add SSL proxy list *ssl_list1* to service *ssl_serv1*, enter:

```
(config-service[ssl_serv1])# add ssl-proxy-list ssl_list1
```

To remove the SSL proxy list from the service, enter:

```
(config-service[ssl_serv1])# remove ssl-proxy-list ssl_list1
```

Specifying the SSL Session ID Cache Size

Use the **session-cache-size** command to reconfigure 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. By default, the SSL session cache can hold 10000 sessions. If necessary for your SSL service, you can increase the SSL session cache size to 100000. The valid entries are 0 (SSL session cache disabled) to 100000 sessions.

Note Cisco Systems does not recommend specifying a zero value for the **session-cache-size** command to ensure that the SSL session ID is reused. Specifying an SSL session cache and cache timeout allows the reuse of the master key on subsequent connections between the client and the CSS SSL module, which can speed up the SSL negotiation process and improve CSS performance.

If you specify 0 as the SSL session cache size the SSL module associated with the SSL service does not cache any SSL session IDs. If you choose to disable the SSL session cache, ensure the following are properly configured to turn off the use of SSL session ID:

- Set the **ssl-server** *number* **session-cache timeout** setting in the SSL proxy list to 0 (disabled).
- Disable the **advanced-balance ssl** command in the content rule to disable SSL sticky.

For example, to specify an SSL session cache size of 20000 sessions, enter:

```
(config-service[ssl_serv1])# session-cache-size 20000
```

To reset the SSL session cache size to the default of 10000 sessions, enter:

```
(config-service[ssl_serv1])# no session-cache-size
```

Activating the SSL Service

Once you configure an SSL proxy list service, use the **active** command to activate the service. Activating a service puts it into the resource pool for load-balancing SSL content requests between the client and the server.

Before activating an SSL service:

- You must add an SSL proxy list to an **ssl-accel** type service before you can activate the service. If no list is configured when you enter the **active** command, the CSS logs the error message `Must add at least one ssl-proxy-list to an ssl-accel type service` and does not activate the service.
- The SSL proxy list added to the service must be active before you can activate the service. If the list is suspended, the CSS logs the error message `Ssl-proxy-list(s) must be active to activate service` and does not activate the service.

Once the service is ready to activate, the CSS initiates the transfer of appropriate SSL configuration data for each SSL proxy list to a specific SSL module and activates the service. If there is an error in transfer, the CSS logs the appropriate error and does not activate the service.

No modifications may be made to an active SSL proxy list. If modifications are necessary, first suspend the ssl service to make changes to the SSL proxy list entries.

To activate service *ssl_serv1*, enter:

```
(config-service[ssl_serv1])# active
```

Suspending the SSL Service

Use the **suspend** command to suspend an SSL service and remove it from the pool for future load-balancing SSL content requests. Suspending an SSL service does not affect existing content flows, but it prevents additional connections from accessing the service for its content.

You must suspend a service prior to modifying an SSL proxy list.

To suspend service ssl_*serv1*, enter:

```
(config-service[ssl_serv1])# suspend
```

Configuring an SSL Content Rule

To configure a content rule, add one or more SSL services, and activate the content rule, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 3, Configuring Content Rules. No network traffic is sent to an SSL module until you activate an SSL content rule to define where the content physically resides, where to direct the request for content (which SSL service), and which load-balancing method to use.

Ensure that each VIP address and port configured in the content rule matches a VIP address and port configured in one of the SSL proxy list entries in an associated SSL service. When you activate a content rule with a configured SSL service, the CSS verifies that there is a match. If a match is not found, the CSS logs the error message `Not all content VIP:Port combinations are configured in an ssl-proxy-list for sslAccel type of service` and does not activate the content rule. Verify the configured VIP addresses used in the content rule and SSL proxy list, and modify as necessary.

When using two or more SSL modules, Cisco Systems recommends that you use stickiness based on SSL version 3 session ID for a Layer 5 content rule specify the following in the content rule:

- Enable the content rule to be sticky based on SSL using the **advanced-balance ssl** command.
- Specify the SSL application type using the **application ssl** command.

The Layer 5 SSL sticky content rule ensures SSL session ID reuse to eliminate the rehandshake process (which speeds up the SSL negotiation process) and to increase overall performance.

Note If the 32K sticky table becomes full (which means that 32K simultaneous users are on the site) the table wraps and the first users in the table become "unstuck". This may be due to a combination of number of flows and the duration of the sticky period, which can quickly use up the available space in the sticky table. This problem can typically occur in a CSS that contains multiple SSL modules. An SCM with 288M memory module can support a 128K sticky table.

Note If you specify the **sticky-inact-timeout** command 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.

Showing SSL Proxy Configuration Information

Use the **show ssl-proxy-list** command to display information about SSL proxy lists. You can display general information about all SSL proxy lists or detailed information about a specific SSL proxy list.

You can enter the **show ssl-proxy-list** commands from the specified command modes to display configuration information for an SSL proxy list:

- **show ssl-proxy-list**:
  - In ssl-proxy-list mode, this command displays detailed configuration information for the specified SSL proxy list.
  - In global, content, owner, service, SuperUser, and User modes, this command displays general configuration information for all existing SSL proxy lists.
- **show ssl-proxy-list ssl-server** *number* - Displays detailed configuration information for the SSL proxy list and a specific virtual SSL server in the list. This command is available in ssl-proxy-list mode.
- **show ssl-proxy-list** *list_name* - Displays detailed configuration information for the specified SSL proxy list and all virtual SSL servers associated with the list. This command is available in all modes.
- **show ssl-proxy-list** *list_name* **ssl-server** *number* - Displays detailed configuration information for the SSL proxy list and a specific virtual SSL server in the list. This command is available in all modes.

To view general information about all configured SSL proxy lists, enter:

```
# show ssl-proxy-list
```

Table 9-13 describes the fields in the **show ssl-proxy-list** output.

Table 9-13 Field Descriptions for the show ssl-proxy-list Command

Field	Description
Name	The name of the SSL proxy list
Description	The description for the SSL proxy list
State	The state of the SSL proxy list (active or suspended)

Table 9-13 Field Descriptions for the show ssl-proxy-list Command (continued)

Field	Description
Services Associated	The number of services associated with the SSL proxy list
Rules Associated	The number of content rules associated with the SSL proxy list

To display detailed configuration information about *ssl_list1* from the ssl-proxy-list mode, enter:

```
(ssl-proxy-list[ssl_list1])# show ssl-proxy-list
```

To display detailed configuration information about *ssl_list1* from the global configuration mode, enter:

```
(config)# show ssl-proxy-list ssl_list1
```

Table 9-14 describes the fields in the **show ssl-proxy-list** *list_name* output.

Table 9-14 Field Descriptions for the show ssl-proxy-list list_name Command

Field	Description
Description	The description for the SSL proxy list
Number of SSL Proxy ssl-servers	The total number of virtual SSL servers specified for the SSL proxy list
ssl-server	A unique number for the virtual SSL server
VIP Address	The VIP for the virtual SSL server number (corresponding to an SSL proxy list)
VIP Port	The virtual TCP port for the virtual SSL server number (corresponding to an SSL proxy list)
RSA Certificate	The name of the RSA certificate
RSA Keypair	The name of the RSA key
DSA Certificate	The name of the DSA certificate
DSA Keypair	The name of the DSA key pair

Table 9-14 Field Descriptions for the show ssl-proxy-list list_name Command (continued)

Field	Description
DH Param	The name of the Diffie-Hellman parameter association
Session Cache Timeout	The period of time an SSL session ID remains valid before the CSS requires the full SSL handshake to establish a new SSL connection
SSL Version	The specified SSL (version 3.0), TLS (version 1.0), or SSL and TLS protocol in use
Rehandshake Timeout	The period of time the CSS waits before initiating an SSL rehandshake message
Rehandshake Data	The maximum amount of data to be exchanged between the CSS and the client, after which the CSS transmits the SSL handshake message and reestablishes the SSL session
Virtual TCP Inactivity Timeout	The time period that the CSS waits before terminating a TCP connection with a client when there is little or no activity occurring on the connection
Virtual TCP Syn Timeout	The time period that the CSS waits before terminating a TCP connection with a client that has not successfully completed the TCP three-way handshake with the CSS prior to transferring data
Server TCP Inactivity Timeout	The time period that the CSS waits before terminating a TCP connection with a server when there is little or no activity occurring on the connection
Server TCP Syn Timeout	The time period that the CSS waits before terminating a TCP connection with a server that has not successfully completed the TCP three-way handshake with the CSS prior to transferring data
Cipher Suite(s)	The name of the cipher suite(s) assigned to the SSL content rule (see Table 9-12)
Weight	The priority assigned to the cipher suite

Table 9-14 Field Descriptions for the show ssl-proxy-list list_name Command (continued)

Field	Description
Port	The TCP port of the backend content rule/server through which the backend HTTP connections are sent
Server	The IP address assigned to the backend content rule/server used with the cipher suite

Showing SSL Statistics

Use the **show ssl statistics** command to view the statistics for the cryptography components on one or more SSL modules. If you do not specify any options for this command, SSL statistics appear for all SSL modules in the CSS chassis.

The syntax for this command is:

show ssl statistics {*component*} {**slot** *number*}

The options and variables are:

- *component* - Selects a specific component in the SSL module to display statistics. The components include:
  - **ssl-proxy-server** - Displays counter statistics for the SSL proxy list component that provides SSL termination in the SSL module
  - **crypto** - Displays counter statistics for the cryptography chip
  - **ssl** - Displays counter statistics for the SSL server counter
- **slot** *number* - Displays statistics for a component in a specific SSL module in the CSS chassis (assuming more than one module is installed). Specify **slot** *number* after each **show ssl statistics** command. The valid slot entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506). If no slot number is specified, the **show ssl statistics** command displays statistics for all installed SSL modules.

For example, to view all SSL statistics for the SSL module in slot 5 of the CSS chassis, enter:

```
# show ssl statistics slot 5
```

Table 9-15 describes the fields in the **show ssl statistics** output.

Table 9-15 Field Descriptions for the show ssl statistics Command

Field	Description
Component	Indicates the specific component in the SSL module for which statistics are displayed. The SSL statistic functions include: • **ssl-proxy-server** - Displays counter statistics for the SSL proxy list component that provides SSL termination in the SSL module. • **crypto** - Displays counter statistics for the cryptography chip in the SSL module. • **ssl** - Displays counter statistics for the OpenSSL software.
Slot	Indicates the slot number of the SSL module for which statistics are displayed. Valid slots are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506).
SSL Proxy List Statistics	
Handshake started for incoming SSL connections	Number of times the handshake process was initiated for incoming SSL connections from a client to the SSL module.
Handshake completed for incoming SSL connections	Number of times the handshake process was completed for incoming SSL connections from a client to the SSL module.
Handshake started for outgoing SSL connections	Number of times the handshake process was initiated for outgoing SSL connections from the SSL module to a client.
Handshake completed for outgoing SSL connections	Number of times the handshake process was completed for outgoing SSL connections from a client to the SSL module.
Crypto Statistics	
RSA Private	Number of RSA private key calculations requested.
RSA Public	Number of RSA public key calculations requested.

Table 9-15 Field Descriptions for the show ssl statistics Command (continued)

Field	Description
DH Shared	Number of Diffie-Hellman shared secret key calculations requested.
DH Public	Number of Diffie-Hellman public key calculations requested.
DSA Sign	Number of DSA signings requested.
DSA Verify	Number of DSA verifications requested.
SSL MAC	Number of SSL MAC calculations requested.
TLS HMAC	Number of TLS HMAC calculations requested.
3DES	Number of 3 DES calculations requested.
ARC4	Number of ARC4 calculations requested.
HASH	Number of pure hash calculations requested.
RSA Private Failed	Number of RSA private key calculations that failed.
RSA Public Failed	Number of RSA public key calculations that failed.
DH Shared Failed	Number of Diffie-Hellman shared secret key calculations that failed.
DH Public Failed	Number of Diffie-Hellman public key calculations that failed.
DSA Sign Failed	Number of DSA signings that failed.
DSA Verify Failed	Number of DSA verifications that failed.
SSL MAC Failed	Number of SSL MAC calculations that failed.
TLS HMAC Failed	Number of TLS HMAC calculations that failed.
3DES Failed	Number of 3 DES calculations that failed.
ARC4 Failed	Number of ARC4 calculations that failed.
HASH Failed	Number of pure hash calculations that failed.
Hardware Device Not Found	Number of times that a call was made to the cryptography hardware and no hardware acceleration device was available.

Table 9-15 Field Descriptions for the show ssl statistics Command (continued)

Field	Description
Hardware Device Timed Out	Number of times the cryptography hardware did not complete an acceleration request within the specified time. This function is not currently implemented. This counter should always be 0.
Invalid Crypto Parameter	Number of times a hardware acceleration function was requested with an invalid parameter from the CSS. Invalid parameters include an invalid bit length for the operation, a buffer that is not a multiple of 4 bytes in length, a buffer that does not begin on an even 4-byte boundary, requesting an operation on a buffer with too many fragments or too few fragments (such as with no input), or requesting an illegal (nonsense) function.
Hardware Device Failed	Number of times the hardware acceleration device failed. This counter only increments on a DMA error.
Hardware Device Busy	Number of times he hardware acceleration device was busy and could not accept an acceleration request.
Out Of Resources	Number of times no hardware buffers were available and the cryptography hardware could not accept an acceleration request.
Cancelled -- Device Reset	Number of cancelled status returns due to a CSS reset.

Table 9-15 Field Descriptions for the show ssl statistics Command (continued)

Field	Description
SSL Statistics	
RSA Private Decrypt calls	Number of RSA private decryption calls.
RSA Public Decrypt calls	Number of RSA public encryption calls.
DH Compute key calls	Number of Diffie-Hellman Compute key calls.
DH Generate key calls	Number of Diffie-Hellman Generate key calls.
DSA Verify calls	Number of DSA Verifications calls.
DSA Sign calls	Number of DSA Signing calls.
MD5 raw hash calls	Number of MD5 pure hash calls.
SHA1 raw hash calls	Number of SHA1 pure hash calls.
3-DES calls	Number of 3-DES calls.
RC4 calls	Number of RC4 calls.
SSL MAC (MD5) calls	Number of SSL Message Authentication Code (MAC) computations using MD5 algorithm.
SSL MAC (SHA1) calls	Number of SSL MAC computations using SHA algorithm.
TLS MAC (MD5) calls	Number of TLS MAC computations using MD5 algorithm.
TLS MAC (SHA1) calls	Number of TLS MAC computations using SHA algorithm.
Level 1 Alerts Received	Number of Level 1 alerts received.
Level 2 Alerts Received	Number of Level 2 alerts received.
Level 1 Alerts Sent	Number of Level 1 alerts transmitted.
Level 2 Alerts Sent	Number of Level 2 alerts transmitted.
SSL received bytes from TCP	Number of bytes SSL received from TCP.
SSL transmitted bytes to TCP	Number of bytes SSL transmitted to TCP.

Table 9-15 Field Descriptions for the show ssl statistics Command (continued)

Field	Description
SSL received Application Data bytes	Number of Application Data bytes received by the SSL module.
SSL transmitted Application Data bytes	Number of Application Data bytes transmitted by the SSL module.
SSL received non-application data bytes	Number of non-application data (handshake, alert, and change cipher) bytes received by the SSL module.
SSL transmitted non-application data bytes	Number of non-application data (handshake, alert, and change cipher) bytes transmitted by the SSL module.
RSA Private Decrypt failures	Number of RSA Private Decrypt calls that failed.
MAC failures for packets received	Number of times the MAC could not be verified for the incoming SSL messages.
Rehandshake TimerAlloc failed	Number of times unable to allocate timer for Rehandshake Timer.

Clearing SSL Statistics

Use the **clear ssl statistics** command to clear the SSL statistics counters for all SSL modules in the CSS chassis. The reset statistics appear as 0 in the **show ssl statistics** display.

To clear SSL statistics counters for a specific module, use the **clear ssl statistics** command and specify the **slot** *number* following the command. The valid slot entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506).

To clear the SSL statistics counter, enter:

```
# clear ssl statistics
```

Showing SSL Flows

Use the **show ssl flows** command to display information about the active flows for each VIP address, port, and SSL module. The output displays TCP proxy flows, active SSL flows (a subset of TCP proxy flows), and SSL flows occurring during the handshake phase of the protocol (a subset of active SSL flows).

The syntax for this command is:

show ssl flows {**slot** *number*}

The **slot** *number* option displays information about the active flows for a specific SSL module in the CSS chassis (assuming more than one module is installed). The valid slot entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506). If no slot number is specified, the **show ssl flows** command displays statistics for all installed SSL modules.

To view SSL flows for all SSL modules in the CSS, enter:

```
# show ssl flows
```

To view SSL flows for a specific SSL module in the CSS chassis (for example, installed in slot 5), enter:

```
# show ssl flows slot 5
```

Table 9-16 describes the fields in the **show ssl flows** output.

