CISCO SYSTEMS

# Cisco Content Services Switch Administration Guide

Software Version 8.10
November 2005

**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

Text Part Number: OL-8240-01

# CONTENTS

FIGURES

TABLES

# Preface

This guide provides instructions for the administration of the Cisco 11500 Series Content Services Switches (CSS). It describes how to perform administration tasks on the CSS, including managing the CSS software, upgrading your CSS software, and so on. Information in this guide applies to all CSS models except where noted.

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

This preface contains the following major sections:

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

## Audience

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

- Web master
- System administrator
- System operator

## How to Use This Guide

This guide is organized as follows:

| Chapter | Description |
|---|---|
| Chapter 1, Managing the CSS Software | Copy the running-configuration and startup-configuration files, specify file storage locations for a two-disk 11500 series CSS, and unpack and remove an ArrowPoint Distribution Image (ADI). This chapter also includes an overview of the CSS system software. |
| Chapter 2, Specifying the CSS Boot Configuration | Set the primary and secondary boot configuration for the CSS. |
| Chapter 3, Configuring User Profiles | Configure user profiles in the default-profile file. |
| Chapter 4, Using the CSS Logging Features | Configure logging for the CSS. This chapter also provides information displaying and interpreting log messages. |
| Chapter 5, Configuring Simple Network Management Protocol (SNMP) | Configure SNMP on the CSS. This chapter also includes a summary of all CSS Enterprise Management Information Base (MIB) objects. |
| Chapter 6, Configuring Remote Monitoring (RMON) | Configure RMON on the CSS. |

| Chapter | Description |
|---|---|
| Chapter 7, Using an XML Document to Configure the CSS | Use extended markup language (XML) to configure a CSS. |
| Chapter 8, Using the CSS Scripting Language | Use the CSS scripting language to automate configuration tasks and create script keepalives. This chapter includes example scripts. |
| Appendix A, Upgrading Your CSS Software | Upgrade your CSS software manually or use the upgrade script. |
| Appendix B, Using the Offline Diagnostic Monitor Menu | Use the Offline Diagnostic Monitor (Offline DM) menu. |

## Related Documentation

In addition to this document, the CSS documentation set includes the following:

| Document Title | Description |
|---|---|
| *Release Note for the Cisco 11500 Series Content Services Switch* | This release note provides information on operating considerations, caveats, and command-line interface (CLI) commands for the Cisco 11500 series CSS. |
| *Cisco 11500 Series Content Services Switch Hardware Installation Guide* | This guide provides information for installing, cabling, and powering the Cisco 11500 series CSS. In addition, this guide provides information about CSS specifications, cable pinouts, and hardware troubleshooting. |

| Document Title | Description |
|---|---|
| *Cisco Content Services Switch Getting Started Guide* | This guide describes how to perform initial administration and configuration tasks on the CSS, including:<br>• Booting the CSS for the first time and a routine basis, and logging in to the CSS<br>• Configuring the username and password, Ethernet management port, static IP routes, and the date and time<br>• Configuring DNS server for hostname resolution<br>• Configuring sticky cookies with a sticky overview and advanced load-balancing method using cookies<br>• Installing the CSS Cisco View Device Manager (CVDM) browser-based user interface used to configure the CSS<br>• Finding information in the CSS documentation with a task list<br>• Troubleshooting the boot process |
| *Cisco Content Services Switch Routing and Bridging Configuration Guide* | This guide describes how to perform routing and bridging configuration tasks on the CSS, including:<br>• Management ports, interfaces, and circuits<br>• Spanning-tree bridging<br>• Address Resolution Protocol (ARP)<br>• Routing Information Protocol (RIP)<br>• Internet Protocol (IP)<br>• Open Shortest Path First (OSPF) protocol<br>• Cisco Discovery Protocol (CDP)<br>• Dynamic Host Configuration Protocol (DHCP) relay agent |

| Document Title | Description |
|---|---|
| *Cisco Content Services Switch Content Load-Balancing Configuration Guide* | This guide describes how to perform CSS content load-balancing configuration tasks, including:<br>• Flow and port mapping<br>• Services<br>• Service, global, and script keepalives<br>• Source groups<br>• Loads for services<br>• Server/Application State Protocol (SASP)<br>• Dynamic Feedback Protocol (DFP)<br>• Owners<br>• Content rules<br>• Sticky parameters<br>• HTTP header load balancing<br>• Content caching<br>• Content replication |
| *Cisco Content Services Switch Global Server Load-Balancing Configuration Guide* | This guide describes how to perform CSS global load-balancing configuration tasks, including:<br>• Domain Name System (DNS)<br>• DNS Sticky<br>• Content Routing Agent<br>• Client-Side Accelerator<br>• Network proximity |
| *Cisco Content Services Switch Redundancy Configuration Guide* | This guide describes how to perform CSS redundancy configuration tasks, including:<br>• VIP and virtual interface redundancy<br>• Adaptive session redundancy<br>• Box-to-box redundancy |

| Document Title | Description |
|---|---|
| *Cisco Content Services Switch Security Configuration Guide* | This guide describes how to perform CSS security configuration tasks, including:<br>• Controlling access to the CSS<br>• Secure Shell Daemon protocol<br>• Radius<br>• TACACS+<br>• Firewall load balancing |
| *Cisco Content Services Switch SSL Configuration Guide* | This guide describes how to perform CSS SSL configuration tasks, including:<br>• SSL certificate and keys<br>• SSL termination<br>• Back-end SSL<br>• SSL initiation<br>• HTTP data compression |
| *Cisco Content Services Switch Command Reference* | This reference provides an alphabetical list of all CLI commands including syntax, options, and related commands. |

## Symbols and Conventions

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

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

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

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

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

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

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

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

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

- 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

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

### Cisco.com

You can access the most current Cisco documentation at this URL:

http://www.cisco.com/techsupport

You can access the Cisco website at this URL:

http://www.cisco.com

You can access international Cisco websites at this URL:

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

## Product Documentation DVD

Cisco documentation and additional literature are available in the Product Documentation DVD package, which may have shipped with your product. The Product Documentation DVD is updated regularly and may be more current than printed documentation.

The Product Documentation DVD is a comprehensive library of technical product documentation on portable media. The DVD enables you to access multiple versions of hardware and software installation, configuration, and command guides for Cisco products and to view technical documentation in HTML. With the DVD, you have access to the same documentation that is found on the Cisco website without being connected to the Internet. Certain products also have .pdf versions of the documentation available.

The Product Documentation DVD is available as a single unit or as a subscription. Registered Cisco.com users (Cisco direct customers) can order a Product Documentation DVD (product number DOC-DOCDVD=) from Cisco Marketplace at this URL:

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

## Ordering Documentation

Beginning June 30, 2005, registered Cisco.com users may order Cisco documentation at the Product Documentation Store in the Cisco Marketplace at this URL:

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

Nonregistered Cisco.com users can order technical documentation from 8:00 a.m. to 5:00 p.m. (0800 to 1700) PDT by calling 1 866 463-3487 in the United States and Canada, or elsewhere by calling 011 408 519-5055. You can also order documentation by e-mail at tech-doc-store-mkpl@external.cisco.com or by fax at 1 408 519-5001 in the United States and Canada, or elsewhere at 011 408 519-5001.

## Documentation Feedback

You can rate and provide feedback about Cisco technical documents by completing the online feedback form that appears with the technical documents on Cisco.com.

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

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

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

We appreciate your comments.

## Cisco Product Security Overview

Cisco provides a free online Security Vulnerability Policy portal at this URL:

http://www.cisco.com/en/US/products/products_security_vulnerability_policy.html

From this site, you can perform these tasks:

- Report security vulnerabilities in Cisco products.
- Obtain assistance with security incidents that involve Cisco products.
- Register to receive security information from Cisco.

A current list of security advisories and notices for Cisco products is available at this URL:

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

If you prefer to see advisories and notices as they are updated in real time, you can access a Product Security Incident Response Team Really Simple Syndication (PSIRT RSS) feed from this URL:

http://www.cisco.com/en/US/products/products_psirt_rss_feed.html

## Reporting Security Problems in Cisco Products

Cisco is committed to delivering secure products. We test our products internally before we release them, and we strive to correct all vulnerabilities quickly. If you think that you might have identified a vulnerability in a Cisco product, contact PSIRT:

- Emergencies—security-alert@cisco.com

  An emergency is either a condition in which a system is under active attack or a condition for which a severe and urgent security vulnerability should be reported. All other conditions are considered nonemergencies.

- Nonemergencies—psirt@cisco.com

In an emergency, you can also reach PSIRT by telephone:

- 1 877 228-7302
- 1 408 525-6532

**Tip** We encourage you to use Pretty Good Privacy (PGP) or a compatible product to encrypt any sensitive information that you send to Cisco. PSIRT can work from encrypted information that is compatible with PGP versions 2.*x* through 8.*x*.

Never use a revoked or an expired encryption key. The correct public key to use in your correspondence with PSIRT is the one linked in the Contact Summary section of the Security Vulnerability Policy page at this URL:

http://www.cisco.com/en/US/products/products_security_vulnerability_policy.html

The link on this page has the current PGP key ID in use.

## Obtaining Technical Assistance

Cisco Technical Support provides 24-hour-a-day award-winning technical assistance. The Cisco Technical Support & Documentation website on Cisco.com features extensive online support resources. In addition, if you have a valid Cisco service contract, Cisco Technical Assistance Center (TAC) engineers provide telephone support. If you do not have a valid Cisco service contract, contact your reseller.

## Cisco Technical Support & Documentation Website

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

http://www.cisco.com/techsupport

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

http://tools.cisco.com/RPF/register/register.do

**Note** Use the Cisco Product Identification (CPI) tool to locate your product serial number before submitting a web or phone request for service. You can access the CPI tool from the Cisco Technical Support & Documentation website by clicking the **Tools & Resources** link under Documentation & Tools. Choose **Cisco Product Identification Tool** from the Alphabetical Index drop-down list, or click the **Cisco Product Identification Tool** link under Alerts & RMAs. The CPI tool offers three search options: by product ID or model name; by tree view; or for certain products, by copying and pasting **show** command output. Search results show an illustration of your product with the serial number label location highlighted. Locate the serial number label on your product and record the information before placing a service call.

## Submitting a Service Request

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

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

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

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

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

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

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

## Definitions of Service Request Severity

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

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

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

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

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

## Obtaining Additional Publications and Information

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

- Cisco Marketplace provides a variety of Cisco books, reference guides, documentation, and logo merchandise. Visit Cisco Marketplace, the company store, at this URL:

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

- *Cisco Press* publishes a wide range of general networking, training and certification titles. Both new and experienced users will benefit from these publications. For current Cisco Press titles and other information, go to Cisco Press at this URL:

  http://www.ciscopress.com

- *Packet* magazine is the Cisco Systems technical user magazine for maximizing Internet and networking investments. Each quarter, Packet delivers coverage of the latest industry trends, technology breakthroughs, and Cisco products and solutions, as well as network deployment and troubleshooting tips, configuration examples, customer case studies, certification and training information, and links to scores of in-depth online resources. You can access Packet magazine at this URL:

  http://www.cisco.com/packet

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

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

  or view the digital edition at this URL:

  http://ciscoiq.texterity.com/ciscoiq/sample/

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

  http://www.cisco.com/ipj

- Networking products offered by Cisco Systems, as well as customer support services, can be obtained at this URL:

  http://www.cisco.com/en/US/products/index.html

- Networking Professionals Connection is an interactive website for networking professionals to share questions, suggestions, and information about networking products and technologies with Cisco experts and other networking professionals. Join a discussion at this URL:

  http://www.cisco.com/discuss/networking

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

  http://www.cisco.com/en/US/learning/index.html

# Managing the CSS Software

This chapter describes how to manage the software running on the CSS. Information in this chapter applies to all CSS models, except where noted.

This chapter contains the following major sections:

- CSS Software Overview
- Creating an FTP Record
- Using the Running-Config and Startup-Config Files
- Configuring Disks in a Two-Disk CSS
- Unpacking and Removing an ADI
- Archiving Files to the Archive Directory
- Restoring Files from the Archive Directory
- Enabling and Copying Core Dumps
- Showing CSS Configurations

## CSS Software Overview

The CSS software contains the files needed to run the CSS, including boot files, directories for archiving and logging files, and MIB files. This software is pre-installed on the CSS conventional hard disk or on an optional Flash disk, which is a Flash memory-based storage device. The CSS software is approximately 50 MB, and you can install a maximum of two software versions.

The CSS software image is available from the Cisco Systems website (www.cisco.com) as an ArrowPoint Distribution Image (ADI), network boot ZIP (.zip) image, or GZIP-compressed (adi-gz) image.

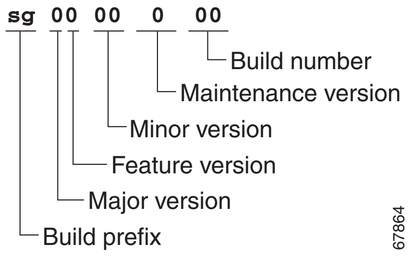
You can install the CSS software on an FTP server, which the CSS accesses through the File Transfer Protocol (FTP). The CSS accesses the ADI or GZIP file containing the CSS software from an FTP server, copies the file to the CSS disk, and unpacks it. The CSS then boots from the disk.

You can also install the CSS software on a network-mounted drive on a remote system, which the CSS accesses through FTP. Network boot uses a special ZIP version of WebNS that ends with a .zip extension. Instead of the CSS disk, the network file system contains the CSS software. This software must be copied and uncompressed on the network drive.

Refer to Chapter 2, Specifying the CSS Boot Configuration, for information on booting the CSS, including from a network boot drive.

The CSS software version format is shown in Figure 1-1.

***Figure 1-1 Software Version for the CSS***



To display the software versions installed on the CSS, use the **show version** and **show installed-software** commands, as described in the "Showing Software Information" section.

From an FTP server, you can view the following directories on the hard disk or Flash disk:

- The log directory contains the following log files:
  - **boot.log** - ASCII log of the boot process
  - **boot.bak** - Backup of the previous boot log
  - **sys.log** - ASCII log of system events (logging to disk is enabled by default to **subsystem all** and **level info**)
  - **sys.log.prev** - Backup of the previous system log file (if any)
- The scripts directory contains default, profile, and sample scripts.
- The core directory contains any core dumps created by the CSS. For information on copying core dumps to an FTP or TFTP server, see the "Enabling and Copying Core Dumps" section.
- The MIB directory contains MIB files that you can load in to SNTP-compliant network management software applications.

**Caution** When you view the CSS software directories installed on a network drive, more directories are listed than those you can view on the hard disk or Flash disk. The additional directories are reserved for internal use. Do not manipulate the files in these directories.

The software directory also contains the startup-config file. The startup-config is an ASCII file containing commands that the CSS executes at startup. This file is created when you:

- Finish using the Configuration Script (refer to the *Cisco Content Services Switch Getting Started Guide*).
- Use the **copy running-config startup-config** or **write memory** command (see the "Saving the Running-Config to the Startup-Config File" section). Both commands save configuration changes to the startup-config file during a CSS session. The **write memory** command also archives the startup configuration file to the archive directory on the CSS (similar to the **archive startup-config** command, see the "Archiving Files to the Archive Directory" section).
- Use FTP to copy a startup-config file to the CSS.

The archive directory contains the files that you archive from the current software by using the **archive** command. These files include the running-config file, startup-config file, log files, profile scripts, and scripts you create. You can view a list of archived files by using the **show archive ?** command.

To restore any archived files to the CSS, use the **restore** command. For more information on the **archive** and **restore** commands, see the "Archiving Files to the Archive Directory" and "Restoring Files from the Archive Directory" sections.

## Creating an FTP Record

The CSS requires a File Transfer Protocol (FTP) record file to access an FTP server from the CSS. A few examples of uses for an FTP record with the CSS include:

- Copy an ADI, script file, or startup configuration file from an FTP server to the CSS
- Copy a running-configuration file, startup-configuration file, log file, script, or a core dump file from the CSS to an FTP or TFTP server
- Define a keepalive method in which the CSS logs in to an FTP server
- Import or export certificates and private keys from or to a Cisco 11500 series CSS
- Associate an FTP access mechanism with a service for demand-based replication activities
- Write a portion or all of the Proximity Database to a file in the log directory on the CSS disk or a file on an FTP server

Use the **ftp-record** command to create the FTP record file. The syntax for this global configuration mode command is:

**ftp-record** *ftp_record ip_address_or_hostname username*
["*password*" | **des-password** *des_password*] {*base_directory*}

**Note** The CSS FTP server supports only the active (normal) FTP mode of operation. It does not support the passive FTP mode of operation.

The variables for this command are as follows:

- *ftp_record* - The name for the FTP record file. Enter an unquoted text string with no spaces and a maximum of 16 characters.
- *ip_address_or_hostname* - The IP address or host name of the FTP server you want to access. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1) or a mnemonic host name (for example, myhost.mydomain.com).
- *username* - A valid login username on the FTP server. Enter a case-sensitive unquoted text string with no spaces and a maximum of 16 characters.
- *password* - The password for the valid login username on the FTP server. Enter a case-sensitive quoted text string with no spaces and a maximum of 16 characters.
- *des_password* - The Data Encryption Standard (DES) encrypted password for the valid login username on the FTP server. Enter a case-sensitive unquoted text string with no spaces and a maximum of 64 characters.
- *base_directory* - An optional base directory for this record. Enter the base directory name as a case-sensitive unquoted text string with no spaces and a maximum of 64 characters.

  The config-path and base directory path in the FTP record associated with a network boot must not contain a pathname that conflicts with a non-network drive name (for example, c: or host:).

For example (using an encrypted password), to create an FTP record called *arrowrecord*, enter:

```
# ftp-record arrowrecord 192.168.19.21 bobo password “secret”
/outgoing
```

To delete the FTP record *arrowrecord* from the CSS, enter:

```
# no ftp-record arrowrecord
```

## Copying Files from an FTP Server

Use the **copy ftp** command to copy files from an FTP server to the CSS. This command is available in SuperUser mode. Before using this command, you must use the **ftp-record** global configuration mode command to create an FTP record file containing the FTP server IP address, username, and password. The syntax for this command is:

**copy ftp** *ftp_record filename* [**boot-image**|**gui-image**|**script***script_filename*|
**startup-config**]

The options and variables for this command are as follows:

- *ftp_record* - Name of the FTP record file that contains the FTP server IP address, username, and password. Enter an unquoted text string with no spaces. To create an FTP record, use the **ftp-record** global configuration mode command.
- *filename* - Name of the file on the FTP server that you want to copy to the CSS. Include the full path to the file. Enter an unquoted text string with no spaces and a maximum length of 32 characters.

  If you are using the **boot-image** keyword to copy an ADI file from an FTP server to the CSS, include the full path to the file including the file extension. Enter an unquoted text string with no spaces and a maximum length of 32 characters. You can also copy a GZIP-compressed version of the ADI file. The CSS uncompresses the file. If there is not enough disk space available, the CSS provides a message.
- **boot-image** - Copies an ADI file from an FTP server. The ADI file contains the CSS software including boot files and logging and archiving directories. To unpack the CSS software in the ADI file, use the **unpack** boot mode command. When you use the **boot-image** keyword, the file you copy to the CSS must be an ADI file. Otherwise, the CSS rejects it.
- **gui-image** - Copies the CiscoView Device Manager (CVDM) zip file onto the CSS hard drive. For more information, see the *Cisco Content Services Switch Getting Started Guide*.
- **script** *script_file* - Copies an FTP file to the script directory. To assign a name to the script file on the CSS, enter an unquoted text string with no spaces and a maximum length of 32 characters.
- **startup-config** - Copies the startup-config file and overwrites the existing configuration file.

## Using the Running-Config and Startup-Config Files

When you make configuration changes, the CSS places those changes in a virtual running configuration file (running-config). Before you log out or reboot the CSS, you must copy the contents of the running-config file to the startup-config file (startup-config) to save configuration changes. The CSS uses the startup configuration file on subsequent reboots.

This section includes the following topics:

- Saving the Running-Config to the Startup-Config File
- Copying the Running- and Startup-Config Files
- Clearing the Running-Config and Startup-Config Files
- Showing the Running Configuration
- Showing the Startup Configuration
- Creating a Running-Config or Startup-Config File Using a Text Editor
- Finding an IP Address in the Running-Config File

### Saving the Running-Config to the Startup-Config File

To save the running-config file to the startup-config file on the CSS disk, use one of the following commands:

- **copy running-config startup-config** - Copies the contents of the running-config file to the startup-config file. The CSS uses the startup configuration upon reboot. If you do not copy the contents of the running-config file to the startup-config file before you reboot, changes to the running configuration are lost. This command is available in SuperUser mode.
- **write memory** - Copies the contents of the running-config file to the startup-config file (similar to the **copy running-config startup-config** command). In addition, the **write memory** command also archives the startup configuration file to the archive directory on the CSS (similar to the **archive startup-config** command, see the “Archiving Files to the Archive Directory” section).

- **copy startup-config running-config** - Copies the contents of the startup-config file to the running-config file and merges the contents with the running-config file. This command is available in SuperUser mode.

## Copying the Running- and Startup-Config Files

The **copy running-config** command can also copy the running configuration to an FTP or TFTP server. This command is available in SuperUser mode.

**Note** If desired, use the **save_config** alias command to automatically copy the contents of the running-config file to the startup-config file, and then archive the startup-config file to the CSS disk.

The syntax for this command is:

**copy running-config** [[**ftp** *ftp_record*|**tftp** *ip_or_host*]*filename*|
**startup-config**]

The options and variables for this command are as follows:

- **ftp** *ftp_record filename* - Copies the running-config file to an FTP server. The name of the FTP record file contains the FTP server IP address, username, and password. Enter an unquoted text string with no spaces. To create an FTP record, use the **ftp-record** global configuration mode command.
- **tftp** *ip_or_host* - Copies the running-config file to a TFTP server. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1) or in mnemonic host-name format (for example, myhost.mydomain.com).
- *filename* - 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.
- **startup-config** - Copies the running-config file to the startup-config file on the CSS disk. In the event of the CSS rebooting, if you do not save changes in the running-config file to the startup-config file, these changes are lost.

The **copy startup-config** command can copy the startup configuration to an FTP or TFTP server. This command is available in SuperUser mode.

The syntax for this command is:

> **copy startup-config** [[**ftp** *ftp_record*|**tftp** *ip_or_host*]*filename*|
> **running-config**]

The options and variable for this command are as follows:

- **ftp** *ftp_record* - Copies the startup-configuration file to an FTP server. The name of the FTP record file contains the FTP server IP address, username, and password. Enter an unquoted text string with no spaces. To create an FTP record, use the **ftp-record** global configuration mode command.
- **tftp** *ip_or_host* - Copies the startup-config file to a TFTP server. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1) or in mnemonic host-name format (for example, myhost.mydomain.com).
- *filename* - 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.
- **running-config** - Copies the startup configuration and merges with the running configuration file on the CSS disk.

## Clearing the Running-Config and Startup-Config Files

To reset the CSS running configuration to the default configuration, use the **clear running-config** command in SuperUser mode. This command takes effect immediately. The **clear running-config** command resets all configurations to their defaults.

Use of the **clear running-config** command is restricted to CSS users who are identified as either administrators or technicians.

For example:

```
# clear running-config
```

To reset the startup configuration to the default configuration, use the **clear startup-config** command in SuperUser mode. This command takes effect upon the next reboot. For example:

```
# clear startup-config
```

## Showing the Running Configuration

To display the CSS running configuration, use the **show running-config** command. Configuration entries within each mode in the running-config file (such as Global, Interface, Circuit, and Service) appear in chronological order, based on the order in which you configure the CSS. The CSS does not display default configurations in the CSS running configuration.

The syntax and options for the **show running-config** command are as follows:

- **show running-config** - Displays all components of the running-config file.
- **show running-config acl** {*index number*} - Displays access control list (ACL) information in the running-config file. For information about a specific ACL, include its index number.
- **show running-config circuit** {*circuit name*} - Displays the circuit components of one or all circuits in the running-config file.
- **show running-config dql** {*dql name*} - Displays domain qualifier list (DQL) information of the running-config file. For information about a specific DQL, enter the DQL name as a case-sensitive unquoted text string.
- **show running-config eql** {*eql name*} - Displays extension qualifier list (EQL) information of the running-config file. For information about a specific EQL, enter the EQL name as a case-sensitive unquoted text string.
- **show running-config global** - Displays the global components of the running-config file.
- **show running-config group** {*group name*} - Displays the valid existing group components of the running-config file. For information about a specific group, enter the group name as a case-sensitive unquoted text string.
- **show running-config header-field-group** {*name*} - Displays the valid existing header-field group components of the running-config file. For information about a specific group, enter *name* as a case-sensitive unquoted text string with a maximum of 16 characters. To see a list of header-field groups, enter **show running-config header-field-group ?**.

- **show running-config interface** *interface name* - Displays a specific interface component of the running-config file.
  - For a CSS 11501, enter the interface name in interface-port format (for example, e2)
  - For a CSS 11503 or CSS 11506, enter the interface name in slot/port format (for example, 3/1)
- **show running-config interfaces** - Displays all the interface components of the running-config file.
- **show running-config keepalive** {*keepalive name*} - Displays the existing keepalive components of the running-config file. For information about a specific keepalive, enter *keepalive_name* as a case-sensitive unquoted text string and a maximum of 32 characters. To see a list of keepalives, enter **show keepalive-summary**.
- **show running-config nql** {*name*} - Displays network qualifier list (NQL) information of the running-config file. For information about a specific NQL, enter the NQL name as a case-sensitive unquoted text string.
- **show running-config owner** {*owner name*} - Displays the valid existing owner components of the running-config file. For information about a specific owner, enter the owner name as a case-sensitive unquoted text string.
- **show running-config rmon-alarm** - Displays RMON alarm information of the running-config file.
- **show running-config rmon-event** - Displays RMON event information of the running-config file.
- **show running-config rmon-history** - Displays RMON history information of the running-config file.
- **show running-config service** {*service name*} - Displays the components of the running-config file for a valid existing service. For information about a specific service, enter the service name as a case-sensitive unquoted text string.
- **show running-config ssl-proxy-list** {*list_name*} - Displays RMON history information of the running-config file. Displays the components of the running configuration for a valid existing SSL-proxy list. For information about a specific list, enter *list_name* as a case-sensitive unquoted text string.

- **show running-config urql** {*urql name*} - Displays the components of the running-config file for existing uniform resource locator qualifier lists (URQL). For information about a specific URQL, enter the URQL name as a case-sensitive unquoted text string.

The following example shows a running-config file. Comments are preceded by an exclamation point (!). Note that the CSS does not display default values in the CSS running configuration or startup configuration even if you manually enter the values.

```
# show running-config
!************************ GLOBAL *********************
ip route 0.0.0.0/0 158.3.7.2
!********************** INTERFACE ********************
interface e1
    bridge vlan 2
interface e2
    bridge vlan 2
!*********************** CIRCUIT *********************
circuit VLAN1
    ip address 10.3.6.58 255.255.255.0
circuit VLAN2
    ip address 158.3.7.58 255.255.255.0
!*********************** SERVICE *********************
service serv1
    ip address 10.3.6.1
    active
service serv2
    ip address 10.3.6.2
    active
!************************ OWNER **********************
owner arrowpoint.com
    content rule1
        ip address 158.3.7.43
        protocol tcp
        port 80
        add service Serv1
        add service Serv2
        active
```

## Showing the Startup Configuration

Once you copy the contents of the running-config file to the startup-config file, use the **show startup-config** command to display the CSS startup configuration. The CSS does not display default configurations in the startup-config file.

Use the **show startup-config line-numbers** command to display the startup-config file with line numbers

The following example shows a CSS startup configuration with line numbers. Comments are preceded by an exclamation point (!).

```
# show startup-config line-numbers

1. !Generated MAR 6 18:56:11
2. configure
3. !********************** CIRCUIT **********************
4. circuit VLAN1
5.  ip address 192.168.2.170 255.255.255.0
6.  ip address 192.168.1.108 255.255.255.0
7. !********************** SERVICE **********************
8. service s1
9.  ip address 192.168.2.4
10. keepalive type none
11. active
12. !*********************** OWNER ***********************
13. owner rose
14. content rule-L3
15.     vip address 192.168.128.108
16.     add service s1
17.     active
18.     content rule-L5
19.     add service s1
20.     vip address 192.168.128.108
21.     url “/*”
22.     active
```

## Creating a Running-Config or Startup-Config File Using a Text Editor

If you create a running- or startup-config file using a text editor, you must arrange the configuration information in the same order as occurs in an automatically created running- or startup-config file. The CSS arranges configuration information in the following categories within the running-config file and the startup-config file:

- Global - Configuration information relating to the CSS (for example, default route IP address)
- Interface - Physical port and VLAN associations
- Circuit - Circuit VLAN IP addresses and subnet masks
- SSL Proxy List - The ssl-proxy-list configuration
- Keepalive - The global keepalive configuration
- Service - Service names, IP addresses, and all service configuration information
- EQL - Extension Qualifier List (EQL) configuration
- Owner - Owner name, content rule name, and content rules
- Group - Source group configurations
- RMON Event - RMON event configurations
- RMON Alarm - RMON alarm configurations
- RMON History - RMON history configurations
- ACL - Access Control List (ACL) configurations
- URQL - Uniform Resource Locator Qualifier List (URQL) configurations

Though the CSS automatically organizes configuration information, the order in which you configure the CSS is important because of interdependencies within CSS functionality. Enter configuration commands for features in the same sequence as they appear in the startup-config file.

## Finding an IP Address in the Running-Config File

To avoid IP address conflicts when you configure the CSS, you can search the CSS running-config file for use of a specific IP address. You can include a netmask for subnet (wildcard) searches. Use the **find ip address** command to search the CSS running-config file for the IP address.

When you use this command, the CSS checks all services, source groups, content rules, ACLs, the management port, the syslog, Application Peering Protocol (APP) sessions, and local interfaces in the running-config file for the specified IP address. If the address is found, the CSS displays the locations of its use. If no addresses are found, the CSS returns you to the command prompt.

This command is available in all modes. The syntax is:

**find ip address** *ip_or_host* {*subnet_mask*|**range** *number*}

The options and variables for this command are as follows:

- *ip_or_host* - IP address in dotted-decimal notation (for example, 192.168.11.1) or enter the host name in mnemonic host-name format (for example, host.domain.com).
- *subnet mask* - The IP subnet mask. Enter the subnet mask as either:
  - A prefix length in CIDR bit-count notation (for example, /24). Enter a prefix length of /16 or greater. Do not include a space to separate the IP address from the prefix length.
  - An IP address in dotted-decimal notation (for example, 255.255.255.0).
- **range** *number* - Defines how many IP addresses you want to find, starting with the *ip_or_host* address. Enter a number from 1 to 65535. The default range is 1.

  For example, if you enter an IP address of 192.168.1.1 with a range of 10, the CSS tries to find the addresses from 192.168.1.1 through 192.168.1.10.

For example:

```
(config)# find ip address 192.168.0.0

Users of IP address 192.168.0.0
Content Rule - 192.168.12.1, layer 3, owner: lml, state:Active
Content Rule - 192.168.12.1, layer 5, owner: lml, state:Active
Service - 192.168.3.6, serv1, state:Active
Service - 192.168.3.7, serv3, state:Active
Interface - 192.168.1.117. VLAN1
Interface - 192.168.2.117. VLAN1
```

## Configuring Disks in a Two-Disk CSS

The CSS 11501 and the Switch Control Module (SCM) in the CSS 11503 and CSS 11506 contain two PCMCIA slots for a hard disk or Flash disk. These disks contain the CSS system software and are used for logging and storing offline system files. The two disks are identified by the PCMCIA slots (slot 0 and slot 1) in which they are installed. Disk 0 is the default storage location for the primary and secondary boot records in the CSS. The default storage location for log files and core dumps in the CSS is the specified disk from which the CSS boots (disk 0 or disk 1).

In addition to specifying the file storage locations, you can also:

- Format the disks
- Copy information such as the scripts, archives, or startup configuration from one disk to the other disk
- Display the mapping configuration of the two disks in slot 0 and slot 1
- Display the specified archive, log, script, or startup configuration file stored on a specific disk
- Delete a specific file (startup configuration, logs, scripts, or archive file) stored on a specific disk

This section includes the following topics:

- Formatting a Disk
- Specifying a Disk for Booting, Logging, and Core Dumps
- Copying Files Between Disks
- Showing the Disk Mapping Configurations
- Showing Files from a Disk
- Clearing Files from a Disk

As an alternate procedure for configuring disks from the CLI, you can use the Advanced Options menu of the Offline DM menu to reformat or set the disk mapping for the disks in slots 0 and 1. Refer to Appendix B, Using the Offline Diagnostic Monitor Menu, for details.

## Formatting a Disk

To format and create the Core and Archive directories on a specified disk, use the **format** command. The **format** command permanently erases all data on the disk. This command is available only in SuperUser mode.

If you wish to retain the startup-config file, ensure you move the file off the CSS before reformatting the disk. Also make sure you have a copy of the CSS software ADI file to reinstall on the CSS.

To format a disk, use the following commands:

- **format** *disk_slot* - Formats the specified disk. The slot number designates which disk you want to format. Valid *disk_slot* selections are 0 (for the disk in slot 0) or 1 (for the disk in slot 1).
- **format** *disk_slot* {**quick**} - Formats the specified disk (0 or 1). The quick option reformats the disk without performing cluster verification.

**Note** Use the quick disk format only when you are certain of the disk integrity.

For example, to format the disk in slot 1, enter:

```
# format 1
```

The CSS queries you about formatting the disk.

```
Formatting the disk results in all disk data being
permanently erased.
Are you sure you want to continue? (yes,no):
```

Enter one of the following:

- **yes** to reformat the disk.
- **no** to end the reformat function. If the disk has unrecoverable errors and you do not reformat it, be aware that the file system may be corrupt and functionality is compromised.

## Specifying a Disk for Booting, Logging, and Core Dumps

By default, disk 0 is the default storage location for the primary and secondary boot records in the CSS. The default storage location for log files and core dumps is the specified disk from which the CSS boots (disk 0 or disk 1). Use the **map** commands to specify the disk (slot 0 or slot 1) that the CSS uses to store the primary boot record, the secondary boot record, the logging output file, and core dumps.

You can mix and match the storage location of these files between the two disks. For example, you can store the primary boot record on disk 0 and the secondary boot record on disk 1, and redirect the storage of output logs and core dumps to disk 1.

The syntax for this global configuration mode command is:

**map** [**core**|**log**|**primary-boot**|**secondary-boot**] *disk_slot*

The options for the **map** command are as follows:

- **core** - Specifies the disk that contains the core dumps
- **log** - Specifies the disk that contains the logging output
- **primary-boot** - Specifies the disk that contains the primary boot record
- **secondary-boot** - Specifies the disk that contains the secondary boot record

Use the **no** form of each command to remove mapping to the specified disk and return the setting to the default disk.

### Selecting a Disk for the Primary Boot Record

Disk 0 is the default storage location for the primary boot record 0. Valid selections are 0 (for the disk in slot 0) and 1 (for the disk in slot 1). Use the **map primary-boot** command to select the disk that contains the primary boot record of the CSS. This command is available only in SuperUser mode.

For example, to select the disk in slot 1 as the storage location for the primary boot record, enter:

```
# map primary-boot 1
```

To return the storage location of the primary boot record back to the disk in slot 0, enter:

```
# no map primary-boot
```

or

```
# map primary-boot 0
```

## Selecting a Disk for the Secondary Boot Record

Disk 0 is the default storage location for the secondary boot record. Valid selections are 0 (for the disk in slot 0) and 1 (for the disk in slot 1). Use the **map secondary-boot** command to select the disk that contains the secondary boot record of the CSS. This command is available only in SuperUser mode.