Table 9-16 Field Descriptions for the show ssl flows Command

Field	Description
SSL Acceleration Flows for Slot	Indicates the slot number of the SSL module for which flows are displayed. Valid slots are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506).
Virtual	The virtual address of the ssl-server.
Port	The virtual TCP port of the ssl-server.

Table 9-16 Field Descriptions for the show ssl flows Command (continued)

Field	Description
TCP Proxy Flows	The number of TCP connections which are currently being proxied through the SSL virtual IP address. These connections could either be in: • The TCP handshake or teardown phase and, therefore, not carrying any SSL traffic • The Established TCP phase and carrying SSL traffic
Active SSL Flows	The current number of TCP Proxy Flows that are carrying active SSL connections. These flows are the Established TCP connections in which an SSL Client Hello message has been received by the CSS. The SSL flows remain in this active state until the teardown process is initiated, either by sending or receiving an SSL Alert message. The Active SSL Flows number is a subset of the TCP Proxy Flows column.
SSL Flows in Handshake	The current number of Active SSL Flows that are in the handshake phase of the SSL protocol but are not yet sending data. This means that an SSL Client Hello message has been received by the CSS but the final finished message still has not been sent. The SSL Flows in Handshake number is a subset of the Active SSL Flows column.

Example SSL Proxy Configurations

This section describes the SSL flow process with the SSL module and includes example proxy configurations. Each configuration section includes a running-configuration example and an accompanying illustration.

This section covers:

- Processing of SSL Flows by the SSL Module
- SSL Transparent Proxy Configuration - One SSL Module
- SSL Transparent Proxy Configuration - Two SSL Modules
- SSL Full Proxy Configuration - One SSL Module

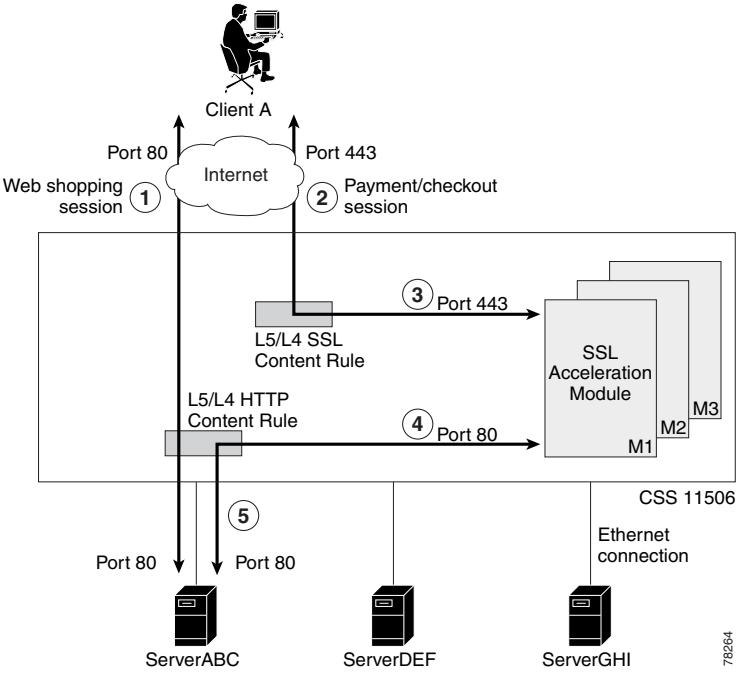
Processing of SSL Flows by the SSL Module

To terminate SSL flows, the SSL module functions as a proxy server, which means that it is the TCP endpoint for inbound SSL traffic. The SSL module maintains a separate TCP connection for each side of the communications, the client side and the server side. The proxy server can perform both TCP and SSL handshakes.

The following example is intended as an overview on the flow process; how the CSS and SSL module translate flows from HTTPS-to-HTTP for inbound packets and from HTTP-to-HTTPS for outbound packets. Figure 9-2 illustrates a CSS with three SSL modules (M1, M2, and M3) configured to off load the SSL traffic from the backend servers (ServerABC, ServerDEF, and ServerGHI). Figure 9-2 also shows the CSS maintaining a consistent stickiness between HTTP and SSL connections from the same client.

1. In a normal Web shopping-cart application, a transaction consists of multiple HTTP connections for shopping or browsing, and a few SSL connections for the final order placement and payment checkout sequence. The client must remain stuck to the same server that holds the customer's database information during the entire transaction. During the initial HTTP connections from a client to a server, the client is stuck to a server by using Layer 5 HTTP cookies or a URL content rule. At checkout, the client transitions to SSL connections.

Figure 9-2 CSS Configuration with Multiple SSL Modules



2. The client transmits the encrypted payment or order information through an SSL connection (TCP SYN received through destination port 443). In this example, when the client connection reaches the CSS, the CSS uses a Layer 5 SSL Session ID sticky content rule to load balance the SSL connection among the three SSL modules (M1, M2, and M3). When the inbound TCP SYN connection reaches the SSL module (the SSL server), it terminates the TCP connections from the client.
3. Once an SSL module is selected (for example, M1), the CSS forwards the SSL packet to that module. The Session ID is saved in the sticky table for subsequent SSL connections from the same client. Once this SSL flow is mapped, the CSS forwards all subsequent packets for this connection to SSL module M1. If there are additional SSL connections associated with this transaction (as determined by the SSL Session ID), the CSS also forwards forwards and maps the packets to SSL module M1.

4. The SSL module terminates the SSL connection and decrypts the packet data. The SSL module then initiates an HTTP connection to a content rule configured on the CSS. The data in this HTTP connection is clear text.
5. The HTTP content rule uses the Layer 5 HTTP cookies or URL sticky content rule on this HTTP request. The cookie or URL string in this clear text HTTP request is used to locate the same server (ServerABC) as the one initially used by the non-SSL HTTP connection in the transactions (for example, online shopping). The CSS forwards the request to the ServerABC and maps this flow. Once the flow is mapped, the return HTTP response from the server is sent to the same SSL module (M1) that sent the original request. The SSL module encrypts the response as an SSL packet (it translates flows from HTTP-to-HTTPS for outbound packets) and sends the packets back to the client through the correct SSL connection.
6. When the TCP connection is finished, the four flows (the two flows between the client and SSL module, and the two flows between the SSL module and the Server) are torn down.

Multiple TCP connections can comprise an entire SSL session. For each of those connections, the same process takes place among the client, SSL module, and server. The SSL Session ID maintains the stickiness between the client and the SSL module and the cookie maintains the stickiness between the SSL module and the servers. In this way, stickiness can be maintained consistently through the entire web transaction.

Note By default, the SSL session cache for the SSL module can hold 10000 sessions. The cache size is the maximum number of SSL session IDs that can be stored in a dedicated session cache on an SSL module. If necessary for your SSL service, use the **session-cache-size** command to reconfigure the size of the SSL session ID cache for the SSL service.

SSL Transparent Proxy Configuration - One SSL Module

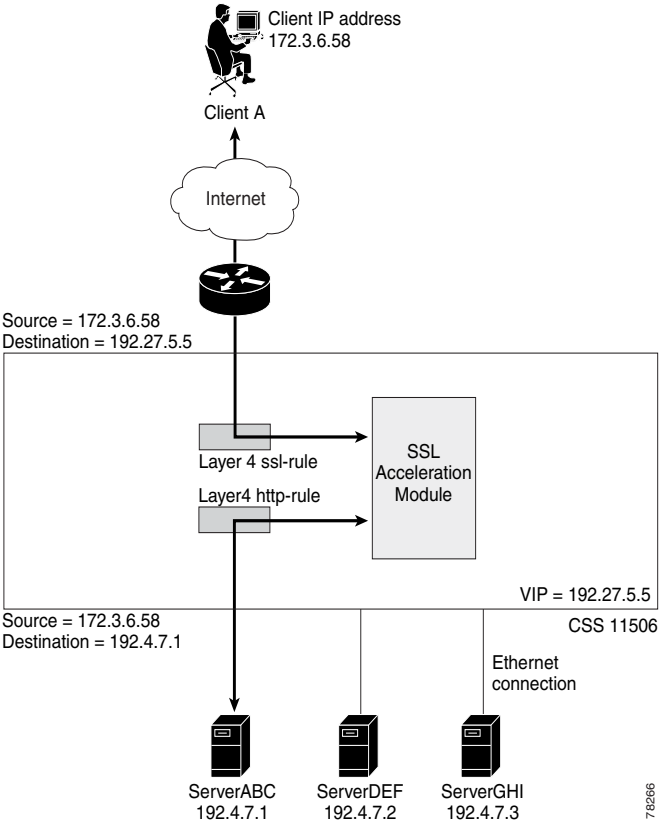
An SSL transparent proxy server is a proxy server that preserves the client's IP address as the source IP address for the backend connection to the server. When you configure an SSL transparent proxy on the CSS, the CSS intercepts and redirects outbound client requests to an HTTP server on the network without changing the source IP address.

This section provides a simple configuration of an SSL transparent proxy between a client, a CSS with a single SSL module, and three HTTP servers (ServerABC, ServerDEF, and ServerGHI). Two content rules are used in this configuration, an SSL content rule and a HTTP content rule. The SSL content rule is for Layer 4 because there is only a single SSL module and there is no need to maintain client-to-server (SSL) stickiness. The use of a Layer 4 content rule in this configuration may improve CSS performance.

Figure 9-3 illustrates this transparent proxy configuration.

For purposes of illustration, the configuration example in Figure 9-3 shows the VIP address for the SSL content rule (ssl-rule) to be the same as the VIP address for the HTTP content rul e (http-rule). These two VIPs do not have to be identical. Depending on the method that you choose to allow access to secure content on your HTTP servers, you may require specification of a different VIP for the clear-text content rule to place it in non-routable address space. In this example, instead of specifying a VIP address of 192.27.5.5 for the http-rule content rule, you could specify a VIP address of 10.1.1.5. The clear-text http-rule will be unreachable from the Internet, which can offer you more flexibility and granularity while allowing the CSS to be seamlessly integrated for secure transactions.

Figure 9-3 Transparent Proxy Configuration with a Single SSL Module




```
!*************************** GLOBAL ****************************
  logging commands enable

  ssl associate dsakey dsakey dsakey.pem
  ssl associate dhparam dhparams dhparams.pem
  ssl associate rsakey rsakey rsakey.pem
  ssl associate cert rsacert rsacert.pem

  ftp-record ssl_record 161.44.174.127 anonymous des-password
deye2gtcld1b6feeeebabfcfagyezc5f /
```

```
!************************* CIRCUIT **************************
circuit VLAN1

  ip address 192.4.8.254 255.255.255.0

circuit VLAN2

  ip address 192.4.7.254 255.255.255.0

!********************** SSL PROXY LIST **********************
ssl-proxy-list test
  ssl-server 111
  ssl-server 111 vip address 192.27.5.5
  ssl-server 111 port 443
  ssl-server 111 rsacert rsacert
  ssl-server 111 rsakey rsakey
  ssl-server 111 cipher rsa-with-rc4-128-md5 192.27.5.5 80
  active

!************************* SERVICE **************************
service ssl_module1
  type ssl-accel
  keepalive type none
  slot 5
  add ssl-proxy-list test
  active

service serverABC
  ip address 192.4.7.1
  protocol tcp
  port 80
  keepalive type http
  active

service serverDEF
  ip address 192.4.7.2
  protocol tcp
  port 80
  keepalive type http
  active

service serverGHI
  ip address 192.4.7.3
  protocol tcp
  port 80
  keepalive type http
  active
```

```
!************************** OWNER ***************************
owner ap.com

content ssl-rule
    vip address 192.27.5.5
    protocol tcp
    port 443
    add service ssl_module1
    active

content http-rule
    vip address 192.27.5.5
    protocol tcp
    port 80
    add service serverABC
    add service serverDEF
    add service serverGHI
    advanced-balance cookies
    active
```

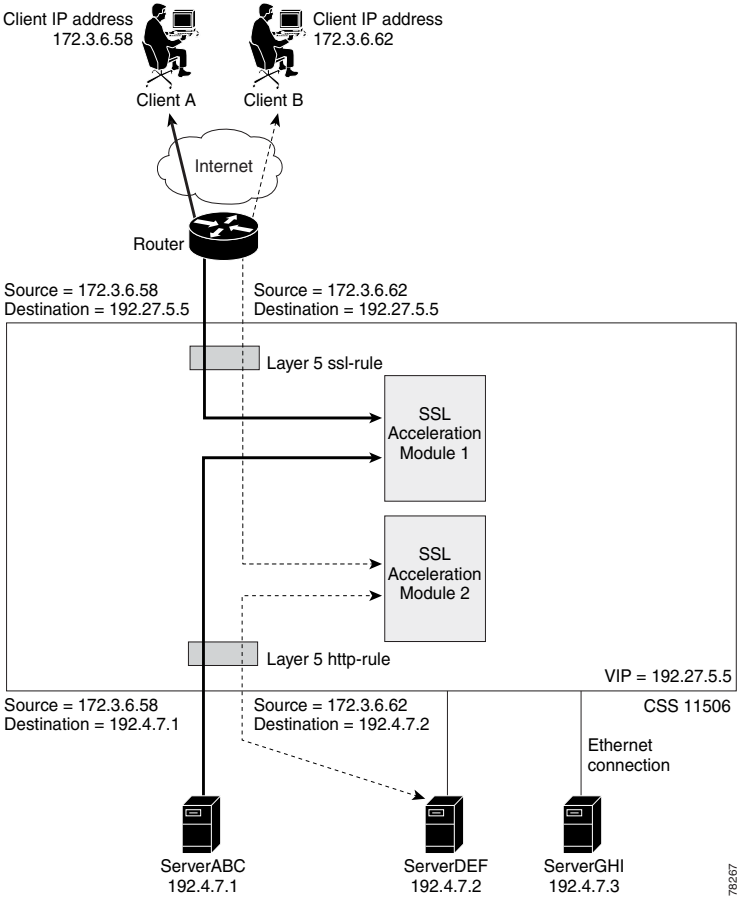
SSL Transparent Proxy Configuration - Two SSL Modules

This section provides an example configuration for an SSL transparent proxy between a client, a CSS with two SSL modules, and three HTTP servers (ServerABC, ServerDEF, and ServerGHI). A Layer 5 SSL sticky content rule is used in the configuration to maintain stickiness of the client to a particular SSL module. The Layer 5 SSL sticky content rule ensures SSL session ID reuse to eliminate the rehandshake process (which speeds up the SSL negotiation process) and to increase overall performance.

Figure 9-4 illustrates this transparent proxy configuration.

For purposes of illustration, the configuration example in Figure 9-4 shows the VIP address for the SSL content rule (ssl-rule) to be the same as the VIP address for the HTTP content rul e (http-rule). These two VIPs do not have to be identical. Depending on the method that you choose to allow access to secure content on your HTTP servers, you may require specification of a different VIP for the clear-text content rule to place it in non-routable address space. In this example, instead of specifying a VIP address of 192.27.5.5 for the http-rule content rule, you could specify a VIP address of 10.1.1.5. The clear-text http-rule will be unreachable from the Internet, which can offer you more flexibility and granularity while allowing the CSS to be seamlessly integrated for secure transactions.

Figure 9-4 Transparent Proxy Configuration with Two SSL Modules




```
!*************************** GLOBAL ****************************
  logging commands enable

  ssl associate dsakey dsakey dsakey.pem
  ssl associate rsakey rsakey rsakey.pem
  ssl associate cert rsacert rsacert.pem
  ssl associate dhparam dhparams dhparams.pem

  ftp-record ssl_record 161.44.174.127 anonymous des-password
deye2gtcld1b6feeeebabfcfagyezc5f /
```

```
!************************* CIRCUIT **************************
circuit VLAN1

  ip address 192.4.8.254 255.255.255.0

circuit VLAN2

  ip address 192.4.7.254 255.255.255.0

!********************** SSL PROXY LIST **********************
ssl-proxy-list test
  ssl-server 111
  ssl-server 111 vip address 192.27.5.5
  ssl-server 111 rsacert rsacert
  ssl-server 111 rsakey rsakey
  ssl-server 111 cipher rsa-with-rc4-128-md5 192.27.5.5 80
  ssl-server 111 port 443
  active

!************************* SERVICE **************************
service ssl_module1
  type ssl-accel
  keepalive type none
  slot 5
  add ssl-proxy-list test
  active

service ssl_module2
  type ssl-accel
  keepalive type none
  slot 6
  add ssl-proxy-list test
  active

service serverABC
  ip address 192.4.7.1
  protocol tcp
  port 80
  keepalive type http
  active

service serverDEF
  ip address 192.4.7.2
  protocol tcp
  port 80
  keepalive type http
  active
```

```
service serverGHI
  ip address 192.14.7.3
  protocol tcp
  port 80
  keepalive type http
  active

!************************** OWNER ***************************
owner ap.com

content ssl-rule
    vip address 192.27.5.5
    protocol tcp
    port 443
    add service ssl_module1
    add service ssl_module2
    application ssl
    advanced-balance ssl
    active

content http-rule
    vip address 192.27.5.5
    protocol tcp
    port 80
    url "/*"
    add service serverABC
    add service serverDEF
    add service serverGHI
    advanced-balance cookies
    active
```

SSL Full Proxy Configuration - One SSL Module

An SSL full proxy server is a proxy server that terminates the client's SSL connections and initiates the backend connection to the HTTP server using a different source IP address than of the client. This configuration does not preserve the client's IP address for the backend connection to the HTTP server.