For example, to select the disk in slot 1 as the storage location for the secondary boot record, enter:

```
# map secondary-boot 1
```

To return the storage location of the secondary boot record back to the disk in slot 0, enter:

```
# no map secondary-boot
```

or

```
# map secondary-boot 0
```

## Selecting a Disk for Core Dumps

The default storage location for core dump files is the disk from which the CSS boots (disk 0 or disk 1). For example, if the CSS boots from disk 1, then disk 1 becomes the default storage location for core dump files. Use the **map core** command to select the disk that stores core dump files when the CSS experiences a fatal error.

Valid selections are 0 (disk in slot 0) and 1 (disk in slot 1). This command is available only in SuperUser mode.

**Note** Core dump information is intended for Customer Support use only.

For example, to select the disk in slot 1 as the storage location for core dumps, enter:

```
# map core 1
```

To return the storage location for core dumps back to boot disk, enter:

```
# no map core
```

## Selecting a Disk for Logging

The default storage location for log files is the disk from which the CSS boots (disk 0 or disk 1). For example, if the CSS boots from disk 0, then disk 0 becomes the default storage location for log files. Use the **map log** command to select the disk on which you want to store log files.

Valid selections are 0 (disk in slot 0) and 1 (disk in slot 1). This command is available only in SuperUser mode.

**Note** Logging to a CSS disk can cause the performance of the CSS to degrade. If logging requires frequent writes to disk (that is, several hundred log messages per day), we recommend that you log to a hard disk and store all other system files on a Flash disk. Although Flash disks generally provide the most reliable way to store information over time, hard disks endure frequent writes to disk better than the Flash disks currently available.

For example, to select the disk in slot 1 as the storage location for log files, enter:

```
# map log 1
```

To return the storage location of log files back to the boot disk, enter:

```
# no map log
```

## Copying Files Between Disks

Use the **copy** command to copy the startup configuration, logs, scripts, archive, and boot image files from one disk (source) to the second disk (destination) in a CSS. The CSS software automatically creates the software directory and hierarchy on the destination disk. This command is available only in SuperUser mode.

The syntax is:

> **copy** *source_disk_slot* {**log** *filename* {*destination filename*}|**logs**|**script** *filename* {*destination filename*}|**scripts**|**archive** *filename* {*destination filename*}|**archives**|**boot-image** *filename*|**startup-config**}

The options and variables for the **copy** command are as follows:

- *source_disk_slot* - Specifies the disk location containing the files you want to copy. Valid entries are 0 (disk in slot 0) and 1 (disk in slot 1). If you want to perform a complete copy of all contents from the source disk to the second disk, enter only the *disk_slot* value. Do not enter values for the additional **copy** command variables.
- **log** *filename* - Copies the specified log file from the source disk to the second disk.
- **log** *filename* {*destination filename*} - Copies the specified log file from the source disk to the second disk using a different destination filename.
- **logs** - Copies all log files from the source disk to the second disk.
- **script** *filename* - Copies the specified script from the source disk to the second disk.
- **script** *filename* {*destination filename*} - Copies the specified script from the source disk to the second disk using a different destination filename.
- **scripts** - Copies all scripts from the source disk to the second disk.
- **archive** *filename* - Copies the specified archive file from the source disk to the second disk.
- **archive** *filename* {*destination filename*} - Copies the specified archive file from the source disk to the second disk using a different destination filename.
- **archives** - Copies all archive files from the source disk to the second disk.

- **boot-image** *filename* - Copies the specified boot image ADI from the source disk to the second disk. If necessary, use the **show installed-software** command to view the names of the boot-images (see the "Showing Software Information" section for details on using the **show installed-software** command).
- **startup-config** - Copies the startup-config file from the source disk to the second disk.

Note the following restrictions for the **copy** command when copying information between two disks in the CSS:

- The source file must exist.
- An equivalent release of CSS software must be present on the destination disk before you copy information to the disk (such as a startup-config file, a log file, or a script). If necessary, copy the boot image to the second disk before copying a startup-config file, log file, or script.

## Showing the Disk Mapping Configurations

Use the **show map** command to display the mapping configuration of the two disks in slot 0 and slot 1 in a CSS. This command displays the disk assignment of the primary-boot record, the secondary-boot record, core dump files, and logging output. This command is available in all modes.

For example:

```
(config)# show map

MSD Mapping:
Primary-Boot:   0
Secondary-Boot: 0
Core:           1
Log:            1
```

## Showing Files from a Disk

Use the **show** command to display the specified archive, log, script, or startup configuration file stored on a specific disk in the CSS. The syntax is:

> **show** *disk_slot* {**log** *filename*|**script** *filename*|**archive** *filename*|
> **startup-config**}

The options and variables for the **show** command are as follows:

- *disk_slot* - Specifies the disk location containing the file to display. The valid entries are 0 (disk in slot 0) and 1 (disk in slot 1).
- **log** *filename* - Displays the contents of a log (or trap log file) from the specified disk.
- **script** *filename* - Displays the contents of the script from the specified disk.
- **archive** *filename* - Displays the contents of the archive filename from the specified disk.
- **startup-config** - Displays the contents of the CSS startup configuration file from the specified disk.

## Clearing Files from a Disk

Use the **clear** command to delete the specified file (startup configuration, logs, scripts, archive file) stored on a specific disk in the CSS. This command is available only in SuperUser mode. The syntax is:

> **clear** *disk_slot* {**log** *filename*|**script** *filename*|**archive** *filename*|
> **startup-config**}

The options and variable for the **clear** command are as follows:

- *disk_slot* - Specifies the disk location containing the file to delete. Valid entries are 0 (disk in slot 0) and 1 (disk in slot 1).
- **log** *filename* - Deletes the specified log (or trap log file) from the disk.
- **script** *filename* - Deletes the specified script from the disk.
- **archive** *filename* - Deletes the specified archive filename from the disk.
- **startup-config** - Deletes the CSS startup configuration file from the disk.

## Unpacking and Removing an ADI

Before unpacking the ADI, you must first copy the ADI to the CSS disk. Use the **copy ftp ftp_record** *filename* **boot-image** command to copy the ADI to the CSS disk. Refer to Chapter 2, Specifying the CSS Boot Configuration, for details.

Use the **unpack** command to unpack the ArrowPoint Distribution Image (ADI) on the CSS disk. Enter the ADI filename as an unquoted text string with a maximum of 32 characters. For example:

```
(config-boot)# unpack ap0720002.adi
```

Use the **remove** command to remove an ArrowPoint Distribution Image (ADI) that is not currently running on the CSS. For a dual-disk CSS, you need to identify the specified disk.

**Warning** **Ensure you do not delete the software version that you are currently running in the CSS.**

To remove a software version installed on the CSS, use the following commands:

- **remove** *software version* **-** Enter the ADI filename as an unquoted text string with a maximum of 32 characters.
- **remove** *disk_slot software version* - Enter the slot location of the disk (0 or 1) in a dual disk CSS, followed by the ADI filename as an unquoted text string with a maximum of 32 characters.

To display a list of ADIs installed on your CSS, enter **remove ?**. To display the ADI you are currently running, use the **version** command.

To remove an ADI, enter:

```
(config-boot)# remove ap0720001
```

To remove an ADI from a disk in slot 1 of a dual-disk CSS, enter:

```
(config-boot)# remove ap0720001 1
```

## Archiving Files to the Archive Directory

Archiving is useful when you update software and want to save a startup-config file, running-config file, log, or script from a previous release of software. The archive directory on the CSS disk stores the archive files. Use the **archive** command and options to archive the specific files residing on the CSS.

The syntax for this command is:

> **archive** [[**startup-config**|**log** *log_filename*|**script** *script_filename*]
> *archive_filename*}|**running-config** *archive_filename*]

The options for this command are as follows:

- **archive startup-config** - Archives the startup-config file
- **archive running-config** - Archives the running-config file
- **archive log** - Archives a log file
- **archive script** - Archives a script or user-profile file

To display the contents of the archive directory, enter **show archive ?**. Archive files include running-config and startup-config files, scripts, and user profiles.

You must archive your startup-config file, custom scripts, and user-profile files before you upgrade the CSS software or these files will be overwritten during the upgrade. Once the CSS completes the upgrade and reboots, use the **restore** command to restore these files from the archive directory.

This section includes the following topics:

- Archiving the Startup-Config File
- Clearing the Archive Directory
- Archiving the Running-Config File
- Archiving Scripts

**Note** If you booted your CSS from a network-mounted system and your hard drive does not work, the CSS suspends all archive-related functions.

## Archiving the Startup-Config File

Use the **archive startup-config** command to archive the startup-config file. Enter the archive filename as an optional name you want to assign to the archive file. Enter an unquoted text string with a maximum of 32 characters. The syntax for this command is:

**archive startup-config** {*archive_filename*}

## Archiving the Running-Config File

Use the **archive running-config** command to archive the running-config file. Enter the archive filename as the name you want to assign to the archive file. The archive filename is an unquoted text string with a maximum of 32 characters. The syntax for this command is:

**archive running-config** *archive_filename*

**Note** You can also use the **save_config** alias command to automatically copy the running-config to the startup-config, and then archive the startup-config.

## Archiving a Log File

Use the **archive log** command to archive a log file. The syntax for this command is:

**archive log** *log_filename* **{***archive_filename***}**

The variables are as follows:

- *log_filename* - The filename of the log to archive. To see a list of log files, enter **archive log ?**.
- *archive_filename* - (Optional) The name you want to assign to the archive file. Enter an unquoted text string with a maximum of 32 characters.

## Archiving Scripts

Use the **archive script** command to archive a script file or user-profile file. The syntax for this command is:

**archive script** *script_filename* {*archive_filename*}

The variables are as follows:

- *script_filename* - The filename of the script to archive. To see a list of scripts, enter **archive script ?**.
- *archive_filename* - (Optional) The name you want to assign to the archive file. Enter an unquoted text string with a maximum of 32 characters.

## Clearing the Archive Directory

Use the **clear archive** command to clear a file in the archive directory. Enter the archive filename as the name of the archive file to clear. To list the archive files, enter **clear archive ?**. The syntax for this command is:

**clear archive** *archive_filename*

# Restoring Files from the Archive Directory

The archive directory resides on the CSS disk (hard or Flash disk) to store logs, scripts, and startup-config files. Use the **restore** command to restore files previously archived in the CSS archive directory.

The syntax for this command is:

**restore** *archive_filename* [**log** {*log_filename*} |**script** {*script_filename*}|**startup-config**]

The options for this command are as follows:

- **restore** *archive_filename* **log** - Restores an archived log file to the log subdirectory.
- **restore** *archive_filename* **script** - Restores an archived script file or user-profile file to the script subdirectory.
- **restore** *archive_filename* **startup-config** - Restores an archived startup-config file to the startup configuration.

This section includes the following topics:

- Restoring an Archived Log File
- Restoring an Archived Script File
- Restoring an Archived Startup-Config File

**Note** If you booted your CSS from a network-mounted system and your hard drive does not work, the CSS suspends all restore-related functions.

## Restoring an Archived Log File

Use the **restore log** command to restore an archived log file to the log subdirectory. The syntax for this command is:

**restore** *archive_filename* **log** {*log_filename*}

The variables are as follows:

- *archive_filename* - The name of the archived log file. Enter an unquoted text string. To see a list of archived files, enter **restore ?**.
- *log_filename* - (Optional) The name you want to assign to the restored log file. Enter an unquoted text string with a maximum of 32 characters.

For example, to restore the log file *arrowlog* to the log subdirectory and rename the log file to *arrowpointlog*, enter:

```
# restore arrowlog log arrowpointlog
```

## Restoring an Archived Script File

Use the **restore** *archive_filename* **script** command to restore an archived script file or user-profile file to the script subdirectory. The syntax for this command is:

**restore** *archive_filename* **script** {*script_filename*}

The variables are as follows:

- *archive_filename* - The name of the archived file. Enter an unquoted text string. To see a list of archived files, enter **restore ?**.
- *script_filename* - (Optional) The name you want to assign to the file. Enter an unquoted text string with a maximum of 32 characters.

For example, to restore the script *arrowscript* to the script subdirectory, enter.

```
# restore arrowscript script
```

## Restoring an Archived Startup-Config File

Use the **restore** *archive_filename* **startup-config** command to restore an archived file to the startup configuration.

**Caution** The restored file overwrites the startup configuration.

The syntax for this command is:

**restore** *archive_filename* **startup-config**

Enter the archived startup-config filename as an unquoted text string. To see a list of archived files, enter **restore ?**.

For example, to restore the archived startup-config file *arrowstart* as the current startup-config file, enter:

```
# restore arrowstart startup-config
```

## Enabling and Copying Core Dumps

A core dump occurs when the CSS experiences a fatal error. The CSS allows you to enable or disable core dumps. Core dumps are enabled by default.

When the CSS experiences a fatal error and core dumps are enabled, the CSS:

- Writes information about the fatal error to the Core directory of the volume root (for example, c:\core) on either the hard or Flash disk. The CSS stores one dump file per slot for each card type until the disk (Flash or hard disk) is full. Files can be 10 to 20 MB in size.
- Reboots automatically.

**Note** Core dump information is for Cisco Technical Assistance Center (TAC) use only.

When the CSS experiences a fatal error and core dumps are disabled, the CSS reboots automatically. The CSS does not write information to the hard disk or the Flash disk.

For a Flash disk-based system, if the core dump file is older than 15 minutes, the file may be overwritten. If you want to save the core dump file for later examination, archive the file to another directory or disk before it is overwritten. For details on using the **archive log** command, see the "Archiving the Startup-Config File" section.

This section includes the following topics:

- Enabling and Disabling Core Dumps
- Showing Core Dumps
- Copying Core Dumps to an FTP or TFTP Server

## Enabling and Disabling Core Dumps

To disable core dumps, enter:

```
(config)# dump disable
```

To reenable core dumps (the default setting), enter:

```
(config)# dump enable
```

## Showing Core Dumps

Use the **show core** command to display the core dump files stored in the Core directory of the volume root (for example, c:\core) on the hard disk or Flash disk. This command is available in all modes except User mode.

Use the **show core** *disk_slot* command to display the core dump files stored in the Core directory of the volume root of a specific disk in the CSS 11501, CSS 11503, or CSS 11506. Valid selections are 0 (for the disk in slot 0) or 1 (for the disk in slot 1).

For example:

```
# show core

SCP0101_4.80_115... OCT 31 15:06:26       16708412
SCP0101_4.80_109... OCT 29 16:56:16       37806459
SCP0101_4.80_116... NOV  1 15:54:28       38403870
```

## Copying Core Dumps to an FTP or TFTP Server

Use the **copy core** command to copy core dumps from the CSS to a File Transfer Protocol (FTP) or Trivial File Transfer Protocol (TFTP) server. This command is available in SuperUser mode. A core dump occurs when the CSS experiences a fatal error.

To see a list of core dumps, enter the **copy core ?** command.

**Note** The CSS FTP server supports only the active (normal) FTP mode of operation. It does not support the passive FTP mode of operation.

## Copying Core Dumps to an FTP Server

Use the **copy core ftp** command to copy a core dump to an FTP server. This command is available only in SuperUser mode.

Before you copy a core dump from the CSS to an FTP server, create an FTP record file containing the FTP server IP address, username, and password. For information on configuring an FTP record, see the "Creating an FTP Record" section.

The syntax for this command is:

**copy core** *coredump_filename* **ftp** *ftp_record filename*

The variables are as follows:

- *coredump_filename* - The name of the core dump on the CSS. Enter an unquoted text string with no spaces and a maximum of 32 characters.
- *ftp_record* - The name of the FTP record file that contains the FTP server IP address, username, and password. Enter an unquoted text string with no spaces and a maximum of 32 characters.
- *filename* - The name you want to assign to the file on the FTP server. Include the full path to the file. Enter an unquoted text string with no spaces and a maximum of 32 characters.

For example:

```
# copy core dumpfile ftp ftpserv1 starlogthurs
```

## Copying Core Dumps to a TFTP Server

Use the **copy core tftp** command to copy a core dump to an TFTP server. This command is available only in SuperUser mode.

The syntax for this command is:

**copy core** *coredump_filename* **tftp** *ip_address_hostname filename*

The variables are as follows:

- *coredump_filename* - The name of the core dump on the CSS. Enter an unquoted text string with no spaces and a maximum of 32 characters.
- *ip_address_hostname* - The IP address or host name of the TFTP server to receive the file. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1) or in mnemonic host-name format (for example, myhost.mydomain.com). If you wish to use a host name, you must first set up a host table using the **host** command.
- *filename* - The name you want to assign to the file on the TFTP server. Include the full path to the file. Enter an unquoted text string with no spaces and a maximum of 32 characters.

## Showing CSS Configurations

The CSS CLI provides a comprehensive set of **show** commands that display CSS configurations. The **show** commands are mode-independent; that is, they are available in each mode. The CSS does not show configuration default values in the individual show output, even when you specify a CLI command to configure a default value.

To display the list of **show** commands, enter:

```
(config)# show ?
```

This section includes the following topics:

- Showing Software Information
- Showing Hardware Information
- Showing Unique Device Identifier Information
- Showing System Resources
- Showing System Uptime

- Showing Disk Information
- Showing User Information
- Showing Current Logins

## Showing Software Information

To display the software versions installed on the CSS, use the following commands:

- **show version** - Displays details about the currently installed software version, including the version of Flash software code, whether the software is set to primary or secondary, your license number, and the version of the CiscoView Device Manager (CVDM), if it is installed on the CSS.
- **show installed-software version-limit** - Displays the maximum number of software versions allowed on your CSS.
- **show installed-software -** Displays a list of currently installed software on the CSS.
- **show installed-software** *disk_slot* - Displays a list of currently installed software on a specific disk in a dual-disk CSS. Valid selections are 0 (for the disk in slot 0) or 1 (for the disk in slot 1).

**Note** Use the **version** command in SuperUser mode to display the version of software currently running on the CSS. This display also shows the version of Flash software code, whether the software is set to primary or secondary, your license number, and the installed version of CVDM.

For example:

```
# show version
Version:              sg0810002 (8.10.0.02)
Network Path:         e:/adi_directory/
Config Path:          e:/adi_directory/
Flash (Locked):       8.10.0.03
Flash (Operational):8.10.0.02
Type:                 PRIMARY
License Cmd Set(s):   Standard Feature Set
                      Enhanced Feature Set
                      SSH Server
CVDM Version:        cvdm-css-1.0
```

## Showing Hardware Information

Use the **show chassis** command to display a chassis configuration for the CSS. The syntax and options for this command are as follows:

- **show chassis** - Displays a summary of the chassis configuration.
- **show chassis slot** *number* - Displays the operational parameters for a slot in a CSS 11503 or CSS 11506 chassis. Enter an integer value for the chassis slot number.
- **show chassis verbose** - Displays detailed information about the chassis configuration.
- **show chassis flash** - Displays the operational and locked Flash software code on the CSS 11501, and the CSS 11503 or CSS 11506 SCM and I/O modules. An asterisk (*) character before a Flash version of code and build number indicates that it is active.
- **show chassis inventory** - Displays the physical configuration of the CSS including part and serial numbers.
- **show chassis session-processors** - Displays the weight and power summary of the session processors in the CSS chassis.

To display the CSS Unique Device Identifier (UDI) information, see the "Showing Unique Device Identifier Information" section.

For example, to view a summary of the CSS chassis configuration, enter:

```
# show chassis
```

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

***Table 1-1 Field Descriptions for the show chassis Command***

| Field | Description |
|---|---|
| Product Name | The model number of the CSS. |
| SW Version | The software version currently running on the CSS. |
| Serial Number | The serial number of the chassis Flash memory device. |
| Base MAC Address | The MAC address for the chassis. |
| Slot/Module Number | The number of the CSS 11501, CSS 11503 or CSS 11506 chassis slot in which the module resides. |
| Module Name | The name of the module installed in the CSS. |
| Status | The operational status of the module. The possible states are as follows:<br>• primary<br>• backup<br>• powered-off<br>• powered-on<br>• bad<br>• unknown<br>• empty slot |
| Slot/Port | The slot and port number on the CSS 11503 or CSS 11506 (for example, 2/1). |
| Port Number | The port number on the CSS 11501 (for example, 1). |
| Name | The name of the interface port on the CSS 11501 or the module installed in the CSS 11503 or CSS 11506. |
| Status | The operational status of the interface port/module. The possible states are as follows:<br>• Online<br>• Offline |

Table 1-2 describes the fields in the **show chassis slot** command output.

*Table 1-2 Field Descriptions for the show chassis slot Command*

| Field | Description |
|---|---|
| Product Name | The model number of the CSS. |
| SW Version | The software version currently running on the CSS. |
| Serial Number | The serial number of the chassis Flash memory device. |
| Base MAC Address | The MAC address for the chassis. |
| Slot Number | The number of the CSS 11503 or CSS 11506 chassis slot in which the module resides. |
| Type | The name and product number of the installed module. |
| Serial Number | The serial number of the module. |
| Number of Ports | The total number of ports in an I/O module. |
| Status | The operational status of the module. The possible states are as follows: <br>• primary <br>• backup <br>• powered-off <br>• powered-on <br>• bad <br>• unknown <br>• empty slot |
| Port Number | The Ethernet port number. |

*Table 1-2 Field Descriptions for the show chassis slot Command (continued)*

| Field | Description |
|---|---|
| Port Name | The port name. |
| Operational Status | The status of the port. The possible states are as follows:<br>• online<br>• offline-ok<br>• offline-bad<br>• bad<br>• going-online<br>• going-offline<br>• inserted<br>• post<br>• post-ok<br>• post-fail<br>• post-bad-comm<br>• any<br>• unknown-state |

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

*Table 1-3 Field Descriptions for the show chassis verbose Command*

| Field | Description |
|---|---|
| Product Name | The model number of the CSS. |
| SW Version | The software version currently running on the CSS. |
| Serial Number | The serial number of the chassis Flash memory device. |
| Base MAC Address | The MAC address for the chassis. |
| Module(s) Found | The number of modules installed in the chassis. |
| Power Supplies Found | The number of power supplies installed in the chassis. |
| Fan(s) Found | The number of fans installed in the chassis. |
| Slot/Subslot | The number of the CSS 11503 or CSS 11506 chassis slot in which the module resides. |
| Module Name | The name of the module installed in the CSS 11501. |
| Operational | The active Flash code on the CSS. |
| Locked | The inactive Flash code available on the CSS. |
| Slot Number | The number of the CSS 11503 or CSS 11506 chassis slot in which the module resides. |
| Module Number | The number of the CSS 11501 chassis slot in which the module resides. |
| Type | The name and product number of the installed module. |
| Serial Number | The serial number of the module. |
| Number of Ports | The total number of ports in an I/O module. |

*Table 1-3 Field Descriptions for the show chassis verbose Command (continued)*

| Field | Description |
|---|---|
| Status | The operational status of the module. The possible states are as follows:<br>• primary<br>• backup<br>• powered-off<br>• powered-on<br>• bad<br>• unknown<br>• empty slot |
| Port Number | The Ethernet port number. |
| Port Name | The port name. |
| Operational Status | The status of the port. The possible states are as follows:<br>• online<br>• offline-ok<br>• offline-bad<br>• bad<br>• going-online<br>• going-offline<br>• inserted<br>• post<br>• post-ok<br>• post-fail<br>• post-bad-comm<br>• any<br>• unknown-state |

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

*Table 1-4 Field Descriptions for the show chassis flash Command*

| Field | Description |
|---|---|
| Product Name | The model number of the CSS. |
| SW Version | The currently running software version on the CSS. |
| Serial Number | The serial number of the chassis Flash. |
| Base MAC Address | The MAC address for the chassis. |
| Slot/Subslot | The number of the CSS 11503 or CSS 11506 chassis slot in which the module resides. |
| Module Name | The name of the module installed in the CSS 11501. |
| Operational | The active Flash code on the CSS. |
| Locked | The inactive Flash code available on the CSS. |

Table 1-5 describes the fields in the **show chassis inventory** command output.

*Table 1-5 Field Descriptions for the show chassis inventory Command*

| Field | Description |
|---|---|
| Product Name | The model number of the CSS. |
| SW Version | The software version currently running on the CSS. |
| Serial Number | The serial number of the chassis Flash memory device. |
| Base MAC Address | The MAC address for the chassis. |
| Slot | The number of the CSS 11503 or CSS 11506 chassis slot in which the module resides. |
| Module | The number of the CSS 11501 chassis slot in which the module resides. |
| Part | The name of the board in the CSS 11501 chassis. |
| Module/Part Name | The name of the module installed in the CSS. |
| Serial | The serial number of the module. |

Table 1-6 describes the fields in the **show chassis session-processors** command output.

*Table 1-6 Field Descriptions for the show chassis session-processor Command*

| Field | Description |
|---|---|
| Chassis Total Weight | The combined relative weights of all active session processors in the CSS chassis. |
| SP Modules Total/Active | The total number of installed modules that contain session processors, and the number of active modules that contain session processors. |
| Name | The name of the module installed in the CSS. |
| Slot | The number of the CSS 11503 or CSS 11506 chassis slot in which the module resides. |
| Module | The number of the CSS 11501 chassis slot in which the module resides. |
| Slot | For a CSS 11503 or CSS 11506, the number of the chassis slot in which the session processor resides. |
| Sub | For a CSS 11503 or CSS 11506, the number of the chassis module subslot in which the session processor resides. |
| Weight | A value assigned to an SP based on its ability to provide session processing. An active SP has a relative weight assignment greater than 0. A weight of 0 prevents the SP from performing any session processing. |
| Power Percentage (%) | A value calculated from an SP-assigned weight relative value that represents the session processor share of the total session processing capacity in the chassis. |

## Showing Unique Device Identifier Information

Use the **show inventory** command to display the Cisco Unique Device Identifier (UDI) information for the CSS chassis and the modules in each chassis slot. The UDI information includes:

- Product Identifier (PID) - The product identifier used to order the chassis or module. For example, the CSS5-SAM identifies the CSS Session Accelerator module.
- Version Identifier (VID) - The version of the chassis or module, identifying its sequence in a series of modifications.
- SN (Serial Number) - The serial number for the chassis or module.

**Note** This command does not display the fan and power supply modules, or modules that do not have a PID. If the chassis or module does not have a VID or has a VID in an incorrect format, the VID field is blank. If the chassis or module does not have a serial number, the SN field is blank.

The syntax for this command is:

**show inventory**

For example, to view the UDI information on the CSS chassis and modules, enter:

```
# show inventory
```

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

***Table 1-7 Field Descriptions for the show inventory Command***

| Field | Description |
|---|---|
| NAME | The chassis or the slot number of the module. |
| DESCR | A product description of the chassis or module. The description can have up to 60 characters. |
| PID | The product identifier of the chassis or module. The PID is an alphanumeric identifier used to order the chassis or module. The PID can have up to 18 characters. |

*Table 1-7 Field Descriptions for the show inventory Command*

| Field | Description |
|---|---|
| VID | The 3-character version identifier for the chassis or module. The VID displays in the V*nn* format, where *nn* is the number of the version. For example, V11 is the 11th occurence in a series of modification.<br><br>If the chassis or module does not have a VID or has a VID in an incorrect format, this field is blank. |
| SN | The 11-character serial number for the chassis or module. If the chassis or module does not have a serial number, this field is blank. |

## Showing System Resources

Use the **show system-resources** command to display information about the size of the installed memory and free memory available on the:

- CSS 11501.
- CSS 11503 or CSS 11506 SM and SCM module. The CSS displays system resources for the primary SCM. To view a specific slot in the CSS, use the **show system-resources** *slot_number* command. Enter a number to specify the slot number (*slot_number*) in the CSS.

**Note** Issuing the **show system-resources** command can cause CSS CPU usage to increase. If the CSS 11503 or 11506 has multiple modules installed, using this command increases CPU usage accordingly. The increased CPU usage is a direct result of the computational overhead that occurs when the CSS polls the modules and calculates CPU usage.

Table 1-8 describes the fields in the **show system-resources** command output.

*Table 1-8 Field Descriptions for the show system-resources Command*

| Field | Description |
|---|---|
| Installed Memory | The total memory size in the CSS |
| Free Memory | The amount of free memory available |
| CPU | The utilized percentage of the CPU |
| **Buffer Statistics** | |
| Buffer Pool | The buffer pool index |
| Size | The size, in bytes, of each data buffer in the buffer pool |
| Total | The total number of buffers in the buffer pool |
| Available | The current number of available buffers in the buffer pool |
| Failures | The number of failures to obtain a buffer from the buffer pool |
| Low Buffer Count | The lowest recorded number of available buffers |

Use the **show system-resources cpu_summary** command to display a summary of the CPU utilization by all modules in the CSS 11501, CSS 11503, or CSS 11506 chassis.

Table 1-9 describes the fields in the **show system-resources cpu_summary** command output.

*Table 1-9 Field Descriptions for the show system-resources cpu_summary Command*

| Field | Description |
|---|---|
| Name/Module | The name of the module installed in the CSS. |
| Slot | For a CSS 11503 or CSS 11506, the number of the chassis slot in which the module resides. |
| Sub | For a CSS 11503, the number of the chassis module subslot in which the memory resides. |

*Table 1-9 Field Descriptions for the show system-resources cpu_summary Command (continued)*

| Field | Description |
|---|---|
| Module | The number of the module in the CSS 11501 chassis. |
| CPU% | The percentage of the total CPU capacity that is currently in use. |

## Showing System Uptime

Use the **show uptime** command to display the length of time the CSS has been running. The time is displayed in *hour*:*minute*:*second* format. For the CSS 11503 or CSS 11506, this command shows the length of time each module has been running.

To display how long the CSS has been running, enter:

```
# show uptime
Uptime:
10 days 03:25:22
```

## Showing Disk Information

Use the **show disk** command to view general information about the CSS hard disk or Flash disk. The information includes the total number of clusters on the disk, the free space available, and the number of files, folders, and bad clusters on the disk.

To display specific CSS disk information, use the following **show disk** commands:

- **show disk** - Displays disk information for the hard disk or Flash disk. If the CSS includes two disks, the **show disk** command lists information for both disks.
- **show disk** *disk_slot* - Displays disk information for a specific slot in a dual-disk CSS. Valid selections are 0 (for the disk in slot 0) or 1 (for the disk in slot 1). The default is the disk from which the CSS booted.

For example, to display the CSS disk information for the disk in slot 1, enter:

```
# show disk 1
```

Table 1-10 describes the fields in the **show disk** command output for the CSS.

***Table 1-10 Field Descriptions for the show disk Command***

| Field | Description |
|---|---|
| Total # of Clusters | The total number of clusters on the disk |
| Bytes Per Cluster | The number of bytes in each cluster |
| Free Clusters | The number of available clusters on the disk |
| Bad Clusters | The number of bad clusters on the disk |
| Free Bytes | The available disk space, in bytes and megabytes |
| Max Contiguous Free Bytes | The maximum number of contiguous free bytes (and megabytes) found on the disk |
| Files | The number of files on the disk |
| Folders | The number of folders on the disk |
| Total Bytes in Files | The total number of bytes in all of the files found on the disk |

*Table 1-10 Field Descriptions for the show disk Command (continued)*

| Field | Description |
|---|---|
| Lost Chains | The total number of lost chains found on the disk |
| Total Bytes in Lost Chains | The total number of bytes in all of the lost chains found on the disk |

## Showing User Information

Use the **show user-database** command to view CSS operating information related to a single user, or to multiple users. This command displays user information related to login privileges, the type of user, and directory access privileges.

To display all users currently defined in the CSS, enter:

```
(config)# show user-database
```

To display information for a specific user, enter:

```
(config)# show user-database picard
```

Table 1-11 describes the fields in the **show user-database** command output.

*Table 1-11 Field Descriptions for the show user-database Command*

| Field | Description |
|---|---|
| Virtual Authentication | Identifies if users must enter a username and password to log in to the CSS. |
| Console Authentication | Identifies if console port authentication of locally defined usernames and passwords logging in to the CSS in enabled. |
| Username | The name of the user. |
| Privilege Level | The privilege level of the user. |

*Table 1-11 Field Descriptions for the show user-database Command (continued)*

| Field | Description |
|---|---|
| Type | The type of user. Types are as follows: • Administrator (administrative username, created using the **username-offdm** command) • Technician (technician username, created using the **username-technician** command) If the field is blank, the user is neither an administrator nor a technician. **Note** The **username-offdm** command is for use by system administrative personnel only. The **username-technician** command is for use by technical personnel only. |

*Table 1-11 Field Descriptions for the show user-database Command (continued)*

| Field | Description |
|---|---|
| Directory Access | The directory access privileges for the listed usernames (as specified through the **dir-access** option of the **username** command). There are a series of access privilege codes assigned to the seven CSS directories in the following order: Script, Log, Root (installed CSS software), Archive, Release Root (configuration files), Core, and MIBs directories. By default, users have both read- and write-access privileges (B) to all seven directories. The levels for each of the CSS directories can be one of the following access privilege codes:<br>• R - Read-only access to the CSS directory<br>• W - Write-only access to the CSS directory<br>• B - Both read- and write-access privileges to the CSS directory (default for all users)<br>• N - No access privileges to the CSS directory<br>For example, BBNBNBB indicates that the user has no access to the root and release root directories, but has read and write access to the script, log, archive, core, and MIB directories. |

## Showing Current Logins

Use the **show lines** command to display currently connected lines or sessions. A connected line is a console or Telnet session. This command is available in all modes.

To display currently connected lines or sessions, enter:

```
(config)# show lines
```

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

*Table 1-12 Field Descriptions for the show lines Command*

| Field | Description |
|---|---|
| Line | The type of session. The * indicates your current session. |
| User | The login name of the user. |
| Login | The amount of time that the user has been logged in on the CSS. |
| Idle | The amount of time that the session has been idle. |
| Location | The location where the session is occurring. |

## Where to Go Next

Chapter 2, Specifying the CSS Boot Configuration, provides information on how to setup the boot configuration for the CSS, including configuring an FTP record and specifying the primary and secondary location from which the CSS accesses the boot image.

# Specifying the CSS Boot Configuration

This chapter describes how to set the boot configuration, both the primary and secondary boot files, for the CSS. Information in this chapter applies to all CSS models, except where noted.

This chapter contains the following major sections:

- Boot Setup Quick Start
- Accessing Boot Mode
- Specifying the Primary Boot Configuration
- Specifying the Secondary Boot Configuration
- Configuring a Boot Configuration Record for the Passive SCM
- Showing the Boot Configuration
- Booting the CSS from a Network Drive

As an alternate procedure for managing the CSS boot configuration from the CLI, you can use the Offline DM menu. Refer to Appendix B, Using the Offline Diagnostic Monitor Menu, for details.

## Boot Setup Quick Start

Table 2-1 provides a quick overview of the steps required to configure the CSS to boot from a primary boot file and from a secondary boot file. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI command, see the sections following Table 2-1.

*Table 2-1 Boot Setup Quick Start*

| Task and Command Example |
|---|
| **1.** Create a File Transfer Protocol (FTP) record file to use when accessing an FTP server from the CSS. This step is optional.<br>`# ftp-record arrowrecord 192.168.19.21 bobo “secret” /outgoing`<br>**Note** Refer to Chapter 1, Managing the CSS Software, for details on creating an FTP record. |
| **2.** Access boot mode.<br>`(config)# boot` |
| **3.** Specify the primary boot configuration.<br>`(config-boot)# primary boot-file ap0720002`<br>`(config-boot)# primary boot-type boot-via-ftp arrowrecord` |
| **4.** Specify the secondary boot configuration.<br>`(config-boot)# secondary boot-file ap0720001`<br>`(config-boot)# secondary boot-type boot-via-disk` |
| **5.** Exit from boot mode.<br>`(config-boot)# exit` |
| **6.** Save your configuration changes to the startup-config file (recommended). If you do not save the running configuration, all configuration changes are lost upon reboot.<br>`(config)# copy running-config startup-config` |

## Accessing Boot Mode

Boot configuration mode contains all the commands necessary to boot the CSS and maintain the software revision. To access this mode, use the **boot** command from global configuration mode.

To access boot mode, enter:

```
(config)# boot
```

The CSS enters boot mode.

```
(config-boot)#
```

## Specifying the Primary Boot Configuration

You can configure a primary location from which the CSS accesses the boot image. Use the **primary** command to specify the primary boot configuration. The options for this boot-mode command are as follows:

- **primary boot-file** - Specifies the primary boot file
- **primary boot-type** - Specifies the primary boot method: local disk, using FTP, or a network-mounted file system using FTP
- **primary config-path** - Specifies the path to a network CSS configuration

This section includes the following topics:

- Specifying the Primary Boot File
- Specifying the Primary Boot Type
- Specifying the Primary Configuration Path

### Specifying the Primary Boot File

Use the **primary boot-file** command to specify the primary boot file. Enter the primary boot file as an unquoted text string with no spaces and a maximum of 64 characters.

To specify the primary boot filename, enter:

```
(config-boot)# primary boot-file ap0720002
```

To display a list of boot filenames, enter:

```
(config-boot)# primary boot-file ?
```

To remove the primary boot file, enter:

```
(config-boot)# no primary boot-file
```

## Specifying the Primary Boot Type

Use the **primary boot-type** command to specify the location from which the CSS accesses the primary boot image upon system reboot or when you download new software.

The syntax for this boot mode command is:

**primary boot-type** [**boot-via-disk**|**boot-via-ftp** *ftp_record*|
**boot-via-network** *ftp_record*]

The options and variables for this command are as follows:

- **boot-via-disk** - Boots the CSS from a software version that resides on the CSS disk.
- **boot-via-ftp** *ftp_record* - Downloads an ADI file containing CSS software that you want to install on the CSS disk. The CSS accesses the ADI or GZIP file containing the CSS software from an FTP server, copies the file to the disk, and unpacks it. The CSS then boots from the disk.
- **boot-via-network** *ftp_record* - Uses FTP to boot the CSS from software located on a network-mounted file system on a remote system. Instead of the CSS disk, the network file system contains the CSS software. The CSS boots from this file system and loads the configuration in to memory.

**Note** A network boot requires that the CSS contains an operational disk.

The *ftp_record* variable is the name of the FTP record file that contains the FTP server IP address, username, and password. Enter an unquoted text string with no spaces. Refer to Chapter 1, Managing the CSS Software, for details on creating an FTP record.

For example, to configure the primary boot-type to **boot-via-disk**, enter:

```
(config-boot)# primary boot-type boot-via-disk
```

To remove the primary boot type, enter:

```
(config-boot)# no primary boot-type
```

## Primary Boot Configuration Considerations

When you select **primary boot-type boot-via-ftp** or **primary boot-type boot-via-network**, make sure you properly connect the Ethernet Management port on the CSS to the network. The locations of the Ethernet Management port on the CSS are listed below.

- CSS 11503 and CSS 11506 - SCM 10 Mbps-Ethernet Management port
- CSS 11501 - Front Panel 10 Mbps-Ethernet Management port

When you select **primary boot-type boot-via-network**, make sure you:

- Locate the remote system on the network where you will copy the CSS software.
  - Make sure the CSS can access the system via FTP.
  - Copy the CSS software Zip file from www.cisco.com onto the designated network server.
  - Create a directory and unzip the file in to the directory. This directory will contain all of the boot files and directories.
- Create an FTP record on the CSS to the directory that contains the CSS software on the network drive. Refer to Chapter 1, Managing the CSS Software, for details on creating an FTP record.

**Note** Be aware of the following network boot restrictions: a network boot is not supported on UNIX workstations, and the War-FTP daemon is not supported for network-booting the system software.

A network boot requires that the CSS contains an operational disk.

## Specifying the Primary Configuration Path

An alternate configuration path allows multiple CSSs to use the same boot image while keeping their configuration information in separate directories. Use the **primary config-path** command to specify the alternate path to a network configuration for the network boot method. Note that the CSS must be able to access the configuration path through an FTP server as defined in the FTP record for the network boot method.

When using an alternate configuration path, make sure the path leads to a directory containing the script, log, and information subdirectories, and to the startup-config file. These subdirectories must contain the files in the corresponding subdirectories of the unzipped boot image. First, create these subdirectories on the FTP server, then copy the files from the boot image to the subdirectories.

Enter the configuration pathname as an unquoted text string with no spaces and a maximum of 64 characters.

To configure the primary configuration path, enter:

```
(config-boot)# primary config-path f:/bootdir/
```

To remove the primary network configuration path, enter:

```
(config-boot)# no primary config-path
```

## Specifying the Secondary Boot Configuration

You can configure a secondary location from which the CSS accesses the boot image when the primary boot configuration fails. Use the **secondary** command to specify the secondary boot configuration. The CSS uses the secondary boot configuration when the primary boot configuration fails. The options for this boot mode command are as follows:

- **secondary boot-file** - Specifies the secondary boot file
- **secondary boot-type** - Specifies the boot method: local disk or FTP
- **secondary config-path** - Specifies the path to a network configuration using FTP

This section includes the following topics:

- Specifying the Secondary Boot File
- Specifying the Secondary Boot Type
- Specifying the Secondary Configuration Path

## Specifying the Secondary Boot File

To specify the secondary boot file that the CSS uses when the primary boot configuration fails, use the **secondary boot-file** command. Enter the boot file as an unquoted text string with no spaces and a maximum of 64 characters.

To specify the secondary boot filename, enter:

```
(config-boot)# secondary boot-file ap0720001
```

To display a list of secondary boot filenames, enter:

```
(config-boot)# secondary boot-file ?
```

To remove the secondary boot file, enter:

```
(config-boot)# no secondary boot-file
```

## Specifying the Secondary Boot Type

Use the **secondary boot-type** command to specify the secondary boot configuration.

The syntax for this boot mode command is:

**secondary boot-type** [**boot-via-disk**|**boot-via-ftp** *ftp_record*|
**boot-via-network** *ftp_record*]

The options and variables for this command are as follows:

- **boot-via-disk** - Boots the CSS from a software version that resides on the CSS disk.
- **boot-via-ftp** *ftp_record* - Downloads an ADI file containing CSS software that you want to install on the CSS disk. The CSS accesses the ADI or GZIP file containing the CSS software from an FTP server, copies the file to the disk, and unpacks it. The CSS then boots from the disk.

- **boot-via-network** *ftp_record* - Uses FTP to boot the CSS from software located on a network-mounted file system on a remote system. Instead of the CSS disk, the network file system contains the CSS software. The CSS boots from this file system and loads the configuration in to memory.

**Note** A network boot requires that the CSS contains an operational disk.

The *ftp_record* variable is the name of the FTP record file that contains the FTP server IP address, username, and password. Enter an unquoted text string with no spaces. Refer to Chapter 1, Managing the CSS Software, for details on creating an FTP record.

For example, to specify the secondary boot type as **boot-via-disk**, enter:

```
(config-boot)# secondary boot-type boot-via-disk
```

To remove the secondary boot type, enter:

```
(config-boot)# no secondary boot-type
```

## Secondary Boot Configuration Considerations

When you select **secondary boot-type boot-via-ftp** or **secondary boot-type boot-via-network**, make sure you properly connect the Ethernet Management port on the CSS to the network. Note the locations of the Ethernet Management port on the CSS as listed below.

- CSS 11503 and CSS 11506 - SCM 10 Mbps-Ethernet Management port
- CSS 11501 - Front Panel 10 Mbps-Ethernet Management port

When you select **secondary boot-type boot-via-network**, make sure you:

- Locate the remote system on the network where you will copy the CSS software.
  - Make sure the CSS can access the system via FTP.
  - Copy the CSS software Zip file from www.cisco.com onto the designated network server.
  - Create a directory and unzip the file in to the directory. This directory will contain all of the boot files and directories.

- Create an FTP record on the CSS to the directory that contains the CSS software on the network drive.

**Note** Be aware of the following network boot restrictions: a network boot is not supported on UNIX workstations, and the War-FTP daemon is not supported for network-booting the system software.

A network boot requires that the CSS contains an operational disk.

## Specifying the Secondary Configuration Path

An alternate configuration path allows multiple CSSs to use the same boot image while keeping their configuration information in separate directories. Use the **secondary config-path** command to specify the alternate path to a network configuration for the network boot method. Note that the CSS must be able to access the configuration path through an FTP server as defined through the FTP record for the network boot method.

When using an alternate configuration path, make sure the path leads to a directory containing the script, log, and information subdirectories, and to the startup-config file. These subdirectories must contain the files in the corresponding subdirectories of the unzipped boot image. First, create these subdirectories on the FTP server, then copy the files from the boot image to the subdirectories.

Enter the configuration pathname as an unquoted text string with no spaces and a maximum of 64 characters.

To configure the secondary configuration path, enter:

```
(config-boot)# secondary config-path f:/bootdir/
```

To remove the secondary network configuration path, enter:

```
(config-boot)# no secondary config-path
```

## Configuring a Boot Configuration Record for the Passive SCM

You can configure the individual components of the boot configuration record on the passive SCM installed in a CSS 11506 chassis. A passive module is a standby module in case of an active module failure. Use the **passive** command to configure the boot configuration record for the current passive SCM. The boot configuration record consists of the IP address, subnet mask, boot method, and boot file.

Using the **sync** options for the **passive** command, copy the boot configuration record from the active SCM to the passive SCM. In most CSS configurations, the active and passive SCMs have the same boot record.

The **passive** command also allows you to configure the individual components of the boot configuration record on the passive SCM. For example, you can configure a boot record on the passive SCM that has a software version that differs from the active SCM. The boot configuration record allows you to run a new software version on the active SCM and have an older software version on the passive SCM.

You can also configure a different IP address on the passive SCM to track an active-to-passive state transition between the SCMs. You can track active-to-passive state transitions through a network management station, where you can receive SNMP host traps.

The **passive** command and its options affect only the current passive SCM. When you configure the passive SCM, the set values are loaded in to its NVRAM. If the passive SCM transitions to the active state, it continues to retain these values, but is no longer affected by these commands; boot commands are not saved in the running-config file.

This section includes the following topics:

- Configuring the Passive SCM Gateway Address
- Configuring the Passive SCM IP Address
- Configuring the Passive SCM Primary Boot File
- Configuring the Passive SCM Primary Boot Type
- Configuring the Passive SCM Primary Configuration Path
- Configuring the Passive SCM Secondary Boot File
- Configuring the Passive SCM Secondary Boot Type

- Configuring the Passive SCM Secondary Configuration Path
- Configuring the Passive SCM Subnet Mask
- Copying Configuration Information from the Active SCM to the Passive SCM

## Configuring the Passive SCM Gateway Address

Use the **passive gateway address** command to configure an Ethernet management port default gateway to load a boot file on a CSS across different subnets for the passive SCM. Enter the IP address for the passive SCM to be used upon CSS boot up. Do not enter an all-zero IP address.

For example:

```
(config-boot)# passive gateway address 172.16.3.6
```

To change the passive SCM boot gateway address, reenter the **passive gateway address** command.

## Configuring the Passive SCM IP Address

Use the **passive ip address** command to configure the boot IP address for the passive SCM. Enter the IP address for the passive SCM to be used upon CSS boot up. Do not enter an all-zero IP address.

For example:

```
(config-boot)# passive ip address 172.16.3.6
```

To change the passive SCM boot IP address, reenter the **passive ip address** command.

## Configuring the Passive SCM Primary Boot File

Use the **passive primary boot-file** command to specify the primary boot image for the passive SCM. Enter the filename of the primary boot image for the passive SCM as an unquoted text string with no spaces and a maximum of 64 characters. To display a list of filenames, enter **passive primary boot-file ?**.

For example:

```
(config-boot)# passive primary boot-file ap0720002
```

To remove the primary boot file from the passive SCM, enter:

```
(config-boot)# no passive primary boot-file
```

## Configuring the Passive SCM Primary Boot Type

Use the **passive primary boot-type** command to specify the location from which the CSS accesses the primary boot image for the passive SCM upon system reboot or when you download new software.

The syntax for this boot mode command is:

**passive primary boot-type** [**boot-via-disk**|**boot-via-ftp** *ftp_record*|
**boot-via-network** *ftp_record*]

The options and variables for the **passive primary boot-type** are as follows:

- **boot-type boot-via-disk** - Boots the CSS from a software version that currently resides on the CSS disk.
- **boot-type boot-via-ftp** *ftp_record* - Downloads an ADI file containing CSS software that you want to install on the CSS disk. The CSS accesses the ADI or GZIP file containing the CSS software from an FTP server, copies the file to the disk, and unpacks it. The CSS then boots from the disk.
- **boot-type boot-via-network** *ftp_record* - Uses FTP to boot the CSS from software located on a network-mounted file system on a remote system. Instead of the CSS disk, the network file system contains the CSS software. The CSS boots from this file system and loads the configuration in to memory.

**Note** A network boot requires that the CSS contains an operational disk.

The *ftp_record* variable is the name of the FTP record file that contains the FTP server IP address, username, and password. Enter an unquoted text string with no spaces. Refer to Chapter 1, Managing the CSS Software, for details on creating an FTP record.

For example:

```
(config-boot)# passive primary boot-type boot-via-ftp arecord
```

To remove the primary boot type from the passive SCM, enter:

```
(config-boot)# no passive primary boot-type
```

## Configuring the Passive SCM Primary Configuration Path

An alternate configuration path allows multiple CSSs to use the same boot image while keeping their configuration information in separate directories. Use the **passive primary config-path** command to specify an alternate path to a network configuration for a passive SCM network boot method. Note that the CSS must be able to access the configuration path through an FTP server as defined through the FTP record for the network boot method.

When using an alternate configuration path, make sure the path leads to a directory containing the script, log, and information subdirectories, and the startup-config file. These subdirectories must contain the files in the corresponding subdirectories in the unzipped boot image. First, create these subdirectories on the FTP server, then copy the files from the boot image to the subdirectories.

Enter the configuration path for network configuration. Enter an unquoted text string with no spaces and a maximum of 64 characters. For example:

```
(config-boot)# passive primary config-path c:/bootdir/
```

To remove the primary network configuration path, enter:

```
(config-boot)# no passive primary config-path
```

## Configuring the Passive SCM Secondary Boot File

Use the **passive secondary boot-file** command to specify the secondary boot image for the passive SCM. Enter the name of the boot file for the primary boot image as an unquoted text string with no spaces and a maximum of 64 characters. To display a list of boot filenames, enter **passive secondary boot-file ?**. For example:

```
(config-boot)# passive secondary boot-file ap0720001
```

To remove the secondary boot file from the passive SCM, enter:

```
(config-boot)# no passive secondary boot-file
```

## Configuring the Passive SCM Secondary Boot Type

Use the **passive secondary boot-type** command to specify the secondary boot configuration for the passive SCM. The secondary boot configuration is used when the primary configuration fails.

The syntax for this boot mode command is:

**passive secondary boot-type** [**boot-via-disk**|**boot-via-ftp** *ftp_record*|
**boot-via-network** *ftp_record*]

The options and variables for the **passive secondary boot-type** command are as follows:

- **boot-type boot-via-disk** - Boots the CSS from a software version that resides on the CSS disk.
- **boot-type boot-via-ftp** *ftp_record* - Downloads an ADI file containing CSS software that you want to install on the CSS disk. The CSS accesses the ADI or GZIP file containing the CSS software from an FTP server, copies the file to the disk, and unpacks it. The CSS then boots from the disk.
- **boot-type boot-via-network** *ftp_record* - Uses FTP to boot the CSS from software located on a network-mounted file system on a remote system. Instead of the CSS disk, the network file system contains the CSS software. The CSS boots from this file system and loads the configuration in to memory.

**Note** A network boot requires that the CSS contains an operational disk.

The *ftp_record* variable is the name of the FTP record file that contains the FTP server IP address, username, and password. Enter an unquoted text string with no spaces. Refer to Chapter 1, Managing the CSS Software, for details on creating an FTP record.

For example:

```
(config-boot)# passive secondary boot-type boot-via-disk
```

To remove the secondary boot type from the passive SCM, enter:

```
(config-boot)# no passive secondary boot-type
```

## Configuring the Passive SCM Secondary Configuration Path

An alternate configuration path allows multiple CSSs to use the same boot image while keeping their configuration information in separate directories. The CSS must be able to access the configuration path through an FTP server as defined through the FTP record for the network boot method. Use the **passive secondary config-path** command to specify the secondary alternate path to a network configuration for a passive SCM network boot method.

When using an alternate configuration path, make sure that the path leads to a directory containing the script, log, and information subdirectories, and the startup-config file. These subdirectories must contain the files in the corresponding subdirectories of the unzipped boot image. First, create these subdirectories on the FTP server, then copy the files from the boot image to the subdirectories.

Enter the configuration path as an unquoted text string with no spaces and a maximum of 64 characters.

For example:

```
(config-boot)# passive secondary config-path c:/bootdir/
```

To remove the primary network configuration path, enter:

```
(config-boot)# no passive secondary config-path
```

## Configuring the Passive SCM Subnet Mask

Use the **passive subnet mask** command to configure the system boot subnet mask for the passive SCM.

For example:

```
(config-boot)# passive subnet mask 255.255.0.0
```

## Copying Configuration Information from the Active SCM to the Passive SCM

To copy the primary and secondary boot configuration record from NVRAM of the active SCM to the passive SCM, use the **passive sync** command. For the CSS 11506, the **passive sync** command also copies the startup- configuration file and synchronizes the clock time from the active SCM to the passive SCM. This command is available in boot mode.

To synchronize specific boot configuration, startup configuration, or clock time information between the active SCM and the passive SCM in a CSS 11506, use the following commands:

- **passive sync boot-config** - Copies the boot configuration record from the active SCM to the passive SCM.
- **passive sync startup-config** - Copies the startup-config file from the active SCM to the passive SCM.
- **passive sync image** - Copies the ADI of the boot-image file from the active SCM to the passive SCM.
- **passive sync time** - Synchronizes the clock time of the passive SCM with the active SCM.

To copy the primary and secondary boot configuration record, startup configuration, and clock time on a CSS 11506, enter:

```
(config-boot)# passive sync
```

To copy the boot configuration record from the active SCM to the passive SCM in a CSS 11506, enter:

```
(config-boot)# passive sync boot-config
```

## Showing the Boot Configuration

Use the **show boot-config** command to display the boot configuration. For example:

```
(config-boot)# show boot-config

!********************** BOOT CONFIG **********************
ip address 172.16.36.58
subnet mask 255.0.0.0
primary boot-file ap0720001
primary boot-type boot-via-disk
```

## Booting the CSS from a Network Drive

Network booting enables you to boot the CSS from a network drive using a .zip file of the CSS software version located on www.cisco.com. When you configure the CSS for network boot, the CSS must contain an operational disk (hard or Flash).

Use your customer login and password to access www.cisco.com. From this location, you can access the page listing the network boot .zip file versions of CSS software. Click an image to download the software.

**Note** Be aware of the following network boot restrictions: a network boot is not supported on UNIX workstations, and the War-FTP daemon is not supported for network-booting the system software. In addition, network booting does not support the use of core dumps from the CSS.

Perform a network boot if you want multiple CSSs to use the same boot image while keeping their own configuration information. Provide an alternate path for the location of the configuration information. This information must exist on the same network file system as the boot image.

When using an alternate configuration path, make sure the path leads to a directory containing the script, log, and information subdirectories. These subdirectories must contain the files in the corresponding subdirectories in the boot image. Create these subdirectories, then copy the files from the boot image.

This section includes the following topics:

- Configuring Network Boot for a Primary SCM
- Configuring Network Boot for a Passive SCM
- Showing Network Boot Configurations

## Configuring Network Boot for a Primary SCM

To configure network boot for a primary SCM on the CSS 11503 or CSS 11506:

1. Make sure the SCM management port has access to the network drive from which you are booting the CSS. The SCM mounts the drive, and reads and writes to the network drive.
2. Use FTP to install the software .zip file to the network drive base directory specified in the FTP record. This network directory must be the same directory that you use to boot the CSS.
3. Unzip the file. You must use the .zip distribution format for network loading.
4. Configure the FTP record. Refer to Chapter 1, Managing the CSS Software, for details on creating an FTP record. Note that the config-path and the base directory path in the FTP record associated with the network boot must contain a pathname that is distinct from a non-network drive name (for example, c: or host:).

   For example:

   ```
   # ftp-record bootrecord 192.168.19.21 bobo encrypted-password
   "secret" e:/adi_directory/
   ```

   This directory must contain the unzipped files.
5. Configure the CSS to boot from a network drive. For example:

   ```
   (config-boot)# primary boot-type boot-via-network bootrecord
   ```

6. Optionally, configure a primary configuration path to allow multiple CSSs to use the same boot image while keeping their configuration information in separate directories. The CSS must be able to access the configuration path through the FTP server as defined in the FTP record. For example:

   ```
   (config-boot)# primary config-path e:/adi_directory/
   ```

## Configuring Network Boot for a Passive SCM

To configure network boot for a passive SCM on the CSS 11503 or CSS 11506:

1. Configure an FTP record for the passive SCM, if not already configured (see the "Configuring a Boot Configuration Record for the Passive SCM" section).
2. Make sure the passive SCM management port has access to the network drive from which you are booting the CSS. If the primary SCM fails, the passive SCM connects to the remote disk and loads the software configuration.
3. Configure the CSS to boot from a network drive. For example:

   ```
   (config-boot)# passive primary boot-type boot-via-network
   bootrecord
   ```

   To display a list of configured FTP records, reenter the command and specify the **?** character. For example:

   ```
   (config-boot)# passive primary boot-type boot-via-network
   bootrecord ?
   ```

4. Optionally, configure a primary configuration path to allow multiple CSSs to use the same boot image while keeping their configuration information in separate directories. Your FTP daemon must support the drive mapping. Also, the CSS must be able to access the configuration path through the FTP server as defined in the FTP record. For example:

   ```
   (config-boot)# primary config-path e:/adi_directory/
   ```

## Showing Network Boot Configurations

Use the **show version** command to display the network boot configuration. For example:

```
(config)# show version

Version:               sg0730002 (7.30.0.02)
Network Path:          e:/adi_directory/
Config Path:           e:/adi_directory/
Flash (Locked):        7.20.0.03
Flash (Operational):7.30.0.02
Type:                  PRIMARY
Licensed Cmd Set(s):Standard Feature Set
                       Enhanced Feature Set
                       Secure Management
```

**Note** Use the **version** command in SuperUser mode to display the network boot configuration.

To display network boot configuration information, use the **show boot-config** command. For example:

```
(config)# show boot-config

!*********************** BOOT CONFIG ***********************
secondary config-path   e:/adi_directory/
secondary boot-type     boot-via-network Secondary-Boot
primary boot-file       sg0730002
primary boot-type       boot-via-network
subnet mask             255.0.0.0
ip address              192.168.4.226
```

## Where to Go Next

Chapter 3, Configuring User Profiles, provides information about how to configure CSS user profiles in the default-profile file.

# Configuring User Profiles

This chapter describes how to configure user profiles. Information in this chapter applies to all models of the CSS, except where noted.

This chapter contains the following major sections:

- User Profiles Overview
- User Profile Configuration Quick Start
- Configuring Idle Timeout
- Using Expert Mode
- Changing the CLI Prompt
- Modifying the History Buffer
- Configuring a Pre-Login Banner
- Configuring a Login Banner
- Copying and Saving User Profiles

## User Profiles Overview

The CSS contains a default-profile file that resides in the scripts directory on the CSS disk (hard disk or Flash disk). This file contains settings that are user-specific; that is, they apply uniquely to each user when that user logs in.

You can customize the following settings for each user:

- CLI prompt
- Expert mode
- History buffer
- Terminal parameters
- Login banner

Though the settings are user-specific, each default setting applies to all users until the user saves the default-profile file to a *username*-profile (where *username* is the current login username). You can continue using the default-profile file to ensure all users logging in to a CSS use the same settings. See the "Configuring a Login Banner" section for information on saving the default-profile file.

If you change a user setting and want to save this setting in the scripts directory of the current ADI, use the **copy profile** command. If you do not save this setting, the CSS stores the setting temporarily in a running profile. If you attempt to log out of the CSS without saving profile changes, the CSS prompts you that profile changes have been made and allows you to save or discard the changes.

You can also use the **save_profile** alias command to save your user-profiles settings to the scripts directory and then archive them in the CSS archive directory.

When you upgrade the ADI, the CSS does not contain all user profiles from the current ADI directory. If you wish to save user profiles permanently, use the **archive script** command. This command saves a user-profile file from the scripts directory to the archive directory. The archive directory is not overwritten during a software upgrade. Then you can restore them to the scripts directory with the **restore** command.

To access the CSS disk, FTP to the CSS. Use the appropriate commands to access the scripts directory and list the contents of the default-profile file. When logged into the CSS, use the **show profile** command to display either the default-profile file or your *username*-profile file.

For example:

```
# show profile

@prompt CSS11503
@no expert
alias all reboot "@configure;boot;rebo"
alias all shutdown "@configure;boot;shutd"
alias all logon "@configure;logging line \${LINE};exit"
alias all logoff "@configure;no logging line \${LINE};exit"
alias all aca-load "@script play service-load"
alias all dnslookup "@script play dnslookup"
alias super save_config "copy running-config startup-config;archive
startup-config"
alias super setup "script play setup"
alias super upgrade "script play upgrade"
alias super monitor "script play monitor"
alias super save_profile "copy profile user-profile;archive script
admin-profile
"
set CHECK_STARTUP_ERRORS "1" session
```

## User Profile Configuration Quick Start

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

*Table 3-1 User Profile Configuration Quick Start*

| Task and Command Example |
|---|
| **1.** Set the length of time a session can be idle before the CSS terminates a console or Telnet session.<br>`# terminal idle 15` |
| **2.** Set the number of output lines the CLI displays on the terminal screen.<br>`# terminal length 35` |

*Table 3-1 User Profile Configuration Quick Start (continued)*

| Task and Command Example |
|---|
| **3.** (Optional) Turn on the display of the --More-- prompt at the bottom of the terminal screen to indicate that additional CLI commands are to follow on the screen.<br>`# terminal more` |
| **4.** Define how the CSS displays subnet masks in show screens.<br>`# terminal netmask-format bitcount` |
| **5.** Set the total amount of time a session can be logged in before the CSS terminates a console or Telnet session.<br>`# terminal timeout 30` |
| **6.** Globally set the total amount of time all sessions can be active before the CSS terminates a console or Telnet session.<br>`(config)# idle timeout 15` |
| **7.** (Optional) Enable expert mode to *disable* the CSS from prompting you for confirmation when you make changes. Expert mode is available in SuperUser mode and is off by default.<br>`# expert` |
| **8.** (Optional) Change the default CLI prompt (CSS product model number followed by the # symbol).<br>`CSS11506# prompt CSS1-lab`<br>`CSS1-lab#` |
| **9.** (Optional) Modify the CLI history buffer length.<br>`# history length 80` |
| **10.** (Optional) Create and display a custom banner that appears after you log in to a CSS.<br>`# set BANNER “mybanner” session` |
| **11.** Copy the running profile from the CSS to the default-profile file, an FTP server, a TFTP server, or your user-profile file. For example:<br>`# copy profile default-profile`<br>**Note** This command is available only in SuperUser mode. |

## Configuring User Terminal Parameters

You can control the configure the output to the system terminal screen. Use the **terminal** command to configure terminal parameters. These parameters control output to the system terminal screen. Terminal parameters are user-specific; that is, they apply uniquely to each CSS user.

Use the **copy profile user-profile** command to add terminal command parameters to your user profile so that the parameters are used each time you log in. Otherwise, you must reenter the commands for the parameters to take effect each time you log in.

This section includes the following topics:

- Configuring Terminal Idle
- Configuring Terminal Length
- Configuring the More Terminal Prompt
- Configuring Terminal Netmask-Format
- Configuring Terminal Timeout

### Configuring Terminal Idle

By default, the terminal idle time is disabled. Use the **terminal idle** command to set the length of time a session can be idle before the CSS terminates a console or Telnet session. This command is available in the User and SuperUser modes. Enter an idle time between 0 and 65535 minutes. The default value is 0 (disabled).

To set a terminal idle time, enter:

```
# terminal idle 15
```

To restore the terminal idle time to the default state of disabled, enter:

```
# no terminal idle
```

## Configuring Terminal Length

By default, the CSS displays 25 lines of CLI output on the terminal screen. Use the **terminal length** command to set the number of output lines the CLI displays on the terminal screen. This command is available in User and SuperUser modes. Enter the number of lines you want the CLI to display, from 2 to 65535.

To set the line number to 35, enter:

```
# terminal length 35
```

To reset the number of lines to the default of 25 lines, enter:

```
# no terminal length
```

## Configuring the More Terminal Prompt

When you enter the question mark (?) character at the command line to get help about a command, the CSS displays 24 lines on the terminal. The --More-- prompt indicates that additional CLI commands are to follow. By default, the CSS disables the display of the --More-- prompt.

To display the --More-- prompt at the bottom of the terminal screen, use the **terminal more** command. Press the **Space** bar to continue viewing the next series of commands (or press the **Return** key to display only the next line). This command is available in User and SuperUser modes. The default is enabled.

You can also toggle the terminal more function on and off within a session by using the Esc-M key sequence.

To enable support for the --More-- terminal prompt, enter:

```
# terminal more
```

To disable support for the --More-- terminal prompt, enter:

```
# no terminal more
```

## Configuring Terminal Netmask-Format

By default, the CSS displays dotted-decimal subnet masks in show screens. Use the **terminal netmask-format** command to determine how the CSS displays subnet masks. This command is available in User and SuperUser modes. The options for this command are as follows:

- **terminal netmask-format bitcount** - Displays masks in classless interdomain routing (CIDR) bitcount (for example, /24).
- **terminal netmask-format decimal** - Displays masks in dotted-decimal format (for example, 255.255.255.0). This is the default format.
- **terminal netmask-format hexadecimal** - Displays masks in hexadecimal format (for example, OXFFFFFFOO).

For example, to display subnet masks in bit-count format, enter:

```
# terminal netmask-format bitcount
```

To restore the default display format (**decimal**), enter:

```
# no terminal netmask format
```

## Configuring Terminal Timeout

By default, the CSS does not have a time limit for a console or Telnet session. Use the **terminal timeout** command to set the total amount of time a session can be logged in before the CSS terminates a console or Telnet session. This command is available in User and SuperUser modes. Enter a timeout value between 0 and 65535 minutes. The default value is 0 (disabled).

For example, to set the terminal timeout value to 30 minutes, enter:

```
# terminal timeout 30
```

To restore the terminal timeout value to the default state of disabled, enter:

```
# no terminal timeout
```

## Configuring Idle Timeout

By default, the CSS does not have a idle timeout period for all active Telnet, console, FTP, SSH, and web management sessions. Use the **idle timeout** command to globally set the total amount of time all console, Telnet, SSH or FTP sessions can be active before the CSS terminates them.

Enter a timeout value between 0 and 65535 minutes. By default, the idle timeout is disabled (set to 0).

**Note** To override the idle timeout value for a specific Telnet, console, SSH, or FTP session, configure the **terminal timeout command**. Terminal commands are user-specific; that is, they apply uniquely to each CSS user.

To set an idle timeout value for Telnet, console, SSH, or FTP sessions, enter:

```
(config)# idle timeout 15
```

We recommend that you configure the Telnet idle timeout for at least 30 minutes. Setting this value to 30 minutes:

- Cleans up idle Telnet sessions
- Helps prevent busy conditions due to a high number of active Telnet session

To disable the terminal timeout for Telnet, console, SSH, or FTP sessions, enter:

```
(config)# no idle timeout
```

To set the idle timeout for all active web management sessions, use the **idle timeout web-mgmt** command.

**Note** The **web-mgmt** option does not apply to the CVDM available in the CSS software release 8.10 or greater.

To set an idle timeout value for all active web management sessions, enter:

```
(config)# idle timeout web-mgmt 15
```

To disable the web management timeout period, enter:

```
(config)# no idle timeout web-mgmt
```

## Using Expert Mode

Expert mode allows you to turn the CSS confirmation capability on or off. Expert mode is available in SuperUser mode and is off by default. When expert mode is off, the CSS prompts you for confirmation when you:

- Execute commands that could delete or change operating parameters
- Exit a terminal session (console or Telnet) without copying the running configuration to the startup configuration
- Create services, owners, and content rules

**Caution** Turning expert mode on *disables* the CSS from prompting you for confirmation when you make changes. When you exit from the CSS, all configuration changes are automatically saved to the profile and to the running-config file. You are not prompted for confirmation to save the changes.

To enable expert mode, enter:

```
# expert
```

To allow the CSS to prompt you for confirmation before executing configuration commands, enter:

```
# no expert
```

For example, when you enter the command to create an owner and expert mode is off, the CSS prompts you to verify the command, enter:

```
(config)# owner arrowpoint.com
Create owner <arrowpoint.com>, [y/n]:y
(config-owner[arrowpoint.com])#
```

## Changing the CLI Prompt

The CLI default prompt appears as the CSS product model number followed by the number (#) symbol. The CSS adds a **#** to the prompt automatically to indicate SuperUser mode. To change the default prompt, enter the **prompt** command. For example:

```
CSS11506# prompt CSS1-lab
CSS1-lab#
```

You can enter a maximum of 15 characters.

To save the new prompt, add it to your user- or default-profile file. To restore the prompt to the default, use the **no** form of the **prompt** command.

For example:

```
CSS11506# no prompt
```

## Modifying the History Buffer

The history buffer stores 20 lines of the most recent CLI commands that you enter. Use the **history** command to modify the CLI history buffer length. Enter the number of lines you want in the history buffer as an integer from 0 to 256. The default is 20. This command is available only in SuperUser mode.

To set the history buffer to 80 lines, enter:

```
# history length 80
```

To disable the history function (setting of 0), enter:

```
# history length 0
```

To restore the history buffer to the default of 20 lines, enter:

```
# no history length
```

## Displaying the History Buffer

Use the **show history** command to display the contents of the history buffer. The history buffer is cleared automatically upon reboot.

For example:

```
# show history

    history
    show history
    show ip routes
    show ip summary
    show ip stat
    clock
    clock date
    clock time
    show history
```

## Configuring a Pre-Login Banner

You can configure a custom banner that displays before you log in when you connect to a CSS. The banner is an ASCII text file that you provide and it must reside in the CSS script directory. This banner is a general banner that is the same for all users. For example, you could create a banner that includes the name of your company or a department within your company.

To configure a pre-login banner, use the **prelogin-banner** command in global configuration mode. This command has the following syntax:

**prelogin-banner** "*filename*"

The *filename* variable is the name of the ASCII text file that contains the pre-login banner text. Enter a quoted text string with a maximum of 32 characters.