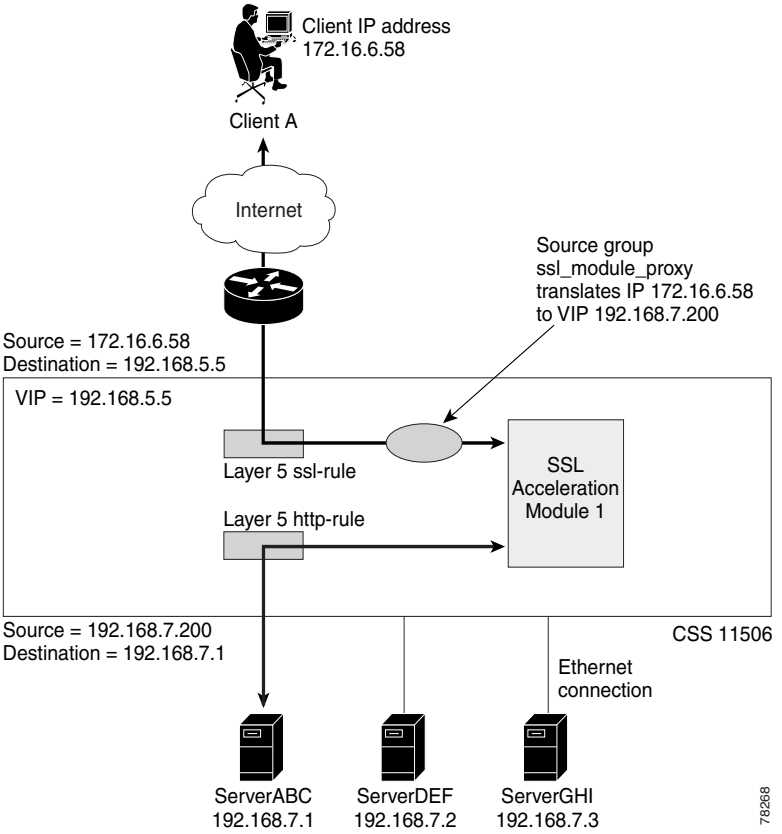
This section provides an example configuration for an SSL full proxy between a client, a CSS with a single SSL module, and three HTTP servers (ServerABC, ServerDEF, and ServerGHI). A Layer 5 sticky content rule is used in the configuration. For the CSS to implement a full proxy configuration with an SSL module, the configuration includes a source group that is used to isolate the SSL module traffic and to NAT its source address.

Figure 9-5 illustrates this full proxy configuration.

For purposes of illustration, the configuration example in Figure 9-5 shows the VIP address for the SSL content rule (ssl-rule) to be the same as the VIP address for the HTTP content rul e (http-rule). These two VIPs do not have to be identical. Depending on the method that you choose to allow access to secure content on your HTTP servers, you may require specification of a different VIP for the clear-text content rule to place it in non-routable address space.

In this example, instead of specifying a VIP address of 192.168.5.5 for the http-rule content rule, you could specify a VIP address of 10.1.1.5. The clear-text http-rule will be unreachable from the Internet, which can offer you more flexibility and granularity while allowing the CSS to be seamlessly integrated for secure transactions.

Figure 9-5 Full Proxy Configuration Using a Single SSL Module




```
!*************************** GLOBAL ****************************
  logging commands enable

  ssl associate dsakey dsakey dsakey.pem
  ssl associate dhparams dhparams dhparams.pem
  ssl associate rsakey rsakey rsakey.pem
  ssl associate cert rsacert rsacert.pem

  ftp-record ssl_record 161.44.174.127 anonymous des-password
deye2gtcld1b6feeeebabfcfagyezc5f /
```

```
!************************** CIRCUIT **************************
circuit VLAN1

  ip address 192.168.5.254 255.255.255.0

circuit VLAN2

  ip address 192.168.7.254 255.255.255.0

!********************** SSL PROXY LIST **********************
ssl-proxy-list test
  ssl-server 111
  ssl-server 111 vip address 192.168.5.5
  ssl-server 111 port 443
  ssl-server 111 rsacert rsacert
  ssl-server 111 rsakey rsakey
  ssl-server 111 cipher rsa-with-rc4-128-md5 192.168.5.5 80
  active

!************************** SERVICE **************************
service ssl_module1
  type ssl-accel
  keepalive type none
  slot 5
  add ssl-proxy-list test
  active

service ssl_module2
  type ssl-accel
  keepalive type none
  slot 6
  add ssl-proxy-list test
  active

service serverABC
  ip address 192.168.7.1
  protocol tcp
  port 80
  keepalive type http
  active
```

```
service serverDEF
  ip address 192.168.7.2
  protocol tcp
  port 80
  keepalive type http
  active

service serverGHI
  ip address 192.168.7.3
  protocol tcp
  port 80
  keepalive type http
  active

!************************** OWNER ***************************
owner ap.com

content ssl-rule
    vip address 192.168.5.5
    protocol tcp
    port 443
    add service ssl_module1
    add service ssl_module2
    application ssl
    advanced-balance ssl
    active

content http-rule
    vip address 192.168.5.5
    protocol tcp
    port 80
    url "/*"
    add service serverABC
    add service serverDEF
    add service serverGHI
    advanced-balance cookies
    active

!************************** GROUP ***************************
group ssl_module_proxy
  add destination service serverABC
  add destination service serverDEF
  add destination service serverGHI
  vip address 192.168.7.200
  active
```

Configuring Firewall Load Balancing

This chapter describes how to configure the Content Services Switch Firewall Load Balancing feature. Information in this chapter applies to all CSS models, except where noted.

This chapter includes the following sections:

Firewall Load Balancing Overview

Firewall load balancing enables you to configure a maximum of 15 firewalls per CSS. Configuring multiple firewalls can overcome performance limitations and remove the single point of failure when all traffic is forced through a single firewall. The firewall load balancing feature ensures that the CSS will forward all packets with the same source and destination IP addresses through the same firewall. The CSS accomplishes this task by performing an XOR on the source and destination IP address.

Because the CSS can exist on either side of a firewall, it can balance traffic over multiple firewalls simultaneously. Each firewall is active and available in the load balancing firewall algorithm. The CSS uses the source and destination IP addresses in the algorithm to calculate which firewall to use for each flow.

Firewall load balancing acts as a Layer 3 device. Each connection to the firewall is a separate IP subnet. All flows between a pair of IP addresses, in either direction, traverse the same firewall. Firewall load balancing performs routing functions; it does not apply content rules to firewall load balancing decisions.

Note Firewalls cannot perform Network Address Translation (NAT). If your configuration requires NATing, you must configure a content rule or source group on the CSS to provide this function.

To configure firewall load balancing, you must define the following parameters for each path through the firewalls on both local and remote CSSs:

- Firewall index (identifies the physical firewall), local firewall IP address, remote firewall IP address, and CSS VLAN IP address
- Static route that the CSS will use for each firewall

Refer to the sections that follow for information on configuring firewall load balancing.

Firewall Synchronization

Firewall solutions providing Stateful Inspection, such as Check Point™ FireWall-1®, create and maintain virtual state for all connections through their devices, even for stateless protocols such as UDP and RPC. This state information, including details on Network Address Translation (NAT), is updated according to the data transferred. Different firewall modules running on different machines, such as those in a firewall load balancing environment, can then share this information by mutually updating each other on the different state information of their connections.

Firewall synchronization (as shown in Figure 10-1) provides a significant benefit whereby each firewall device is aware of all connections in a firewall load balanced environment, making recovery of a failed firewall immediate and transparent to its users.

Note You should refer to your specific firewall documentation for details on configuring firewall synchronization. In the case of a FireWall-1 device, you can find detailed configuration information in the *Check Point Software FireWall-1 Architecture and Administration* guide, in the chapter Active Network Management.

Configuring Firewall Load Balancing

Use the **ip firewall** command to define firewall parameters. You must define these parameters for each path through the firewalls on both local and remote CSSs.

A CSS must exist on each side of the firewall to control which firewall is selected for each flow. Within the firewall configuration, you must configure both the local and remote CSSs with the same firewall index number.

To avoid dropping packets, the CSS directs all packets between a pair of IP addresses across the same firewall. This applies to packets flowing in either direction. If a failure occurs on one path, all traffic will use the remaining path or balance traffic on the remaining paths.

Note You must define the firewall index before you define the firewall route or the CSS will return an error message. To configure the route, see the **ip route... firewall** command.

The syntax for this global configuration mode command is:

ip firewall *index local_firewall_address remote_firewall_address remote_switch_address*

The variables are:

Note Enter all IP addresses in dotted-decimal notation (for example, 192.168.11.1).

- *index* - The index number to identify the firewall. Enter a number from 1 to 254.
- *local_firewall_IP address* - The IP address of the firewall on a subnet connected to the CSS.
- *remote_firewall_IP address* - The IP address of the firewall on the remote subnet that connects to the remote CSS.
- *remote_switch_IP address* - The IP address of the remote CSS.

For example:

```
(config)# ip firewall 1 192.168.27.1 192.168.28.1 192.168.28.3
```

To delete a firewall index, enter:

```
(config)# no ip firewall 1
```

Caution When you delete a firewall index, all routes associated with that index are also deleted.

Configuring a Keepalive Timeout for a Firewall

Use the **ip firewall timeout** *number* command to specify the number of seconds the CSS waits to receive a keepalive message from the remote CSS before declaring the firewall to be unreachable. The two CSS switches at the endpoints of the firewall configuration must use the same firewall keepalive timeout value. Otherwise, routes on one CSS may not failover simultaneously with those on the other CSS. This could permit asymmetric routing to occur across the firewalls.

The timeout range is 3 to 16 seconds. The default is 3 seconds.

Note The amount of time required for a firewall path to become available is unaffected by this command, and remains at three seconds.

For example, to set a timeout of 16 enter:

```
(config)# ip firewall timeout 16
```

To reset the firewall timeout to the default value of three seconds, enter:

```
(config)# no ip firewall timeout
```

Configuring an IP Static Route for a Firewall

Use the **ip route... firewall** command to configure a static route for firewalls. You can optionally set the administrative distance for the IP route.

Note You must define the firewall index before you define the firewall static route or the CSS will return an error message. To configure the firewall index, see the **ip firewall** command.

The syntax for this command is:

ip route *ip_address subnet_mask* **firewall** *index distance*

The variables are:

- *ip_address* - The destination network address. Enter the IP address in dotted-decimal notation (for example, 192.168.11.1).
- *subnet_mask* - The IP subnet mask. Enter the mask in either:
  - CIDR bitcount notation (for example, /24). Do not enter a space to separate the IP address from the prefix length.
  - Dotted-decimal notation (for example, 255.255.255.0).
- *index* - An existing index number for the firewall route. For information on configuring a firewall index, see the **ip firewall** command.
- *distance* - The optional administrative distance. Enter an integer from 1 to 254. A smaller number is preferable. The default value is 1.

Note The CLI prevents you from configuring IP static routes that are firewall routes and IP static routes that are not firewall routes with the same destination addresses and administrative costs. Make either the costs or the addresses unique between firewall and non-firewall routes.

For example:

```
(config)# ip route 192.168.2.0/24 firewall 1 2
```

To remove a firewall route, enter:

```
(config)# no ip route 192.168.2.0/24 firewall 1
```

Configuring OSPF Redistribute Firewall

Use the **ospf redistribute firewall** command to advertise firewall routes from other protocols through OSPF. Redistribution of these routes makes them OSPF external routes.

You can optionally:

- Define the network cost for the route by including the **metric** option. Enter a number from 1 to 16,777,215. The default is 1.
- Define a 32-bit tag value to advertise each external route by including the **tag** option. You can use it to communicate information between autonomous system boundary routers (ASBRs).
- Advertise the routes as ASE type1 by including the **type1** option. The default is ASE type2. The difference between type1 and type2 is how the cost is calculated. For a type2 ASE, only the external cost (metric) is considered when comparing multiple paths to the same destination. For type1 ASE, the combination of the external cost and the cost to reach the ASBR is used.

For example:

```
(config)# ospf redistribute firewall metric 3 type1
```

To stop advertising firewall routes, enter:

```
(config)# no ospf redistribute firewall
```

Configuring RIP Redistribute Firewall

Use the **rip redistribute firewall** command to advertise firewall routes from other protocols through RIP. You may also include an optional metric that the CSS uses when advertising this route. Enter a number from 1 to 15. The default is 1.

For example, to advertise a firewall route through RIP, enter:

```
(config)# rip redistribute firewall 3
```

Note By default, RIP advertises RIP routes and local routes for interfaces running RIP. This command also advertises other routes.

To stop advertising firewall routes, enter:

```
(config)# no rip redistribute firewall
```

Firewall Load Balancing Static Route Configuration Example

This section describes how to configure firewall load balancing for two firewalls between two CSSs. To configure a static route for firewall load balancing, you must define the following parameters for each path through the firewalls on both the local (client) and a remote (server) CSSs:

- Firewall index (identifies the physical firewall), local firewall IP address, remote firewall IP address, and CSS VLAN IP address. You must configure the **ip firewall** command before you configure the static route or the CSS will report an error.
- Static route each CSS will use for each firewall.

To configure CSS-A (the client side of the network configuration) as shown in Figure 10-1:

1. Use the **ip firewall** command to define firewall 1. For example:

```
(config)# ip firewall 1 192.168.28.1 192.168.27.1 192.168.27.3
```

2. Use the **ip route** command to define the static route for firewall 1. For example:

```
(config)# ip route 192.168.2.0/24 firewall 1
```

3. Use the **ip firewall** command to define firewall 2. For example:

```
(config)# ip firewall 2 192.168.28.2 192.168.27.2 192.168.27.3
```

4. Use the **ip route** command to define the static route for firewall 2. For example:

```
(config)# ip route 192.168.2.0/24 firewall 2
```

To configure CSS-B (the server side of the network configuration) as shown in Figure 10-1:

1. Use the **ip firewall** command to define firewall 1. For example:

```
(config)# ip firewall 1 192.168.27.1 192.168.28.1 192.168.28.3
```

2. Use the **ip route** command to define the static route for firewall 1. For example:

```
(config)# ip route 0.0.0.0/0 firewall 1
```

3. Use the **ip firewall** command to define firewall 2. For example:

```
(config)# ip firewall 2 192.168.27.2 192.168.28.2 192.168.28.3
```

4. Use the **ip route** command to define the static route for firewall 2. For example:

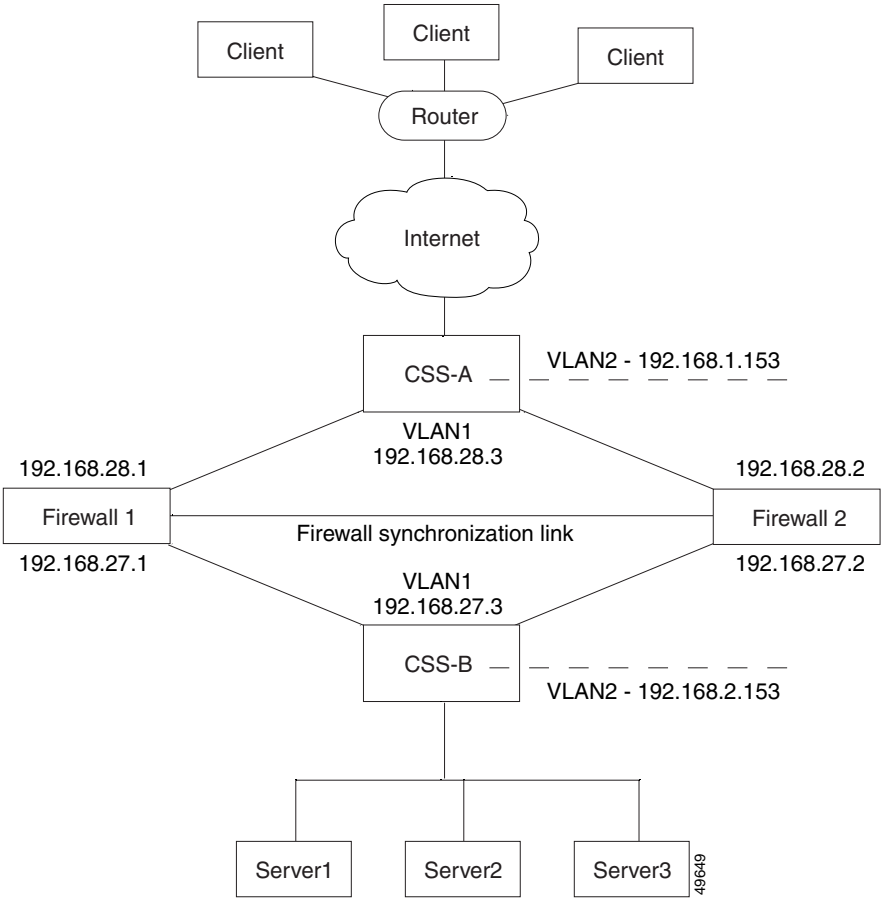
```
(config)# ip route 0.0.0.0/0 firewall 2
```

Firewall configurations are displayed in the IP portion of the running-config. For example:

```
(config)# show running-config
```

Figure 10-1 illustrates the configuration defined in the firewall commands.