For example, to configure a pre-login banner file called newBanner:

1. Use any text editor (for example, Notepad or Wordpad) to create a custom banner called newBanner and save it as a text file. The maximum line width is 80 characters.
2. FTP the text file to the CSS script directory as follows:
   a. From the directory that contains the banner text file, FTP to the CSS. For example, enter: **ftp 192.168.12.5**.
   b. At the FTP prompt, log in to the CSS.
   c. Enter **cd script** to change to the CSS script directory.
   d. Enter **put newBanner newBanner**.

      FTP transfers the banner file to the CSS script directory.
3. To complete the configuration, enter the following command at the CSS CLI:

   ```
   (config)# prelogin-banner newBanner
   ```

   The next time you connect to the CSS, the custom banner appears.

To reset the default behavior of the CSS to no pre-login banner, enter:

```
(config)# no prelogin-banner
```

## Configuring a Login Banner

The CSS banner is an ASCII text file that you provide and it must reside in the CSS /script directory. Because this feature is part of your user profile, you can have your own custom banner associated with your login name.

Use the **set** command with the BANNER environment variable to create a custom banner that appears after you log in to a CSS. To configure a login banner:

1. Use any text editor (for example, Notepad or Wordpad) to create a custom banner and save it as a text file in the CSS script directory. The maximum line width is 80 characters.
2. FTP the banner file to the CSS script directory. See the "Configuring a Pre-Login Banner" section.

3. At the CSS CLI, set the environment variable BANNER and point to the mybanner file:

```
# set BANNER "mybanner" session
```

   The keyword **session** causes the CSS to create the environment variable every time you log in.

4. To complete the configuration, enter the following command:

```
# copy profile user-profile
```

   The next time you log in to the CSS, your custom banner appears.

To provide the same banner for all users, replace the command in Step 4 with:

```
# copy profile default-profile
```

## Copying and Saving User Profiles

To copy the running profile from the CSS to the default-profile file, an FTP server, a TFTP server, or your user-profile file, use the **copy profile** command. This command is available only in SuperUser mode.

If you exit the CSS without copying changes in the running profile to your *username*-profile or default-profile file, the CSS prompts you that the profile has changed and queries whether you want to save your changes. If you respond with **y**, the CSS copies the running profile to your *username*-profile or the default-profile.

This section includes the following topics:

- Copying the Running Profile to the Default-Profile
- Copying the Running Profile to a User Profile
- Copying the Running Profile to an FTP Server
- Copying the Running Profile to a TFTP Server

## Copying the Running Profile to the Default-Profile

To copy the running profile to the default profile, use the **copy profile default-profile** command. This command is available only in SuperUser mode.

For example:

```
# copy profile default-profile
```

## Copying the Running Profile to a User Profile

To copy the changes made to the running profile to the user profile, use the **copy profile user-profile** command. This command is available only in SuperUser mode. This command creates a file *username*-profile if one does not exist (where *username* is the current username).

For example:

```
# copy profile user-profile
```

## Copying the Running Profile to an FTP Server

To copy the running profile to an FTP server, use the **copy profile ftp** command. This command is available only in SuperUser mode. The syntax is:

**copy profile ftp** *ftp_record filename*

The variables for this command are as follows:

- *ftp_record* - The name of the FTP record file that contains the server IP address, username, and password. Enter an unquoted text string with no spaces and a maximum of 32 characters.
- *filename* - 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.

For example:

```
# copy profile ftp arrowrecord \records\arrowftprecord
```

## Copying the Running Profile to a TFTP Server

To copy the running profile to a TFTP server, use the **copy profile tftp** command. This command is available only in SuperUser mode. The syntax is:

**copy profile tftp** *ip_or_host filename*

The variables for this command are as follows:

- *ip_or_host* - The IP address or host name of the server to receive the file. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1) or in mnemonic host-name format (for example, myhost.mydomain.com).
- *filename* - 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 of 32 characters.

For example:

```
# copy profile tftp 192.168.3.6 \home\bobo\bobo-profile
```

## Where to Go Next

Chapter 4, Using the CSS Logging Features, describes how to enable logging, set up the log buffer, and determine where to send the activity information. This chapter also provides information on interpreting sys.log messages and a description of frequently queried messages.

# Using the CSS Logging Features

This chapter describes how to enable logging, set up the log buffer, determine where to send the activity information, and display and interpret log messages. Information in this chapter applies to all CSS models, except where noted.

This chapter contains the following major sections:

## Logging Overview

The CSS generates log messages to assist you with debugging and monitoring operations. By default, the CSS saves boot and subsystem event log messages to log files on its hard or Flash disk. The content of these files is recorded in ASCII text. You can also configure the CSS to send log messages to an active CSS session, e-mail address, or another host system.

The maximum size of a log file is 50 MB for hard disk-based systems, and 10 MB for Flash disk-based systems.

The boot log messages are the result of the boot process. The CSS saves these messages in the boot.log file.

The subsystem log messages are subsystem events that occur during the operation of the CSS. The CSS saves these messages in the sys.log file, created when the first loggable subsystem event occurs. The CSS determines which subsystem messages to log by its configured logging level. By default, the CSS logs events on all subsystems with a level of warning. The warning level designates that the CSS logs fatal, alert, critical, error, and warning messages for the subsystem.

You have the option to log subsystem messages at a different level than the default warning level. The level you specify instructs the CSS to log subsystem activity that occurs at that level and the activity greater than that level.

In addition to informational messages, the CSS also logs notice, warning, error, critical, alert, and fatal messages.

You can display or copy a log file using the **show log** or **copy log** command, respectively. For details on the **show log** command, see the "Showing Log Files" section. For details on the **copy log** command, see the "Copying Log Files to an FTP or TFTP Server" section. You need SuperUser privileges to use the **show log** command.

The CSS provides logging capabilities for debugging and system monitoring by generating the log files described in Table 4-1.

***Table 4-1 CSS Log File Descriptions***

| | **Log File Destination** | | |
|---|---|---|---|
| **Log File** | **Default Location** | **Alternate Location** | **Records** |
| boot.log | Hard disk and console or Flash disk and console | None | Results of the boot process. |
| boot.bak | Hard disk and console or Flash disk and console | None | Backup of a boot log file. Each time you reboot the CSS, the software renames the current boot log file to boot.log.prev and starts a new boot log file. The CSS overwrites an existing backup boot log file when a boot log file is renamed. |
| sys.log | Hard disk or Flash disk | Console<br>syslogd<br>VTY1<br>VTY2 | Log information for user-defined subsystem or CLI commands. By default, logging is enabled and logs subsystem **all** with level **warning**. The CSS creates sys.log to record this log information. |

Table 4-1 CSS Log File Descriptions (continued)

| Log File | Log File Destination | | Records |
|---|---|---|---|
| | Default Location | Alternate Location | |
| sys.log.prev | Hard disk or Flash disk | Console<br>syslogd<br>VTY1<br>VTY2 | Backup of a system log file. When a system log file reaches its maximum size (50 MB, for a hard disk-based CSS; 10 MB, for a Flash disk-based CSS), the software renames the system log file to sys.log.prev and starts a new system log file. The CSS overwrites an existing backup system log file when a system log file is renamed. When you reboot a CSS, the software continues to use the existing system log file until the file reaches its maximum size. |

## CSS Logging Quick Start Table

If you are familiar with the CSS logging functions, see Table 4-2 for the commands and command options required to configure and enable logging.

You configure all logging commands from configuration mode except for the clear log command. The **clear log** command is available only in SuperUser mode at the root prompt (#).

*Table 4-2 Configuring and Enabling Logging*

| Step | Logging Option | Example |
|---|---|---|
| **1.** Specify the disk buffer size. | *size* - Size of the disk buffer (0 to 64000) | **logging buffer 1000** |
| **2.** Select a CSS subsystem and determine which type of activity to log (default **all**) and level (default **warning**). | **subsystem** - Valid subsystems:<br>**acl**, **all**, **app**, **boomerang**, **buffer**, **cdp**, **chassis**, **circuit**, **csdpeer**, **dhcp**, **dql**, **fac**, **flowagent**, **flowmgr**, **fp-driver**, **hfg**, **ipv4**, **keepalive**, **natmgr**, **netman**, **netmgr**, **nql**, **ospf**, **pcm**, **portmapper**, **proximity**, **publish**, **radius**, **redundancy**, **reporter**, **replicate**, **rip**, **security**, **ssl-accel**, **slr**, **sntp**, **sshd**, **syssoft**, **urql**, **vlanmgr**, **vpm**, **vrrp**, **wcc**<br>**level** - Valid levels:<br>**fatal-0, alert-1**, **critical-2**, **error-3**, **warning-4**, **notice-5**, **info-6, debug-7** | **logging subsystem rip level alert-1** |
| **3.** Specify the destination (disk, host, line) where you wish to log subsystem activity. | **disk** *filename* - New or existing filename in the log directory | **logging disk stubs** |
| | **host** *ip* or *host* - IP address of the syslog daemon on the host or a host name | **logging host 192.168.11.3**<br>**logging host myhost.domain.com** |
| | **log** *line* - CSS active session | **logging line vty1** |

*Table 4-2 Configuring and Enabling Logging (continued)*

| Step | Logging Option | Example |
|---|---|---|
| **4.** Optionally, enable the CSS to send log messages to an e-mail address and specify a level. | **sendmail** *email address* of mail recipient<br>*IP address* or *hostname* of SMTP host<br>**level** - Valid levels for the CSS:<br>**fatal-0**, **alert-1**, **critical-2**, **error-3**, **warning-4**, **notice-5**, **info-6**, **debug-7** | **logging sendmail us@arrowpoint.com 172.16.6.58 critical-2** |
| **5.** Show the log file. | *filename* - Log file to display | **show log stubs** |

The following running-configuration example shows the results of entering the commands in Table 4-2.

```
!************************** GLOBAL **************************

  logging buffer 1000
  logging subsystem rip level alert-1
  logging disk stubs
  logging sendmail us@cisco.com 172.16.6.58 critical-2
```

## Specifying Logging Buffer Size

The logging buffer size is the amount of information the CSS buffers in memory before outputting the information to disk. The larger you configure the buffer size, the less frequently the CSS outputs the contents to disk. Specifying a buffer size is required only if you specify logging to disk as the log file destination.

To set the disk buffering size, use the **logging buffer** command. Specify the buffer size from 0 to 64000 bytes. The default is 0, where the CSS sends the logging output directly to the log file.

To set the buffer size to 1000 bytes, enter:

```
(config)# logging buffer 1000
```

To send the logging output directly to the log file, enter:

```
(config)# no logging buffer
```

## Configuring Logging for a Subsystem

This section describes how to select a CSS subsystem and log activity for the subsystem. This section includes the following topics:

- Enabling and Disabling Logging for a Subsystem
- Configuring a Log Message for a Subsystem at a Logging Level
- Logging ACL Activity
- Sending Log Messages to an E-Mail Address

### Enabling and Disabling Logging for a Subsystem

By default, the logging levels for all CSS subsystems is set at warning-4. The level you specify instructs the CSS to log subsystem activity that occurs at that level and the activity greater than that level. For example, if you wish to log informational messages (info-6), the CSS also logs notice, warning, error, critical, alert, and fatal error levels.

Use the **logging subsystem** command to select a CSS subsystem and determine which type of activity to log.

To reset logging for a subsystem to the default logging level (warning-4), enter the **no** version of the logging command. For example:

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

The following example enables logging for the chassis subsystem with a critical-2 error level. The CSS logs all critical, alert, and fatal errors for the chassis.

```
(config)# logging subsystem chassis level critical-2
```

Table 4-3 defines the CSS subsystems for which you can enable logging.

*Table 4-3 Logging Subsystems*

| Subsystem | Definition |
|---|---|
| **acl** | Access control list (ACL) |
| **all (default)** | All CSS subsystems |
| **app** | Application Peering Protocol (APP) |
| **boomerang** | DNS Content Routing Agent (CRA) |
| **buffer** | Buffer manager |
| **cdp** | Cisco Discovery Protocol (CDP) |
| **chassis** | Chassis manager |
| **circuit** | Circuit manager |
| **csdpeer** | Content Server Database (CSD) peer |
| **dhcp** | Dynamic Host Configuration Protocol (DHCP) |
| **dql** | Domain Qualifier List (DQL) |
| **fac** | Flow Admission Control (FAC) |
| **flowagent** | Flow agent |
| **flowmgr** | Flow manager subsystem |
| **fp-driver** | Fathpath driver |
| **hfg** | Header Field Group (HFG) |
| **ipv4** | Internet Protocol version 4 (IPv4) |
| **keepalive** | Keepalive |
| **natmgr** | NAT manager |
| **netman** | Network management |
| **nql** | Network Qualifier List (NQL) |
| **ospf** | Open Shortest Path First (OSPF) protocol |
| **pcm** | Proximity CAPP Messaging (PCM) |
| **portmapper** | Port mapper |
| **proximity** | Proximity |

*Table 4-3 Logging Subsystems (continued)*

| Subsystem | Definition |
|---|---|
| **publish** | Publish |
| **radius** | Remote Authentication Dial-In User Service (RADIUS) |
| **redundancy** | CSS redundancy |
| **reporter** | Reporter |
| **replicate** | Content replication |
| **rip** | Routing Information Protocol (RIP) |
| **security** | Security manager |
| **slr** | Session Level Redundancy |
| **sntp** | Simple Network Time Protocol (SNTP) |
| **sshd** | SSHD |
| **ssl-accel** | Secure Socket Layer (SSL) Acceleration |
| **syssoft** | System software |
| **urql** | Uniform Resource Locator Qualifier List (URQL) |
| **vlanmgr** | VLAN manager |
| **vpm** | Virtual pipe manager |
| **vrrp** | Virtual Router Redundancy Protocol |
| **wcc** | Web conversation control |

Table 4-4 defines the logging levels you can set for the specified CSS subsystem. The logging levels are listed in order of severity, with a fatal-0 level being the most severe errors and an info-6 level being the least severe error.

*Table 4-4 Subsystem Logging Levels*

| Level | Definition |
|---|---|
| **fatal-0** | Fatal errors only. |
| **alert-1** | Alert errors, including fatal errors. |

*Table 4-4 Subsystem Logging Levels (continued)*

| Level | Definition |
|---|---|
| **critical-2** | Critical errors, including alert and fatal errors. The following trap events log at the critical level: link down, cold start, warm start, service down, service suspended. |
| **error-3** | General errors, including critical, alert, and fatal errors. |
| **warning-4** (default) | Warning messages, including all lower levels (error, critical, alert, and fatal. |
| **notice-5** | Notice messages, including all trap events (except for events logged at critical) and all lower levels except for info and debug. |
| **info-6** | Informational messages, including all lower levels except for debug. |
| **debug-7** | Debug messages, including all other error levels. The debug-7 log level may degrade the performance of the CSS. When you enter this option, the CSS prompts you with the following message: `Logging at the debug level may degrade the CSS performance. Continue, [y/n]:` Enter **y** to verify that you want to set the log level to debug-7. Enter **n** to cancel the executing of the debug-7 log level. |

## Configuring a Log Message for a Subsystem at a Logging Level

To define a log message for a subsystem at a particular logging level, use the **cliLogMessage subsystem** command. The syntax for this global configuration mode command is:

**cliLogMessage subsystem** *name* "*message*" **level** *level*

The variables and options are as follows:

- *name* - The name of a CSS subsystem. Enter one of the subsystem names, as shown in Table 4-3. To see a list of subsystems, enter:

  ```
  cliLogMessage subsystem ?
  ```

- **level** *level* - The log level for the message. Enter one of the levels, from 0 to 7, as shown in Table 4-4. To see a list of levels, enter:

  ```
  cliLogMessage subsystem name "message" level ?
  ```

## Logging ACL Activity

When you configure the CSS to log ACL activity, the CSS logs the event of the packet matching the clause and ACL. The CSS sends log information to the location you specified in the **logging** command.

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

To configure logging for an ACL clause:

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

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

2. Enable logging for:

   - A new clause, by entering the **log** option at the end of the clause. For example:

     ```
     (config-acl[7])# clause 1 deny udp any eq 3 destination any eq
     3 log
     ```

   - An existing clause, by using the **clause log enable** command:

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

To disable ACL logging for a specific clause, enter:

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

To globally disable logging for all ACL clauses, enter:

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

## Sending Log Messages to an E-Mail Address

To send the log activity of a subsystem to an e-mail address, use the **logging sendmail** command. The syntax for this global configuration mode command is:

**logging sendmail** *email_address ip_address level* {*domain*}

The variables are as follows:

- *email_address* - The e-mail address for the recipient. Enter the e-mail address as an unquoted text string with a length of 1 to 30 characters.
- *IP_address* - The IP address for the SMTP host. Enter the IP address in dotted-decimal notation (for example, 192.168.11.1).
- *level* - The type of information to log. The valid levels are defined in Table 4-4.
- *domain* - (Optional) The domain name for the SMTP host. Enter an unquoted text string with a maximum length of 64 characters (for example, arrowpoint.com). Do not insert an @ sign before the domain name. The CSS automatically prepends the @ sign to the domain name.

To turn off logging to an e-mail address, enter:

```
(config)# no logging sendmail email_address
```

## Specifying a Log File Destination

To specify a destination where the CSS logs subsystem activity, use the **logging** command. You can specify the following locations for log files:

- **disk** *filename* - New or existing filename in the disk log directory
- **host** *ip* or *host* - IP address of the syslog daemon on the host or a host name
- **log** *line* - CSS active session

Logging to a CSS disk causes the performance of the CSS to degrade.If logging requires frequent writes to disk (that is, several hundred log messages per day), the most reliable configuration is to log to a hard disk and store all other system files on a Flash disk. Although Flash disks generally provide the most reliable way to store information over time, hard disks endure frequent writes to disk better than the Flash disks currently available.

To prevent excessive writes to the CSS disk, consider disabling logging to the sys.log file on disk (see the "Disabling Logging to the sys.log and boot.log Files on the Disk" section). You can continue sending CSS log information to the sys.log file at an alternate location. To do this, use either the **logging host** command to send log information to a syslog daemon on a host system (see the "Specifying a Host for a Log File Destination" section) or the **logging line** command to send (but not save) log information to an active CSS line (see the "Specifying a Line for a Log File Destination" section).

This section includes the following topics:

## Specifying a Log File on the Disk

To send log information to a specific file on the CSS disk, use the **logging disk** command. Specify a log filename. Enter a text string from 0 to 32 characters. The filename can be new or an existing name.

For example:

```
(config)# logging disk stubs
```

When you specify the **logging disk** command, the CSS:

- Stops writing default log information to the sys.log file
- Creates the filename you specify in the disk log directory
- Sends subsystem and level information to the log file specified

You can have only one active log file on the disk at a time. If you wish to send subsystem information to a different log file on the disk, reenter the logging disk command with a different filename.

**Caution** Logging to a CSS disk causes the performance of the CSS to degrade.

To stop logging to the specified file and reenable logging to the sys.log file on the CSS, enter:

```
(config)# no logging disk
```

## Disabling Logging to the sys.log and boot.log Files on the Disk

Disabling logging to the sys.log and boot.log files is useful when you want to prevent excessive writes to the CSS disk (for example, to the Flash disk) or to increase the performance of the CSS. Use the **logging to-disk** command to disable logging to the sys.log and boot.log files on the CSS disk (hard or Flash).

The options for the **logging to-disk** command are as follows:

- **logging to-disk disable** - Disables writing default log information to the CSS sys.log and boot.log files on the CSS disk. The **logging to-disk disable** command affects the sys.log and boot.log files only, and does not affect a disk log file specified through the **logging disk** command. You can still use the **logging disk** *filename* command to send log information to a specific filename on the CSS disk. To disable all logging to the CSS disk, first enter the **no logging disk** command and then enter the **logging to-disk disable** command. When you enter the **no logging disk** command, the CSS does not reenable logging to the sys.log and boot.log files. You must specify the **logging to-disk enable** command to reactivate the sys.log and boot.log files.
- **logging to-disk enable** - Resets logging back to disk and resumes writing default log information to the CSS sys.log and boot.log files.

You are prompted to reboot the CSS after issuing the **logging to-disk disable** or **logging to-disk enable** commands for the command to take effect.

**Note** You can continue sending CSS log information to the sys.log and boot.log files at an alternate location. To send log information to an alternate location, use either the **logging host** command to send log information to a syslog daemon on a host system (see the "Specifying a Host for a Log File Destination" section) or the **logging line** command to send (but not save) log information to an active CSS line (see the "Specifying a Line for a Log File Destination" section).

To disable logging to the CSS sys.log and boot.log files on the CSS disk (Flash disk or hard disk), enter:

```
(config)# logging to-disk disable
```

To resume logging back to the CSS disk, enter:

```
(config)# logging to-disk enable
```

## Specifying a Host for a Log File Destination

To send CSS log information to a syslog daemon running on the host system, use the **logging host** command. The syslog daemon receives and displays the CSS log messages on the host system.

The syntax for this configuration mode command is:

**logging host** *ip_or_host* **facility** *number* **log-level** *number*

The options and variables for this command are as follows:

- *ip_or_host* - Specifies the address of a syslog daemon on the host. Enter the IP address in dotted-decimal notation (for example, 192.168.11.1) or the mnemonic host name (for example, myhost.mydomain.com).
- **facility** *number* - Specifies the syslog daemon facility level. Facilities are considered service areas, such as printing, e-mail, or network. Enter a number from 0 to 7. For more information on the syslog daemon and facility levels, refer to your syslog daemon documentation.
- **log-level** *number* - Specifies the level of the CSS subsystem log messages to be sent to the syslog daemon on the host. The valid log levels for the CSS include: fatal-0, alert-1, critical-2, error-3, warning-4 (default), notice-5, info-6, debug-7. The logging levels are listed in order of severity, with a fatal-0 level being the most severe error and an info-6 level being the least severe error. Refer to Table 4-4 for a definition of the different logging levels.

  The **logging host log-level** *number* must be equal to or less than the log level you configure for the **logging subsystem** command (see the "Configuring Logging for a Subsystem" section). If the log-level value is less than the logging subsystem level, the CSS only sends the message level specified in the **log-level** option. If the log-level is greater than the logging subsystem level, the CSS only sends the level of messages specified in the **logging subsystem** command.

The CSS continues to send log information to the sys.log file on the CSS disk (hard or Flash disk) when the **logging host** command is entered. To disable logging to the sys.log and boot.log files on the CSS disk, use the **logging to-disk disable** command (see the "Disabling Logging to the sys.log and boot.log Files on the Disk" section).

For example, to send log information to a host at IP address 192.168.11.1 with a facility level of 3 and a log-level of error-3:

```
(config)# logging host 192.168.11.1 facility 3 log-level error-3
```

To turn off logging to a host, enter:

```
(config)# no logging host
```

## Specifying a Line for a Log File Destination

To send log information to an active CSS session, use the **logging line** command and specify a valid log line on the CSS. Enter the line as a case-sensitive text string with a maximum of 32 characters.

The CSS continues to send log information to the sys.log file on the CSS disk (hard or Flash disk) even when the **logging line** command is entered. To disable logging to the sys.log file on the CSS disk, use the **logging to-disk disable** command (see the "Disabling Logging to the sys.log and boot.log Files on the Disk" section).

To display a list of active CSS lines, enter the **logging line** command as shown. The asterisk (*) denotes your current session.

```
(config)# logging line ?

console      Login Name:  Location:local
*vty1        Login Name:  admin Location:10.0.3.35
```

To send subsystem information to your monitor, enter:

```
(config)# logging line vty1
```

To turn off logging, enter the **no logging line** command.

```
(config)# no logging line vty1
```

## Logging CLI Commands

When you want to keep track of all CLI commands entered from the CSS, you can log them to the sys.log file. To log the CLI commands:

1. Set the logging level of the netman subsystem to info-6. Enter:

```
(config)# logging subsystem netman info-6
```

2. Enable logging of commands through the **logging commands enable** command. This command logs each CLI command to the sys.log file. Enter:

```
(config)# logging commands enable
```

To disable logging CLI commands to the sys.log file, enter:

```
(config)# no logging commands
```

## Showing Log Files

Use the **show log** command to display the contents in a log or trap log file, a list of all log files, or the state of logging for CSS facilities. You can use the **show log** command in all modes, including User mode.

When you use the **show log** command to send the log activity to your current session, and you want to stop sending log activity, press any key on the terminal or workstation. The **show log** command performs the same function as the **logging line** command. You cannot run these commands at the same time.

This section includes the following topics:

- Showing Log Activity
- Showing Log Lists
- Showing the Log State

### Showing Log Activity

Use the **show log** command and its options to send the log activity to your current session or to display the contents in a log or trap log file. You can use the **show log** command in all modes, including User mode.

The syntax for the **show log** command is:

**show log** {*log_filename*|**traplog** {**tail** *lines*} {**line-numbers**}}

The options and variables for this command are as follows:

- *log_filename* - Specifies the name of the log file. Enter an unquoted text string with no spaces. To see a list of log files with their creation dates, enter: **show log ?**
- **traplog** - (Optional) Displays all SNMP traps that have occurred. A trap log file is an ASCII file in the log directory containing generic and enterprise traps. By default, the following events generate level critical-2 messages:
  - Link Up
  - Link Down
  - Cold Start
  - Warm Start
  - Service Down
  - Service Suspended

  All other SNMP traps generate level notice-5 messages.

  When a traplog file reaches its maximum size (50 MB for a hard disk-based CSS, 10 MB for a flash disk-based CSS), the CSS renames the traplog file to traplog.prev as a backup file and starts a new traplog file. The CSS overwrites the backup traplog file when it renames the traplog file. Each time the CSS reboots, it continues to use the existing traplog file until it reaches its maximum size.

**Note** When traps are disabled, the CSS still produces a log message for any event that would normally generate a trap.

- **tail** *lines* - (Optional) Displays the bottom and most recent portion of the log file. The CSS displays the log file, starting from the beginning of the file. The top of the file lists the older messages and the bottom lists the most recent messages. You can specify the number of lines to display (to a maximum of 1000 lines), starting at the end of the log file. Enter a number from 1 to 1000.
- **line-numbers** - (Optional) Includes the line numbers when displaying the contents of the log file.

To send the log activity to your current session, enter:

```
# show log
Displaying Log events.
Press any key to abort...
APR 14 16:28:09 5/1 2398 NETMAN-7: HTTPC:HTTPC_Open:
ERROR->connect <-1,0> <192.20.1.7> <80>
APR 14 16:28:15 5/1 2399 NETMAN-7: HTTPC:HTTPC_Open:
ERROR->connect <-1,0> <192.20.1.7> <80>
APR 14 16:28:21 5/1 2400 NETMAN-7: HTTPC:HTTPC_Open:
ERROR->connect <-1,0> <192.20.1.7> <80>
APR 14 16:28:27 5/1 2401 NETMAN-7: HTTPC:HTTPC_Open:
ERROR->connect <-1,0> <192.20.1.7> <80>
```

To display information in a specific log file, enter the **show log** command with a valid log filename. For example:

```
# show log stubs
SEP 22 09:59:18 5/1 918 NETMAN-7: SNMP:SET RSP (3803)
SEP 22 09:59:53 5/1 919 NETMAN-7: SNMP:SET  (3804)
SEP 22 09:59:53 5/1 920 NETMAN-7: SNMP:  1
apLogHostIpAddress.[1.2.3.4] VT_IPADDRESS  <1.2.3.4>
SEP 22 09:59:53 5/1 921 NETMAN-7: SNMP:  2
apLogHostIpAddress.[1.2.3.4] VT_IPADDRESS  <1.2.3.4>
```

To view the content of the sys.log file, enter:

```
(config)# show log sys.log
```

To view the bottom and most recent portion of the file, use the **tail** option with the **show log** command. For example, to view the most recent 500 lines in the sys.log file, enter:

```
(config)# show log sys.log tail 500
```

## Showing Log Lists

Use the **show log-list** command to display a list of all log files. You can use the **show log-list** command in all modes, including User mode. For example:

```
(config)# show log-list
```

## Showing the Log State

Use the **show log-state** command to display the state of logging for CSS facilities. You can use the **show log-state** command in all modes, including User mode. For example:

```
(config)# show log-state
```

Table 4-5 describes the fields in the **show log-state** command output.

***Table 4-5 Field Descriptions for the show log-state Command***

| Field | Description |
|---|---|
| **Subsystems:** | |
| acl | Access Control List (ACL) |
| app | Application Peering Protocol (APP) |
| boomerang | DNS Content Routing Agent (CRA) |
| buffer | Buffer manager |
| cdp | Cisco Discovery Protocol (CDP) |
| chassis | Chassis manager |
| circuit | Circuit manager |
| csdpeer | Content Server Database (CSD) peer |
| dhcp | Dynamic Host Configuration Protocol (DHCP) |
| dql | Domain Qualifier List (DQL) |
| fac | Flow Admission Control (FAC) |
| flowagent | Flow agent |
| flowmgr | Flow manager subsystem |
| fp-driver | Fathpath driver |
| hfg | Header Field Group (HFG) |
| ipv4 | Internet Protocol version 4 (IPv4) |
| keepalive | Keepalive |
| natmgr | NAT manager |
| netman | Network management |

*Table 4-5 Field Descriptions for the show log-state Command (continued)*

| Field | Description |
|---|---|
| nql | Network Qualifier List (NQL) |
| ospf | Open Shortest Path First (OSPF) |
| pcm | Proximity CAPP Messaging (PCM) |
| portmapper | Port mapper |
| proximity | Proximity |
| publish | Publish |
| radius | Remote Authentication Dial-In User Service (RADIUS) |
| redundancy | CSS redundancy |
| reporter | Reporter |
| replicate | Content replication |
| rip | Router Information Protocol (RIP) |
| security | Security manager |
| slr | Session Level Redundancy |
| sntp | Simple Network Time Protocol (SNTP) |
| sshd | SSHD |
| ssl-accel | Secure Sockets Layer (SSL) Acceleration |
| syssoft | System software |
| urql | Uniform Resource Locator Qualifier List (URQL) |
| vlanmgr | VLAN manager |
| vpm | Virtual pipe manager |
| vrrp | Virtual Router Redundancy Protocol |
| wcc | Web conversation control |
| **Levels:** | |
| debug | Log all errors and messages (Verbose) |
| info | Log informational messages, including errors at the notice level |

*Table 4-5 Field Descriptions for the show log-state Command (continued)*

| Field | Description |
|---|---|
| notice | Log notice messages, including errors at the warning level |
| warning | Log warning errors (default), including errors at the error level |
| error | Log errors, including errors at the critical level |
| critical | Log critical errors, including errors at the alert level |
| alert | Log alert errors, including errors at the fatal level |
| fatal | Log fatal errors only (Quiet) |
| **File:** | |
| Filename: | Name of the log file |
| Current size: | Current size of the log file |
| Log to Disk | Identifies whether logging to disk (Flash disk or hard disk) is Enabled or Disabled. |

## Copying Log Files to an FTP or TFTP Server

To copy log files from the CSS to a File Transfer Protocol (FTP) or Trivial File Transfer Protocol (TFTP) server, use the **copy log** command. The **copy log** command is available only in SuperUser mode.

This section includes the following topics:

- Copying Log Files to an FTP Server
- Copying Log Files to a TFTP Server

## Copying Log Files to an FTP Server

To copy a log file to an FTP server, use the **copy log ftp** command. Before you copy a log file from the CSS to an FTP server, create an FTP record file containing the FTP server IP address, username, and password. Refer to Chapter 1, Managing the CSS Software, for information on configuring an FTP record.

The syntax is:

**copy log** *logfilename* **ftp** *ftp_record filename*

The options and variables for this command are as follows:

- *logfilename* - Specifies the name of the log file on the CSS. Enter an unquoted text string with no spaces and a maximum of 32 characters. To see a list of log files, enter the **copy log ?** command.
- **ftp** - Copies a log file to an FTP server.
- *ftp_record* - Specifies the name of the FTP record file that contains the FTP server IP address, username, and password. Enter an unquoted text string with no spaces and a maximum of 16 characters. To create an FTP record, see Chapter 1, Managing the CSS Software.
- *filename* - Specifies the name you want to assign to the file on the FTP server. Include the full path to the file. Enter an unquoted text string with no spaces and a maximum of 32 characters.

For example, to copy the *starlog* log file to an FTP server:

```
# copy log starlog ftp ftpserv1 starlogthurs
```

## Copying Log Files to a TFTP Server

To copy a log file to a TFTP server, use the **copy log tftp** command. The syntax is:

**copy log** *log_filename* [**ftp** *ftp_record*|**tftp** *ip_or_host*] *filename*

The options and variables for this command are as follows:

- *log_filename* - The name of the log file on the CSS. Enter an unquoted text string with no spaces and a maximum of 32 characters. To see a list of log files, enter the **copy log ?** command.
- **tftp** - Copies a log file to a TFTP server.

- *ip_or_host* - The IP address or host name of the TFTP server to receive the file. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1) or in mnemonic host-name format (for example, myhost.mydomain.com). If you wish to use a host name, you must first set up a host table using the **host** command.
- *filename* - The name you want to assign to the file on the TFTP server. Include the full path to the file. Enter an unquoted text string with no spaces and a maximum of 32 characters.

For example, to copy the *starlog* log file to a TFTP server:

```
# copy log starlog tftp tftpserv1 starlogthurs
```

## Interpreting sys.log Log Messages

The following example shows a sys.log message. This section describes the parts of a log message using this example.

```
FEB 16 14:01:13 5/1 2453 VLANMGR-7: Transmit sfm STP BPDU on bPort 1,
egressLp 0x1f00 VlanLpSend() ret:0
```

A log message consists of the following components:

- The time stamp indicates when the log message event occurred. In this example, the time stamp is FEB 16 14:01:13.
- The physical interface indicates the *slot/port* (for example, 3/1) where the event occurred in the CSS.
- The counter records the incremental occurrence of each message. The count of this message is 2,453.
- The subsystem name and level is the CSS subsystem assigned to the message and the level of the message. Because this example is a subsystem message, the subsystem is the VLAN Manager and the log level is 7, which is a debug level (VLANMGR-7). See the "Configuring Logging for a Subsystem" section for a list of CSS subsystems.
- The log message indicating the event has occurred. The remaining string in the example is the event that occurred.

  ```
  Transmit sfm STP BPDU on bPort 1, egressLp 0x1f00 VlanLpSend()
  ret:0
  ```

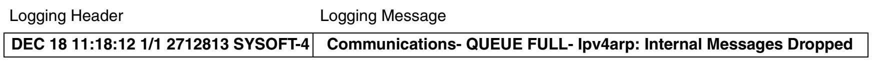
You can define a log message for a subsystem at a particular logging level through the **cliLogMessage subsystem** command. For more information, see the "Configuring a Log Message for a Subsystem at a Logging Level" section.

## Interpreting Undeliverable Messages

Undeliverable messages, such as IMM, EVENT, and LOCAL, appear in the CSS log when a queue on the CSS becomes full or overutilized to help clarify the nature of the problem.

Undeliverable messages consist of a logging header and a logging message, as shown in Figure 4-1.

***Figure 4-1 Undeliverable Message Format***



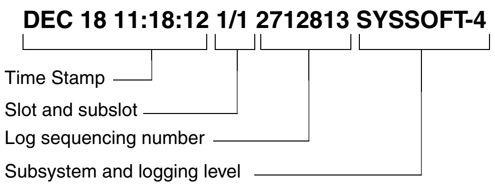
| Logging Header | Logging Message |
|---|---|
| DEC 18 11:18:12 1/1 2712813 SYSOFT-4 | Communications- QUEUE FULL- Ipv4arp: Internal Messages Dropped |

The logging header contains:

- A time stamp with the date and time
- The CSS slot and subslot number
- A logging sequence counter indicating that log messages were dropped due to excessive logging or CSS processor load
- The subsystem and its log level

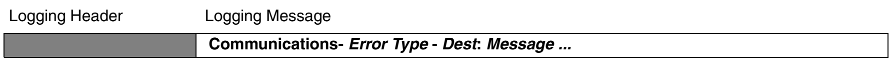
Figure 4-2 shows an example of a logging header.

***Figure 4-2 Logging Header in a Log Message***


DEC 18 11:18:12 1/1 2712813 SYSSOFT-4
Time Stamp
Slot and subslot
Log sequencing number
Subsystem and logging level

The logging message that follows the logging header defines the error type and the destination, followed by a message as shown in Figure 4-3.

***Figure 4-3 Logging Message***



The error type indicates one of the following conditions:

- QUEUE FULL - The receiving queue has no room to accept messages
- QUEUE DELETED - The CSS was trying to place a message in a valid queue but the queue was destroyed
- QUEUE INVALID - A destination message queue handle was not a valid object
- QUEUE UNKNOWN - The CSS was trying to determine the destination queue and the lookup failed

The destination indicates one of the following:

- String decoded name of the destination message queue
- Hexadecimal value if the error type is QUEUE DELETED, QUEUE INVALID, or QUEUE UNKNOWN
- INTERNAL, which indicates a LOCAL message passed between the tasks on the same processor

The *Message...* section in the logging message provides additional information concerning the problem. The log level for the undeliverable message determines how much information is in the logging message, and how often the message occurs in the log. You can set the log level to Warning-4, Info-6, or Debug-7.

By default, the log level for undeliverable messages is Warning-4. These messages occur every two seconds per message queue that is experiencing the problem. The message in the logging messages provides only the following information:

```
Internal Messages Dropped.
```

When you change the log level to Info-6, the undeliverable message still occurs every two seconds per message queue that is experiencing the problem. However, the logging message displays a message similar to the following:

```
Internal Messages dropped 5 times since the previous log for a total
of 21 times since bootup.
```

This message provides additional information such as:

- How many times the internal messages were dropped since the previous logging of the message
- The total number of dropped messages since the CSS bootup

When you require more detailed information, set the log level to Debug-7. An undeliverable message appears in the log each time the message occurs and an identifier appears in the body of the message. The logging message displays a message similar to the following:

```
Message (IMM:Base Class-IPV4_ARP, Identifier 1) from 1/1 (the other
CSS) failed to reach destination 'Ipv4Arp' on 1/1 (this CSS)
```

**Note** The Debug-7 log level displays debug messages and all other error levels. Be aware that selecting the Debug-7 log level may adversely affect the performance of the CSS.

The fields in this message include a message type, details based on the message type, source information, and destination information. Figure 4-4 illustrates the fields in the logging message displayed above.

***Figure 4-4 Logging Message Fields***

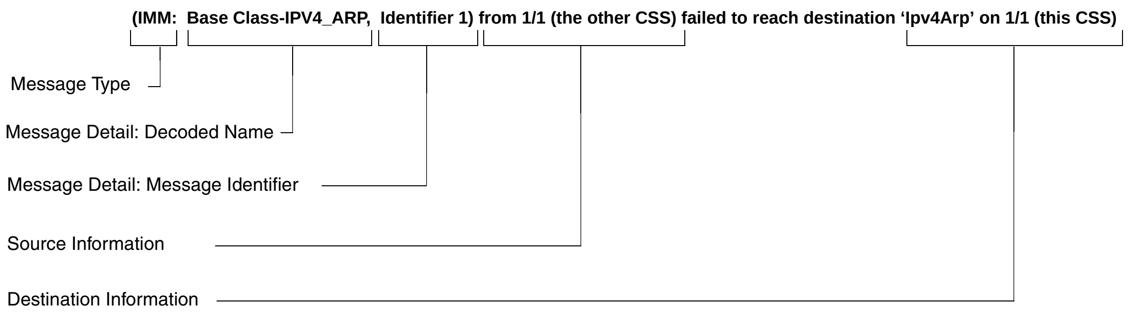

(IMM: Base Class-IPV4_ARP, Identifier 1) from 1/1 (the other CSS) failed to reach destination 'Ipv4Arp' on 1/1 (this CSS)
Message Type
Message Detail: Decoded Name
Message Detail: Message Identifier
Source Information
Destination Information

Table 4-6 describes the fields in a logging message.

*Table 4-6 Message Fields in a Log Level Debug-7 Logging Message*

| Message Fields | Possible Entries and Description |
|---|---|
| Message Type | **IMM** - Message passed both as an interprocessor or intraprocessor message.<br>**LOCAL** - Message passed between the tasks on the same processor.<br>**EVENT** - Message to be distributed to a registered set of recipients throughout the entire CSS. |
| Message Detail | For a LOCAL message type, there is no detail.<br>For an EVENT message type, the details can be one of the following:<br>• String decoded name of the event when the event is known. For example:<br>`(Event:Ipv4ArpChangeEvent)`<br>• Hexadecimal encoded name of the event when the event is out of range. For example:<br>`(Event:unknown type-0x00a00005)`<br>For an IMM message type, the details include the Base Class followed by one of the following:<br>• The string decoded name of the base class IMM message and a message identifier that is the decimal instance in the class. For example:<br>`(IMM:Base Class-IPV4_ARP, Identifier 1)`<br>• Unknown, and the hexadecimal value of the message type field. For example:<br>`(IMM:Base Class- Unknown, unknown type-0x00a00005)` |

*Table 4-6 Message Fields in a Log Level Debug-7 Logging Message (continued)*

| Message Fields | Possible Entries and Description |
|---|---|
| Source Information | The origin of the message. The information includes the slot and subslot numbers, and whether they are on either the local CSS or the remote CSS, such as in an Adaptive Session Redundancy (ASR) configuration. For example: `from 1/1 (other CSS)`<br><br>A LOCAL message type also includes information for the local processor context in string format. For example: `from 1/1- ‘EventAgent’ (this CSS)` |
| Destination Information | The destination as displayed at the beginning of the logging message. The destination information identifies where the message is going. This information includes the slot and subslot numbers as they appear in the logging header. For example: `failed to reach destination Ipv4Arp on 1/1 (this CSS)` |

Table 4-7 provides a listing of the frequently queried IMM Base Class messages and Identifiers that may appear as the message details for the different CSS subsystems.

*Table 4-7 IMM Message Identifiers*

| Base Class Decoded Name | Description and Message Identifier | Subsystem |
|---|---|---|
| CHASSIS | IMM message undeliverables or queue drops on the Chassis Manager Presence queue. The log messages sent to this queue occur during boot time processing of boards through the CSS Chassis Manager and Online Diagnostic Monitor subsystems. The master SCM should be the only destination.<br>The Identifiers for each assigned message type are:<br>• 0 - Occurs during the boot process to indicate that boards are present for processing.<br>• 1 - Occurs only once for each module during the boot process to indicate that the boards are ready for processing.<br>• 2 - Module state change notifications from the Chassis Manager and Online Diagnostic Monitor subsystems. This Identifier appears during the boot process or when specific modules fail.<br>• 3 - Submodule state change notifications from the Chassis Manager and Online Diagnostic Monitor subsystems. This Identifier appears during the boot process or when submodules fail.<br>• 4 - Initiates a Chassis Manager timeout to allow for modules to become operational. The timeout is sent from the Chassis Manager. This Identifier appears during the boot process. | Chassis Manager |

*Table 4-7 IMM Message Identifiers (continued)*

| Base Class Decoded Name | Description and Message Identifier | Subsystem |
|---|---|---|
| DHCP | Log messages that represent an overflow of messages on the Dynamic Host Configuration Protocol (DHCP) main receive queue residing on the SCM.<br>The Identifier is 1 for this message type. | DHCP Task |
| IPV4_ARP | IMM message undeliverables or queue drops on the ARP_Q queue residing on the SCM.<br>The Identifiers for each assigned message type are:<br>• 0 - Updates received from the Session Processor<br>• 1 - ARP requests initiated by the Session Processor | IPv4ARP Task |
| IPV4_RDNMGR_TID | IMM message undeliverables or queue drops on the IP Redundancy Manager queue residing on the SCM.<br>The Identifiers for each assigned message type are:<br>• 0 - VRRP software callback<br>• 1 - VRRP one second timer<br>• 2 - Remove service from the virtual routers<br>• 3- Redundancy VIP from the virtual routers | IP Redundancy Manager |
| IPV4_SLAVE_RX | IMM message undeliverables or queue drops on the SfmForwRx_Q queue residing on the Session Manager. This queue receives broadcast/multicast traffic, ICMP keepalives, and UDP/TCP fragments.<br>The Identifier is 1 for this message type. | IPv4 Slave Forwarding Rx Task |

Table 4-7 IMM Message Identifiers (continued)

| Base Class Decoded Name | Description and Message Identifier | Subsystem |
|---|---|---|
| SYS_IMM | IMM message undeliverables or queue drops on the ImmRxQ queue for pings and on the SysImmPing queue for ping acknowledgements. The SysImmPing queue is not permanent and is created and deleted dynamically as SysImmPing commands are issued.<br>The Identifiers for each assigned message type are:<br>• 0 - Pings<br>• 1 - Ping acknowledgements | Syssoft IMM |

## Frequently Queried Log Messages

Table 4-8 lists the frequently queried log messages for the Cisco 11500 series CSS. This table includes information on the possible cause and corrective action, if required. Log messages are divided by logging subsystem, with messages listed alphabetically.

*Table 4-8 Cisco 11500 Series CSS Log Messages*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| **acl Subsystem** | |
| `ACL-7: ACL match 2:254 Discarding`<br>`ACL-7: TCP SrcPort: 1043 DestPort: 21`<br>`ACL-7: Source: 172.20.57.2`<br>`ACL-7: Dest: 172.20.48.35` | Incoming traffic matches an ACL statement. The CSS examines, and then drops, the packet.<br><br>The log message appears for a packet that has an ACL statement applied by the flow manager. This log message indicates that load balancing can take place. |
| `ACL-7: ACL rule match 2:254 Discarding packet, Log Enabled` | Incoming traffic matches an ACL statement. The CSS examines, and then drops, the packet.<br><br>The log message appears for a packet that has an ACL statement applied by the IPV4 module. This log message indicates that the CSS may not set up flows for the packet (certain source or destination ports do not create a flow). This log message could also be related to issues with ICMP or RIP. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| **chassis Subsystem** | |
| `CHMGR: Missing backup power supply.` | The power supply lost AC power from the source. A CSS 11501 and CSS 11503 contains one power supply. The CSS 11506 contains up to three power supplies, but requires two functioning power supplies to guarantee service. If the following message appears first, then you can assume that the problem is with the AC power source, not the power supply.<br>`CHMGR: Cannot locate power supply: PS`*`number`*`.`<br>The *PSnumber* variable indicates which power supply cannot be found or has failed.<br>To determine whether the Cisco 11500 series CSS power supplies are working properly, both LEDs on the front of each power supply should be lit. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `CHMGR: Cannot locate power supply: PS`*`number`*`.` | The CSS chassis cannot find the power supply. The CSS 11501 and CSS 11503 contain one power supply. The CSS 11506 contains up to three power supplies but requires two functioning power supplies to guarantee service. The PS*number* variable indicates which power supply cannot be found or has failed. If you know that the power source is supplied to the chassis and correctly flowing to it, then the problem may be the power supply.<br><br>To determine whether the Cisco 11500 series CSS power supplies are working properly, make sure that both LEDs on the front of each power supply are lit. |

Table 4-8 Cisco 11500 Series CSS Log Messages (continued)

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| **circuit Subsystem** | |
| `CIRCUIT-7: Circuit status message for circuit 1023 sent to CE 20202c01 cause code is 7` | Codes indicate the status of interfaces within VLANs. The logical port cause and command codes are as follows:<br>**Cause** **Code**<br>CM_CIRCUIT_CREATED 1<br>CM_IP_REGISTER 2<br>CM_IP_NOT_REGISTER 3<br>CM_IP_MODIFIED 4<br>CM_LP_STATE_CHG 5<br>CM_CIRCUIT_REMOVED 6<br>CM_LP_ADDED 7<br>CM_LP_REMOVED 8<br>CM_LP_MODIFIED 9<br>CM_LP_FAILOVER 10<br>CM_CIRCUIT_DOWN 11<br>This log message indicates that a port has been added to a VLAN. This log message can occur when the association to a VLAN changes as the port transitions from an up to a down state.<br>Use the **show circuit** command to list the VLANs (refer to the *Cisco Content Services Switch Routing and Bridging Guide*). Check the status of the ports of the VLAN or determine whether the VLAN is active. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| **csdpeer Subsystem** | |
| `CSDPEER-7: LR Send list too small !!!` | The number of domain names sent by the peer exceeds the size of the CSS list. You can configure this parameter through the **(config) dns-peer** command (see the *Cisco Content Services Switch Global Server Load-Balancing Configuration Guide*). We recommend that you configure the receive and send slots with the same value. The default slot value is 255. |
| **flowmgr Subsystem** | |
| `FLOWMGR-4: Flow manager received an illegal message with code 10` | One of the Ethernet ports received a high number of malformed packets, resulting in an overflow of the fastpath. In this case, the flow manager received a badly formatted control message from the fastpath. This problem may be due to intermittent hardware, which results in the fastpath corrupting the packets, or the problem is related to the fastpath receiving streams of malformed packets and leaking some of those packets to the flow manager.<br><br>Use the **show ether-errors** command to display information for a port that is experiencing many errors (refer to the *Cisco Content Services Switch Routing and Bridging Guide*). Try disconnecting the port or changing ports and determine whether the errors stop. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `FLOWMGR-4: Flow manager received an illegal message with code 255` | One of the CSS Ethernet ports encountered a significant number of errors and a few malformed packets reached the flow manager. |
| `FLOWMGR-4: Flow manager received an illegal message with code 194` | The flow manager received an illegal message from the fastpath. This log message may occur due to hardware problems or a port receiving an excessive number of malformed packets. Use the **show mibii** or **show ether-errors** command to look for errors on one of the CSS ports. |
| `FLOWMGR-6: FM_Tcp: Handling generic FMTCP flow Re-Transmit ERROR`<br>`FLOWMGR-6: FM_ReTransTimeout: Re-Transmit timeout ERROR`<br>`FLOWMGR-6: FM_Tcp: Handling generic FMTCP flow Re-Transmit ERROR`<br>`FLOWMGR-6: FM_ReTransTimeout: Re-Transmit timeout ERROR`<br>`FLOWMGR-6: FM_Tcp: Handling generic FMTCP flow Re-Transmit ERROR` | If the CSS handles a content request for a Layer 5 rule that spans more than three TCP packets, after the CSS decides on the server to use, it sends the TCP packets in TCP slow start form. Here is an example of a five-TCP segment content request:<br>`Segment 1 -->`<br>`Segment 2 -->`<br>`(wait for an ACK)`<br>`<--- ACK`<br>`Segment3 ->`<br>`Segment4 ->`<br>`Segment5 ->`<br>`<-- Content`<br>If the CSS does not receive an ACK from the server within three seconds, it will refuse any remaining packets and terminate the connection with a reset. At that point, the log message is generated. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `FLOWMGR-6: \n FM_UtilGenericTcpFlowReject: Handling Generic Flow REJECT` | The CSS rejects a connection, either due to issues with the client side or the server side of the connection.<br><br>When this log message occurs as a result of the client side connection, the issue could be due to a content request that spans more than the configured maximum (default is 6). Other reasons could include:<br>• A delayed ACK could not be sent to the client (at 200ms)<br>• A delayed ACK is sent to the client, and the client responds back with a TCP SYN/FIN/RST handshake sequence.<br>• The client side closes down unexpectedly.<br><br>When this log message occurs as a result of the server side connection, the issue could be due to the CSS sending the spanned content request to the server and did not get an acknowledgement from the server or received an unexpected response (for example, due to the flow being torn down to the client side).<br><br>If this message appears frequently in the log, contact Cisco Systems TAC. |
| `FLOWMGR-7: Allocation for a vector-loaded flow, where theFlow = 840ef5b0` | This is an informational message. No further action is required. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `FLOWMGR-7: Exceeded outflow SYN count` | For a Layer 5 rule, the CSS is trying to establish a connection with the backend server. The CSS sent four SYNs to the backend server and did not get a response.<br><br>For the CSS to establish a connection with the backend server, the CSS must receive the following TCP/IP handshake:<br><br>`SYN->`<br>`<-SYN/ACK`<br>`ACK->`<br>`GET->`<br><br>After receiving the GET message, the CSS opens the backend connection. At that point, the log message is generated.<br><br>When several of these log messages occur, there might be a malfunctioning server. The server problem could be from keepalives, or from regular TCP HTTP traffic. Make sure the port 80 sockets are not full on the servers. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| **fp-driver** | |
| `FP_DRV-4: PrismImmFastPath::Send: Could not allocate an MCID. Remote message send aborted.` | The CSS MIPS processor attempts to send a group message, but the processor cannot obtain an multicast ID (MCID) from the Multicast ID Module (MID). The MID keeps track of the reference count on a buffer when the CSS sends a packet to multiple locations. The fast path uses an MCID to reference count the buffer with a contained packet that is being flooded to all ports in a VLAN.<br><br>For MCIDs to be depleted in the CSS, there must be 1024 reference counted packets queued up in some combination of hardware queues and software queues.<br><br>To fill up hardware queues, the CSS must receive a large number of packets, which it then chooses to flood out all ports. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `FP_DRV-4: PrismImmFastPath::Send: Could not allocate an MCID. Remote message send aborted.`<br><br>(continued from previous page) | In the case of software queues, it might be possible for some task on the MIPS processor to deplete the MCID pool by sending a large number of messages to a group that contains both local and remote members. If the local member has a very large queue, the queue could fill up before running the recipient task, processing the messages, and freeing the buffers.<br><br>Of the two potential causes, the most likely causes is the CSS receiving a large number of packets which it must flood out all ports.<br><br>This log message typically occurs when the CSS loses a specific route and forwards the flow out the default gateway. The default gateway then forwards the flow back to the CSS because the routing table lists the CSS as the next hop. As a result, the packet is continuously routed out the default gateway and back to the CSS. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| **ipv4 Subsystem** | |
| `IPV4-4:Ipv4IfMgrCctUpdateMsg: IF config for circuit 1015 not found`<br>`CIRCUIT-4: Error, Circuit 1015 does not exist.` | You deleted a circuit but the circuit is still referenced by an ACL or to another configuration parameter. Verify the CSS configuration and make the necessary modifications to remove references to the deleted circuit. |
| `IPV4-4: Ipv4ReceivePacket: out of mbufs[, count` *`number,`* `current ingress ce 0x120fa00]` | An *mbuf* is a data structure in BSD UNIX-based IP stacks (such as the VxWorks stack) that is used for buffering. This log message indicates the CSS received a packet that was addressed to a CSS IP address, and when attempting to send the packet up the VxWorks IP stack, the CSS had no remaining buffers.<br><br>These buffers are separate from those used for flow setup and forwarding purposes. They are used only when traffic is sent to the CSS itself, (for example, during a Telnet session).<br><br>The CSS logs only one message per second that includes the count of how many messages were dropped that second.<br><br>You can set a flag in debug mode to enable detailed logging (shown in the brackets) of the packet header and ingress logical port of messages dropped during an out of mbufs condition.<br><br>If you receive this message, contact Cisco Systems TAC. |

Table 4-8 Cisco 11500 Series CSS Log Messages (continued)

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `IPV4-4: Ipv4SfmArpTx: unknown circuit in buffer (2001)` | An ARP TX task is running on the SFM, receiving packets from the SFM and transmitting them to the proper egress ports. This message includes the circuit number (2001, in this example). If the circuit number for this ARP was down or inactive while the ARP was still being queued in the SFM, the message would appear in the log. An action caused the circuit to be removed while data for the circuit was still in the buffer.<br><br>Determine whether all physical interfaces in a circuit VLAN are going up and down, or a configuration change occurred on the VLAN at the time of the message. |
| `IPV4-4: Ipv4SfmForwRx: bad IP version received (0)` | The IPV4 receive task received a packet and the IP version is displayed in parentheses (). The CSS discards any packet that is not Ipv4 version 4. In this example, the IP version is 0. If you see many of these messages, the problem could be an improperly configured device or a DoS attack. |
| `IPV4-4: Ipv4SfmForwTx: No VC for buffer (0x00000000)` | The IPV4 transmit task has a buffer to transmit to the switch fabric processor, however the CSS still needs to create the Virtual Circuit to the fastpath. This message typically occurs if the CSS Ethernet port changes state.<br><br>This is an informational message. No further action is required. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `IPV4-4 Ipv4SfmForwTx: unknown logical port in buffer <0x05c01f00>` | A link became unavailable while an ARP or other IPV4 packets were in transit. When this occurred, the CSS chose a logical port to transmit from, formatted the packets, and then attempted to transmit the packets. At the point when the CSS attempted to transmit packets, the logical port was no longer available.<br><br>This is an informational message. No further action is required. |
| `IPV4-4: Ipv4ApIoctl: unknown command: 1074031872` | This is an informational message. No further action is required. |
| `IPV4-4: Ipv4SfmForwRx: buffer length (872) less than IP length (1004)` | IP packets have been corrupted and the IP header Total Length value does not match with the actual length of the packet. In this case, the SFP receives less total bytes than expected from the IP header length. This message may be related to hardware problems or errors on the line (corrupted packets). Use the **show mibii** or **show ether-errors** command to look for errors on one of the CSS ports. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `IPV4-4: (RIP) VIP Redundancy callback on unregistered address 0.0.0.0 range 0` | The CSS is running VIP redundancy and is using the global Routing Information Protocol (RIP) to advertise VIPs on the CSS to other routers (configured through the **rip advertise** command). In this case, RIP sends the redundancy manager in the CSS the VIP and the range, and requests to be informed of any changes in the VIP redundancy status.<br><br>If the redundancy manager monitors a change in VIP redundancy status, it contacts RIP with the VIP address and the range. To ensure the proper advertisement of the VIPs, RIP verifies the VIP address and range. This log message occurs when RIP is unable to find the VIP address received in the callback message from the redundancy manager.<br><br>Check the IP address specified in the **rip advertise** command and verify that the VIPs are configured properly for VIP and virtual interface redundancy. |
| `IPV4-0: Ipv4SfmProcessArpFrame: ARP packet with unknown ingress port 0x0fc01f00` | A CSS Ethernet port became unavailable while an ARP packet is in transit from the fastpath to the IPV4 destination. As a result, the ARP packet is dropped.<br><br>This is an informational message. No further action is required. |

Table 4-8 Cisco 11500 Series CSS Log Messages (continued)

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `IPV4-4: Ipv4SfmCmDeleteFlow: -1 response from VccRemoveVc, egress 0x09c01f00` | A CSS Ethernet port became unavailable. As a result, the IPV4 module was unable to delete a Virtual Circuit established through the switch fabric to the fastpath. This is an informational message. No further action is required. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `IPV4-4: Ipv4SfmProcessArpFrame: bad ARP packet received` | The CSS detects that it has received an invalid ARP packet. The following messages can appear in the log to clarify why the CSS logs the receipt of an invalid ARP packet.<br>`IPV4-4: ffff53ff01ff ff0077fa6503 0806`<br>`IPV4-4: HW type: 0x0000  Proto type: 0x0000`<br>`IPV4-4: HLEN 0x00  PLEN 0x00 OPTPA-TSI-CSS1# 0x0000`<br>`IPV4-4: Sender HA 000000000000`<br>`IPV4-4: Sender IP 0.0.0.0`<br>`IPV4-4: Target HA 000000000000`<br>`IPV4-4: Target IP 0.0.0.0`<br>The first line in the log message identifies the destination MAC address of the packet, the source address of the packet, and the type of IP packet. In the example above, the destination MAC address is ff-ff-53-ff-01-ff and the source MAC address is ff-00-77-e7-65-03. In addition, 0806 equals ETHERTYPE_ARP.<br>The second line in the message identifies the hardware type and protocol type. In this example, the CSS logs the messages because both the hardware type and protocol type are **0000**. The CSS views this value as an indication of an invalid packet. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `IPV4-4: Duplicate IP address detected: 192.168.163.129 00-08-e2-10-38-54`<br>`IPV4-4: Incoming CE 0x3001f04, incoming (0 based) SLP 0xc` | A duplicate IP address has been detected by the CSS. Typically, two level 4 IPV4 messages appear to provide assistance in finding the duplicate IP address that was detected by the CSS.<br><br>The first message states that the CSS received a packet with a source IP address that is also configured on the CSS. The message identifies the duplicate source IP address and its corresponding MAC address as an aid to locate the device with the duplicate IP address.<br><br>The second message is intended to assist you in locating the port on the CSS that has received the duplicate IP address. Use the **flow statistics** command to locate the interface on the CSS. The **flow statistics** command should correspond the CE value listed in the log message with a port. |
| **keepalive Subsystem** | |
| `KAL-7: kal_ServiceNotify: kalIndex = 24 kalSvcEvent=3`<br>`KAL-7: kal_ServiceNotify: kalIndex = 31 kalSvcEvent=4`<br>`KAL-7: kal_ServiceNotify: kalIndex = 49 kalSvcEvent=5` | The CSS is configured with HTTP keepalives (HEAD or GET) and the servers transition between states. The service event (kalSvcEvent) values are as follows:<br>• 3 = Alive<br>• 4 = Dying<br>• 5 = Dead<br>Check the status of the server. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| **netman Subsystem** | |
| `NETMAN-1: TRAP:Authentication:Generated by: 192.168.36.252` | The CSS is configured to transmit SNMP trap messages and a user attempts to access the CSS with an incorrect SNMP community string. In this example, the CSS sends a trap to the configured SNMP trap receiver stating that a client with IP address 192.168.36.252 is trying to access the CSS with an incorrect community string.<br><br>This log message also appears when a user attempts to access the CSS using SNMP and SNMP is not configured on the CSS. Refer to Chapter 5, Configuring Simple Network Management Protocol (SNMP), for configuration information. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `NETMAN-2: Sshd:do_authenticated:ERROR-> TSM Rejects connection` | Remote access is being initiated to the CSS CLI. If the CSS security manager rejects the log in, the session terminates.<br><br>The security manager can reject the log in when:<br>• The maximum number of concurrent security manager users had been exceeded (128 concurrent users).<br>• The CSS could not re-register (if you had a session that just ended and the flow cleanup was not performed, and you attempted to re-register too soon).<br>• The CSS ran out of memory and could not allocate a control block. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `NETMAN-2: Generic:LINK DOWN for 13/1`<br>`CIRCUIT-6: Port 13/1 is down for circuit VLAN1`<br>`NETMAN-2: Generic:LINK DOWN for 13/2`<br>`CIRCUIT-6: Port 13/2 is down for circuit VLAN1`<br>`NETMAN-2: Generic:LINK DOWN for 13/3`<br>`CIRCUIT-6: Port 13/3 is down for circuit VLAN1`<br>`NETMAN-2: Generic:LINK DOWN for 13/4`<br>`CIRCUIT-6: Port 13/4 is down for circuit VLAN1`<br>`SYSSOFT-3: ONDM: Timeout downloading image to EPIF 0 from the switch.`<br>`SYSSOFT-3: ONDM: Timeout downloading image to EPIF 0 from the switch.` | EPIF 0 belongs to the first four ports on a FEM. This log message usually relates to a problem with the SFM not getting the code to the FEM or the FEM not reading the SFM properly.<br>In this case, there is a communications problem between the SFP 9/2 and the FEM.<br>`JAN  5 00:31:43 arrowpoint1.com 9/2 385390 SYSSOFT-3: ONDM: Timeout downloading image to EPIF 0 from the switch.`<br>`JAN  5 00:31:45 arrowpoint1.com 9/2 385407 SYSSOFT-3: ONDM: Timeout downloading image to EPIF 0 from the switch.`<br>Reseat the SFM in slot 9, then reseat the FEM in slot 13 that is controlled by the SFM. Cycle power to the CSS. |
| `NETMAN-2: Enterprise:Service Transition:ServerA -> down`<br>`NETMAN-5: Enterprise:Service Transition:ServerA -> alive` | This is an information message when the service has changed state. Check the status of the server based on the keepalive parameters. |
| `NETMAN-4: SNMPAPI:SNMPAPI_Set:SET failure` | A user on the CLI, connected to the CSS either through the console or by Telnet, has entered an incorrect command. A Telnet or console session displays this message, stating that the command was incorrect. |

*Table 4-8      Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `NETMAN-5: Enterprise: Login Failure:vty2 10.6.3.171 Mandy` | SNMP Enterprise login failure traps are enabled and an invalid username and password have been entered. Refer to Chapter 5, Configuring Simple Network Management Protocol (SNMP), for details on SNMP and the CSS. |
| `NETMAN-5: Generic:SNMP Authentication > Failure from x.x.x.x` | A user is trying to use SNMP to poll the CSS, but has entered the wrong community string. Refer to Chapter 5, Configuring Simple Network Management Protocol (SNMP), for details on specifying a community string. |
| `NETMAN-5: Enterprise:Service > Transition:nexthop00001 -> down` | The next hop IP address can not be reached by the CSS. When you configure a static route, an internal service is automatically created by the CSS. When the service is up, the static route is included in the routing table. If the service is unavailable, the service is removed from the routing table.<br><br>Make sure that all of the routes included in the CSS routing table are available. Some routes may have transitioned between states. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `NETMAN-5: Generic:LINK UP for 3/1`<br>`SYSSOFT-7: NP55_connection.c 512: Connection already open or reserved`<br>`SYSSOFT-3: NP55 Driver: Connection already open or reserved`<br>`SYSSOFT-2: VccAddVc:open conn failed w/ stat = -1; iVc 320; eVc 290`<br>`FLOWMGR-7: FM_GetIpv4Vc: Warning VCC_FP_IPV4_DC failed` | The flow manager tried to reallocate a Virtual Circuit that was already established.<br><br>These messages occur when the port is coming up. They do not represent a problem. The messages are most noticeable at the end of boot time if you connected through the console. |
| `NETMAN-7: clm_ProcessStdAction:ERROR->Action<clms_dir>not found`<br>`NETMAN-7: CLM:ERROR from clm_DispatchActionRoutine()` | An invalid CLI command was entered. In this example, a user entered the **dir** command in debug mode and specified an invalid directory. For example:<br><br>`(debug)#` **`dir d:`** |
| `NETMAN-7: SNMP:UNKNOWN RSP (493512`<br>`NETMAN-7: SNMP:(493512) Index = 1 <NO_SUCH_NAME>` | A valid SNMP agent (community string matched) is attempting to set an invalid object and the CSS does not recognize the object. |
| `NETMAN-7: TSM:tsm_SendToCLA:ERROR->Write` | This security manager message is associated with line data moving through the stack after a line has been disconnected (Telnet application disconnect). This message is at DEBUG level for developer information purposes only. |
| `NETMAN-7: ASUPPORT:as_SyncTask:ERROR->No registered reciever for MT:5/1.0 W` | This is an informational message. No further action is required. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| **portmapper Subsystem** | |
| `PORTMAPPER-5: PortUnmap no Port mapping found.` | A source group is running out of portmappers. Use the **portmap** command to increase the portmappers on the source group so that the message does not appear, and users or services do not see a performance or network address translation problem. |
| **publish Subsystem** | |
| `PUBLISH-1: Unable to allocate tree memory <4150000>` | The CSS is looking for a segment of memory that is approximately 4150000 bytes and it cannot locate an available block of memory. In some cases, this message may be caused by a replication misconfiguration. Review the configuration and verify that it reflects what was intended. Verify that files are being replicated properly.<br><br>If this message is seen with significantly smaller memory requests, the system memory may not sufficient in size to meet the requirements of the configuration. To isolate the issue, monitor the available memory under non-replication conditions to determine a baseline and then repeat this process while replicating to isolate this issue.<br><br>Use the **show system-resources** command to view information about the installed and free memory in the CSS. To make additional memory available, reboot the CSS. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| **radius Subsystem** | |
| `RADIUS-7: Auth Primary`<br>`RADIUS-7: The id is 63`<br>`RADIUS-7: Return Auth Primary`<br>`RADIUS-7: RADIUS no memory available` | The RADIUS server does not have the correct attributes set up for the CSS. Refer to *Cisco Content Services Switch Security Configuration Guide*, for background information on setting up an ACS radius server. |
| `RADIUS-4: RADIUS Authentication failed with reason code 2` | In this message, the different codes include:<br>`#define PW_ACCESS_REQUEST 1`<br>`#define PW_ACCESS_ACCEPT 2`<br>`#define PW_ACCESS_REJECT 3`<br>`#define PW_ACCOUNTING_REQUEST 4`<br>`#define PW_ACCOUNTING_RESPONSE 5`<br>`#define PW_ACCOUNTING_STATUS 6`<br>`#define PW_ACCESS_CHALLENGE 11`<br>In the example above, code 2 indicates that the CSS received the Accept response from the RADIUS server but may have rejected the Accept response, perhaps due to an invalid username or password.<br>This log message only appears when logging debug messages (debug-7) for the radius subsystem. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| **sntp Subsystem** | |
| `SNTP-6: Sntp Server has incorrect mode 29` | This message indicates potential issues with the SNTP server. Ensure that the SNTP server is transmitting time updates as “server” to the CSS. SNTP server updates are the only SNTP server updates supported by the CSS. |
| **syssoft Subsystem** | |
| `SYSSOFT-2: VccAddVc:open conn failed w/ stat = -1; iVc 320; eVc 290` | This message occurs when a port transitions from an up to a down state. Check autonegotiation, for a defective cable, or for malfunctioning hardware. |
| `SYSSOFT-3: ONDM: Could not open file <wsscm.sys>`<br>`SYSSOFT-3: ONDM: Could not download Sub-module 8/1.`<br>`SYSSOFT-3: ONDM: Could not open file <wssfm.sys>`<br>`SYSSOFT-3: ONDM: Could not download Sub-module 6/2.`<br>`SYSSOFT-3: ONDM: Could not download Sub-module 6/1.`<br>`SYSSOFT-3: ONDM: Could not download Sub-module 5/2`<br>`SYSSOFT-3: ONDM: Could not download Sub-module 5/1.`<br>`SYSSOFT-3: ONDM:No Sfm proxy for Slot 2.`<br>`SYSSOFT-3: ONDM:No Sfm proxy for Slot 1.` | The CSS could not find the image file to load on the disk. There is something wrong with the disk or the file was deleted from the directory.<br>Contact Cisco Systems TAC. |
| `SYSSOFT-4: SYS:SysImmBind:Bind Collision TSM:5/1.1 W` | This is an informational message. No further action is required. |

Table 4-8 Cisco 11500 Series CSS Log Messages (continued)

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `SYSSOFT-4: Invalid Target(0x03087a01) for Chassis Type, Message being dropped.` | The CSS attempts to request the sending of a message to a slot and subslot that does not exist in the chassis. The log message indicates the incorrect address in hexadecimal and the message has been dropped. This message most likely occurs when you issue a command to slot 4 and subslot 1 on a CSS 11503.<br>No corrective action is required. |
| `SYSSOFT-4: Event not deliverable, msgq id =0x8cc48980, event id = 29, event name = BridgeMacAddrEvent` | The CSS was unable to deliver a certain process because a queue was full. Every message signifies that a event has been dropped because the queue full condition. This message appears when the fastpath (network processor) performs a source MAC address lookup and cannot find an entry. The fastpath then sends a MAC address learn message to the SCM. If the SCM recieves too many messages before it has time to process them, the messages fill the queue. |

*Table 4-8 Cisco 11500 Series CSS Log Messages (continued)*

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| `SYSSOFT-4: Event not deliverable, msgq id = 0x865c2110, event id = 4, event name = Ipv4RouteChangeEvent` | The CSS was unable to deliver a certain process because a queue was full. Every message signifies that a event has been dropped because the queue full condition. This message happens whenever a route change is detected. The RIP process, the OSPF process, the caretaker processes (one for each SFM, which try to keep the SFM and SCM route tables in sync), the static route process, and the ARP process register for this event.<br><br>Look for any routes transitioning state, locally attached stations or servers going up and down, or a large number of ARP requests being performed. |
| `SYSSOFT-7: MPOOL:mpoolAutoAlloc:WARN->Overrun on MPOOL 3 321` | This message typically appears at boot up as an informational message to let you know that additional CSS memory is being allocated. No further action is required. |

Table 4-8 Cisco 11500 Series CSS Log Messages (continued)

| Log Message (sys.log: Subsystem Name, Level, and Message) | Cause and Resolution |
|---|---|
| **vlanmgr Subsystem** | |
| `VLANMGR-4: DeleteMacAddr() called with VlanID = 0 for MacAddr 0- 0- 0- 0- 0- 0` | The VLAN Manager is being asked to delete a MAC address entry from the forwarding table with a VLAN ID and MAC address of all zeros. |
| **vpm Subsystem** | |
| `VPM removed Vc 8000b00 based on failure of port 3401f00.<010>` | The CSS is reclaiming the resources used by a specific Ethernet port because the port is unavailable. The CSS reclaims resources when a port is unresponsive to an internal check, or when a circuit is unavailable. No addressing information is available for that Ethernet port. Use the **show interface** command to display information for the Ethernet ports and detemine which port is the problem. |
| **wcc Subsystem** | |
| `WCC-7: Route Change for IP Address ( x.x.x.x)` | This is an informational message, stating that an ARP came in on a different port. |

## Where to Go Next

Chapter 5, Configuring Simple Network Management Protocol (SNMP) describes how to configure Simple Network Management Protocol (SNMP) features of your CSS.