Figure 10-1 Firewall Load Balancing Example



Configuring Firewall Load Balancing with VIP/Interface Redundancy

Configure FWLB with VIP redundancy and virtual IP interface redundancy to provide the following benefits:

- Very fast failover (typically 1 to 3 seconds)
- No single point of failure
- All CSSs forward traffic (active-backup configuration)

Note For details on configuring VIP/Interface Redundancy, refer to Chapter 6, Configuring VIP and Virtual IP Interface Redundancy.

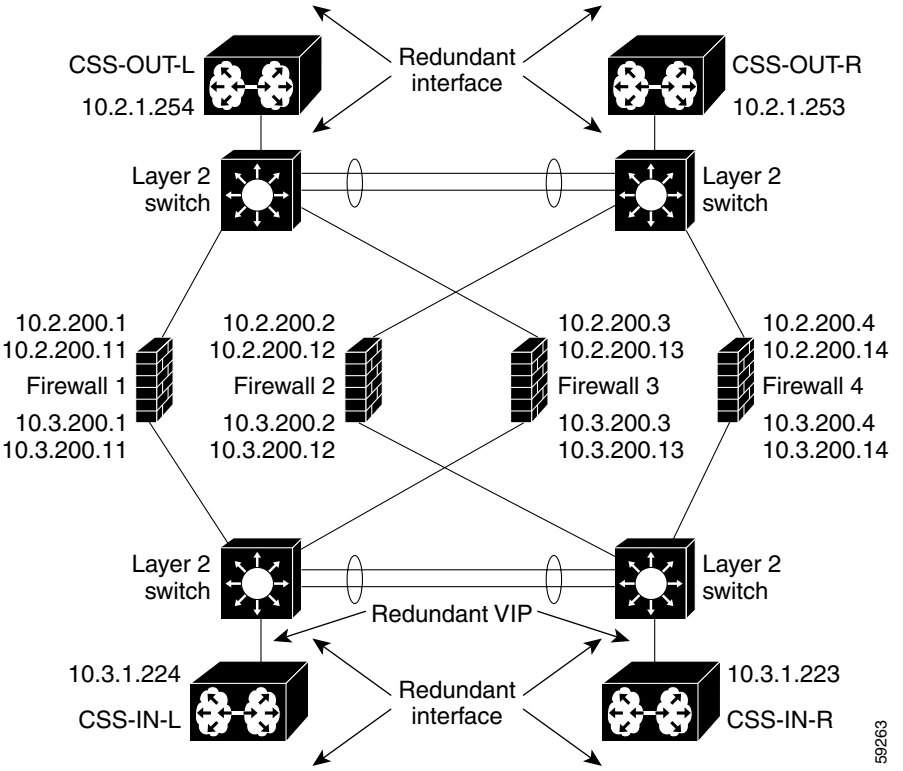
This configuration consists of two redundant CSSs and two L2 devices on either side of the firewall. If a CSS fails, the redundant CSS on the same side of the firewall assumes the additional load.

Note When you configure FWLB with VIP and virtual interface redundancy, do not configure shared VIPs. Shared VIPs are not supported by the FWLB topology.

You must configure the VIPs on the CSS that has the services directly connected to it or connected through a Layer 2 device. Do not configure content rules with VIPs on a CSS when the services are located on the other side of the firewall and connected to another CSS participating in FWLB. This type of configuration will result in asymmetric paths and could cause firewalls performing stateful inspection to tear down connections.

In Figure 10-2, odd-numbered firewalls are connected to the L2 switches servicing the CSS-OUT-L and CSS-IN-L CSSs. Even-numbered firewalls are connected to the L2 switches servicing the CSS-OUT-R and CSS-IN-R CSSs.

Figure 10-2 Firewall Load Balancing with VIP/Interface Redundancy Configuration



Each firewall must have two addresses on either side of it. The first address is used for the next hop on the lower-cost static (primary) path. The second address is used for the next hop on the higher-cost floating-static (secondary) path.

Set the floating-static paths with a higher cost (typically a cost of 10) than those associated with the static paths (typically a cost of 1). If a CSS fails (for example, CSS-OUT-L), CSS-OUT-R will use the higher cost path to send traffic to CSS-IN-L.

If the firewall supports it, you can use multinetting by configuring multiple addresses on the firewall. If the firewall does not support multiple addresses per physical interface, use the ap-kal-fwlb-multinet script to simulate multiple addresses for the firewall. The script takes arguments of "realAddress secondaryAddress". The script creates a static ARP entry for each firewall interface.

Note You can also enter the static ARP entries manually. However, the benefit of the script is that it will change the ARP entries if you replace the firewall and the MAC address changes.

Failover time is very fast at 1 to 3 seconds, because:

- Floating-static path is already up
- Firewall path information has been exchanged
- Circuits are up

If an L2 switch fails, traffic will rehash over every other firewall. If there are an even number of firewalls, 50 percent of the traffic will rehash to the same firewalls.

If you configure redundant interfaces on both sides of a CSS, use critical services to ensure that if one interface fails over to backup, the other interface does the same. If you are implementing multiple interfaces, use firewall interfaces as critical services on external CSSs, and firewall interfaces (configured as service type redundancy-up) and backend servers on internal CSSs.

For details on configuring critical services, refer to Chapter 6, Configuring VIP and Virtual IP Interface Redundancy, in the section "Configuring IP Critical Services". For details on configuring redundant uplink services, refer to Chapter 7, Configuring Redundant Content Services Switches, in the section "Configuring Multiple Redundant Uplink Services".

Example Firewall/Route Configurations

The following **ip firewall** and **ip route** example configurations are valid for Figure 10-2 with four active firewalls.

CSS-OUT-L Configuration

```
ip firewall 1 10.2.200.1 10.3.200.1 10.3.1.224
ip firewall 2 10.2.200.2 10.3.200.2 10.3.1.224
ip firewall 3 10.2.200.3 10.3.200.3 10.3.1.224
ip firewall 4 10.2.200.4 10.3.200.4 10.3.1.224
ip firewall 11 10.2.200.11 10.3.200.11 10.3.1.223
ip firewall 12 10.2.200.12 10.3.200.12 10.3.1.223
ip firewall 13 10.2.200.13 10.3.200.13 10.3.1.223
ip firewall 14 10.2.200.14 10.3.200.14 10.3.1.223
ip route 10.3.0.0 255.255.0.0 firewall 1 1
ip route 10.3.0.0 255.255.0.0 firewall 2 1
ip route 10.3.0.0 255.255.0.0 firewall 3 1
ip route 10.3.0.0 255.255.0.0 firewall 4 1
ip route 10.3.0.0 255.255.0.0 firewall 11 10
ip route 10.3.0.0 255.255.0.0 firewall 12 10
ip route 10.3.0.0 255.255.0.0 firewall 13 10
ip route 10.3.0.0 255.255.0.0 firewall 14 10
```

CSS-OUT-R Configuration

```
ip firewall 11 10.2.200.11 10.3.200.11 10.3.1.223
ip firewall 12 10.2.200.12 10.3.200.12 10.3.1.223
ip firewall 13 10.2.200.13 10.3.200.13 10.3.1.223
ip firewall 14 10.2.200.14 10.3.200.14 10.3.1.223
ip firewall 1 10.2.200.1 10.3.200.1 10.3.1.224
ip firewall 2 10.2.200.2 10.3.200.2 10.3.1.224
ip firewall 3 10.2.200.3 10.3.200.3 10.3.1.224
ip firewall 4 10.2.200.4 10.3.200.4 10.3.1.224
ip route 10.3.0.0 255.255.0.0 firewall 11 1
ip route 10.3.0.0 255.255.0.0 firewall 12 1
ip route 10.3.0.0 255.255.0.0 firewall 13 1
ip route 10.3.0.0 255.255.0.0 firewall 14 1
ip route 10.3.0.0 255.255.0.0 firewall 1 10
ip route 10.3.0.0 255.255.0.0 firewall 2 10
ip route 10.3.0.0 255.255.0.0 firewall 3 10
ip route 10.3.0.0 255.255.0.0 firewall 4 10
```

CSS-IN-L Configuration

```
ip firewall 1 10.3.200.1 10.2.200.1 10.2.1.254
ip firewall 2 10.3.200.2 10.2.200.2 10.2.1.254
ip firewall 3 10.3.200.3 10.2.200.3 10.2.1.254
ip firewall 4 10.3.200.4 10.2.200.4 10.2.1.254
ip firewall 11 10.3.200.11 10.2.200.11 10.2.1.253
ip firewall 12 10.3.200.12 10.2.200.12 10.2.1.253
ip firewall 13 10.3.200.13 10.2.200.13 10.2.1.253
ip firewall 14 10.3.200.14 10.2.200.14 10.2.1.253
ip route 0.0.0.0 0.0.0.0 firewall 1 1
ip route 0.0.0.0 0.0.0.0 firewall 2 1
ip route 0.0.0.0 0.0.0.0 firewall 3 1
ip route 0.0.0.0 0.0.0.0 firewall 4 1
ip route 0.0.0.0 0.0.0.0 firewall 11 10
ip route 0.0.0.0 0.0.0.0 firewall 12 10
ip route 0.0.0.0 0.0.0.0 firewall 13 10
ip route 0.0.0.0 0.0.0.0 firewall 14 10
```

CSS-IN-R Configuration

```
ip firewall 11 10.3.200.11 10.2.200.11 10.2.1.253
ip firewall 12 10.3.200.12 10.2.200.12 10.2.1.253
ip firewall 13 10.3.200.13 10.2.200.13 10.2.1.253
ip firewall 14 10.3.200.14 10.2.200.14 10.2.1.253
ip firewall 1 10.3.200.1 10.2.200.1 10.2.1.254
ip firewall 2 10.3.200.2 10.2.200.2 10.2.1.254
ip firewall 3 10.3.200.3 10.2.200.3 10.2.1.254
ip firewall 4 10.3.200.4 10.2.200.4 10.2.1.254
ip route 0.0.0.0 0.0.0.0 firewall 11 1
ip route 0.0.0.0 0.0.0.0 firewall 12 1
ip route 0.0.0.0 0.0.0.0 firewall 13 1
ip route 0.0.0.0 0.0.0.0 firewall 14 1
ip route 0.0.0.0 0.0.0.0 firewall 1 10
ip route 0.0.0.0 0.0.0.0 firewall 2 10
ip route 0.0.0.0 0.0.0.0 firewall 3 10
ip route 0.0.0.0 0.0.0.0 firewall 4 10
```

Displaying Firewall Flow Summaries

Use the **show flows** command to display the flow summary for a source IP address, or for a specific source address and its destination IP address on a:

- Switch Processor (SP) in an 11500 series CSS. You can display up to 4096 flows per SP.
- Switch Fabric Processor (SFP) in an 11000 series CSS. You can display up to 200 flows per SFP. You can display up to 800 flows on a CSS 11800 with four SFPs.

This information allows you to:

- Identify which firewall is used for a particular flow
- View flows to ensure the proper operation of firewall load balancing

The syntax is:

show flows *source_address destination_address*

The variables are:

- *source_address* - The source IP address for the flows. Enter the address in dotted-decimal format (for example, 192.168.11.1).
- *destination_address* - The destination IP address. Enter the address in dotted-decimal format (for example, 192.168.11.1).

For example:

```
(config)# show flows 192.165.22.1 192.163.2.3
```

To display the flows for a specific source IP address, enter:

```
(config)# show flows 192.165.22.1
```

To display the flows for specific source and destination IP addresses, enter:

```
(config)# show flows 192.165.22.1 192.163.2.3
```

Table 10-1 describes the fields in the **show flows** output.

Table 10-1 Field Descriptions for the show flow Command

Field	Description
Src Address	The source address for the flow
SPort	The source port for the flow
Dst Address	The destination address for the flow
DPort	The destination port for the flow
NAT Dst Address	The NAT destination address
Prot	The protocol of the flow (TCP or UDP)
InPort	The interface port for the in flow
OutPort	The interface port for the out flow

Displaying Firewall IP Routes

Use the **show ip routes firewall** command to display all static firewall routes. For example:

```
(config)# show ip routes firewall
```

Table 10-2 describes the fields in the **show ip routes firewall** output.

Table 10-2 Field Descriptions for the show ip routes firewall Command

Field	Description
Prefix/length	The IP address and prefix length for the route.
Next hop	The IP address for the next hop.
If	The ifIndex value that identifies the local interface through which the next hop of this route should be reached.
Type	The type of the route entry. The type is remote.
Proto	The protocol for the route, firewall.
Age	The maximum age for the route.
Metric	The metric cost for the route.

Displaying Firewall IP Information

Use the **show ip firewall** command to display the configured values of the IP firewall keepalive timeout and the state of each firewall path configured on the CSS. For example:

```
(config)# show ip firewall
```

Table 10-3 describes the fields in the **show ip routes** output.

Table 10-3 Field Descriptions for the show ip routes firewall Command

Field	Description
IP Firewall KAL Timeout	The number of seconds the CSS waits to receive a keepalive message from the remote CSS before declaring the firewall to be unreachable.
Firewall Index	The index number to identify the firewall.
State	The current state of the connection to the remote switch (Init, Reachable, or Unreachable).
Next Hop	The IP address used for the next hop.
Remote Firewall	The IP address of the firewall on the remote subnet that connects to the remote CSS.
Remote Switch	The IP address of the remote CSS.
Time Since Last KAL Tx	The length of time since the last keepalive message was transmitted.
Time Since Last KAL Rx	The length of time since the last keepalive message was received.

Using the CSS Scripting Language

The CSS includes a rich Scripting Language that you can use to write scripts to automate everyday procedures. The Scripting Language is especially useful for writing scripts used by script keepalives for your specific service requirements. For details on script keepalives, refer to the *Cisco Content Services Switch Basic Configuration Guide*, Chapter 1, Configuring Services.

Note Commands shown in the script examples are bolded for clarity.

You can write scripts that use any command in the command line interface (CLI). Scripts can also take advantage of logical blocks that run specific commands based upon a set of conditionals and expressions. This chapter provides information on:

- Playing a Script
- Using the Echo Command
- Using Commented Lines
- Using Variables
- Using Logical/Relational Operators and Branch Commands
- Special Variables
- Using Arrays
- Capturing User Input
- Using Command Line Arguments
- Using Functions

Script Considerations

When you upgrade the CSS software, the old scripts remain in the old software version's /script directory. For details on copying the scripts from the old software directory to the new software directory, see "Script Upgrade Considerations" later in this chapter.

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.

Playing a Script

Use the **script play** command to execute a script line by line from the CLI. You can also use this command to pass parameter values to a script.

For example, enter:

```
script play MyScript "Argument1 Argument2"
```

A script executes from within the /script directory on your local hard disk or flash disk. Only scripts that you put in this directory are accessible to the **script play** command.

You can display a list of available scripts using the **show script** command.

Using the Echo Command

When you play a script, the default behavior of the script is to display each command and its output. Use the **echo** command to control what appears on the screen during script execution. Because the goal of most scripts is to present clear, useful output, it is good practice to disable the **echo** command by using the **no echo** command. The **no echo** command tells the script engine not to print the commands being processed or their output to the terminal. Once you disable the **echo** command, you must use the **echo** command explicitly to print text to the screen.

Echo is enabled automatically upon script termination. For a further explanation of the **echo** and **no echo** commands, see "Using the "!no echo" Comment" later in this chapter.

Note All of the examples and their outputs shown in the remainder of this chapter assume that the **echo** command is disabled, unless otherwise stated.

Using Commented Lines

To write scripts that other users can understand and maintain, it is important to document your script with comments. Comments are particularly important when other users begin using and/or modifying your scripts. To make this process easier for you, you can add commented lines to your scripts. You denote comments with the exclamation mark (!) as the first character in any line of a script.

For example, enter:

```
! I want a comment here so that I can tell you that I will
! execute a show variables command
show variables
```

This example begins with a comment for the user to read (note the exclamation mark). Any characters are allowed after a comment symbol, but the entire line must be a comment. In the third line, the script will execute a **show variables** command because it does not start with an exclamation mark.

This is an example of a valid statement:

```
! Say hello
echo "Hello"
```

This is an example of an invalid statement:

```
echo "Hello" ! Say hello
```

Using the "!no echo" Comment

Use the commented **no echo** command as the first line in any script to disable the **echo** command without the text "no echo" appearing in the script output. A script actually executes the commented command **no echo**. (If you are familiar with MS-DOS batch files, this command is similar to the **@echo off** DOS command.)

For example, enter:

```
! no echo
echo "Hello"
```

The output is:

```
Hello
```

While the statements:

```
! Print Echo
echo "Hello"
```

Actually print:

```
echo "Hello"Hello
```

This is because the script tells the script engine to print the **echo** command to the screen. The result is that the script engine prints the **echo** command and its argument ("Hello") to the screen, followed by the output (Hello) of the command. This is usually not a desired result, so you typically start most scripts with the **!no echo** command as the first line.

Using Variables

The CLI supports user-defined variables that you can use to construct commands, command aliases, and scripts. Variables are case-sensitive and can of type integer or character. They can exist as either single elements or arrays of elements. All arrays are of type character, but their elements can be used in integer expressions. For details on arrays, see "Using Arrays" later in this chapter.

Within a script, you can create and remove variables. During script termination, the script engine automatically removes script-created variables from memory. You can also create a session variable that allows a variable to remain in the session environment after a script has exited. Additionally, if you save a session variable to a user's profile, the variable will be recreated in the CLI session environment upon login. For details on saving a session variable to a user profile, refer to the *Cisco Content Services Switch Administration Guide*, Chapter 6, Configuring User Profiles and CSS Parameters.

A variable name can contain 1 to 32 characters. Its value is quoted text that normally contains alphanumeric characters. Spaces within the quoted text delineate array elements.

Creating Variables

Use the **set** command to create a variable and set the variable with a value. For example, enter:

```
set MyVar "1"
```

This command sets the variable MyVar to a value of 1. You can also set the variable in memory without a value. For example, enter:

```
set MyVar ""
```

This will set the variable equal to NULL (no value). This is different from a value of 0, which is a value. You can set the variable so that it can be used across all CLI sessions. For example, enter:

```
set MyVar "1" session
```

Saving a variable marked with the session keyword to your user profile allows you to use the variable across CLI sessions.

To use a variable, you must supply a variable indicator to allow the CLI to recognize the variable. The CLI searches each line of text it processes for the variable indicator "$". The next character *must* be the left brace character ({) followed by the variable name. You terminate the variable name with the right brace character (}). For example, if you want to print your variable to the screen using the **echo** command, enter:

```
set MyVar "CSS11506"
echo "My variable name is: ${MyVar}"
```

The output is:

```
My Variable name is: CSS11506
```

Variable Types

The CLI stores a variable as either type integer or character. The CLI determines a variable's type by the alphanumeric characters in its value. If any non-numeric characters are present, the variable's type is character. If all the characters are numeric, the variable's type is integer.

Arithmetic operations on quoted numbers such as "100" are possible, but are not possible on variables like "CSS11506" because the CLI knows that "CSS11506" is not a numeric value. You can retrieve the variable type by appending [*] at the end of the command string. For example, enter:

```
set MyVar "CSS11506"
echo "My variable type is ${MyVar}[*]."
set Number "100"
echo "My variable type is ${Number}[*]."
```

The output from this script is:

```
My variable type is char.
My variable type is int.
```

"Int" means integer (numeric with no decimal precision) and "char" means character (any printable ASCII character). Anything that is not an integer is a character. A variable that is defined as "3.14" is a character because the CLI does not consider a period (.) to be a number.

Removing Variables

Use the **no set** command to remove a variable from memory.

For example, enter:

```
no set MyVar
```

If MyVar exists, this line removes the variable from memory. If the variable does not exist, the CLI displays an error message informing you of this invalid action.

Note To permanently remove a session variable, you must also remove it from the user's profile.

Modifying Integer Variables

You can modify a variable's value in the following ways:

- Using the No Set and Set Commands
- Using Arithmetic Operators
- Using the Increment and Decrement Operators

Using the No Set and Set Commands

Use the **no set** command to remove a variable from memory, then use the **set** command with the same variable name to reset the variable with a different value. You can also issue a **set** command on the same variable multiple times without using the **no set** command in between **set** commands.

Example 1:

```
set MyVar "1"
no set MyVar
set MyVar "2"
```

Example 2:

```
set MyVar "1"
set MyVar "2"
```

Example 3:

```
set MyVar1"1"
set Myvar2 "2"
set MyVar1 "${MyVar2}"
```

Note You can also apply the **set** and **no set** commands to character variables.

Using Arithmetic Operators

Use the **modify** command with arithmetic operators (-, +, /, *, or modulus) to change the value of a variable. For example, enter:

```
set MyVar "100"
modify MyVar "+" "2"
echo "Variable value is ${MyVar}."
modify MyVar "-" "12"
echo "Variable value now is ${MyVar}."
modify MyVar "*" "6"
echo "Variable value now is ${MyVar}."
modify MyVar "/" "6"
echo "Variable value now is ${MyVar}."
modify MyVar "MOD" "10"
echo "Variable modulus value now is ${MyVar}"
```

The output is:

```
Variable value is 102.
Variable value now is 90.
Variable value now is 540.
Variable value now is 90.
Variable modulus value now is 0.
```

For simple arithmetic operations, the **modify** command takes an operator in quotes (for example, "/", "*", "+", "-", or "MOD") and a new value in quotes. This value does not have to be a constant (for example, "5" or "10"), but can be another variable (for example, "${Var1}" or "${Var2}"). The modulus operator "MOD" divides the variable by the specified value (in the above example "10") and sets the variable value to the remainder.

The following sections describe additional operations you can perform on variables using the **modify** command. For more information on the **modify** command, refer to the *Cisco Content Services Switch Command Reference*.

Using the Increment and Decrement Operators

The scripting language provides two operators for incrementing and decrementing variable values. The increment operator "++" adds 1 to the variable value and the decrement operator "--" subtracts 1 from the variable value. Use these operators with the **modify** command.

For example, enter:

```
set MyVar "1"
echo "Variable is set to ${MyVar}."
modify MyVar "++"
echo "Variable is set to ${MyVar}."
modify MyVar "--"
echo "Variable is set to ${MyVar}."
```

The output is:

```
Variable is set to 1.
Variable is set to 2.
Variable is set to 1.
```

These two operators make it possible to add or subtract a value without having to type an addition modification command. So you can replace:

```
modify MyVar "+" 1
```

With:

```
modify MyVar "++"
```

Note Both the increment and the decrement operators work only with integer variables. If you use them with character variables, such as "CSS11506", an error occurs.

Using Logical/Relational Operators and Branch Commands

Use the **if** and **while** branch commands to build structured command blocks. The **if** command creates script branches based on the results of previous commands. The **while** command is a looping mechanism. Both commands facilitate efficient scripts with fewer repeated actions.

Use both these branch commands with the **endbranch** command, which indicates to the CLI the end of a logical block of commands. Any branches created without a terminating **endbranch** command produce a script logic error and possibly a script syntax error. For information on script errors, see "Syntax Errors and Script Termination" later in this chapter.

Note You can nest a maximum of 32 levels of branch commands.

Boolean Logic and Relational Operators

You can use each of the following operators with an **if**, **while**, or **modify** command.

The Boolean logic operators used with branch commands are:

- **AND** - Logical AND
- **OR** - Logical OR

The relational operators used with branch commands are:

- **GT** - Greater than
- **LT** - Less than
- **==** - Equal to
- **NEQ** - Not equal to
- **LTEQ** - Less than or equal to
- **GTEQ** - Greater than or equal to

Using the If Branch Command

Use the **if** command to create a branch in a script based on the result of a previous command. If the previous result satisfies the expression in the **if** statement, the script engine executes every line between the **if** and **endbranch** commands. Otherwise, the script engine ignores the intervening commands and execution continues following the **endbranch** statement.

```
set MyVar "1"
if MyVar "==" "1"
    echo "My variable is equal to ${MyVar}!!!"
endbranch
if MyVar "NEQ" "1"
    echo "My variable is not equal to 1 (oh well)."
endbranch
```

In the example above, the script tests the variable MyVar to see if it is equal to "1". If it is equal to this value, the **echo** command between the **if** command and the **endbranch** command is executed. Note that the variable MyVar does not have the typical variable indicator symbol ($) in front of it. This is because the **if** command requires that a constant value or a variable name immediately follow the command.

An exception to this rule applies when the **if** command references an array element. In this case, you must use the normal variable syntax, including the variable indicator ($) and the braces ({ }). For information on arrays, see "Using Arrays" later in this chapter.

For example, the following logical block is valid:

```
if 12 "==" "${MyVar}"
    echo "We made it!"
endbranch
```

However, this logical block is not valid:

```
if "12" "==" "${MyVar}"
    echo "We made it!"
endbranch
```

Because the **if** command expects to see a constant or a variable name (without the variable indicator), the text string "12" does not satisfy this requirement.

You can also test a variable for a NULL value. For example, enter:

```
if MyVar
    echo "MyVar is equal to ${MyVar}"
endbranch
```

Using the While Branch Command

Use the **while** branch command to execute the same commands repeatedly based on the result of an associated expression. When the expression results in a true value (greater then 1), the script engine executes the commands within the branch. If the result is a false value (denoted by the value of 0), then the script breaks out of the loop and continues execution of all the commands following the **endbranch** command. For example, enter:

```
set Counter "0"
while Counter "NEQ" "5"
    echo "Counter is set to ${Counter}."
    modify Counter "++"
endbranch
echo "We're done!"
```

The output of this logical block is:

```
Counter is set to 0.
Counter is set to 1.
Counter is set to 2.
Counter is set to 3.
Counter is set to 4.
We're done!
```

Until the expression is *not* satisfied, notice that the script jumps to the beginning of the loop and evaluates the expression each time it reaches the **endbranch** command. For the first five times through the loop, Counter is set to 0, 1, 2, 3, and 4, respectively, because the script continually increments the variable. However, on the sixth time through the loop, Counter equals a value of 5, which does not satisfy the expression "While Counter is not equal to 5". The expression produces a false result, which causes the loop to terminate.

As with the **if** command, an **endbranch** command terminates a **while** command logical block.

Special Variables

Within the CLI, there are sets of predefined variables that you can use in scripts to add more power to your scripts. Some of these predefined variables change how scripts react, while others are merely informational variables. None of the special variables requires user intervention for cleanup. However, it is good practice to remove both the CONTINUE_ON_ERROR and the EXIT_MSG variables following the execution of their relevant command blocks.

Informational Variables

- **LINE** - Line name (for example, pty1, console, etc.).
- **MODE** - Current mode of command (for example, configure, boot, service).
- **USER** - The currently logged in user (for example, admin, bob, janet).
- **ARGS** - A list of arguments passed to a script from the CLI. See "Using Command Line Arguments" later in this chapter.
- **UGREP** - A line of text obtained using the **grep -u** command.
- **CHECK_STARTUP_ERRORS** - A session variable that determines whether or not a user is informed of startup errors upon login.

CONTINUE_ON_ERROR Variable

Use the CONTINUE_ON_ERROR variable to control how a script that is executing in an interactive CLI session handles command failure. By default, a script terminates when it encounters an error. For example, if you use the **echo** command to print out information from a script and spell the command incorrectly, the script exits with a syntax error and prints out the line in which the error occurred. For example, enter:

```
! Spell echo incorrectly
eco "This will not print"
```

The output is:

```
Error in script playback line:2
>>>eco "Hello"
     ^
%% Invalid input detected at '^' marker.
Script Playback cancelled.
```

Because the script contains a spelling error, the script exits with a message telling you what went wrong.

However, there may be cases where you want a script to continue on error. You can override the default behavior by setting the CONTINUE_ON_ERROR variable. When you set this variable (regardless of its value), a script continues to execute even after it encounters syntax errors or other errors.

Note Exercise caution when using this variable because the CLI ignores syntax errors when the variable is set. You should set and then unset this variable in scripts where you expect a command to fail.

For example, enter:

```
set CONTINUE_ON_ERROR "1"
! Spell echo incorrectly
eco "This will not print"
echo "This will print"
no set CONTINUE_ON_ERROR
```

The output is:

```
This will print
```

Notice in the above example that the script does not print "Script Playback cancelled" and then terminate. This is because the CONTINUE_ON ERROR variable is set. In most situations, it is important that you issue a **no set** command on the CONTINUE_ON_ERROR variable. If you want the script to continue on specific errors, then you have to set the variable and then unset it when you are done. If you do not perform a **no set** command on the variable, then any other syntax errors that occur in the script will not cause an early termination. Remember, setting this variable to a value of 0 does not disable it. To disable the variable's functionality, you must unset it.

STATUS Variable

Use the STATUS variable to return the exit status of the previously executed CLI command. In most cases, except for the **grep** command, an exit status of 0 indicates that a command was successful, while a non-zero value indicates that a command failed. The CLI sets the STATUS variable automatically after each command completes execution.

Note Using the **grep** command sets the STATUS variable equal to the number of lines that satisfied the search. For details on the **grep** command, see "Using the Grep Command" later in this chapter.

Typically, it is not necessary to examine the STATUS variable because a script will terminate if a command does not execute properly. However, if you set the CONTINUE_ON_ERROR variable, you can use the STATUS variable to test the results of a command.

For example, enter:

```
set CONTINUE_ON_ERROR "1"
eco "Hello world"
if STATUS "NEQ" "0"
    echo "Failure to execute command correctly"
endbranch
```

In the above example, the STATUS variable is set to a non-zero value. This value is specific to the type of error that occurred. In this case, the script receives a general syntax error, which informs you that the command being executed failed. This is a typical example and one that you should watch closely when using the CONTINUE_ON_ERROR variable. In most circumstances, you will want to catch syntax errors as real errors.

Note When writing scripts, keep in mind that the value of the STATUS variable changes as each command executes. If you intend to use a STATUS value later in a script, you should save the value of the STATUS variable in another variable.

In the following example, notice that the error that is most likely to occur is not a syntax error, but an error with the command being initiated. In this case, a nonzero value results in a "Failure to connect to remote host" message. This could be a special case where you want to catch the error and then perform another action instead.

```
set CONTINUE_ON_ERROR "1"
socket connect host 1.1.1.1 port 9
if STATUS "NEQ" "0"
    echo "Failure to connect to remote host"
endbranch
no set CONTINUE_ON_ERROR
```

EXIT_MSG Variable

Use the EXIT_MSG variable to tell the CLI to print out a custom message when a script exits. Typically, you set this variable to a string value that is printed when the script terminates. Set this variable to prepare for potential errors, and unset it using the **no set** command before the script exits cleanly. This variable allows you to take advantage of the CLI's exit upon error behavior, while permitting the flexibility of customizing the error message. For example, enter:

```
set EXIT_MSG "Failure to connect to host"
socket connect host 1.1.1.1 port 9
no set EXIT_MSG
```

The example above shows how you can create a custom error message that will print "Failure to connect to host" if the **socket connect** command returns a non-zero STATUS variable. When this occurs (unless the CONTINUE_ON_ERROR variable is set), the script terminates automatically and the CLI prints the EXIT_MSG string to the screen.

If the **socket connect** command succeeds, then the CLI executes the next command in the script. In this case, the script performs a **no set** EXIT_MSG command. This allows the script to terminate normally without printing an exit message to the screen, which would be inappropriate because no error occurred.

SOCKET Variable

When you use the **socket** commands, the SOCKET variable is set automatically (see "Using Socket Commands" later in this chapter). The SOCKET variable contains the connection ID associated with a host. When you connect to a remote host, the SOCKET variable is set so that you can send and receive messages by referring to this variable. When you make multiple connections using the **socket** commands, save the SOCKET variable to another variable or it will be overwritten.

For example, enter:

```
set EXIT_MSG "Failure to connect to host"
socket connect host 1.1.1.1 port 80
no set EXIT_MSG
set EXIT_MSG "Send: Failure"
socket send ${SOCKET} "GET /index.html\n\n"
no set EXIT_MSG
! Save current socket ID
set OLD_SOCKET "${SOCKET}"
! The new socket connect command will overwrite the old
! ${SOCKET} variable
set EXIT_MSG "Failure to connect to host"
socket connect host 1.1.1.1 port 80
no set EXIT_MSG
set EXIT_MSG "Send: Failure"
socket send ${SOCKET} "GET /index.html\n\n"
no set EXIT_MSG
set EXIT_MSG "Waitfor: Failed"
socket waitfor ${OLD_SOCKET} "200 OK"
socket waitfor ${SOCKET} "200 OK"
! Finished, cleanup
no set EXIT_MSG
socket disconnect ${OLD_SOCKET}
socket disconnect ${SOCKET}
```

Using the Show Variable Command

Use the **show variable** command to display all the variables currently set in the CSS software environment.