# Configuring Simple Network Management Protocol (SNMP)

This chapter provides information on configuring Simple Network Management Protocol (SNMP) features of your CSS. It also provides a brief overview of SNMP, an Application Layer protocol used extensively in the communications industry. Information in this chapter applies to all CSS models except where noted.

This chapter contains the following major sections:

## SNMP Overview

SNMP is a set of network management standards for IP-based internetworks. SNMP includes a protocol, a database-structure specification, and a set of management data objects. SNMP implementations typically consist of a management application running on one or more network management systems (NMSs), and agent applications, usually executing in firmware on various network devices.

SNMP has two major standard revisions, SNMPv1 and SNMPv2. The CSS supports both SNMPv1 and SNMPv2C (SNMP version 2C), a standard Management Information Base (MIB-II) object, along with an extensive set of enterprise MIB objects. MIBs are discussed in the "Management Information Base (MIB) Overview" section.

This section includes the following topics:

- Managers and Agents
- SNMP Manager and Agent Communication

**Note** By default, SNMP access to the CSS is enabled through the **no restrict snmp** command. For details, see the "Preparing to Configure SNMP on the CSS" section.

## Managers and Agents

SNMP uses software entities called *managers* and *agents* to manage network devices:

- The *manager* monitors and controls all other SNMP-managed devices (network nodes) in the network. There must be at least one SNMP manager in a managed network. The manager is installed on a workstation somewhere in the network.
- An *agent* resides in a managed device (a network node). The agent receives instructions from the SNMP manager, and also sends management information back to the SNMP manager as events occur. The agent can reside on routers, bridges, hubs, workstations, or printers, to name just a few network devices.

There are many different SNMP management applications, but they all perform the same basic task: They allow SNMP managers to communicate with agents to monitor, configure, and receive alerts from the network devices. You can use any SNMP-compatible NMS to monitor and control a CSS.

## SNMP Manager and Agent Communication

There are several ways that the SNMP manager and the agent communicate.

- The manager can:
  - Retrieve a value (a GET action).

    The SNMP manager requests information from the agent, such as the number of users logged on to the agent device, or the status of a critical process on that device. The agent gets the value of the requested MIB object and sends the value back to the manager.
  - Retrieve the value immediately after the variable you name (a GET-NEXT action).

    The SNMP manager retrieves values from within a MIB. Using the get-next function, you do not need to know the exact MIB object instance you are looking for; the SNMP manager takes the variable you name and then uses a sequential search to find the desired variables.
  - Retrieve a number of values (a GET-BULK action).

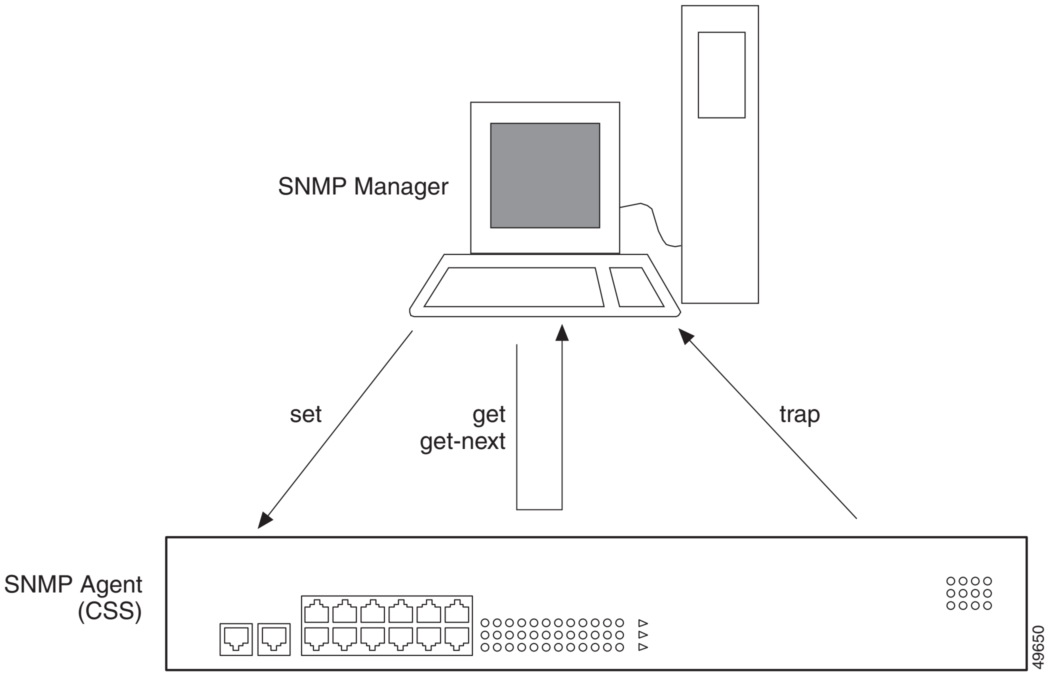
    The SNMP manager performs a number of get-next actions that you specify.
  - Change a setting on the agent (a SET action).

    The SNMP manager requests the agent to change the value of the MIB object. For example, you could run a script or an application on a remote device with a set action.

- An agent can send an unsolicited message to the manager at any time if a significant, predetermined event takes place on the agent. This message is called a *trap*. For details on SNMP traps (and associated MIB objects) supported by the CSS software, see the "CSS SNMP Traps" section.

  When a trap condition occurs, the SNMP agent sends an SNMP trap message to the device specified as the *trap receiver* or *trap host*. The SNMP Administrator configures the trap host (usually the SNMP management station) to perform the action needed when a trap is detected. Figure 5-1 illustrates SNMP manager and agent communication.

***Figure 5-1 SNMP Manager and Agent Interaction***



## Management Information Base (MIB) Overview

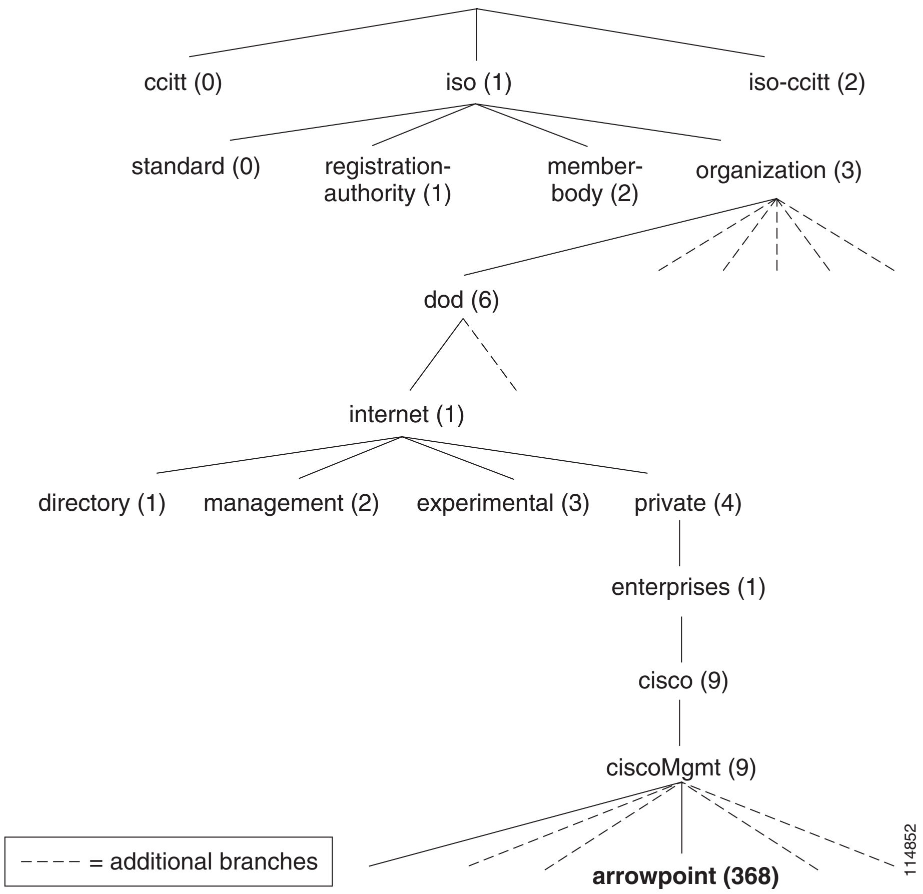
SNMP obtains information from the network through a Management Information Base (MIB). The MIB is a database of code blocks called *MIB objects*. Each MIB object controls one specific function, such as counting how many bytes are transmitted through an agent's port. The MIB object comprises *MIB variables*, which define the MIB object name, description, default value, and so forth.

The collection of MIB objects is structured hierarchically. The MIB hierarchy is referred to as the *MIB tree*. The MIB tree is defined by the International Standards Organization (ISO). The MIB is installed on the SNMP manager and is present within each agent in the SNMP network.

At the top of the tree is the broadest information about a network. Each branch and sub-branch of the tree gets progressively more specific, and the lowest branches of the tree contain the most specific MIB objects; the leaves contain the actual data. Figure 5-2 shows an example of how the MIB tree objects become more specific as the tree expands.

**Note** There are two versions of the MIB tree as defined by ISO: MIB-I and MIB-II. MIB-II has more variables than MIB-I. Refer to the MIB-II standard in RFC 1213, "Management Information Base for Network Management of TCP/IP-based Internets: MIB-II."

***Figure 5-2 Top of the MIB Tree***



This section includes the following topics:

- MIB Variables
- MIB Extensions (Enterprise MIBs)
- Updating MIB Files

## MIB Variables

There are two types of MIB variables:

- Scalar - Variables that define an object with a single representation. This means that the object describes a particular characteristic of the entire system. An example of a scalar variable is **SysDescr**, which provides a system-wide description of the CSS.
- Tabular - Variables that define an object with multiple representations. This means that the object can have different values, depending on the qualifier. For example, one tabular object could show bytes per interface, temperature per board, or hits per service.

As shown in Figure 5-2, a number is associated with a MIB object name. This number is called the *object identifier* (OID), and it uniquely identifies the MIB object in the MIB tree. (The dotted lines represent other branches not relevant to this discussion.)

For example, note in Figure 5-2 that the MIB object labeled arrowpoint (368), which contains the MIB objects specific to the CSS, can be labeled:

```
iso.organization.dod.internet.private.enterprises.cisco.ciscoMgmt.
arrowpoint
```

or

```
1.3.6.1.4.1.9.9.368
```

## MIB Extensions (Enterprise MIBs)

The MIB tree has a special branch set aside for specific vendors to build their own extensions; this special branch is called the *Enterprise MIB branch*. The CSS MIBs are included in the following directories and ZIP files on the CSS disk:

- SNMPv1 MIBs - /mibs/v1/cssmibsv1.zip
- SNMPv2C MIBs - /mibs/v2/cssmibsv2.zip

The MIB files in this branch comprise the CSS Enterprise MIBs (the highlighted MIB identifier in Figure 5-2). The enterprise MIB files are categorized along functional boundaries.

For a list of MIB branches under the CSS Enterprise MIB, see the "CSS MIBs" section.

## Updating MIB Files

This section describes the procedure to follow when you want to update the MIB files on your management station.

### Loading the Standard MIBs

Before you can load the CSS MIBs on your management station, you must load the following standard MIBs.

SNMP v1 Standard MIBs

- RFC-1212
- RFC-1215
- INET-ADDRESS-MIB
- SNMP-FRAMEWORK-MIB
- SNMPv2-TC-v1
- RFC1155-SMI
- SNMPv2-SMI-v1 (RFC 1493)

SNMP v2 Standard MIBs

- SNMPv2-SMI
- SNMPv2-TC
- SNMP-FRAMEWORK-MIB
- SNMPv2-CONF
- INET-ADDRESS-MIB
- BRIDGE-MIB

To update the standard MIBs on your management station after you upgrade the CSS software:

1. Transfer the standard MIBs to your management station.
2. Load the standard MIBs into the management application.

## Loading the CSS MIBs

We recommend that you update the CSS Enterprise MIBs after you upgrade the CSS software. CSS MIBs are included in the CSS GZIP file. During the software upgrade, the MIBs are loaded into the CSS /mibs directory. For details about the CSS MIBs, see the "CSS MIBs" section.

To update the CSS MIBs on your management station after you upgrade the CSS:

1. Transfer the CSS MIBs using FTP from the CSS MIBs (/v1 or /v2) directory to your management station.

   **Note** The CSS implementation of FTP does not support the **mget** command, which is used for multiple file transfers.

2. Load the CSS MIBs into the management application.

**Note** The /v2 directory contains the full set of CSS v2 MIBs. The /v1 directory contains only those CSS v1 MIBs that will compile using a v1 compiler. If you do not find a MIB that you need in the /v1 directory, use the corresponding MIB in the /v2 directory.

## SNMP Communities

Each SNMP device or member is part of a *community*. An SNMP community determines the access rights for each SNMP device.

You supply a name to the community. After that, all SNMP devices that are assigned to that community as *members* have the same access rights. The access rights that the CSS supports are:

- read - Allows read-only access to the MIB tree for devices included in this community
- read-write - Allows both read and write access to the MIB tree for devices included in this community

## Preparing to Configure SNMP on the CSS

Once you have set up your SNMP management application, you are ready to configure SNMP settings on the CSS. You can configure two basic areas of SNMP functionality on the CSS: SNMP functions and RMON functions.

**Note** Refer to Chapter 6, Configuring Remote Monitoring (RMON), for information on configuring RMON.

To control SNMP access to the CSS, use the **no restrict snmp** and **restrict snmp** commands. Access through SNMP is enabled by default. The options for this global configuration mode command are:

- **no restrict snmp** - Enables SNMP access to the CSS (default setting)
- **restrict snmp** - Disables SNMP access to the CSS

Before you set up SNMP on your network consider the following items when planning your SNMP configuration:

- Decide which types of information the SNMP manager needs (if your application is using an SNMP manager). Choose the particular MIB objects that you want through the management software.
- Decide how many trap hosts you need. In some network configurations, you may want to have a primary trap host with one other workstation also receiving traps for redundancy. In a distributed or segmented network, you may want to have more trap hosts enabled. You can configure up to five trap hosts per SNMP agent; that is, one agent can report to a maximum of five hosts.
- Designate a management station or stations. The CSS is an agent in the SNMP network scheme. The agent is already embedded in the CSS when you boot up the device. All you need to do is configure the SNMP parameters on the CSS.

## Defining the CSS as an SNMP Agent

This section describes how to define the CSS as an SNMP agent. It includes the following topics:

- SNMP Agent Configuration Quick Start
- Configuring an SNMP Community
- Configuring an SNMP Contact
- Configuring an SNMP Location
- Configuring an SNMP Name
- Configuring SNMP Generic Traps
- Configuring SNMP Auth-Traps
- Configuring SNMP Enterprise Traps
- Configuring SNMP Reload-Enable
- Configuring an SNMP Trap Host
- Configuring SNMP Trap Source

### SNMP Agent Configuration Quick Start

Table 5-1 provides a quick overview of the steps required to configure the CSS as an SNMP agent. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI command, see the sections following Table 5-1.

***Table 5-1 Quick Start for Defining the CSS as an SNMP Agent***

| Task and Command Example |
|---|
| **1.** Define the SNMP community strings for each access type, read-only (for a GET action) or read-write (for a GET and SET action). This step is required for using SNMP on the CSS.<br>`(config)# snmp community public read-only`<br>`(config)# snmp community private read-write` |
| **2.** (Optional) Provide the SNMP contact name.<br>`(config)# snmp contact “fred n mandy”` |

*Table 5-1 Quick Start for Defining the CSS as an SNMP Agent (continued)*

| Task and Command Example |
|---|
| **3.** (Optional) Provide an SNMP contact location.<br>`(config)# snmp location "Operations"` |
| **4.** (Optional) Provide the SNMP device name.<br>`(config)# snmp name "arrowpoint.com"` |
| **5.** Assign trap receivers and SNMP community (required if configuring SNMP traps). You can specify a maximum of five trap hosts. By default, all traps are disabled.The **snmp trap-host** IP address corresponds to the SNMP host configured to receive traps. The community information provided at the end of the **trap-host** command is included in the trap and can be used by the management station to filter incoming traps.<br>`(config)# snmp trap-host 172.16.3.6 trap`<br>`(config)# snmp trap-host 172.16.8.4 trap` |
| **6.** (Optional) Turn on generic traps. This step is required if you plan to use authentication failure traps.<br>`(config)# snmp trap-type generic` |
| **7.** (Optional) Turn on authentication failure traps. This step requires that you turn on generic traps. See step 6. An authentication failure occurs if an unauthorized SNMP manager sends an invalid or incorrect community name to an SNMP agent. If an authentication failure occurs, the agent sends an authentication trap to the trap host (or hosts depending on how many trap hosts are configured).<br>`(config)# snmp auth-traps` |
| **8.** (Optional) Enable global enterprise traps. This step is required if you plan to use enterprise traps.<br>`(config)# snmp trap-type enterprise`<br>Enable a specific enterprise trap type. For example, you can set a trap to notify the trap host of failed login attempts. Login failure traps provide the username and source IP address of the person who failed to log in.<br>`(config)# snmp trap-type enterprise login-failure` |

*Table 5-1 Quick Start for Defining the CSS as an SNMP Agent (continued)*

| Task and Command Example |
|---|
| **9.** (Optional) Configure the trap host for reload enable ability. Reload enable allows a management station with the proper WRITE community privilege to reboot the CSS.<br>`(config)# snmp reload-enable 100` |
| **10.** (Optional) Configure special enterprise trap thresholds to notify the trap host of Denial of Service (DoS) attacks on your system. For example, you can set a trap threshold to notify the trap host of DoS attacks with illegal addresses, either source or destination.<br>`(config)# snmp trap-type enterprise dos-illegal-attack trap-threshold 1` |

The following running-configuration example shows the results of entering the commands in Table 5-1.

```
!************************** GLOBAL **************************
  snmp trap-type enterprise

  snmp community techpubs read-write
  snmp contact "fred n mandy"
  snmp location "Operations"
  snmp name "arrowpoint.com"
  snmp trap-host 172.16.3.6 trap
  snmp trap-host 172.16.8.4 trap
  snmp trap-type generic
  snmp auth-traps
  snmp trap-type enterprise login-failure
  snmp reload-enable 100
  snmp trap-type enterprise dos-illegal-attack trap-threshold 1
```

## Configuring an SNMP Community

To set or modify SNMP community names and access privileges, use the **snmp community** command. You may specify a maximum of five communities.

**Caution** You must define the community strings for each access type (read-only or read-write) before you use SNMP on the CSS. The CSS is inaccessible until you specify a read community string.

The syntax for this global configuration mode command is:

**snmp community** *community_name* [**read-only**|**read-write**]

The variables and options for this command are:

- *community_name* - The SNMP community name for this system. Enter an unquoted text string with no space and a maximum of 12 characters.
- **read-only** - Allows read-only access for this community.
- **read-write** - Allows read-write access for this community.

For example:

```
(config)# snmp community sqa read-write
```

To remove a community name, enter:

```
(config)# no snmp community sqa
```

## Configuring an SNMP Contact

To set or modify the contact name for the SNMP system, use the **snmp contact** command. You can specify only one contact name. The syntax for this global configuration mode command is:

**snmp contact** *"contact_name"*

Enter the contact name as a quoted text string with a maximum of 255 characters including spaces. You can also include information on how to contact the person; for example, a phone number or e-mail address.

For example:

```
(config)# snmp contact "Fred N. Mandy"
```

To remove the specified SNMP contact name and reset it to the default of "Cisco Systems, Content Network Systems", enter:

```
(config)# no snmp contact
```

## Configuring an SNMP Location

To set or modify the SNMP system location, use the **snmp location** command. You can specify only one location. The syntax for this global configuration mode command is:

**snmp location** *"location"*

Enter the location as the physical location of the system. Enter a quoted text string with a maximum of 255 characters.

For example:

```
(config)# snmp location "sqa_lab1"
```

To remove the specified SNMP system location and reset it to the default of "Customer Premises," enter:

```
(config)# no snmp location
```

## Configuring an SNMP Name

To set or modify the SNMP name for this system, use the **snmp name** command. You can specify only one name. The syntax for this global configuration mode command is:

**snmp name** *"name"*

Enter the SNMP name as the unique name assigned to a system by the administrator. Enter a quoted text string with a maximum of 255 characters. The standard name convention is the system's fully qualified domain name (for example, sqa@arrowpoint.com).

For example:

```
(config)# snmp name "sqa@arrowpoint.com"
```

To remove the SNMP name for a system and reset it to the default of "Support", enter:

```
(config)# no snmp name
```

## Configuring an SNMP Trap Host

To set or modify the SNMP host to receive traps from a CSS, use the **snmp trap-host** command. You can specify a maximum of five hosts. The syntax for this global configuration mode command is:

**snmp trap-host** *ip_or_host community_name* {**snmpv2**}

The variables and option for this command are:

- *ip_or_host* - The IP address or host name of an SNMP host that has been configured to receive traps. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1) or in mnemonic host-name format (for example, myhost.mydomain.com).
- *community_name* - The community name to use when sending traps to the specified SNMP host. Enter an unquoted text string with no spaces and a maximum of 12 characters.
- **snmpv2** - Specifies that traps be sent in SNMPv2 format.

For example:

```
(config)# snmp trap-host 172.16.3.6 sqalab1 snmpv2
```

To remove a specified trap host, enter:

```
(config)# no snmp trap-host 172.16.3.6
```

## Configuring SNMP Trap Source

To set the Agent-Address field in the traps generated by the CSS, use the **snmp trap-source** command. This command applies only to SNMPv1 and does not apply to SNMPv2C. The syntax of this global configuration mode command is:

**snmp trap-source** [**egress-port**|**management** |**specified** *source_ip_address*]

The options and variable for this command are:

- **egress-port** - Uses the VLAN circuit IP address configured on the egress port to set the Agent-Address field of the trap. You do not need to enter an IP address because the address is determined dynamically by the CSS.
- **management** - Uses the Ethernet management port IP address to set the Agent-Address field of the trap. This option is available only on a Cisco 11500 series CSS.
- **specified** *source_ip_address* - Allows you to enter the IP address to be used in the Agent-Address field of the traps. Enter the IP address in dotted-decimal notation (for example, 192.168.11.1).

For example:

```
(config)# snmp trap-source egress-port
```

To return SNMP source traps to the default of the management port IP address, enter:

```
(config)# no snmp trap-source
```

## Configuring SNMP Generic Traps

By default, SNMP generic trap types are disabled on the CSS. Use the **snmp trap-type generic** command to enable SNMP generic trap types. The generic SNMP traps consist of cold start, warm start, link down, link up, and authentication failure.

**Note** For details on SNMP traps (and associated MIB objects) loaded as part of the CSS software, see the "CSS SNMP Traps" section.

For example:

```
(config)# snmp trap-type generic
```

To disable a generic trap, enter:

```
(config)# no snmp trap-type generic
```

**Note** Note that the CSS sends only SNMP v1 trap types.

## Configuring SNMP Auth-Traps

By default, generation of SNMP authentication traps is disabled on the CSS. Use the **snmp auth-traps** command to enable SNMP authentication traps. The CSS generates these traps when an SNMP management station attempts to access your system with invalid community names.

**Note** Enable SNMP generic traps using the **snmp trap-type generic** command. This command must be run before the system can generate authentication traps. For details, see the "Configuring SNMP Generic Traps" section.

**Note** For details on SNMP traps (and associated MIB objects) loaded as part of the CSS software, see the "CSS SNMP Traps" section.

For example, enter both of the following commands:

```
(config)# snmp trap-type generic
(config)# snmp auth-traps
```

To disable generation of authentication traps, enter:

```
(config)# no snmp auth-traps
```

## Configuring SNMP Enterprise Traps

By default, SNMP enterprise traps are disabled on the CSS. Use the **snmp trap-type enterprise** command to enable SNMP enterprise trap types. Once you enable enterprise traps, you can then configure the CSS to generate enterprise traps when:

- Denial of Service attack events occur
- A login fails
- A CSS service, power supply, or reporter transitions state
- A module is inserted into a powered-on CSS chassis
- An Inter-Switch Communications (ISC) LifeTick failure message occurs

Use the **no** form of the **snmp trap-type enterprise** command to prevent the CSS from generating a trap when a specific condition occurs.

**Note** Note that the CSS sends only SNMP v1 trap types.

For details on SNMP traps (and associated MIB objects) loaded as part of the CSS software, see the "CSS SNMP Traps" section. For information on configuring Denial of Service enterprise traps, see the "Configuring Denial of Service (DoS)" section.

The syntax for this global configuration mode command is:

**snmp trap-type enterprise** {*dos_attack_type* {**trap-threshold** *threshold_value*}|**chmgr-module-transition**|**chmgr-ps-transition** |**isc-lifetick-failure**|**isc-state-transition**|**login-failure**|**reload**|**redundancy-transition** |**reporter-transition**|**service-transition**}

The options for this command are as follows:

- **snmp trap-type enterprise** - Enables enterprise traps. You must enable enterprise traps before you configure an enterprise trap option.
- *dos_attack_type* - (Optional) Generates SNMP enterprise traps when a Denial of Service (DoS) attack event occurs. One trap is generated each second when the number of attacks during that second exceeds the threshold for the configured DoS attack type. See the "Configuring Denial of Service (DoS)" section for details.
- **trap-threshold** *threshold_value* - (Optional) Overrides a default trap threshold. For the *threshold_value*, enter a number from 1 to 65535. See the "Configuring Denial of Service (DoS)" section for details.
- **chmgr-module-transition** - Generates SNMP enterprise traps if a module (for example, SCM, FEM, GEM) is inserted into or removed from a powered-on CSS 11503 or CSS 11506.
- **chmgr-ps-transition** - Generates SNMP enterprise traps when the CSS 11503 or CSS 11506 power supply changes state (powered off or on, or removed from the CSS).
- **isc-lifetick-failure** - Generates SNMP enterprise traps when an ISC LifeTick failure message occurs on a Cisco 11500 series CSS. A LifeTick message occurs four times a second between ports in an Adaptive Session Redundancy (ASR) configuration. If a port does not receive a LifeTick message within one second from its corresponding port due to a software or hardware failure, an ISC LifeTick failure message occurs.
- **isc-state-transition** - In an ASR configuration with dual Inter-Switch Communication (ISC) links, generates SNMP enterprise traps when the active ISC link goes down and the standby link becomes active. For more information about ASR and ISC, refer to the *Cisco Content Services Switch Redundancy Configuration Guide*.
- **login-failure** - Generates SNMP enterprise traps when a CSS login failure occurs. The CSS also generates an alert-level log message.
- **reload** - Generates SNMP enterprise traps when a CSS reboot occurs. The CSS also generates a trap when a reboot is initiated directly through SNMP.
- **redundancy-transition** - Generates SNMP enterprise traps when the CSS redundancy transitions state.

- **reporter-transition** - Generates SNMP enterprise traps when the CSS reporter transitions state. A trap is generated when the reporter is activated or suspended, or the VRID peering virtual routers or critical phy interfaces change state.
- **service-transition** - Generates SNMP enterprise traps when a CSS service transitions state. A trap is generated when a service fails and when a failed service resumes proper operation.

For example, to enable an SNMP enterprise trap when a CSS login failure occurs, enter:

```
(config)# snmp trap-type enterprise
(config)# snmp trap-type enterprise login-failure
```

To disable all enterprise traps, enter:

```
(config)# no snmp trap-type enterprise
```

To disable a specific enabled enterprise trap, use the **no** form of the **snmp trap-type enterprise** command. For example, to prevent the CSS from generating traps when a power supply fails, enter:

```
(config)# no snmp trap-type enterprise chmgr-ps-transition
```

## Configuring SNMP Reload-Enable

To reboot the CSS using SNMP, use the **snmp reload-enable** command. The syntax and options for this global configuration mode command are:

- **snmp reload-enable** - Allows any SNMP write to the apSnmpExtReloadSet object to force a CSS reboot. The reload object, apSnmpExtReloadSet, is located at 1.3.6.1.4.1.9.9.368.1.22.7. You can find this object in the CSS Enterprise MIB, snmpext.mib.
- **snmp reload-enable** *reload_value* - Allows an SNMP write equal to the *reload_value* to force a CSS reboot.

**Note** This command requires that you enable enterprise traps first using the **snmp trap-type enterprise** command. See the "Configuring SNMP Enterprise Traps" section.

Enter the *reload_value* as the object used to control apSnmpExtReloadSet, providing the SNMP-based reboot. When the object is set to 0, an SNMP reboot is not allowed. When the object is set to a value from 1 to 2147483646, that value is written to the apSnmpExtReloadSet object to cause a reboot. When the *reload_value* object is set to 2147483647, a reboot may be caused with any write value to apSnmpExtReloadSet. For security purposes, this object always returns 0 when read.

**Note** When you reboot the CSS using the **snmp reload-enable** command, the CSS does not prompt you to save the running-config file or to verify that you want to reboot. Before you enter this command, be sure that you have saved any changes to your running-config file and that you want to reboot the CSS.

For example:

```
(config)# snmp reload-enable
```

To prevent users from rebooting the CSS using SNMP (default behavior), enter:

```
(config)# no snmp reload-enable
```

## Configuring Denial of Service (DoS)

You can configure special enterprise traps to notify the trap host of Denial of Service (DoS) attacks on your system. You can also use the CLI to display detailed information about DoS attacks and reset the DoS statistics for your CSS to zero.

Ensure you first enable SNMP enterprise traps using the **snmp trap-type enterprise** command before you configure the CSS to generate SNMP enterprise traps when a DoS attack event occurs. For information, see the "Configuring SNMP Enterprise Traps" section.

This section includes the following topics:

- DoS Quick Start
- Defining a DoS SNMP Trap-Type
- Displaying DoS Configurations

## DoS Quick Start

Table 5-2 provides a quick overview of the steps required to configure special enterprise traps to notify the trap host of DoS attacks on your system. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI command, see the sections following Table 5-2.

*Table 5-2 Denial of Service Configuration Quick Start*

| Task and Command Example |
|---|
| **1.** Enable enterprise traps if not already enabled.<br>`(config)# snmp trap-type enterprise` |
| **2.** Set the trap threshold to notify the trap host of DoS attacks with illegal addresses, either source or destination.<br>`(config)# snmp trap-type enterprise dos-illegal-attack trap-threshold 1` |
| **3.** Set the trap threshold to notify the trap host of DoS LAND attacks.<br>`(config)# snmp trap-type enterprise dos-land-attack trap-threshold 1` |
| **4.** Set the trap threshold to notify the trap host of DoS smurf attacks.<br>`(config)# snmp trap-type enterprise dos-smurf-attack trap-threshold 1` |
| **5.** Set the trap threshold to notify the trap host of DoS SYN attacks.<br>`(config)# snmp trap-type enterprise dos-syn-attack trap-threshold 10` |
| **6.** Display information about DoS attacks.<br>`(config)# show dos summary`<br>`(config)# show dos` |
| **7.** Reset the DoS statistics for a CSS to zero, as required.<br>`(config)# zero dos statistics` |

The following running-configuration example shows the results of entering the commands in Table 5-2.

```
!************************** GLOBAL **************************
  snmp trap-type enterprise

  snmp community techpubs read-write
  snmp contact "fred n mandy"
  snmp location "Operations"
  snmp name "arrowpoint.com"
  snmp trap-host 172.16.3.6 trap
  snmp trap-host 172.16.8.4 trap
  snmp trap-type generic
  snmp auth-traps
  snmp trap-type enterprise login-failure
  snmp reload-enable 100
  snmp trap-type enterprise dos-illegal-attack trap-threshold 1
  snmp trap-type enterprise dos-land-attack trap-threshold 1
  snmp trap-type enterprise dos-smurf-attack trap-threshold 1
  snmp trap-type enterprise dos-syn-attack trap-threshold 10
  snmp trap-type enterprise dos-illegal-attack trap-threshold 1
```

## Defining a DoS SNMP Trap-Type

To enable the CSS to generate SNMP enterprise traps when a DoS attack event occurs, use the **snmp trap-type enterprise** command. One trap is generated each second when the number of attacks during that second exceeds the threshold for the configured DoS attack type. For details on SNMP traps (and associated MIB objects) loaded as part of the CSS software, see the "CSS SNMP Traps" section.

Ensure you first enable SNMP enterprise traps using the **snmp trap-type enterprise** command before you configure the CSS to generate SNMP enterprise traps when a DoS attack event occurs. For information, see the "Configuring SNMP Enterprise Traps" section.

The syntax for this global configuration mode command is:

**snmp trap-type enterprise** *dos_attack_type* {**trap-threshold** *threshold_value*}

The *dos_attack_type* variable is the type of DoS attack event to trap. The options include:

- **dos-illegal-attack** - Generates traps for illegal addresses, either source or destination. Illegal addresses are loopback source addresses, broadcast source addresses, loopback destination addresses, multicast source addresses, or source addresses that you own. The default trap threshold for this type of attack is 1 per second.
- **dos-land-attack** - Generates traps for packets that have identical source and destination addresses. The default trap threshold for this type of attack is 1 per second.
- **dos-smurf-attack** - Generates traps when the number of pings with a broadcast destination address exceeds the threshold value. The default trap threshold for this type of attack is 1 per second.
- **dos-syn-attack** - Generates traps when the number of TCP connections that are initiated by a source, but not followed with an acknowledgment (ACK) frame to complete the 3-way TCP handshake, exceeds the threshold value. The default trap threshold for this type of attack is 10 per second.

Use the **trap-threshold** option to override a default trap threshold. For the *threshold_value*, enter a number from 1 to 65535.