The CLI uses the following special variables in its operation to control session behavior and to enhance interaction with CLI commands and the user:

- The USER variable is set automatically to the username starting the CLI session at login time.
- The LINE variable is set automatically to the line which the user is connected to at login time.
- The MODE variable is set automatically to the current mode as the user navigates the hierarchy of CLI modes.
- The STATUS variable is set automatically to return the exit status of the previously executed CLI command. In most cases, with the exception of the "grep" command, an exit status of 0 indicates a command was successful, and a non-zero value indicates failure.
- The CHECK_STARTUP_ERRORS variable, if set within a profile script, indicates the user should be informed of startup-errors upon login. If the startup-errors file is found in the log directory, the screen displays the "***Startup Errors occurred on boot.***" message.
- The CONTINUE_ON_ERROR variable controls how a script executing in an interactive CLI session handles a command error. When you set this variable in a script with the **set** command, the execution of a script continues when errors are detected. If you do not set this variable in a script, the script terminates when an error occurs.

  You should exercise caution when using this variable. Syntax errors are ignored when it is set. You should set this variable in the script where you expect a command to fail and then disable it with the **no set** command.

For example, enter:

```
show variable
```

The output is:

```
$MODE = super
$LINE = console
$CHECK_STARTUP_ERRORS = 1 *Session
$UGREP =   Weight:  1           Load:  255
$SOCKET = -1
$USER = admin
$STATUS = 0
```

Notice in this example that there are several variables already defined in the environment. You can also display a specific variable by invoking the **show variable** command with a parameter that represents the variable name you want to see.

For example, enter:

```
show variable LINE
```

The output is:

```
$LINE = console
```

Using Arrays

A variable can hold subvalues (elements) within its memory space. Such a variable is commonly called a *variable array* or just an *array.* An array can hold numeric values, strings, or both. To create an array, simply create a variable using the **set** command and separate all of the array elements by spaces. For example, enter:

```
set WeekDays "Sun Mon Tues Wed Thurs Fri Sat"
```

You can print the array like any other variable. For example, enter:

```
echo "Days of the week: ${WeekDays}."
```

The output is:

```
Days of the week: Sun Mon Tues Wed Thurs Fri Sat.
```

However, if you want to print out each day separately, you must refer to a single element within the array. For example, enter:

```
echo "The first day of the week is ${WeekDays}[1]."
echo "The last day of the week is ${WeekDays}[7]."
```

The output is:

```
The first day of the week is Sun.
The last day of the week is Sat.
```

Notice that you reference the array elements by putting the element number within brackets ([and]) to tell the CLI which element you want to use. You append the brackets to the end of the variable name (including variable indicator and braces).

Note The CSS Scripting Language numbers elements starting at 1, not at 0. Some scripting/programming languages "zero index" their arrays; this scripting system does not.

If you reference an element beyond the boundary of the array, you receive a syntax error. For example, enter:

```
echo "The last day of the week is ${WeekDays}[8]"
%% Error variable syntax
```

This occurs because there is not an eighth element in the WeekDays array.

Element Numbers

To find out how many elements are in an array, pass the pound symbol (#) rather then an element value. For example, if you want to know how many days are in a week, enter:

```
set WeekDays "Sun Mon Tues Wed Thurs Fri Sat"
echo "There are ${WeekDays}[#] days in a week."
```

The output is:

```
There are 7 days in a week.
```

You can use this method to figure out how many times to perform a **while** command. For example, enter:

```
set WeekDays "Sun Mon Tues Wed Thurs Fri Sat"
set Counter "1"
while Counter "LTEQ" "${WeekDays}[#]"
    echo "Counter is set to ${Counter}."
    modify Counter "++"
endbranch
```

The output is:

```
Counter is set to 1.
Counter is set to 2.
Counter is set to 3.
Counter is set to 4.
Counter is set to 5.
Counter is set to 6.
Counter is set to 7.
```

Using Var-shift to Obtain Array Elements

You may need to print out all the days of the week to the screen. While it is possible to print out each array element by hardcoding the element values, it is not always practical or possible to do this. In this case, it is obvious that there are seven days in the week, but there may be cases where the number of elements in a variable are unknown until the script is executed.

Use the **var-shift** command to push an element out of an array one at a time. For example, enter:

```
set WeekDays "Sun Mon Tues Wed Thurs Fri Sat"
while ${WeekDays}[#] "GT" "0"
    ! Push the 1st element out of the array and shift all
    ! elements up one position.
    echo "Day: ${WeekDays}"
    var-shift WeekDays
endbranch
```

The output of this logical block is:

```
Day: Sun
Day: Mon
Day: Tues
Day: Wed
Day: Thurs
Day: Fri
Day: Sat
```

You cannot use a variable as the index to obtain a given element in an array. So the following statement is invalid:

```
set Pet "Dog Cat"
set Index "1"
echo "First Pet is: ${Pet}[${Index}]"
modify Index "+" 1
echo "Second Pet is: ${Pet}[${Index}]"
```

However, the following statement can solve this problem:

```
set Pet "Dog Cat"
echo "First Pet is: ${Pet}[1]"
var-shift Pet
echo "Second Pet is: {$Pet}[1]"
```

Notice in the second Pet example that there is one less variable needed (you do not need the Index variable) and the first element index (${Pet}[1]) is the only one used.

The **var-shift** command deletes the original data within the variable. The variable Pet from the example above now contains only the element "Cat". To solve this problem, save the contents of the original variable in a temporary variable.

For example, enter:

```
set Pet "Dog Cat"
set Temp "${Pet}"
echo "First Pet is: ${Temp}[1]"
var-shift Temp
echo "Second Pet is: {$Temp}[1]"
no set Temp
```

By using the Temp variable, you leave the original variable untouched. If you decide you no longer need the Temp variable, remove it from memory with the **no set** command.

Capturing User Input

Use the **input** command to capture user input in a variable. You can use this command to create a script to assist users in setting up configuration files or preparing a CSS in some predefined way. For example, enter:

```
! Ask the user for his/her full name
echo "What is your full name?"
input FULL_NAME
echo "Hello ${FULL_NAME}!"
```

In the example above, notice that the **input** command has a variable argument called FULL_NAME. Notice that you do not need to set FULL_NAME prior to the **input** command. The command creates the variable and fills it with the data that the user supplies. Also, note that the user's input is terminated by a carriage return.

A user can enter any alphanumeric characters. If a user presses only the Enter key and does not type any characters, then the script creates the variable with a NULL value. This allows you to test the user input to verify that the user typed something. In the following example, the script continues to ask the user the same question until the user types "y".

```
echo -n "\please enter the character 'y' to exit."
input DATA
while DATA "NEQ" "y"
    echo -n "Please enter the character 'y' to exit: "
    input DATA
    echo "\n"
endbranch
```

In the example above, notice the use of the **echo** command with the *-n* parameter. This allows you to prompt the user with a message without forcing a new line at the end of the sentence so that the user's data appears on the same line as the **echo** command output. In the **echo** command quoted string, you can embed "\n", a C-programming style character that puts a line feed in the output to make it more readable.

Using Command Line Arguments

The CLI allows a user to pass command line arguments as quoted text to a script using the **script play** command (see "Playing a Script"). Use the special reserved ARGS variable to access the command line arguments that a user passed to a script.

To print out all the arguments that a user passed to a script, enter:

```
echo "You passed the arguments: ${ARGS}"
```

If you want to access each argument individually, you can use the ARGS variable as an array, where each argument passed on the command line is separated by spaces. For example, enter:

```
echo "Your first argument passed is: ${ARGS}[1]"
```

The script below (called NameScript) prints a user's first and last names. The script requires that the user pass his/her first and last name (in that order) in quoted text to the script. For example, enter:

```
!no echo
if ${ARGS}[#] "NEQ" "2"
    echo "Usage: NameScript \'First_Name Last_Name\'"
    exit script 1
endbranch
echo "First Name: ${ARGS}[1]"
echo "Last Name: ${ARGS}[2]"
exit script 0
```

This script first tests to see if the user passed any arguments. If the user did not pass exactly two arguments, then the script prints usage information to the screen and exits. If the user passed two arguments, the script assumes that the first argument is the user's first name and the second argument is the user's last name. Finally, the script prints the user's first name and last name to the screen.

For example, to play NameScript, enter:

```
script play NameScript "John Doe"
```

The output is:

```
First Name: John
Last Name: Doe
```

Using Functions

Functions provide a way to organize your scripts into subroutines or modules. You can then call these functions as your script requires them.

Using the Function Command

Use the **function** command to modularize your scripts both for ease of reading and simplification. This command allows both the creation and calling of script functions. For example, enter:

```
echo "Calling the PrintName function"
function PrintName call
echo "End"
! Function PrintName: Prints the name John Doe
function PrintName begin
echo "My Name is John Doe"
function PrintName end
```

The output is:

```
Calling the PrintName function
My Name is John Doe
End
```

Notice that the command issued between the commands **function PrintName begin** and **function PrintName end** executes before the last **echo** statement in the script. Also note that the script automatically terminates after the last valid line before the function definition.

Using Function Arguments

You can pass a list of arguments to a function. (This is similar to passing command line arguments to a script.) You can also use those arguments when the script calls the function. For example, enter:

```
echo "Calling the PrintName function"
function PrintName call "John Doe"
echo "End"
! Function PrintName: Prints the name John Doe
function PrintName begin
echo "My Name is ${ARGS}"
function PrintName end
```

The output is:

```
Calling the PrintName function
My Name is John Doe
End
```

Notice that the script uses the ARGS variable to hold the passed arguments. Just as command line arguments work with the **script play** command, function arguments work with the **function call** command.

If you pass command line arguments to a script in the **script play** command and use function arguments in the script, then the ARGS variable is stored until the function returns from its execution.

For example, suppose you pass two arguments "Billy Bob" to a script using the **script play** command and the script calls a function called PrintName with different arguments, enter:

```
echo "I was passed the arguments ${ARGS}"
function PrintName call "John Doe"
echo "The original arguments are ${ARGS}"
! Function PrintName: Prints the name John Doe
function PrintName begin
echo "My Name is ${ARGS}"
function PrintName end
```

The output is:

```
I was passed the arguments Billy Bob
My Name is John Doe
The original arguments are Billy Bob
```

Although you use the ARGS variable twice in this script, ARGS had two different values because the function call held its own ARGS variable value independently of the main script's ARGS variable value.

Now you can modularize your scripts for easier reading and maintenance.

Using the SCRIPT_PLAY Function

Use the SCRIPT_PLAY function to call a script from another script. The syntax is:

function SCRIPT_PLAY **call** *"ScriptName arg1 arg2..."*

When you use the SCRIPT_PLAY function, use caution when dealing with the **exit script** command. For details on exiting from a script, see "Exiting a Script Within Another Script" later in this chapter.

Bitwise Logical Operators

The CSS Scripting Language provides two operations for bit manipulation that apply only to numeric values. The bitwise logical operators are:

- **BAND** - Bitwise AND operation
- **BOR** - Bitwise OR operation

You can manipulate the bits of a numeric variable using the **modify** command. For example, if you have a numeric value of 13 and want to find out if the second and fourth bits of the number are turned on (value of 1), you can do so using a bitwise AND operation as follows:

```
        00001101 (13)
AND     00001010 (10)
```

Result is 00001000 (8)

Notice that the fourth bit is common between the values 13 and 10, so the resulting value contains only the common bits. To script this, do the following:

```
set VALUE "13"
modify VALUE "BAND" "10"
echo "Value is ${VALUE}"
```

The output is:

```
Value is 8
```

Note The script overwrites the original value of the variable VALUE. You can use the BOR operator in a similar manner. The mechanics of AND and OR logical operations is beyond the scope of this document.

Syntax Errors and Script Termination

You can create complex scripts using the CSS Scripting Language. During the development of a script, you may encounter a few syntax errors. When a script exits because of a syntax error, the CLI indicates the script line number and the script text where the syntax error occurred.

The CLI also allows you to exit from a script and specify the exit value (zero or non-zero) using the **exit script** command. This lets you know exactly why a script failed. You can then use this information to initiate a decision process to handle the error condition.

Syntax Errors

When you play a script that contains a misspelled command, an unknown command, or a failed command, the script displays a syntax error on the screen. The error message contains the number and the text of the line in the script that caused the failure. The CSS software sets the STATUS variable to the appropriate error code (a non-zero value).

For example, enter:

```
Error in script playback in line: 1
>>>eco "Hello"
     ^
%% Invalid input detected at '^' marker.
Script Playback cancelled.
```

If a script issues a command that returns a non-zero STATUS code, this will also result in a syntax error. For example, if you play a script that issues a **socket connect** command and the host refuses the connection or there is a hostname resolution failure, the script terminates with a syntax error.

For example, enter:

```
!no echo
socket connect host 192.168.1.1 port 84 tcp
socket disconnect ${SOCKET}
```

This script works well as long as neither command fails. However, suppose that the host 192.168.1.1 does not exist. Playing the script produces an error as follows:

```
Error in script playback line:2
>>>socket connect host 192.168.1.1 port 84 tcp
CSS11506#
Script Playback cancelled.
```

The script failed because of an error in line 2. Notice that the command is not misspelled and all syntax is correct. However, the command issued a non-zero error code because of its failure to connect. The next command, **socket disconnect**, never executes because of the failure of the first command.

The CLI considers this type of error a "syntax error". To discover what went wrong, issue the questionable command directly on the CLI.

For example, enter:

```
socket connect host 192.168.1.1 port 84 tcp
%% Failed to connect to remote host
```

The CLI displays the reason why the command failed.

Note If there are no misspellings in a script, it is good practice to test the commands on the CLI.

Script Exit Codes

When a script terminates, it returns an exit code. The software places the exit code value in the STATUS variable (for reference after the script is invoked). There are two possible script exit code values: zero (success) and non-zero (failure).

To ensure a successful exit code, use the **exit script** command with a value of zero (the default). The integer value is optional with this command.

For example, enter:

```
! Exit Cleanly
exit script 0
```

There may be some cases where you want to indicate that a script failed to run properly. One example of this is when your script requires that a user enter one or more command line arguments. There is no syntax checking that will prove that the user supplied the correct arguments, but you can check if the user supplied the correct number of arguments. For example, enter:

```
if ${ARGS}[#] "NEQ" "2"
    echo "Usage: PingScript \'HostName\'"
    exit script 1
endbranch
```

If this script fails to find exactly two arguments on the command line, it exits with status code 1 (failure). If you were to check the STATUS variable at this point, it would be set to a value of 1.

Note All commands in the CLI write an exit code to the STATUS variable after they execute. If you want to use the STATUS value, you must save it in another variable or use it right away.

For example, enter:

```
script play PingScript
echo "Status: ${STATUS}"
echo "Status: ${STATUS}"
```

The output is:

```
Usage: PingScript "HostName"
Status: 1
Status: 0
```

Because the script contains an **exit script** command with a value of 1, the first **echo** command returns a STATUS value of 1 to indicate that the script failed. The second **echo** command returns a STATUS value of 0 because the first **echo** command executed successfully.

Exiting a Script Within Another Script

It is possible to play a script from another script using the function SCRIPT_PLAY. For details on playing a script, see "Using the SCRIPT_PLAY Function" earlier in this chapter. In this case, be careful when dealing with the **exit script** command in the secondary script.

If script A invokes script B and script B issue an **exit script** command, both scripts will exit. Therefore, it is important that a script calling another script either removes any **exit script** commands in the second script or makes other arrangements to handle this behavior.

Using the Grep Command

Use the **grep** command with the -u option to search for specified data and place the last line of the search results in a variable called UGREP. For example, to create a script to search for the Keepalive field in the **show service** command on a service called S1, enter:

```
!no echo
show service S1 | grep -u "Keepalive"
echo "The line is: ${UGREP}"
```

The output is:

```
The line is: Keepalive: (SCRIPT a-kal-pop3 10 3 5)
```

Because the **show service** screen contains the field Keepalive, the entire line is stored in the UGREP variable. You can also extract each space-separated element by treating the UGREP variable as an array. For example, to extract the first block of text, enter:

```
!no echo
show service S1 | grep -u "Keepalive"
echo "The first element in the line is: ${UGREP}[1]"
```

The output is:

```
The first element in the line is: Keepalive:
```

Specifying Line Numbers for Search Results

The search results from a **grep** command can produce multiple lines. In this case, you can specify the line number you want to set in the UGREP variable by issuing **grep** -u*[n]*, where *[n]* is an optional line number. By default, the last line of the search results is saved in the UGREP variable. Suppose that you want to issue a **grep** command on the **show service** command for the service S1 and search for all the colon (:) characters in the **show service** screen. For example, enter:

```
!no echo
show service S1 | grep -u ":"
echo "The line is: ${UGREP}"
```

The output is:

```
The line is: Weight: 1       Load: 255
```

By default, this is the last line in the **show service** screen that satisfies the search criteria, but it is not the only line that qualifies. To search for a specific line, for example the first line that satisfies the search criteria, use the *[n]* option. For example, enter:

```
!no echo
show service S1 | grep -u1 ":"
echo "The line is: ${UGREP}"
```

The output is:

```
The line is: Name: S1           Index: 1
```

This is the first line that satisfies the search criteria. Notice the -u1 option, which tells the script to search for the colon character (:) and set the UGREP variable equal to the first qualified search result.

STATUS Results from the Grep Command

The STATUS code result from the **grep** command equals the number of qualified search results. This is important to note because other script commands return a zero result to indicate success. The grep command returns just the opposite. If it finds 14 matches, then the STATUS variable is set to 14. If it finds no matches, then the STATUS variable is set to 0.

If you search for the string “Keepalive” as illustrated in the previous section, you will find one result. In this case, the STATUS variable is set to 1.

You can combine the status code returned from the **grep** command with a **while** loop to step through all the search results line-by-line.

For example, enter:

```
show service S1 | grep ":"
set endIndex "${STATUS}"
set index "1"
while index "LTEQ" "${endIndex}"
    show service S1 | grep -u${index} ":"
    echo "${UGREP}"
    modify index "++"
endbranch
```

Using Socket Commands

Use **socket** commands in a script keepalive to assist you in building a structured protocol. The socket commands allow ASCII or hexadecimal send and receive functionality. Each command that has an optional **raw** keyword converts the data from standard ASCII to hexadecimal. For example, “abcd” is “61626364” in ASCII. In hex, it is “0x61 0x62 0x63 0x64”.

Socket Connect

Use the **socket connect** command to performs either a TCP connection handshake (SYN-SYNACK...) to a specific IP address/port or a UDP connection by reserving the host/port. The socket value is received in a ${SOCKET} variable in the script.

The syntax for this command is:

socket connect host *ip_address* **port** *number* {**tcp** {*timeout*}|**udp**} {**session**}

Note The software can open a maximum of 64 sockets simultaneously across all scripts on a CSS.

The options and variables are:

- **host** - Keyword that must be followed by the hostname or IP address of the remote CSS.
- *ip_address* - The hostname or the IP address of the remote CSS.
- **port** - Keyword that must be followed by the port on which to negotiate a connection.
- *number* - The port number on which to negotiate a connection.
- **tcp** - Keyword that specifies a connection using TCP.
- **udp** - Keyword that specifies a connection using UDP.
- *timeout* - Timeout value for making the network connection in milliseconds. If the time limit expires before the connection has been successfully made, then the attempt fails. This applies only to a TCP connection, because UDP is "connectionless". Enter an interger from 1 to 60000 ms (1 to 60 seconds). The default is 5000 ms (5 seconds).
- **session** - Keyword that tells the socket to remain open until the session ends. If a script opens sockets in the session and does not close them, the sockets remain open until you log out.

Socket Send

Use the **socket send** command to write data through a previously connected TCP connection.

The syntax for this command is:

socket send *socket_number "string"* {**raw** | **base64**}

- *socket_number* - Socket file descriptor (integer form). This descriptor is returned from the **socket connect** command.
- *string* - Quoted text string with a maximum of 128 characters.
- **raw** -The optional keyword that causes the software to interpret the string values as hexadecimal bytes rather than as a simple string. For example, the software converts "0D0A" to "0x0D 0x0A" (carriage return, line feed).
- **base64** - Encodes the string in the base-64 numbering system before sending it through the connection. This option is useful for HTTP basic authentication when connecting to a password-protected web site.

Socket Receive

Use the **socket receive** command to fill up the socket's 10-KB internal buffer with data from the remote host. Once the buffer is full, the command locks the buffer so that no new data can be placed in the buffer. You can use the **socket inspect** command to send all data residing in this 10-KB buffer to standard output.

Note The software removes all previous data from the 10-KB internal buffer before it stores new data.

The syntax for this command is:

socket receive *socket_number* {*timeout*} {**raw**}

The options and variables are:

- *socket_number* - Socket file descriptor (integer form). This descriptor is returned by the **socket connect** command.
- *socket_number* - Socket file descriptor (integer form). This descriptor is returned by the **socket connect** command.
- *timeout* - The optional timeout value that specifies the number of milliseconds the CSS software waits before the script locks the internal 10-KB buffer and resumes execution. Enter an integer from 1 to 15,000 ms. The default is 100 ms.
- **raw** - The optional keyword that causes the software to interpret the string values as hexadecimal bytes rather than as a simple string. For example, the software converts "0D0A" to "0x0D 0x0A" (carriage return, line feed).

Socket Waitfor

Use the **socket waitfor** command to fill up the socket's 10-KB internal buffer with data from the remote host. This command is similar to **socket receive** except that it returns immediately upon finding the specified *string* argument. Once the CSS finds the specified string, it returns a ${STATUS} value of 0 (success). Otherwise, it returns 1. You can further view the retrieved data using the **socket inspect** command, as described later in this section.

The syntax for this command is:

socket waitfor *socket_number* [**anything** {*timeout*}|*"string"* {*timeout*} {**case-sensitive**} {**offset** *bytes*} {**raw**}]

The options and variables are:

- *socket_number* - Socket file descriptor (integer form). The descriptor value is returned by the **socket connect** command.
- **anything** - Any incoming data returns the call within the timeout period. If any data is found, the command returns immediately and does not wait the entire timeout period.
- *timeout* - The optional timeout value that specifies the number of milliseconds the CSS waits to find the *string* argument. Enter an integer from 1 to 15000 ms. The default is 100 ms.
- *string* - The specific string that the CSS must find to result in a ${STATUS} value of 0. Once the CSS finds the string, the command returns immediately and does not wait the entire timeout period specified by the *integer* argument.
- **case-sensitive** - The optional keyword specifying that the string comparison is case sensitive. For example, "User:" is not equivalent to "user:".
- **offset** *bytes* - The optional keyword and value indicating the number of bytes after the beginning of the received data to find the string. For example, if you specify a string value of a0 and an offset of 10, then the CSS will look for "a0" 10 bytes after the beginning of the received data.
- **raw** - The optional keyword that causes the software to interpret the string values as hexadecimal bytes rather than as a simple string. For example, the software converts "0D0A" to "0x0D 0x0A" (carriage return, line feed).

Socket Inspect

This command inspects the socket's internal data buffer for actual data. If the software finds data, it displays the data to standard output. If the displayed characters are non-printable, the software represents them with a period character (.) for readability. (See the example under the *pretty* argument, below.)

The syntax for this command is:

socket inspect *socket_number* {**pretty**}{**raw**}

The options and variables are:

- *socket_number* - Socket file descriptor (integer form). This descriptor is returned by the **socket connect** command.
- **raw** - Displays the string values as hexadecimal bytes instead of a simple string. For example, instead of printing “ABCD” to standard output, it prints “41424344” (1-byte hexadecimal equivalent).
- **pretty** - Prints each line with both the hexadecimal and the ASCII equivalent for each byte of data. The software prints up to 16 bytes on each line. For example, enter: “0x41 0x42 0x43 0x44 0x10 0x05 ABCD.“

Socket Disconnect

This command closes the connection to the remote host by sending RST (reset) to the remote host so that it knows that the CSS is finished sending data.

The syntax for this command is:

socket disconnect *socket_number* {**graceful**}

- *socket_number* - Socket file descriptor (integer form). This descriptor is returned by the **socket connect** command.
- **graceful** - Closes the connection gracefully by sending FIN (finished) rather than RST to the remote host.

Socket Administration

You can have a maximum of 32 open (in use) sockets on a CSS concurrently. If you issue a **socket connect** command and do not issue a **socket disconnect** command for that socket's file descriptor (saved in the ${SOCKET} variable), then the socket remains open until you issue a **socket disconnect** command with that socket’s file descriptor as the value for the *socket_number* argument.

If you open sockets within a script, the sockets close automatically when the script ends, unless you passed the *session* argument in the **socket connect** command. If you open sockets within a session, the sockets close when the session ends (user logs out).

Use the **show sockets** command to list all the socket file descriptors that are currently in use.

Table 11-1 describes the fields in the **show sockets** output.

Table 11-1 Field Descriptions for the show sockets Command

Field	Description
Socket ID	Socket file descriptor
Remote Host:Port	The connected Host address and Port pair
User	The line identifier as displayed through the **show lines** command
Time	How long the descriptor has been open

Note If a remote host times out or closes a socket, the socket architecture cleans up the socket and removes it from the list of used sockets. This cleanup occurs only after you attempt another transfer on a socket that the remote host has already closed. Otherwise, the socket remains idle.

Using the **socket receive** command to retrieve data buffers 10 KB of data at one time. This buffer remains unchanged until you issue another **socket receive** command. At that point, the software clears the buffer and then refills it with more data from the remote host. Each socket descriptor (created by the **socket connect** command) has its own 10-KB buffer.

Script Upgrade Considerations

When you upgrade to a new version of CSS software, all script files that you have modified in the previous software version's /script directory need to be copied to the new software version's /script directory or else the scripts remain in the old /script directory and the CSS will not find them.

Follow these steps *before* upgrading your CSS software:

1. Use the **archive script** command in SuperUser mode to archive each script file. For details on archiving a script, refer to the *Cisco Content Services Switch Administration Guide*, Chapter 1, Logging in and Getting Started.
2. Upgrade your CSS software. For details on upgrading the CSS software version, refer to the *Cisco Content Services Switch Administration Guide*, Appendix A, Upgrading Your CSS Software.
3. Use the **restore script** command in SuperUser mode to restore the scripts to the /script subdirectory of the new software version. For details for restoring a script, refer to the *Cisco Content Services Switch Administration Guide*, Chapter 1, Logging in and Getting Started.

Using the Showtech Script

Use the **showtech** script to gather information designed to assist the Cisco Technical Assistance Center (TAC) in analyzing your CSS. The output of the script displays a set of CSS status and configurations settings that the TAC can use for problem resolution. Use the output-capturing capabilities of your application connected to the CSS to save the script output for analysis.

You can also save the script output to the file log/showtech.out by entering **y** at the prompt as shown below. You can then copy the output file and send it to the TAC if necessary.

To run the script, enter:

```
# script play showtech

Save output to disk [y/n]? y
Output will be saved to log/showtech.out
Please wait...
```

If you enter **n**, then the output appears on your screen.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: showtech
!
! Description:
!       show tech-support script
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

set CONTINUE_ON_ERROR "1"

no terminal more
llama

show clock
show disk
show running-config
flow statistics
show service-internal
show service summary
show service
show system-resources
show dump-status
show core
show circuit all
show arp
show ip route
show phy
show summary
show rule
show group
show ether-errors
show keepalive
show ip stat
show rmon
show bridge status
show bridge forwarding
show interface
show virtual-routers
show critical-services
show redundancy
show chassis inventory
show chassis verbose
show log sys.log tail 200
exit
terminal more

set CONTINUE_ON_ERROR "0"

exit script 0
```

Script Keepalive Examples

The following sections provide examples of script keepalives. You can use them as is or modify them for your applications.

Example of a Custom TCP Script Keepalive with Graceful Socket Close

Use the following script keepalive to open and gracefully close (using a FIN rather than a RST) a socket on user-specified TCP ports.

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-tcp-ports
! Parameters: Service Address, TCP Port(s)
!
!Description:

! This script will open and close a socket on the user specified
! ports.
! The close will be a FIN rather than a RST. If one of the ports fails
! the service will be declared down
!
! Failure Upon:
! Not establishing a connection with the host on one of the specified
! ports.
!
! Notes:  Does not use output
!     Will handle out of sockets scenario.
!
! Tested: KGS 12/18/01
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

set OUT-OF-SOCKETS "785"
set NO-CONNECT "774"

! Make sure the user has a qualified number of arguments
if ${ARGS}[#] "LT" "2"
  echo "Usage: ap-kal-tcp-ports \'ipAddress tcpPort1 [tcpPort2
tcpPort3...]\'"
  exit script 1
endbranch
```

```
set SERVICE "${ARGS}[1]"
!echo "SERVICE = ${ARGS}[1]"
var-shift ARGS

while ${ARGS}[#] "GT" "0"
  set TCP-PORT "${ARGS}[1]"
  var-shift ARGS
  function SOCKET_CONNECT call
! If we're out of sockets, exit and look for sockets on the next KAL
interval
  if RETURN "==" "${OUT-OF-SOCKETS}"
    set EXIT_MSG "Exceeded number of available sockets, skipping until
next interval."
    exit script 0
  endbranch

! Valid connection, look to see if it was good
  if RETURN "==" "${NO-CONNECT}"
    set EXIT_MSG "Connect: Failed to connect to
${SERVICE}:${TCP-PORT}"
    exit script 1
  endbranch
endbranch

no set EXIT_MSG
exit script 0

function SOCKET_CONNECT begin
  set CONTINUE_ON_ERROR "1"
  socket connect host ${SERVICE} port ${TCP-PORT} tcp 2000
  set SOCKET-STAT "${STATUS}"
  set CONTINUE_ON_ERROR "0"
  socket disconnect ${SOCKET} graceful
  function SOCKET_CONNECT return "${SOCKET-STAT}"
function SOCKET_CONNECT end
```

Default Script Keepalives

The script keepalives listed below are included in the /script directory of your CSS and defined in the sections following the list:

- SMTP KEEPALIVE
- NetBIOS Name Query (Microsoft Networking))
- HTTP List Keepalive
- POP3 Keepalive
- IMAP4 Keepalive
- Pinglist Keepalive
- Finger Keepalive
- Time Keepalive
- Setcookie Keepalive
- HTTP Authentication Keepalive
- DNS Keepalive
- Echo Keepalive
- HTTP Host Tag Keepalive
- Mailhost Keepalive

SMTP KEEPALIVE

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-smtp
! Parameters: HostName
!
! Description:
!  This script will log into an SMTP server and send a 'helo'
!  to make sure the SMTP server is stable and active.
!
! Failure Upon:
!   1. Not establishing a connection with the host.
!   2. Failure to get a good status code after saying 'helo'
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Make sure the user has a qualified number of arguments
if ${ARGS}[#] "NEQ" "1"
    echo "Usage: ap-kal-smtp \'Hostname\'"
    exit script 1
endbranch

! Defines:
set HostName "${ARGS}[1]"

set EXIT_MSG "Connection Failed"
! Connect to the remote host (use default timeout)
socket connect host ${HostName} port 25 tcp

set EXIT_MSG "Waitfor: Failed"
! Receive the incoming status code 220 "welcome message"
socket waitfor ${SOCKET} "220" 200

set EXIT_MSG "Send: Failed"
! Send the helo to the server
socket send ${SOCKET} "helo ${HostName}\n"

set EXIT_MSG "Waitfor: Failed"
! Wait for status code "250" to be returned
socket waitfor ${SOCKET} "250" 200

! We've successfully logged in, the server is up and running.
! The job was done successfully.
socket disconnect ${SOCKET}

no set EXIT_MSG
exit script 0
```

NetBIOS Name Query (Microsoft Networking)

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-netbios
! Parameters: Hostname
!
! Description:
!   We will make a netbios name query that we know will be
!   a "negative" response.  RFC-1002 NETBIOS states that a hex
!   value of:
!    0x81 Session Request
!    0x82 Positive Session Response
!    0x83 Negative Session Response
!   We will key off of 0x83 which states we failed, but which
!   also means that the service was stable enough to know that
!   we are not a valid machine on the network.
!   This script will send an encoded message for Session Request
!   (0x81) and will invent a CALLER and a CALLED machine name
!   (Caller = this script and CALLED = Server)
!
! Failure Upon:
!   1. Not establishing a connection with the host.
!   2. Not receiving a status code 0x83 (negative response)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if ${ARGS}[#] "NEQ" "1"
    echo "Usage: ap-kal-pop3 \'Hostname\'"
    exit script 1
endbranch

! Defines:
set HostName "${ARGS}[1]"

! Connect to the remote host (default timeout)
set EXIT_MSG "Connection failure"
socket connect host ${HostName} port 139 tcp

! Send a Netbios Session Request (0x81) and its required encoded
!values.
! This value will be sent in RAW Hex
set EXIT_MSG "Send: Failure"
socket send ${SOCKET}
810000442045454550454d454d464a434143414341434143414341434143414341434143414341
434100" raw
! Wait for a response code of 0x83
set EXIT_MSG "Waitfor: Failure"
socket waitfor ${SOCKET} "83" raw
```

```
no set EXIT_MSG
socket disconnect ${SOCKET}
exit script 0
```

HTTP List Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-httplist
! Parameters: Site1 WebPage1 Site2 WebPage2 [...]
!
! Description:
!   This script will connect a list of sites/webpage pairs.  The
!   user must simply supply the site, and then the webpage and
!   we'll attempt to do an HTTP HEAD on that page.
!
! Failure Upon:
!   1. Not establishing a connection with the host.
!   2. Not receiving a status code 200 on the HEAD request on any
!      one site.  If one fails, the script fails.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Make sure the user has a qualified number of arguments
if ${ARGS}[#] "LT" "2"
    echo "Usage: ap-kal-httplist \'WebSite1 WebPage1 WebSite2 WebPage2
...'"
    exit script 1
endbranch

while ${ARGS}[#] "GT" "0"
    set Site "${ARGS}[1]"
    var-shift ARGS

    if ${ARGS}[#] "==" "0"
     set EXIT_MSG "Parameter mismatch: hostname present but webpage
was not"

     exit script 1
    endbranch
    set Page "${ARGS}[1]"
    var-shift ARGS
    no set EXIT_MSG
    function HeadUrl call "${Site} ${Page}"
endbranch
exit script 0
function HeadUrl begin
```