For example, to enable the CSS to generate traps for packets that have identical source and destination addresses, enter:

```
(config)# snmp trap-type enterprise dos-land-attack
```

To prevent the CSS from generating DoS attack event traps, enter:

```
(config)# no snmp trap-type enterprise dos_attack_type
```

## Displaying DoS Configurations

Use the **show dos** command to display detailed information about DoS attacks on each CSS Session Processor (SP). The **show dos** command displays the following information:

- The total number of attacks since booting the CSS
- The types of attacks and the maximum number of these attacks per second
- The first and last occurrence of an attack
- The source and destination IP addresses

A CSS can display a maximum of 50 of the most recent attack events for each SP. For example:

- A CSS 11501 with one SP can display a maximum of 50 events.
- A CSS 11503 with a maximum of three SPs can display a maximum of 150 events.
- A CSS 11506 with a maximum of six SPs can display a maximum of 300 events.

If multiple attacks occur with the same DoS type and source and destination address, an attempt is made to merge them as one event. This merging of events reduces the number of displayed events.

Use the **show dos summary** command to display a summary of information about DoS attacks.

For example:

```
(config)# show dos summary
```

Table 5-3 describes the fields in the **show dos** command output.

*Table 5-3 Field Descriptions for the show dos Command*

| Field | Description |
|---|---|
| Total Attacks | The total number of DoS attacks detected since the CSS was booted. The type of attacks that are listed along with their number of occurrences are:<br>• **SYN Attacks** - TCP connections that are initiated by a source but are not followed with an ACK frame to complete the three way TCP handshake<br>• **LAND Attacks** - Packets that have identical source and destination addresses<br>• **Zero Port Attacks** - Frames that contain source or destination TCP or UDP ports equal to zero<br>**Note** Older SmartBits software may send frames containing source or destination ports equal to zero. The CSS logs them as DoS attacks and drops these frames.<br>• **Illegal Src Attacks** - Illegal source addresses<br>• **Illegal Dst Attacks** - Illegal destination addresses<br>• **Smurf Attacks** - Pings with a broadcast destination address |
| First Attack Detected | The first time a DoS attack was detected. |
| Last Attack Detected | The last time a DoS attack was detected. |

*Table 5-3 Field Descriptions for the show dos Command (continued)*

| Field | Description |
|---|---|
| Maximum per second | The maximum number of events per second. Use the maximum events per second information to set SNMP trap threshold values. The maximum number of events per second is the maximum for each SP.<br>• For a CSS 11506, which may have up to six SPs, the maximum rate per second may be as high as six times the value appearing in this field.<br>• For a CSS 11503, which may have up to three SPs, the maximum rate per second may be as high as three times the value appearing in this field. |
| DoS Attack Event | Details for each detected attack event, up to a maximum of 50 events per SP. |
| First Attack | The first time the attack event occurred. |
| Last Attack | The last time the attack event occurred. |
| Source/Destination Address | The source and destination addresses for the attack event. |
| Event Type | The type of event. |
| Total Attacks | The total number of attack occurrences for the event. |

## Resetting DoS Statistics

To set the Denial of Service (DoS) statistics for the CSS to zero, use the **zero dos statistics** command in any mode. This command sets the values of the DoS statistics in the fields of the **show dos** command to zero. For more information about the **show dos** command, see the "Displaying DoS Configurations" section.

## Displaying the SNMP Configuration

After you configure SNMP, display the SNMP configuration. For example:

```
(config)# show running-config global
```

Refer to Chapter 1, Managing the CSS Software, for details on the **show running-config** command and its output.

## Managing SNMP on the CSS

This section describes the activities that you need to perform to manage SNMP on the CSS. This section includes the following topics:

- Enabling SNMP Manager Access to the CSS
- Using the CSS to Look Up MIB Objects
- Reading Logs
- Setting RMON Alarms

### Enabling SNMP Manager Access to the CSS

By default, the CSS enables SNMP access to its command base. You must first create community strings using the **snmp community** command before you can use SNMP in the CSS. See the "Configuring an SNMP Community" section for details.

**Note** SNMP is not a secure network environment. Do not use SNMP by itself to provide security for your network.

## Using the CSS to Look Up MIB Objects

To look up a MIB object, including the variables that make up the object, perform the following steps:

1. Access global configuration mode by entering:

   ```
   # config
   ```

2. Access rmon-alarm mode by entering:

   ```
   (config)# rmon-alarm index_number
   ```

   where *index_number* is the RMON alarm index. The RMON alarm index identifies the alarm to the CSS. Refer to Chapter 6, Configuring Remote Monitoring (RMON), for information on RMON.

3. Display the MIB object by entering:

   ```
   (config-rmonalarm[1])# lookup object
   ```

   where *object* is the name of the MIB object.

   You can look up a specific object, or you can use the question mark (?) character as a wildcard to help you complete your request.

For example, suppose you want to look up a MIB object but you are not sure of its exact name. You already know that the MIB you want is part of the apFlowMgrExt group of objects. In this case, specify the **lookup** command with the question mark (**?**) character, as shown.

```
(config-rmonalarm[1])# lookup apFlowMgrExt?

apFlowMgrExtDoSAttackEventType
apFlowMgrExtDoSAttackEventCount
apFlowMgrExtDoSAttackIndex
apFlowMgrExtDosTotalSmurfAttacks
apFlowMgrExtDosTotalIllegalSourceAttacks
apFlowMgrExtDosTotalZeroPortAttacks
apFlowMgrExtDosTotalLandAttacks
apFlowMgrExtDosTotalSynAttacks
apFlowMgrExtDosTotalAttacks
apFlowMgrExtIdleTimer
apFlowMgrExtPortIdleValue
apFlowMgrExtPortIdle
apFlowMgrExtReserveCleanTimer
apFlowMgrExtPermanentPort4
apFlowMgrExtPermanentPort3
```

```
apFlowMgrExtPermanentPort2
apFlowMgrExtPermanentPort1
apFlowMgrExtFlowTraceDuration
apFlowMgrExtFlowTraceMaxFileSize
apFlowMgrExtFlowTraceState
```

The previous example shows that using the question mark (?) character as a wildcard returns information about the apFlowMgrExt MIB object. You can also enter the **lookup** command on the exact MIB you want and view its description without using the question mark (?) character. For example:

```
(config-rmonalarm[1])# lookup apFlowMgrExtDOSAttackEventCount

ASN Name:            apFlowMgrExtDOSAttackEventCount
MIB:                 flowmgrext
Object Identifier:    1.3.6.1.4.1.9.9.368.1.36.27.1.6
Argument Type:       Integer
Range:               0-4294967295
Description:
   This is the number of times this DoS attack had occurred.
```

You can also display a list of all the Enterprise MIBs by using the **lookup** command without any MIB object names, as shown in the following example:

```
(config-rmonalarm[1])# lookup ?
```

The **lookup** command omits MIB objects of type *string* and *MAC address*.

## Useful MIB Information

Table 5-4 lists some of the MIB groups that provide useful information about the CSS.

*Table 5-4 CSS MIB Information*

| MIB Name | Description |
|---|---|
| RFC 1398 | Ethernet statistics |
| RFC 1493 | Bridge information |
| RFC 1757 | RMON statistics |
| svcExt.mib | Service variables (including TCP connections) |
| cntExt.mib | Content rule variables (including frame statistics) |

*Table 5-4 CSS MIB Information (continued)*

| MIB Name | Description |
|---|---|
| ownExt.mib | Owner statistics (including frame and bytes counts) |
| cntsvcExt.mib | Services per content rule statistics (including frames, bytes, hits) |
| chassis MgrExt | Provides useful information about the CSS chassis and it allows you to correlate the slot number and port number to the ifIndex number |

## Reading Logs

The traplog file contains all of the traps, both generic and enterprise, that have occurred. The network device writes to the traplog file about whether or not the SNMP trap configuration is enabled.

When a traplog file reaches its maximum size (50 MB for a hard disk-based CSS, 10 MB for a flash disk-based CSS), the CSS renames the traplog file to traplog.prev as a backup file and starts a new traplog file. The CSS overwrites the backup traplog file when it renames the traplog file. Each time the CSS reboots, it continues to use the existing traplog file until it reaches its maximum size.

Use the **show log** command to show the trap log since the last CSS reboot. For example:

```
# show log traplog
```

By default, the following events generate level critical-2 messages:

- Link Up
- Link Down
- Cold Start
- Warm Start
- Service Down
- Service Suspended

All other SNMP traps generate level notice-5 messages by default.

## Setting RMON Alarms

An RMON alarm allows you to monitor a MIB object for a desired transitory state. Refer to Chapter 6, Configuring Remote Monitoring (RMON), for information about commands available in the RMON alarm mode.

## CSS SNMP Traps

Table 5-5 and Table 5-6 list the SNMP v1 and SNMP v2C traps, respectively, supported by the CSS.

*Table 5-5 SNMP v1 Traps*

| Name/MIB | Enterprise Object ID (OID) | Generic | Specific | Parameters |
|---|---|---|---|---|
| coldStart | <sysObjectID> | 0 | 0 | —————————————— |
| warmStart | <sysObjectID | 1 | 0 | —————————————— |
| linkDown | <sysObjectID> | 2 | 0 | ifIndex 1.3.6.1.2.1.2.2.1.1<br>ifOperStatus 1.3.6.1.2.1.2.2.1.8<br>ifAdminStatus 1.3.6.1.2.1.2.2.1.7 |
| linkUp | <sysObjectID> | 3 | 0 | ifIndex 1.3.6.1.2.1.2.2.1.1<br>ifOperStatus 1.3.6.1.2.1.2.2.1.8<br>ifAdminStatus 1.3.6.1.2.1.2.2.1.7 |
| authenticationFailure | <sysObjectID> | 4 | 0 | —————————————— |
| egpNeighborLoss | <sysObjectID> | 5 | 0 | —————————————— |

*Table 5-5 SNMP v1 Traps (continued)*

| Name/MIB | Enterprise Object ID (OID) | Generic | Specific | Parameters |
|---|---|---|---|---|
| apFlowMgrExtDosSynTrap (flowMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.36.1 | 6 | 1 | apFlowMgrExtDOSAttackEventString 1.3.6.1.4.1.9.9.368.1.36.28.1.8<br>apFlowMgrExtDOSAttackEventIntervalCount 1.3.6.1.4.1.9.9.368.1.36.28.1.9<br>apFlowMgrExtDOSAttackEventCount 1.3.6.1.4.1.9.9.368.1.36.28.1.6 |
| apFlowMgrExtDosLandTrap (flowMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.36.1 | 6 | 2 | apFlowMgrExtDOSAttackEventString 1.3.6.1.4.1.9.9.368.1.36.28.1.8<br>apFlowMgrExtDOSAttackEventIntervalCount 1.3.6.1.4.1.9.9.368.1.36.28.1.9<br>apFlowMgrExtDOSAttackEventCount 1.3.6.1.4.1.9.9.368.1.36.28.1.6 |
| apFlowMgrExtDosIllegalTrap (flowMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.36.1 | 6 | 3 | apFlowMgrExtDOSAttackEventString 1.3.6.1.4.1.9.9.368.1.36.28.1.8<br>apFlowMgrExtDOSAttackEventIntervalCount 1.3.6.1.4.1.9.9.368.1.36.28.1.9<br>apFlowMgrExtDOSAttackEventCount 1.3.6.1.4.1.9.9.368.1.36.28.1.6 |
| apFlowMgrExtDosSmurfTrap (flowMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.36.1 | 6 | 5 | apFlowMgrExtDOSAttackEventString 1.3.6.1.4.1.9.9.368.1.36.28.1.8<br>apFlowMgrExtDOSAttackEventIntervalCount 1.3.6.1.4.1.9.9.368.1.36.28.1.9<br>apFlowMgrExtDOSAttackEventCount 1.3.6.1.4.1.9.9.368.1.36.28.1.6 |
| apIpv4RedundancyTrap (apIpv4.mib) | 1.3.6.1.4.1.9.9.368.1.9.1.1 | 6 | 1 | apIpv4TrapEventText 1.3.6.1.4.1.9.9.368.1.9.34.0<br>apIpv4RedundancyState 1.3.6.1.4.1.9.9.368.1.9.19.0<br>apIpv4RedundancyIf 1.3.6.1.4.1.9.9.368.1.9.20.0<br>apIpv4RedundancyMaster 1.3.6.1.4.1.9.9.368.1.9.21.0 |

*Table 5-5 SNMP v1 Traps (continued)*

| Name/MIB | Enterprise Object ID (OID) | Generic | Specific | Parameters |
|---|---|---|---|---|
| apIpv4RedundancyState Transition (apIpv4Redundancy.mib) | 1.3.6.1.4.1.9.9.368.1.9.8.1 | 6 | 1 | apIpv4RedundancyEventText 1.3.6.1.4.1.9.9.368.1.9.8.9.0<br>apIpv4RedundancyVRIntAddr 1.3.6.1.4.1.9.9.368.1.9.8.2.1.2<br>apIpv4RedundancyVRID 1.3.6.1.4.1.9.9.368.1.9.8.2.1.1<br>apIpv4RedundancyVROperState 1.3.6.1.4.1.9.9.368.1.9.8.2.1.13<br>apIpv4RedundancyVRFailReason 1.3.6.1.4.1.9.9.368.1.9.8.2.1.14<br>apIpv4RedundancyVRMasterIP 1.3.6.1.4.1.9.9.368.1.9.8.2.1.8 |
| apSnmpExtReloadTrap (snmpExt.mib) | 1.3.6.1.4.1.9.9.368.1.22.1 | 6 | 1 | apSnmpExtTrapEventText 1.3.6.1.4.1.9.9.368.1.22.27.0 |
| apSnmpExtReporterTraps (snmpExt.mib) | 1.3.6.1.4.1.9.9.368.1.68.1 | 6 | 1 | apReporterTrapEventText 1.3.6.1.4.1.9.9.368.1.68.7.0 |
| apSvcTransitionTrap (svcExt.mib) | 1.3.6.1.4.1.9.9.368.1.15.1 | 6 | 1 | apSvcTrapEventText 1.3.6.1.4.1.9.9.368.1.15.10.0 |
| apTermSessLoginFailureTrap (terminalMgmt.mib) | 1.3.6.1.4.1.9.9.368.1.11.1 | 6 | 1 | apTermSessLoginFailureInfo 1.3.6.1.4.1.9.9.368.1.11.3.0 |
| apChassisMgrExtPsTrap (chassisMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.34.1 | 6 | 1 | apChassisMgrExtTrapPsEventText 1.3.6.1.4.1.9.9.368.1.34.24.0 |
| apChassisMgrModuleTrap (chassisMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.34.1 | 6 | 2 | apChassisMgrExtTrapModuleEventText 1.3.6.1.4.1.9.9.368.1.34.25.0 |
| apEnetISCLifetickTrap (enetExt.mib) | 1.3.6.1.4.1.9.9.368.1.39.1 | 6 | 1 | apEnetISCLifetickEventText 1.3.6.1.4.1.9.9.368.1.39.8.0 |
| apEnetISCStateTransition (enetExt.mib) | 1.3.6.1.4.1.9.9.368.1.39.1 | 6 | 2 | apEnetISCEventText 1.3.6.1.4.1.9.9.368.1.39.13.0<br>apEnetISCState 1.3.6.1.4.1.9.9.368.1.39.10.0<br>apEnetISCPortOneFailureReason 1.3.6.1.4.1.9.9.368.1.39.11.0<br>apEnetISCPortTwoFailureReason 1.3.6.1.4.1.9.9.368.1.39.12.0 |

*Table 5-6 SNMP v2C Traps*

| Name/MIB | Enterprise Object ID (OID) | Parameters |
|---|---|---|
| coldStart | 1.3.6.1.6.3.1.1.5.1 | —————————— |
| warmStart | 1.3.6.1.6.3.1.1.5.2 | —————————— |
| linkDown | 1.3.6.1.6.3.1.1.5.3 | ifIndex 1.3.6.1.2.1.2.2.1.1<br>ifOperStatus 1.3.6.1.2.1.2.2.1.8<br>ifAdminStatus 1.3.6.1.2.1.2.2.1.7 |
| linkUp | 1.3.6.1.6.3.1.1.5.4 | ifIndex 1.3.6.1.2.1.2.2.1.1<br>ifOperStatus 1.3.6.1.2.1.2.2.1.8<br>ifAdminStatus 1.3.6.1.2.1.2.2.1.7 |
| authenticationFailure | 1.3.6.1.6.3.1.1.5.5 | —————————— |
| egpNeighborLoss | 1.3.6.1.6.3.1.1.5.6 | —————————— |
| apFlowMgrExtDosSynTrap (flowMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.36.1.0.1 | apFlowMgrExtDOSAttackEventString 1.3.6.1.4.1.9.9.368.1.36.28.1.8<br>apFlowMgrExtDOSAttackEventIntervalCount 1.3.6.1.4.1.9.9.368.1.36.28.1.9<br>apFlowMgrExtDOSAttackEventCount 1.3.6.1.4.1.9.9.368.1.36.28.1.6 |
| apFlowMgrExtDosLandTrap (flowMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.36.1.0.2 | apFlowMgrExtDOSAttackEventString 1.3.6.1.4.1.9.9.368.1.36.28.1.8<br>apFlowMgrExtDOSAttackEventIntervalCount 1.3.6.1.4.1.9.9.368.1.36.28.1.9<br>apFlowMgrExtDOSAttackEventCount 1.3.6.1.4.1.9.9.368.1.36.28.1.6 |

*Table 5-6 SNMP v2C Traps (continued)*

| Name/MIB | Enterprise Object ID (OID) | Parameters |
|---|---|---|
| apFlowMgrExtDosIllegalTrap (flowMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.36.1.0.3 | apFlowMgrExtDOSAttackEventString 1.3.6.1.4.1.9.9.368.1.36.28.1.8<br>apFlowMgrExtDOSAttackEventInterval Count 1.3.6.1.4.1.9.9.368.1.36.28.1.9<br>apFlowMgrExtDOSAttackEventCount 1.3.6.1.4.1.9.9.368.1.36.28.1.6 |
| apFlowMgrExtDosSmurfTrap (flowMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.36.1.0.5 | apFlowMgrExtDOSAttackEventString 1.3.6.1.4.1.9.9.368.1.36.28.1.8<br>apFlowMgrExtDOSAttackEventInterval Count 1.3.6.1.4.1.9.9.368.1.36.28.1.9<br>apFlowMgrExtDOSAttackEventCount 1.3.6.1.4.1.9.9.368.1.36.28.1.6 |
| apIpv4RedundancyTrap (apIpv4.mib) | 1.3.6.1.4.1.9.9.368.1.9.1.1.0.1 | apIpv4TrapEventText 1.3.6.1.4.1.9.9.368.1.9.34.0<br>apIpv4RedundancyState 1.3.6.1.4.1.9.9.368.1.9.19.0<br>apIpv4RedundancyIf 1.3.6.1.4.1.9.9.368.1.9.20.0<br>apIpv4RedundancyMaster 1.3.6.1.4.1.9.9.368.1.9.21.0 |
| apIpv4RedundancyState Transition (apIpv4Redundancy.mib) | 1.3.6.1.4.1.9.9.368.1.9.8.1.0.1 | apIpv4RedundancyEventText 1.3.6.1.4.1.9.9.368.1.9.8.9.0<br>apIpv4RedundancyVRIntAddr 1.3.6.1.4.1.9.9.368.1.9.8.2.1.2<br>apIpv4RedundancyVRID 1.3.6.1.4.1.9.9.368.1.9.8.2.1.1<br>apIpv4RedundancyVROperState 1.3.6.1.4.1.9.9.368.1.9.8.2.1.13<br>apIpv4RedundancyVRFailReason 1.3.6.1.4.1.9.9.368.1.9.8.2.1.14<br>apIpv4RedundancyVRMasterIP 1.3.6.1.4.1.9.9.368.1.9.8.2.1.8 |
| apSnmpExtReloadTrap (snmpExt.mib) | 1.3.6.1.4.1.9.9.368.1.22.1.0.1 | apSnmpExtTrapEventText 1.3.6.1.4.1.9.9.368.1.22.27.0 |

*Table 5-6 SNMP v2C Traps (continued)*

| Name/MIB | Enterprise Object ID (OID) | Parameters |
|---|---|---|
| apSnmpExtReporterTraps (snmpExt.mib) | 1.3.6.1.4.1.9.9.368.1.68.1.0.1 | apReporterTrapEventText 1.3.6.1.4.1.9.9.368.1.68.7.0 |
| apSvcTransitionTrap (svcExt.mib) | 1.3.6.1.4.1.9.9.368.1.15.1.0.1 | apSvcTrapEventText 1.3.6.1.4.1.9.9.368.1.15.10.0 |
| apTermSessLoginFailureTrap (terminalMgmt.mib) | 1.3.6.1.4.1.9.9.368.1.11.1.0.1 | apTermSessLoginFailureInfo 1.3.6.1.4.1.9.9.368.1.11.3.0 |
| apChassisMgrExtPsTrap (chassisMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.34.1.0.1 | apChassisMgrExtTrapPsEventText 1.3.6.1.4.1.9.9.368.1.34.24.0 |
| apChassisMgrModuleTrap (chassisMgrExt.mib) | 1.3.6.1.4.1.9.9.368.1.34.1.0.2 | apChassisMgrExtTrapModuleEventText 1.3.6.1.4.1.9.9.368.1.34.25.0 |
| apEnetISCLifetickTrap (enetExt.mib) | 1.3.6.1.4.1.9.9.368.1.39.1.0.1 | apEnetISCLifetickEventText 1.3.6.1.4.1.9.9.368.1.39.8.0 |
| apEnetISCStateTransition (enetExt.mib) | 1.3.6.1.4.1.9.9.368.1.39.1.0.2 | apEnetISCEventText 1.3.6.1.4.1.9.9.368.1.39.13.0<br>apEnetISCState 1.3.6.1.4.1.9.9.368.1.39.10.0<br>apEnetISCPortOneFailureReason 1.3.6.1.4.1.9.9.368.1.39.11.0<br>apEnetISCPortTwoFailureReason 1.3.6.1.4.1.9.9.368.1.39.12.0 |

## CSS MIBs

Table 5-7 describes the CSS MIB objects directly under the CSS Enterprise MIB (Object Identifier 1.3.6.1.4.1.9.9.368). The MIBs listed in this table are a representation of the CSS content-specific MIB objects. To find out how you can look up object information, see the "Using the CSS to Look Up MIB Objects" section.

*Table 5-7 MIB Branches Under the CSS Enterprise MIB*

| MIB Filename | MIB Module Description | Related CLI Commands |
|---|---|---|
| aclExt.mib (OID 1.3.6.1.4.1.9.9.368.1.23) | The CSS access control list (ACL) clause table | `(config-acl)# ?` |
| ap64Stats.mib (OID 1.3.6.1.4.1.9.9.368.1.44) | The 64-bit statistical aggregation of RMON (RFC1757), MIB-II (RFC 1213) and EtherErrors (RFC 1398) | `# show rmon ?` `# show mibii ?` `# show ether-errors ?` |
| cisco-apent.mib (OID 1.3.6.1.4.1.9.9.368.1) | CSS Enterprise MIB branch hierarchy | —————————— |
| apIpv4.mib (OID 1.3.6.1.4.1.9.9.368.1.9.1) | MIB support for IPv4 global information, box-to-box redundancy | `(config)# ip ?` |
| apIpv4Arp.mib (OID 1.3.6.1.4.1.9.9.368.1.9.4) | MIB support for IPv4 ARP | `(config)# arp ?` |
| apIpv4Dns.mib (OID 1.3.6.1.4.1.9.9.368.1.9.7) | MIB support for IPv4 DNS resolver configuration | `(config)# dns ?` |
| apIpv4Host.mib (OID 1.3.6.1.4.1.9.9.368.1.9.6) | MIB support for IPv4 host table | `(config)# host ?` |
| apIpv4Interface.mib (OID 1.3.6.1.4.1.9.9.368.1.9.2) | MIB support for IPv4 interfaces, box-to-box redundancy | `(config-ip)# ?` |
| apIpv4Ospf.mib (OID 1.3.6.1.4.1.9.9.368.1.9.3.2) | MIB support for the Open Shortest Path First (OSPF) protocol | `(config)# ospf ?` |
| apIpv4Redundancy.mib (OID 1.3.6.1.4.1.9.9.368.1.9.8) | MIB support for IPv4 redundancy | `(config-ip)# redundancy ?` |
| apIpv4Rip.mib (OID 1.3.6.1.4.1.9.9.368.1.9.3.1) | MIB support for the Routing Information Protocol (RIP) | `(config-ip)# rip ?` |

*Table 5-7 MIB Branches Under the CSS Enterprise MIB (continued)*

| MIB Filename | MIB Module Description | Related CLI Commands |
|---|---|---|
| apIpv4Sntp.mib (OID 1.3.6.1.4.1.9.9.368.1.9.9) | MIB support for the Simple Network Time Protocol (SNTP) | `(config)#` **`sntp ?`** |
| apIpv4StaticRoutes.mib (OID 1.3.6.1.4.1.9.9.368.1.9.5) | MIB support for IPv4 static routes | `(config)#` **`ip route ?`** |
| appExt.mib (OID 1.3.6.1.4.1.9.9.368.1.32) | MIB support for Application Peering Protocol (APP) configurations | `(config)#` **`app ?`** |
| boomClientExt.mib (OID 1.3.6.1.4.1.9.9.368.1.62) | Configuration and monitoring of Content Routing Agent (CRA) parameters | `(config)#` **`dns-boomerang client ?`** |
| bootExt.mib (OID 1.3.6.1.4.1.9.9.368.1.31) | MIB support for system boot administration | `(config-boot)#` **`?`** |
| bridgeExt.mib (OID 1.3.6.1.4.1.9.9.368.1.14) | Configuration and monitoring of bridge-related parameters | `(config)#` **`bridge ?`** |
| cappUdpExt.mib (OID 1.3.6.1.4.1.9.9.368.1.52) | Application Peering Protocol-User Datagram Protocol (APP-UDP) global statistical information and security configuration settings | `(config)#` **`app-udp ?`** |
| cctExt.mib (OID 1.3.6.1.4.1.9.9.368.1.29) | CSS circuit information, box-to-box redundancy | `(config)#` **`circuit ?`** |
| chassisMgrExt.mib (OID 1.3.6.1.4.1.9.9.368.1.34) | MIB for the CSS chassis manager | `#` **`show chassis ?`** |
| cntdnsExt.mib (OID 1.3.6.1.4.1.9.9.368.1.41) | Content rule Domain Name Service (DNS) statistics | `(config)#` **`dns hotlist ?`** |
| cntExt.mib (OID 1.3.6.1.4.1.9.9.368.1.16) | Content rule table | `(config-owner-content)#` **`?`** |
| cnthotExt.mib (OID 1.3.6.1.4.1.9.9.368.1.35) | Content rule hot list | `(config-owner-content)#` **`hotlist ?`** |

*Table 5-7 MIB Branches Under the CSS Enterprise MIB (continued)*

| MIB Filename | MIB Module Description | Related CLI Commands |
|---|---|---|
| cntLctSvcExt.mib (OID 1.3.6.1.4.1.9.9.368.1.67) | MIB support for location cookie. | `(config-owner-content)#` **`add location-service ?`** `(config-owner-content)#` **`remove location-service ?`** |
| cntsvcExt.mib (OID 1.3.6.1.4.1.9.9.368.1.18) | Monitoring of services attached to content rules | `(config-owner-content)#` **`add service ?`** `(config-owner-content)#` **`remove service ?`** |
| csaExt.mib (OID 1.3.6.1.4.1.9.9.368.1.59) | Configuration and monitoring of Client Side Accelerator (CSA) parameters on a CSS | `(config)#` **`dns-server ?`** |
| dfpExt.mib (OID 1.3.6.1.4.1.9.9.368.1.65) | MIB support for Dynamic Feedback Protocol (DFP) statistics and configuration | `(config)#` **`dfp ?`** |
| dnshotExt.mib (OID 1.3.6.1.4.1.9.9.368.1.48) | DNS hot list | `(config)#` **`domain hotlist ?`** |
| dnsServerExt.mib (OID 1.3.6.1.4.1.9.9.368.1.40) | MIB support for DNS server | `(config)#` **`dns-server ?`** |
| domainCacheExt.mib (OID 1.3.6.1.4.1.9.9.368.1.60) | Configuration management for the domain cache on the CSA in the CSS | `(config)#` **`dns-server domain-cache ?`** |
| dqlExt.mib (OID 1.3.6.1.4.1.9.9.368.1.51) | Domain Qualifier Lists (DQLs) | `(config-dql [name])#` **`?`** |
| enetExt.mib (OID 1.3.6.1.4.1.9.9.368.1.39) | Configuration of the PHY state for Ethernet ports | `(config-interface)#` **`phy ?`** |
| eqlExt.mib (OID 1.3.6.1.4.1.9.9.368.1.42) | Extension Qualifier Lists (EQLs) | `(config-eql [name])#` |
| fileExt.mib (OID 1.3.6.1.4.1.9.9.368.1.61) | File extensions to support network management movement to/from the CSS, and to examine and modify the existing file structure | ——————————— |

*Table 5-7 MIB Branches Under the CSS Enterprise MIB (continued)*

| MIB Filename | MIB Module Description | Related CLI Commands |
|---|---|---|
| flowMgrExt.mib (OID 1.3.6.1.4.1.9.9.368.1.36) | MIB for the flow manager module | `(config)# `**`flow ?`** |
| ftpExt.mib (OID 1.3.6.1.4.1.9.9.368.1.30) | MIB support for File Transfer Protocol (FTP) transfer administration records | `(config)# `**`ftp-record ?`** |
| grpExt.mib (OID 1.3.6.1.4.1.9.9.368.1.17) | Configuration of all group-related parameters | `(config-group)# `**`?`** |
| grpsvcExt.mib (OID 1.3.6.1.4.1.9.9.368.1.19) | Groups attached to services | `(config-group)# `**`add service ?`** <br> `(config-group)# `**`remove service ?`** |
| httpExt.mib (OID 1.3.6.1.4.1.9.9.368.1.47) | MIB support for HTTP transfer administration records | — |
| kalExt.mib (OID 1.3.6.1.4.1.9.9.368.1.46) | Configuration of keepalive mode | `(config-keepalive)# `**`?`** |
| logExt.mib (OID 1.3.6.1.4.1.9.9.368.1.20) | CSS logging functionality | `(config)# `**`logging ?`** |
| nqlExt.mib (OID 1.3.6.1.4.1.9.9.368.1.50) | Describes the CSS network qualifier lists (NQLs) | `(config-nql [name])# `**`?`** |
| ownExt.mib (OID 1.3.6.1.4.1.9.9.368.1.25) | Web host owner information | `(config-owner)# `**`?`** |
| plucExt.mib (OID 1.3.6.1.4.1.9.9.368.1.56) | Proximity Lookup Client functionality | `(config)# `**`proximity cache ?`** |
| probeRttExt.mib (OID 1.3.6.1.4.1.9.9.368.1.55) | Tiered Proximity Service RTT Probe Module functionality | `(config)# `**`proximity probe rtt ?`** |
| proxDbExt.mib (OID 1.3.6.1.4.1.9.9.368.1.54) | Tiered Proximity Database (PDB) functionality; contains all configuration, statistic, and metric objects | `(config)# `**`proximity db ?`** |
| publishExt.mib (OID 1.3.6.1.4.1.9.9.368.1.57) | Publisher and subscriber services | `(config-service)# `**`publisher ?`** |

*Table 5-7 MIB Branches Under the CSS Enterprise MIB (continued)*

| MIB Filename | MIB Module Description | Related CLI Commands |
|---|---|---|
| qosExt.mib (OID 1.3.6.1.4.1.9.9.368.1.28) | CSS MIB module quality of service (QoS) class definitions (the QoS class of this known piece of content) | ————————— |
| radiusClientExt.mib (OID 1.3.6.1.4.1.9.9.368.1.12) | CSS extensions to the client side of the Remote Access Dial-in User Service (RADIUS) authentication protocol | `(config)# `**`radius-server ?`** |
| ciscoCssReporter.mib (OID 1.3.6.1.4.1.9.9.368.1.68.1) | MIB support for the CSS reporter, critical phy interfaces related to the CSS reporter, and VRID peering virtual routers related to the CSS reporter | `(config)# `**`reporter ?`**<br>`(config-reporter)# `**`phy ?`**<br>`(config-reporter)# `**`vrid ?`** |
| schedExt.mib (OID 1.3.6.1.4.1.9.9.368.1.45) | MIB support for CLI command scheduler records | `(config)# `**`cmd-scheduler ?`** |
| securityMgrExt.mib (OID 1.3.6.1.4.1.9.9.368.1.13) | CSS MIB objects for the network security manager | `(config)# `**`username ?`** |
| snmpExt.mib (OID 1.3.6.1.4.1.9.9.368.1.22) | SNMP traps and communities | `(config)# `**`snmp ?`** |
| sshdExt.mib (OID 1.3.6.1.4.1.9.9.368.1.43) | MIB support for the Secure Shell Daemon server (SSHD) | `(config)# `**`sshd ?`** |
| sslExt.mib (OID 1.3.6.1.4.1.9.9.368.1.63) | MIB support for Secure Sockets Layer (SSL), file associations for SSL certificates, and keys for the SSL Acceleration Module | `(config)# `**`ssl cert ?`**<br>`(config)# `**`ssl rsakey ?`**<br>`(config)# `**`ssl dakey ?`**<br>`(config)# `**`ssl dhparam ?`** |
| ssllExt.mib (OID 1.3.6.1.4.1.9.9.368.1.64) | MIB support for SSL proxy list elements and cipher suite objects for the SSL Acceleration Module | `(ssl-proxy-list[name])# `**`element ?`** |
| subscribeExt.mib (OID 1.3.6.1.4.1.9.9.368.1.58) | CSS Enterprise subscriber | `(config-service)# `**`subscriber ?`** |

*Table 5-7 MIB Branches Under the CSS Enterprise MIB (continued)*

| MIB Filename | MIB Module Description | Related CLI Commands |
|---|---|---|
| svcExt.mib (OID 1.3.6.1.4.1.9.9.368.1.15) | Configuration and monitoring of all service-related parameters | `(config-service)#` **`?`** |
| tacacsExt.mib (OID 1.3.6.1.4.1.9.9.368.1.66) | CSS extensions to the client side of the Terminal Access Controller Access Control System (TACACS+) authentication protocol | `(config)#` **`tacacs-server ?`** |
| tagExt.mib (OID 1.3.6.1.4.1.9.9.368.1.53) | Content tag lists | `(config)#` **`header-field-group ?`** |
| terminalMgmt.mib (OID 1.3.6.1.4.1.9.9.368.1.11) | MIB support for terminal options | `#` **`terminal ?`** <br> `#` **`restrict ?`** |
| urqlExt.mib (OID 1.3.6.1.4.1.9.9.368.1.49) | Uniform resource locator qualifier lists (URQL) | `(config-urql [name])#` **`?`** |

## Where to Go Next

Chapter 6, Configuring Remote Monitoring (RMON), describes how to describes how to configure RMON on the CSS.

# Configuring Remote Monitoring (RMON)

This chapter provides information on configuring the Remote Monitoring (RMON) features of your CSS. Information in this chapter applies to all CSS models, except where noted.