```
echo "Getting ${ARGS}[1] from Site ${ARGS}[2]\n"
! Connect to the remote Host
set EXIT_MSG "Connect: Failed to connect to ${ARGS}[1]"
socket connect host ${ARGS}[1] port 80 tcp

! Send the head request
set EXIT_MSG "Send: Failed to send to ${ARGS}[1]"
socket send ${SOCKET} "HEAD ${ARGS}[2] HTTP/1.0\n\n"

! Wait for the status code 200 to be given to us
set EXIT_MSG "Waitfor: Failed to wait for '200' on ${ARGS}[1]"
socket waitfor ${SOCKET} " 200 "

no set EXIT_MSG
socket disconnect ${SOCKET}

function HeadUrl end
```

POP3 Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-pop3
! Parameters: HostName UserName Password
!
! Description:
!   This script will connect to a POP3 server and login with the
!   username/password pair specified as argument 2 and 3.  After which
!   it will log out and return.
!
! Failure Upon:
!   1. Not establishing a connection with the host.
!   2. Not being able to log in with supplied username/password.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if ${ARGS}[#] "NEQ" "3"
    echo "Usage: ap-kal-pop3 \'Hostname UserName Password\'"
    exit script 1
endbranch
! Defines:
set HostName "${ARGS}[1]"
set UserName "${ARGS}[2]"
set Password "${ARGS}[3]"

set EXIT_MSG "Connection Failed"
! Connect to the remote host (use default timeout)
socket connect host ${HostName} port 110 tcp
```

```
set EXIT_MSG "Waitfor: Failed"
! Wait for the OK welcome message for 200ms
socket waitfor ${SOCKET} "+OK" 200

set EXIT_MSG "Send: Failed"
! Send the username to the host
socket send ${SOCKET} "USER ${UserName}\n"

set EXIT_MSG "Waitfor: Failed"
! Wait for confirmation
socket waitfor ${SOCKET} "+OK" 200

set EXIT_MSG "Send: Failed"
! Send the password
socket send ${SOCKET} "PASS ${Password}\n"

set EXIT_MSG "Waitfor: Failed"
! Wait for confirmation
socket waitfor ${SOCKET} "+OK" 200

! We've successfully logged in, the server is up and going.
! The job was done successfully.
socket disconnect ${SOCKET}

no set EXIT_MSG
exit script 0
```

IMAP4 Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-imap4
! Parameters: HostName UserName Password
!
! Description:
!   This script will connect to a IMAP4 server and login with the
!   username/password pair specified as argument 2 and 3.  After which
!   it will log out and return.
!
! Failure Upon:
!   1. Not establishing a connection with the host.
!   2. Not being able to log in with supplied username/password.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if ${ARGS}[#] "NEQ" "3"
    echo "Usage: ap-kal-imap4 \'Hostname UserName Password\'"
    exit script 1
```

```
endbranch

! Defines:
set HostName "${ARGS}[1]"
set UserName "${ARGS}[2]"
set Password "${ARGS}[3]"

set EXIT_MSG "Connection Failed"
! Connect to the remote host (use default timeout)
socket connect host ${HostName} port 143 tcp

set EXIT_MSG "Waitfor: Failed"
! Wait for the OK welcome message for 600ms
socket waitfor ${SOCKET} "OK" 600

set EXIT_MSG "Send: Failed"
! Send the username to the host
socket send ${SOCKET} "a1 LOGIN ${UserName} ${Password}\n"

set EXIT_MSG "Waitfor: Failed"
! Wait for confirmation
socket waitfor ${SOCKET} "a1 OK" 200

set EXIT_MSG "Send: Failed"
! Send the password
socket send ${SOCKET} "a2 LOGOUT\n"

set EXIT_MSG "Waitfor: Failed"
! Wait for confirmation
socket waitfor ${SOCKET} "a2 OK" 200

! We've successfully logged in, the server is up and going.
! The job was done successfully.
socket disconnect ${SOCKET}

no set EXIT_MSG
exit script 0
```

Pinglist Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-pinglist
! Parameters: HostName1 HostName2 HostName3, etc.
!
! Description:
!   This script is designed to ping a list of hosts that the user
!   passes in on the command line.
!
! Failure Upon:
!   1. Not being able to ping any one of the hosts in the list
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if ${ARGS}[#] "LT" "1"
    echo "Usage: ap-kal-pinglist \'HostName1 HostName2 HostName3
...\'"
    exit script 1
endbranch

while ${ARGS}[#] "GT" "0"
    set Host "${ARGS}[1]"
    var-shift ARGS
    function PingHost call "${Host}"
endbranch

no set EXIT_MSG
exit script 0

function PingHost begin

! Ping the first host
ping ${ARGS}[1] | grep -u Success
if STATUS "NEQ" "0"
    show variable UGREP | grep 100
    if STATUS "==" "0"
     set EXIT_MSG "Ping: Failure to ping ${ARGS}[1]"
     exit script 1
    endbranch
endbranch

function PingHost end
```

Finger Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-finger
! Parameters: HostName UserName
!
! Description:
!   This script will connect to the finger server on the remote
!   host.  It will query for the UserName and receive the
!   information back.
!
! Failure Upon:
!   1. Not establishing a connection with the host.
!   2. Not being able to send/receive data to the host
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if ${ARGS}[#] "NEQ" "2"
    echo "Usage: ap-kal-finger \'Hostname UserName\'"
    exit script 1
endbranch

! Defines:
set HostName "${ARGS}[1]"
set UserName "${ARGS}[2]"

set EXIT_MSG "Connection Failed"
! Connect to the remote host (use default timeout)
socket connect host ${HostName} port 79 tcp

set EXIT_MSG "Send: Failed"
! Send the username to "finger"
socket send ${SOCKET} "${UserName}\n"
set EXIT_MSG "Waitfor: Failed"
! Wait for data for 100ms (default)
socket waitfor ${SOCKET} "${UserName}"

no set EXIT_MSG
! If the data came in, then we are good to quit
socket disconnect ${SOCKET}

no set EXIT_MSG
exit script 0
```

Time Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-time
! Parameters: HostName
!
! Description:
!   This script will connect to a remote host 'time' service on
!   port 37 and get the current time.  This script currently works
!   strictly with TCP.  [Ref. RFC-868]
!
! Failure Upon:
!   1. Not establishing a connection with the host.
!   2. Not being able to receive incoming data
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Make sure the user has a qualified number of arguments
if ${ARGS}[#] "NEQ" "1"
                echo "Usage: ap-kal-time \'Hostname\'"
                exit script 1
endbranch

! Defines:
set HostName "${ARGS}[1]"

set EXIT_MSG "Connection Failed"
! Connect to the remote host (use default timeout)
socket connect host ${HostName} port 37 tcp 2000

set EXIT_MSG "Receive: Failed"
! waitfor any data for 2000ms
socket waitfor ${SOCKET} anything 2000


! If the data came in, then we are good to quit
socket disconnect ${SOCKET}

no set EXIT_MSG

exit script 0
```

Setcookie Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-setcookie
! Parameters: HostName WebPage cookieString
!
! Description:
!   This script will keepalive a WWW server that is setting
!   cookies in the HTTP response header.  The header value
!   looks like this:
!   Set-Cookie: NAME=VALUE
!
!   The user will be responsible for sending us the name & value
!   in a string like "mycookie=myvalue" so that we can compare
!   the incoming Set-Cookie: request.
!
! Failure Upon:
!   1. Not establishing a connection with the host.
!   2. Not being able to receive the cookie
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if ${ARGS}[#] "NEQ" "3"
    echo "Usage: ap-kal-setcookie \'Hostname WebPage cookieString\'"
    echo "(Where cookieString is a name=value pair like
\'mycookie=myvalue\')"
    exit script 1
endbranch

! Defines:
set HostName "${ARGS}[1]"
set WebPage "${ARGS}[2]"
set CookieData "${ARGS}[3]"

! Connect to the remote host (use default timeout)
set EXIT_MSG "Connection Failed"
socket connect host ${HostName} port 80 tcp

! send our request to the host
set EXIT_MSG "Send: Failure"
socket send ${SOCKET} "GET ${WebPage} HTTP/1.0\n\n"

! Wait for the cookie to come in
set EXIT_MSG "Waitfor: Failure"
socket waitfor ${SOCKET} "${CookieData}"
```

```
! Done
no set EXIT_MSG
socket disconnect ${SOCKET}
exit script 0
```

HTTP Authentication Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-httpauth
! Parameters: HostName WebPage Username-Password
!
! Description:
!   This will keepalive an authentication connection by building
!   a get request with the Authentication field filled with the
!   Username-Password string formatted like so: "bob:mypassword"
!   This is critical to make the authentication base64 hash work
!   correctly.
!
! Note: This script authentication is based on HTTP AUTHENTICATION
!       RFC-2617.  Currently only supported option is "Basic"
!       authentication using base64 encoding.  "Digest" Access is
!       not supported at this time.
!
! Failure Upon:
!   1. Not establishing a connection with the host.
!   2. Not being able authenticated with the Username-Password
!      (not being given a status code of "200 OK"
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if ${ARGS}[#] "NEQ" "3"
    echo "Usage: ap-kal-httpauth \'Hostname WebPage
Username:Password\'"
    echo "(Ie. ap-kal-httpauth \'192.168.1.1 /index.html
bob:mypassword\')"
    exit script 1
endbranch

! Defines:
set HostName "${ARGS}[1]"
set WebPage "${ARGS}[2]"
set UserPass "${ARGS}[3]"

! Connect to the remote Host
set EXIT_MSG "Connection Failure"
socket connect host ${HostName} port 80 tcp
```

```
! Send the GET request for the web page, along with the authorization
! This builds a header block like so:
!
! GET /index.html HTTP/1.0\r\n
! Authorization: Basic bGFiOmxhYnRlc3Qx\r\n\r\n

set EXIT_MSG "Send: Failed"
socket send ${SOCKET} "GET ${WebPage} HTTP/1.0\n"
socket send ${SOCKET} "Authorization: Basic "
socket send ${SOCKET} "${UserPass}" base64
socket send ${SOCKET} "\n\n"

! Wait for a good status code
set EXIT_MSG "Waitfor: Failed"
socket waitfor ${SOCKET} "200 OK"

no set EXIT_MSG
socket disconnect ${SOCKET}
exit script 0
```

DNS Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-dns
! Parameters: Server DomainName
!
! Description:
!   This script will resolve a domain name from a specific DNS
!   server.  This builds a UDP packet based on RFC-1035
!
! Failure Upon:
!   1. Not resolving the hosts's IP from the domain name
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if ${ARGS}[#] "NEQ" "1"
    echo "Usage: ap-kal-finger \'Hostname\'"
    exit script 1
endbranch

set HostName "${ARGS}[1]

! Connect to the remote host
set EXIT_MSG "Connection failed"
socket connect host ${HostName} port 53 udp
```

```
! This may require a little explanation.  Since we just want to see
! if the DNS server is alive we will send a simple DNS Query.  This
! query is hard coded in hexadecimal and sent raw to the DNS server.
! The DNS request has a 12 byte header (as seen for the first 12 bytes
! of hex) and then a DNS name (ie. www.cisco.com).  Lastly it follows
! with some null termination and a few bytes representing query type.
! See RFC-1035 for more.
set EXIT_MSG "Send: failure"
socket send ${SOCKET}
"000201000001000000000000037777770563697363 6f03636f6d0000010001" raw

! Receive some unexplained response.  We don't care what it is because
! an unstable DNS server or a non-existent one would probably not send
! us any data back at all.

set EXIT_MSG "Receive: Failed to receive data"
socket receive ${SOCKET}

no set EXIT_MSG
socket disconnect ${SOCKET}
exit script 0
```

Echo Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-echo
! Parameters: HostName [ udp | tcp ]
!
! Description:
!   This script will send a TCP or UDP echo (depending on what the
!   user has passed to us) that will echo "Hello Cisco" to the
!   remote host, and expect it to come back.  The default protocol
!   is TCP.
!
! Failure Upon:
!   1. Not establishing a connection with the host (TCP Only).
!   2. Not being able to retrieve an echoed message back
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Make sure the user has a qualified number of arguments
if ${ARGS}[#] "NEQ" "2"
    if ${ARGS}[#] "NEQ" "1"
     echo "Usage: ap-kal-echo \'Hostname [ udp | tcp ]\'"
     exit script 1
    endbranch
endbranch
```

```
! Defines:
set HostName "${ARGS}[1]"
set nProtocol "tcp"

! See if the user has specified a protocol
if ${ARGS}[#] "==" "2"
    ! The user specified a protocol, so reset the value
    set nProtocol "${ARGS}[2]"
endbranch

set EXIT_MSG "Connection Failed"
! Connect to the remote host (use default timeout)
socket connect host ${HostName} port 7 ${nProtocol}

set EXIT_MSG "Send: Failed"
! Send the text to echo...
socket send ${SOCKET} "Hello Cisco!\n"

set EXIT_MSG "Waitfor: Failed"
! Wait for the reply from the echo (should be the same)
socket waitfor ${SOCKET} "Hello Cisco!" 200

! We've successfully logged in, the server is up and going.
! The job was done successfully.

socket disconnect ${SOCKET}
no set EXIT_MSG
exit script 0
```

HTTP Host Tag Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-httptag
! Parameters: HostName WebPage HostTag
!
! Description:
!   This script will connect to the remote host and do an HTTP
!   GET method upon the web page that the user has asked for.
!   This script also adds a host tag to the GET request.
!
! Failure Upon:
!   1. Not establishing a connection with the host.
!   2. Not receiving an HTTP status "200 OK"
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
if ${ARGS}[#] "NEQ" "3"
    echo "Usage: ap-kal-httptag \'Hostname WebPage HostTag\'"
    exit script 1
endbranch
! Defines:
set HostName "${ARGS}[1]"
set WebPage "${ARGS}[2]"
set HostTag "${ARGS}[3]"

! Connect to the remote Host
set EXIT_MSG "Connection Failure"
socket connect host ${HostName} port 80 tcp

! Send the GET request for the web page
set EXIT_MSG "Send: Failed"
socket send ${SOCKET} "GET ${WebPage} HTTP/1.0\nHost: ${HostTag}\n\n"

! Wait for a good status code
set EXIT_MSG "Waitfor: Failed"
socket waitfor ${SOCKET} "200 OK"

no set EXIT_MSG
socket disconnect ${SOCKET}
exit script 0
```

Mailhost Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-mailhost
! Parameters: HostName UserName Password
!
! Description:
!   This script will check the status on a mailhost.  The mailhost
!   should be running a POP3 and SMTP service.  We will attempt
!   to keepalive both services, and if one goes down we will report
!   an error.
!
! Failure Upon:
!   1. Not establishing a connection with the host running an SMTP
     service.
!   2. Not establishing a connection with the host running a POP3
     service.
!   3. Failure to get a good status code after saying 'helo' to SMTP.
!   4. Failure to login using POP3.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
if ${ARGS}[#] "NEQ" "3"
    echo "Usage: ap-kal-pop3 \'Hostname UserName Password\'"
    echo "(For checking an SMTP and POP3 service)"
    exit script 1
endbranch

! Defines:
set HostName "${ARGS}[1]"
set UserName "${ARGS}[2]"
set Password "${ARGS}[3]"

!!!!! SMTP !!!!!!

set EXIT_MSG "Connection Failed"
! Connect to the remote host (use default timeout)
socket connect host ${HostName} port 25 tcp

set EXIT_MSG "Waitfor: Failed"
! Receive the incoming status code 220 "welcome message"
socket waitfor ${SOCKET} "220" 200

set EXIT_MSG "Send: Failed"
! Send the helo to the server
socket send ${SOCKET} "helo ${HostName}\n"
set EXIT_MSG "Waitfor: Failed"
! Wait for status code "250" to be returned
socket waitfor ${SOCKET} "250" 200

! We've successfully logged in, the server is up and going.
! The job was done successfully.
socket disconnect ${SOCKET}

!!!!! POP3 !!!!!!

set EXIT_MSG "Connection Failed"
! Connect to the remote host (use default timeout)
socket connect host ${HostName} port 110 tcp

set EXIT_MSG "Waitfor: Failed"
! Wait for the OK welcome message for 200ms
socket waitfor ${SOCKET} "+OK" 200

set EXIT_MSG "Send: Failed"
! Send the username to the host
socket send ${SOCKET} "USER ${UserName}\n"
```

```
set EXIT_MSG "Waitfor: Failed"
! Wait for confirmation
socket waitfor ${SOCKET} "+OK" 200
set EXIT_MSG "Send: Failed"
! Send the password
socket send ${SOCKET} "PASS ${Password}\n"

set EXIT_MSG "Waitfor: Failed"
! Wait for confirmation
socket waitfor ${SOCKET} "+OK" 200

! We've successfully logged in, the server is up and going.
! The job was done successfully.
socket disconnect ${SOCKET}

no set EXIT_MSG
exit script 0
```

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z