This chapter contains the following major sections:

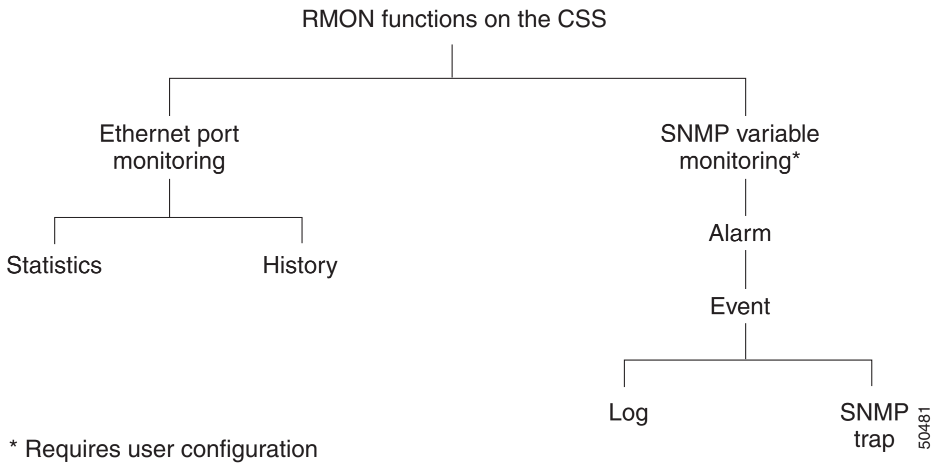
## RMON Overview

RMON allows you to remotely monitor and analyze the activity of packets on CSS Ethernet ports. RMON also allows alarm configuration for monitoring MIB objects, and the event configuration to notify you of these alarm conditions. For detailed information about RMON and its MIB objects, refer to RFC 1757.

The version of RMON provided on the CSS is a subset of the RMON-1 groups (Figure 6-1). The CSS supports the following groups:

- Group 1 - (Statistics) Provides data about all Ethernet ports on a CSS. You cannot configure RMON statistics. You can only view them.
- Group 2 - (History) Provides data about the Ethernet ports over an historical period. Histories are preconfigured for each port. You can configure additional port histories.
- Group 3 - (Alarm) Allows you to create an alarm and configure the conditions, based on a MIB object, to trigger an alarm when changes are detected.
- Group 9 - (Event) Allows you to create an event and configure the event action that is to be performed when an associated alarm occurs.

***Figure 6-1 Supported RMON Functions on the CSS***


RMON functions on the CSS
Ethernet port monitoring
SNMP variable monitoring*
Statistics
History
Alarm
Event
Log
SNMP trap
50481
* Requires user configuration

## RMON Configuration Considerations

Consider the following points before you implement RMON functionality on your CSS:

- You can configure an RMON event, alarm, and history. You cannot configure CSS attributes for RMON statistics. Statistics for the ports are viewable only by using the **show rmon** command.
- You cannot change the configuration for an RMON history after you activate it. If you need to change the RMON history configuration after activation, you must delete it first and then recreate the RMON history with the necessary changes. You can change your RMON history configuration at any time before you activate it.
- You must assign an RMON event to an RMON alarm before the alarm can be activated. The event must exist and must be activated before it can be assigned to an RMON alarm.
- RMON histories are preconfigured for each Ethernet port. Though these histories cannot be deleted or modified, you can add history entries for a port. For more information on the preconfigured histories and adding more history entries, see the "Configuring an RMON History" section.

## Configuring an RMON Event

An RMON event is the action that occurs when an associated RMON alarm is triggered. When an alarm event occurs, it can be configured to generate a log event, a trap to an SNMP network management station, or both. For information on viewing alarm events in log files, see the "Viewing Events in a Log File" section. Refer to Chapter 5, Configuring Simple Network Management Protocol (SNMP), for information on configuring SNMP on your CSS.

The following sections describe how to configure an RMON event.

- RMON Event Configuration Quick Start
- Creating an Index for an RMON Event
- Deleting an RMON Event Index
- Setting the RMON Event Attributes
- Activating an RMON Event
- Suspending an RMON Alarm

## RMON Event Configuration Quick Start

Table 6-1 provides a quick overview of the steps required to configure the attributes for an RMON event. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI command, see the sections following Table 6-1. For information on configuring an alarm and associating this event to an alarm, see the "Configuring an RMON Alarm" section.

*Table 6-1 RMON Event Configuration Quick Start*

| Steps and Possible Settings |
|---|
| **1.** Create an RMON event index from global configuration mode. Enter an integer from 1 to 65534.<br>`(config)# rmon-event 1`<br>`Create Event <1>, [y/n]:y` |
| **2.** Assign an existing SNMP community for this event. The specified *community_name* is the name of an SNMP community configured using the **snmp trap-host** command. This step is required only if the traps are sent to an SNMP network management station.<br>`(config-rmonevent[1])# community moonbase_alpha` |
| **3.** Provide a description for the event. Enter a quoted text string with a maximum of 127 characters including spaces.<br>`(config-rmonevent[1])# description "This event occurs when service connections exceed 100"` |
| **4.** Assign the owner who defined and is using the resources of the event. Enter a quoted string with a maximum of 127 characters including spaces. You must define the owner before you can activate the event.<br>`(config-rmonevent[1])# owner "Boston Tech Lab"` |
| **5.** Specify the type of event notification. The type determines where the notification is sent.<br>`(config-rmonevent[1])# type log-and-trap` |
| **6.** Activate the event.<br>`(config-rmonevent[1])# active` |

The following running-configuration example shows the results of entering the commands in Table 6-1.

```
!************************ RMON EVENT ************************
rmon-event 1
  community moonbase_alpha
  description "This event occurs when service connections exceed 100"
  owner "Boston Tech Lab"
  type log-and-trap
  active
```

## Creating an Index for an RMON Event

The RMON event index identifies the event to the CSS. This index allows you to assign specific configuration attributes to the RMON event. When you create an RMON event index, you access the configuration mode for that event automatically.

Use the **rmon-event** *index* command to create an event index. You can access this command from any configuration mode except the boot and RMON alarm configuration modes. The *index* is a number from 1 to 65534.

**Note** The RMON event index 65535 is administratively predefined and cannot be modified. If you enter this index number, a message similar to the following appears: `%% Index internally used. Administrative control not allowed.`

For example, to create an RMON event with an identifier of 1, access global configuration mode and enter:

```
(config)# rmon-event 1
```

To view a list of existing RMON event configuration identifiers, enter:

```
(config)# rmon-event ?
```

After you create the index for the event, the prompt changes to (config-rmonevent[1]). Define the event as described in the "Setting the RMON Event Attributes" section.

### Modifying the Attributes for an Existing RMON Event Index

Use the **suspend** command to deactivate the RMON event and make attribute changes.

## Deleting an RMON Event Index

If you have an active RMON event index that you no longer need, use the **no rmon-event** command. This command is available in the RMON alarm, RMON event, RMON history, and global configuration modes.

To delete RMON event 1 and its configuration, enter:

```
(config)# no rmon-event 1
Delete Event <1>,[y/n]:y
```

## Setting the RMON Event Attributes

After you create an RMON event index, access RMON event configuration mode for the event identifier and set its attributes. This section includes the following topics:

- Defining an Event Community
- Describing an Event
- Assigning an Owner
- Defining the Notification of an Event

After you set the attributes, activate the event as described in the "Activating an RMON Event" section.

If an RMON event is activated and you want to make modifications to certain event attributes, you must first suspend the RMON event (as described in the "Suspending an RMON Event" section). Ensure the RMON event is not assigned to an RMON alarm.

### Defining an Event Community

When an alarm event occurs and the event is configured to send an SNMP trap, the CSS sends the trap to the trap host with the specified community. If no community is specified the CSS automatically uses the default event community of "public".

Use the **community** *community_name* command to define a community for an unactivated event. The *community_name* variable is the name of the SNMP community you configured using the **snmp trap-host** command (refer to Chapter 5, Configuring Simple Network Management Protocol (SNMP)).

To view a list of currently configured community strings (configured using the **snmp trap-host** command), enter **rmon-event community ?**

For example, to define the SNMP *moonbase_alpha* community for this event, enter:

```
(config-rmonevent[1])# community moonbase_alpha
```

To reset the community back to public, enter:

```
(config-rmonevent[1])# no community
```

## Describing an Event

When an alarm event occurs, the CSS sends a description with the event notification. Because a description is not generated automatically, you must provide one. Use the **description** *"description"* command to provide a description. The *description* variable is the description for the RMON event. Enter a quoted text string with a maximum of 127 characters.

To provide a description for the event, enter:

```
(config-rmonevent[1])# description "This event occurs when service
connections exceed 100"
```

To remove the description from the event, enter:

```
(config-rmonevent[1])# no description
```

## Assigning an Owner

You must define the entity who configured this RMON event and is using the resources assigned to it. Use the **owner** "*owner_name*" command to define the owner. The *owner_name* variable is a quoted text string with a maximum of 127 characters. The owner for the event must be the same as the owner for the alarm.

To define the owner named Boston Tech Lab, enter:

```
(config-rmonevent[1])# owner "Boston Tech Lab"
```

To remove the owner of the RMON event, enter:

```
(config-rmonevent[1])# no owner
```

You must reassign an owner before you can reactivate the RMON event.

## Defining the Notification of an Event

When an RMON event occurs, the event type determines where the CSS sends the event notification.

- A log event type designates that the event notification is made in a CSS log location (for example, CSS disk log file or session). For information on viewing log files, see the "Viewing Events in a Log File" section.

  To define the event as a log type (default), enter:

  ```
  (config-rmonevent[1])# type log
  ```

- A trap event type designates that a trap is sent to a SNMP network management station. To define the event as a trap type, enter:

  ```
  (config-rmonevent[1])# type trap
  ```

**Note** When you want the event to send a trap to a network management station, you need to configure SNMP. Refer to Chapter 5, Configuring Simple Network Management Protocol (SNMP), for information on SNMP.

- You can also designate that the event type is both log and trap. To define the event as both log and trap types, enter:

  ```
  (config-rmonevent[1])# type log-and-trap
  ```

To reset the RMON event type back to log, enter:

```
(config-rmonevent[1])# no type
```

## Activating an RMON Event

After you configure the event attributes, activate the event. However, before you can activate an event, you must specify the owner of the event as described in the "Assigning an Owner" section.

To activate the event, enter:

```
(config-rmonevent[1])# active
```

Note the following when activating an RMON event, once an RMON event is activated and you want to make modifications to certain event attributes, you must first suspend the RMON event. Ensure the RMON event is not assigned to an RMON alarm.

## Suspending an RMON Event

Suspending an RMON event deactivates it, allowing you to make changes to its configuration settings. Use the **suspend** command to suspend an event.

When you suspend an RMON event, ensure that the event is not assigned to an RMON alarm.

For example:

```
(config-rmonevent[1])# suspend
```

## Configuring an RMON Alarm

An RMON alarm allows you to monitor a MIB object for a desired transitory state. An alarm periodically takes samples of the object's value and compares them to the configured thresholds.

RMON allows you to configure two types of sampling, absolute and delta:

- Absolute sampling compares the sample value directly to the threshold. This sampling is similar to a gauge, recording values that go up or down.
- Delta sampling subtracts the current sample value from the last sample taken and then compares the difference to the threshold. This sampling is similar to a counter, recording a value that is constantly increasing.

When a sample value crosses an alarm threshold, an associated event is generated. To limit the number of generated events, only one event is generated when a threshold is crossed. The CSS does not generate additional events until an opposite threshold is crossed. For example, when a rising threshold is crossed, one event is generated. The next event occurs only when a falling threshold is crossed.

When you associate an event to an alarm and an alarm occurs, the event defines the action the CSS takes when an alarm occurs. For more information on events, see the "Configuring an RMON Event" section.

Figure 6-2 is an example of absolute sampling.

*Figure 6-2 Example of Absolute Sampling*

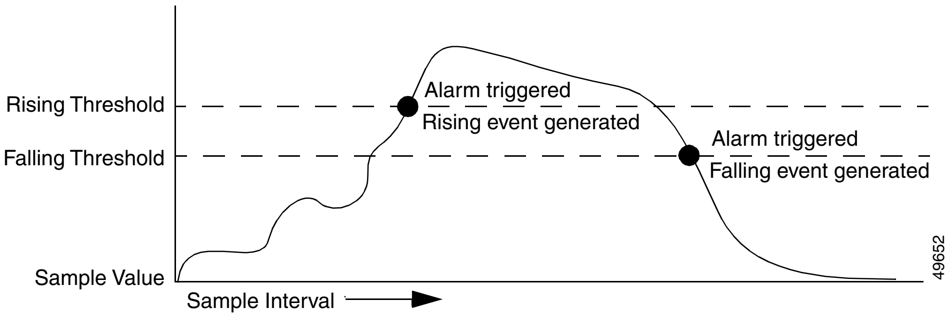
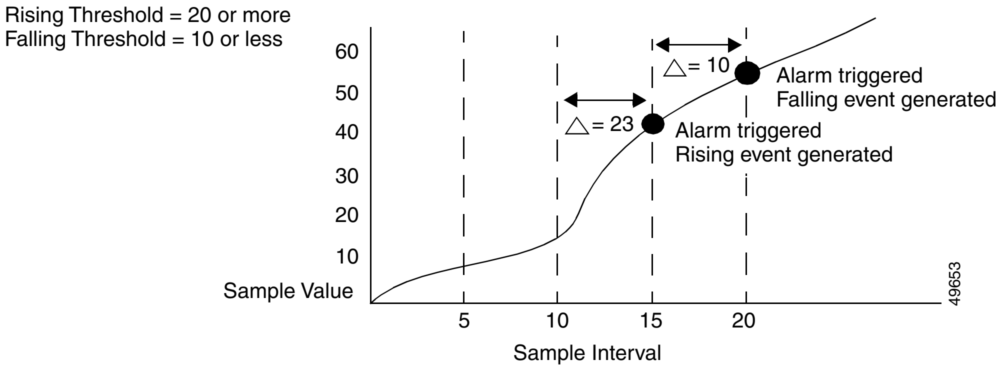


Figure 6-3 is an example of delta sampling.

*Figure 6-3 Example of Delta Sampling*



This section includes the following topics:

- RMON Alarm Configuration Quick Start
- Creating an Index for an RMON Alarm
- Deleting an RMON Alarm Index
- Setting the RMON Alarm Attributes

- Activating an RMON Alarm
- Suspending an RMON Alarm

## RMON Alarm Configuration Quick Start

Table 6-2 provides a quick overview of the steps required to configure the attributes for an RMON alarm. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI command, see the sections following Table 6-2.

*Table 6-2 RMON Alarm Configuration Quick Start*

| Steps and Possible Settings |
|---|
| **1.** Create an RMON alarm index from global configuration mode. Enter an integer from 1 to 65534.<br>`(config)# rmon-alarm 1`<br>`Create Alarm <1>, [y/n]:y` |
| **2.** Assign the owner who defined and is using the resources of the alarm. Enter a quoted string with a maximum of 127 characters including spaces. The owner must be the same as the owner for the event.<br>`(config-rmonalarm[1])# owner "Boston Tech Lab"` |
| **3.** Define the MIB object for the sample variable. For example, for the current number of connections for this service, enter **`apSvcConnections`**. To see a list of objects, use the **sample-variable ?** command. For detailed information about an object, use the **lookup** command.<br>`(config-rmonalarm[1])# sample-variable apSvcConnections` |
| **4.** Define the sampling type. The options are **absolute** or **delta** (default).<br>`(config-rmonalarm[1])# sample-type absolute` |
| **5.** Define the startup alarm type. The options are **falling**, **rising** (default), or **rising-and-falling**.<br>`(config-rmonalarm[1])# startup-type rising-and-falling` |
| **6.** Define the rising threshold. Enter an integer from 0 (default) to 4294967295.<br>`(config-rmonalarm[1])# rising-threshold 100` |

*Table 6-2 RMON Alarm Configuration Quick Start (continued)*

| Steps and Possible Settings |
|---|
| **7.** Associate the rising event with an existing RMON event. Enter an integer from 0 to 65534. If you enter 0, no event is generated.<br>`(config-rmonalarm[1])# rising-event 1` |
| **8.** Define the falling threshold. Enter an integer from 0 (default) to 4294967295.<br>`(config-rmonalarm[1])# falling-threshold 90` |
| **9.** Associate the falling event with an existing RMON event. Enter a number from 0 to 65534. If you enter 0, no event is generated.<br>`(config-rmonalarm[1])# falling-event 2` |
| **10.** Specify the sampling interval for the RMON alarm. The interval is in seconds. Enter an integer from 1 to 65535 (default is 300).<br>`(config-rmonalarm[1])# sample-interval 30` |
| **11.** Activate the alarm.<br>`(config-rmonalarm[1])# active` |

The following running-configuration example shows the results of entering the commands in Table 6-2.

```
!************************ RMON ALARM ************************
rmon-alarm 1
  owner “Boston Tech Lab”
  sample-variable apSvcConnections.
  sample-type absolute
  startup-type rising-and-falling
  rising-threshold 100
  rising-event 1
  falling-threshold 90
  falling-event 2
  sample-interval 30
  active
```

## Creating an Index for an RMON Alarm

The RMON alarm index identifies the alarm to the CSS. This index allows you to assign specific configuration attributes to the RMON alarm. When you create an RMON alarm index, you access the configuration mode for that alarm automatically.

Use the **rmon-alarm** *index* command to create an alarm index. You can access this command from any configuration mode except the boot and RMON history configuration modes. The *index* is an integer from 1 to 65534.

**Note** The RMON alarm index 65535 is administratively predefined and cannot be modified. If you enter this index number, a message similar to the following appears: `%% Index internally used. Administrative control not allowed.`

To create an RMON alarm with an identifier of 1, access global configuration mode and enter:

```
(config)# rmon-alarm 1
```

To see a list of existing RMON alarm configuration identifiers, enter **rmon-alarm ?**.

After you create the identifier for the alarm, the prompt changes to (`config-rmonalarm[1]`). Now you can define the alarm as described in the "Setting the RMON Alarm Attributes" section.

## Modifying the Attributes for an Existing RMON Alarm Index

Before you can modify attributes for an active RMON alarm, you must deactivate the alarm. Use the **suspend** command to deactivate the RMON alarm and make attribute changes.

For example:

```
(config-rmonalarm[1])# suspend
```

## Deleting an RMON Alarm Index

If you have an active RMON alarm index that you no longer need and want to delete it, use the **no rmon-alarm** command. This command is available in the RMON alarm, RMON event, RMON history, and global configuration modes.

For example, to delete RMON alarm 1 and its configuration, enter:

```
(config)# no rmon-alarm 1
Delete Alarm <1>,[y/n]:y
```

## Setting the RMON Alarm Attributes

After you create an RMON alarm index, access RMON alarm configuration mode for an existing inactive alarm identifier and set its attributes. This section includes the following topics:

- Assigning an Owner
- Finding and Defining a Sample Variable
- Defining the Absolute or Delta Sampling Method
- Defining a Rising Threshold and Rising Event
- Defining a Falling Threshold and Index
- Defining a Startup Alarm
- Defining the Sampling Interval

After you set all of the attributes, activate the alarm as described in the "Activating an RMON Alarm" section.

If an RMON alarm is activated and you want to make modifications to certain alarm attributes, you must first suspend the RMON alarm (as described in the "Suspending an RMON Alarm" section).

### Assigning an Owner

Define the owner who configured the RMON alarm and is using the resources assigned to the alarm. To define the owner, use the **owner** "*owner_name*" command. You must reassign an owner before you can reactivate the RMON alarm.

The *owner_name* variable is a quoted text string with a maximum of 127 characters. Enter the same name as the owner of the event.

To define the owner named Boston Tech Lab, enter:

```
(config-rmonalarm[1])# owner "Boston Tech Lab"
```

To remove the owner of the RMON alarm, enter:

```
(config-rmonalarm[1])# no owner
```

## Finding and Defining a Sample Variable

For an alarm condition, RMON samples a configured sample variable associated with a MIB object. MIB objects to consider include:

- svcExt.mib - Contains service objects (for example, apSvcConnections is the MIB object for the current number of TCP connections to this service).
- cntExt.mib - Contains content rule objects (for example, apCntHits is the MIB object for the total number of hits on this service for this content rule).

Refer to Chapter 5, Configuring Simple Network Management Protocol (SNMP), for information on CSS Enterprise MIBs.

Use the **lookup** command to look up a MIB object and view its description. For example, to view the description for the apSvcConnections object, enter:

```
(config-rmonalarm[1])# lookup apSvcConnections
ASN Name:            apSvcConnections
MIB:                 svcext
Object Identifier:   1.3.6.1.4.1.2467.1.15.2.1.20
Argument Type:       Integer
Range:               0-4294967295
Description:
   The current number of TCP connections to this service
```

Use the **sample-variable** *mib_object* command to specify the sample variable for this RMON alarm. For example, to define the apSvcConnections MIB object for the current number of service connections, enter:

```
(config-rmonalarm[1])# sample-variable apSvcConnections
```

To see a list of SNMP variables, use the **sample-variable ?** command. For example:

```
(config-rmonalarm[1])# sample-variable ?

        apSvcLoadInfoTimeout
        apSvcLoadSvcStatRptTimeout
        apSvcLoadEnable
        apSvcLoadDecayInterval
        apSvcLoadStepStatic
        apSvcLoadStepSize
        apSvcLoadThreshold
        ...
```

To remove the sample variable, enter:

```
(config-rmonalarm[1])# no sample-variable
```

## Defining the Absolute or Delta Sampling Method

When you configure an alarm, you can define the sampling method to compare the sample value of a MIB object to either:

- The configured threshold directly. This sampling is similar to a gauge, recording the value as it fluctuates up and down (see Figure 6-2).
- The previous sampling, and then their difference is compared to the configured threshold. This sampling is similar to a counter, recording the value that constantly increases (see Figure 6-3).

Absolute sampling compares the sample value to the configured threshold. For example, if you want to know when 30,000 service connections occur on the CSS during a sample interval, configure the apSvcConnections MIB object with absolute sampling. The apSvcConnections object is the current number of connections on a service. To define an absolute sampling, enter:

```
(config-rmonalarm[1])# sample-type absolute
```

Delta sampling (the default sampling method) compares the current sample value with the previous sample and compares their difference to the configured threshold. For example, if you want to know when the number of content rule hits increase by 100,000 as compared to its previous sampling, configure the apCntHits MIB object with delta sampling. apCntHits is an ever-increasing count of hits.

To define a delta sampling, enter:

```
(config-rmonalarm[1])# sample-type delta
```

To reset the sample type to delta sampling, enter:

```
(config-rmonalarm[1])# no sample-type
```

## Defining a Rising Threshold and Rising Event

If you want to be notified when a sampling is greater than or equal to a specific number, set a rising threshold and associate it with a configured RMON event.

You must create an RMON event before you can associate it with an alarm.

For a single rising alarm event to occur, a sampled value is greater than or equal to the rising threshold value, and the value at the last sampling interval is less than this threshold.

- Use the **rising-threshold** *rising_value* command to set the threshold for the alarm. The *rising_value* variable is the threshold for the rising sample type. Enter an integer from 0 (default) to 4294967295.

  To set the rising threshold value of 100, enter:

  ```
  (config-rmonalarm[1])# rising-threshold 100
  ```

  To reset the rising threshold to 0, enter:

  ```
  (config-rmonalarm[1])# no rising-threshold
  ```

- Use the **rising-event** *rising_index* command to associate a configured event to the RMON alarm when the sampled value exceeds the rising threshold value. The *rising_index* variable is the event index used when a rising threshold is crossed. Enter a previously created RMON event index (see the "Creating an Index for an RMON Event" section). If you enter 0, no event is generated.

  To associate the threshold to RMON event 1, enter:

  ```
  (config-rmonalarm[1])# rising-event 1
  ```

  To see a list of RMON events, enter:

  ```
  (config-rmonalarm[1])# rising-event ?
  ```

  To reset the rising event to 0 (no event is generated), enter:

  ```
  (config-rmonalarm[1])# no rising-event
  ```

## Defining a Falling Threshold and Index

If you want to be notified when a sampling is less than or equal to a specific number, set a falling threshold and associate it to a configured event.

**Note** You must create an RMON event before you can associate it with an alarm.

For a single falling alarm event to occur, a sampled value is less than or equal to the falling threshold value, and the value at the last sampling interval is greater than this threshold.

- Use the **falling-threshold** *falling_value* command to set the threshold for the alarm. The *falling_value* variable is the threshold for the falling sample type. Enter an integer from 0 (default) to 4294967295.

  To set the falling threshold value of 90, enter:

  ```
  (config-rmonalarm[1])# falling-threshold 90
  ```

  To reset the falling threshold to 0, enter:

  ```
  (config-rmonalarm[1])# no falling-threshold
  ```

- Use the **falling-event** *falling_index* command to associate a configured event to the RMON alarm when the sampled value exceeds the falling threshold value. The *falling_index* variable is the event index used when a falling threshold is crossed. Enter a previously created RMON event index (see the "Creating an Index for an RMON Event" section). If you enter 0, no event is generated.

  To associate the threshold to RMON event 2, enter:

  ```
  (config-rmonalarm[1])# falling-event 2
  ```

  To see a list of RMON events, enter:

  ```
  (config-rmonalarm[1])# falling-event ?
  ```

  To reset the falling event to 0, enter:

  ```
  (config-rmonalarm[1])# no falling-event
  ```

## Defining a Startup Alarm

A startup alarm allows the CSS to generate an alarm when the first sample triggers a falling threshold or rising threshold (default).

- A startup falling alarm occurs when the first sample is less than or equal to the falling threshold. To enable this alarm, enter:

  ```
  (config-rmonalarm[1])# startup-type falling
  ```

- A startup rising alarm occurs when the first sample is greater than or equal to the rising threshold. To enable this alarm, enter:

  ```
  (config-rmonalarm[1])# startup-type rising
  ```

- To enable an alarm when either a falling or rising threshold is triggered, enter:

  ```
  (config-rmonalarm[1])# startup-type rising-and-falling
  ```

To reset the startup alarm to a rising threshold alarm, enter:

```
(config-rmonalarm[1])# no startup-type
```

## Defining the Sampling Interval

The sampling interval is the time interval, in seconds, over which the data is sampled and compared with the rising and falling thresholds. Use the **sample-interval** *interval* command to specify the sampling interval for the RMON alarm. The *interval* variable is the number of seconds from 1 to 65535. The default is 300 seconds.

To enter a sampling interval of 60 seconds, enter:

```
(config-rmonalarm[1])# sample-interval 60
```

With delta sampling, set the sampling interval short enough so the sampled variable, which has a tendency to go up and down very fast, does not wrap during a single sampling period.

To reset the sample interval to 300, enter:

```
(config-rmonalarm[1])# no sample-interval
```

## Activating an RMON Alarm

After you configure the alarm attributes, you can activate the alarm. However, before you can activate an alarm, you must specify all attributes for the alarm.

To activate the alarm, enter:

```
(config-rmonalarm[1])# active
```

Note the following when activating an RMON alarm, once an RMON alarm is activated and you want to make modifications to certain alarm attributes, you must first suspend the RMON alarm.

## Suspending an RMON Alarm

Suspending an RMON alarm deactivates it, allowing you to make changes to its configuration settings. To suspend an alarm, use the **suspend** command.

For example:

```
(config-rmonalarm[1])# suspend
```

## Configuring an RMON History

You can configure the operation of the RMON history that periodically samples any CSS Ethernet port for statistical data. All ports are preconfigured with histories for 30-second and 30-minute intervals, and 50 buckets with one sample for each bucket. However, you can create additional histories for a specific port. The creation of an RMON history for a port allows you to configure the time interval to take the sample and the number of samples you want to save.

You can view the statistical information for the history by using the **show rmon-history** command. For more information about viewing the history, see the "Viewing History" section.

This section includes the following topics:

- RMON History Configuration Quick Start
- Creating an Index for an RMON History
- Deleting an RMON History Index
- Setting the RMON History Attributes
- Activating an RMON History Entry

## RMON History Configuration Quick Start

Table 6-3 provides a quick overview of the steps required to configure the attributes for an RMON history. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the options associated with the CLI command, see the sections following Table 6-3.

*Table 6-3 RMON History Configuration Quick Start*

| Steps and Possible Settings |
|---|
| **1.** Create an RMON history from global configuration mode. Enter an integer from 1 to 65535.<br>`(config)# rmon-history 5`<br>`Create History <5>, [y/n]:y` |
| **2.** Assign the owner who defined and is using the history resources. Enter a maximum of 32 characters.<br>`(config-rmonhistory[5])# owner "Boston_Tech_Lab"` |
| **3.** Define the data source object for the Ethernet port. The port is identified by an ifIndex data object identifier.<br>`(config-rmonhistory[5])# data-source ifIndex.3` |
| **4.** Define the time interval for the history. Enter a number from 1 to 3600 seconds. The default is 1800 seconds.<br>`(config-rmonhistory[5])# interval 60` |

*Table 6-3 RMON History Configuration Quick Start (continued)*

| Steps and Possible Settings |
|---|
| **5.** Define the bucket count for the interval. Enter an integer from 1 to 65535. The default is 50.<br>`(config-rmonhistory[5])# requested-buckets 25` |
| **6.** Activate the history.<br>`(config-rmonhistory[5])# active` |

The following running-configuration example shows the results of entering the commands in Table 6-3.

```
!*********************** RMON HISTORY ***********************
rmon-history 5
  owner "Boston_Tech_Lab"
  data-source ifIndex.3
  interval 60
  requested-buckets 25
  active
```

## Creating an Index for an RMON History

The RMON history index identifies the history to the CSS. The RMON history index allows you to assign specific configuration attributes to the RMON history index. When you create an RMON history index, you access the configuration mode for that history automatically.

To create a history index, use the **rmon-history** *index* command from any configuration mode except boot configuration mode. The *index* is an integer from 1 to 65534.

**Note** The RMON history index 65535 is administratively predefined and cannot be modified. If you enter this index number, a message similar to the following appears: `%% Index internally used. Administrative control not allowed.`

To create an RMON history identifier 5, access global configuration mode and enter:

```
(config)# rmon-history 5
```

To see a list of existing RMON history configuration identifiers, enter **rmon-history ?**.

After you create the identifier for the history, the prompt changes to (`config-rmonhistory[1]`). Now you can define the history as described in the "Setting the RMON History Attributes" section.

## Modifying the Attributes for an Existing RMON History Index

If create an RMON history index but have not activated it, you can modify the attributes of the history index (as described in the "Setting the RMON History Attributes" section). Once the RMON history index is activated, you cannot modify its attributes. You must delete the history index (see the "Deleting an RMON History Index" section), recreate it, and respecify the alarm index attributes.

## Deleting an RMON History Index

If you have an active RMON history index that requires changes to its attributes or you no longer need it, delete the RMON history index. Before you delete a history index that requires changes, note the settings for its attributes.

To delete an RMON history configuration identifier, use the **no rmon-history** command. This command is available in the RMON alarm, RMON event, RMON history, and global configuration modes.

For example, to delete RMON history 5 and its configuration, enter:

```
(config)# no rmon-history 5
Delete History <5>,[y/n]:y
```

After you delete the history identifier to change its attributes, recreate it as described in the "Creating an Index for an RMON History" section.

## Setting the RMON History Attributes

After you create an RMON history or access RMON history configuration mode for an existing inactive alarm, set the RMON history attributes. This section includes the following topics:

- Defining the Data Object
- Assigning an Owner
- Defining the Bucket Count
- Defining the Bucket Interval

After you set the attributes, activate the history as described in the "Activating an RMON History Entry" section.

### Defining the Data Object

After you create a history, you must associate it with a CSS Fast Ethernet or Gigabit Ethernet port. To define the data object, use the **data-source** *port* command. The *port* is identified by an ifIndex data object identifier. For example, if your CSS has 12 Ethernet ports, they have data object IDs of ifIndex.1 through ifIndex.12. The Ethernet management port has an ID of ifIndex.14.

To define Ethernet port 4, enter:

```
(config-rmonhistory[5])# data-source ifIndex.4
```

To see a list of data object IDs for all of the CSS Ethernet ports, enter:

```
(config-rmonhistory[5])# show interface
```

### Assigning an Owner

Define the owner who configured the RMON history and is using the resources assigned to it. Use the **owner** *owner_name* command to define the owner. The *owner_name* variable is a quoted text string with a maximum of 32 characters.

For example, to define an owner named Boston Tech Lab, enter:

```
(config-rmonhistory[5])# owner "Boston Tech Lab"
```

### Defining the Bucket Count

You can define a bucket count, which is the number of discrete sampling intervals over which data is saved for a history entry. Use the **requested-buckets** *count* command to define a bucket count. The *count* variable is an integer from 1 to 65535. The default is 50.

To define a bucket count of 25, enter:

```
(config-rmonhistory[5])# requested-buckets 25
```

### Defining the Bucket Interval

You can specify the time interval, in seconds, to take a bucket sample for an RMON history operation. Use the **interval** *value* command to set this interval. Enter an integer from 1 to 3600 seconds. The default is 1800 (30 minutes).

To define a time interval of 60 seconds, enter:

```
(config-rmonhistory[5])# interval 60
```

## Activating an RMON History Entry

After you configure the history attributes, you can activate the history for the port. Use the **active** command to activate an RMON history entry. Before you activate the history, make sure you finish configuring it and are satisfied with the RMON history settings. After you activate a history, you cannot modify its configuration settings. The only way to change the history is to delete it, and then recreate it.

Before activating this command, you must specify the owner of the RMON history entry.

To activate the history, enter:

```
(config-rmonhistory[5])# active
```

## Viewing RMON Information

RMON information includes:

- Ethernet port statistics and history data that you can view from the CSS through **show** commands.
- Alarm event notifications that are sent to log locations on the CSS or an SNMP network management station. Refer to Chapter 5, Configuring Simple Network Management Protocol (SNMP), for information on configuring SNMP on the CSS.

The following sections provide information on:

- Viewing Statistics
- Viewing History
- Viewing Events in a Log File

## Viewing Statistics

RMON statistics provide a summary of data received over the Fast Ethernet or Gigabit Ethernet ports. You can view RMON statistics either in a CSS CLI session through the **show rmon** command, the **show ether-errors** command (refer to the *Cisco Content Services Switch Routing and Bridging Configuration Guide*), or directly through an SNMP network management station by using ether-stats MIB objects (refer to RFC 1398).

The CSS **show rmon** command allows you to display the extended 64-bit RMON statistics for a specific Ethernet port or all Ethernet ports in the CSS. The CSS Enterprise ap64Stats MIB defines these statistics. You can also display the RFC 1757 32-bit statistics by adding the **-32** suffix to the **show rmon** command.

To display the RMON statistics for all ports in the CSS, enter:

```
# show rmon
```

To display the RFC 1757 32-bit statistics, enter:

```
# show rmon-32
```

To display the RMON statistics for a specified port in the CSS, enter:

```
# show rmon port_name
```

The *port_name* variable is the name of the physical port (for example, ethernet-4). Enter the *port_name* variable as a case-sensitive, unquoted text string.

To display the RFC 1757 32-bit statistics, enter **show rmon-32** p*ort_name*.

To see a list of ports, enter:

```
# show rmon ?
```

To display the extended RMON statistics for the Ethernet-4 port in the CSS, enter:

```
# show rmon ethernet-4
```

Table 6-4 lists and describes the fields in the **show rmon** command output.

***Table 6-4 Field Descriptions for the show rmon Command***

| Field | Description |
|---|---|
| Bytes | The total number of received bytes. |
| Packets | The total number of received packets (including bad packets, broadcast packets, and multicast packets). |
| Broadcast Packets | The total number of good received packets that were directed to the broadcast address. The number of broadcast packets does not include multicast packets. |
| Multicast Packets | The total number of good received packets that were directed to a multicast address. The number of multicast packets does not include packets directed to the broadcast address. |
| CRC Alignment Errors | The total number of packets received that had a length (excluding framing bits, but including FCS octets) from 64 and 1518 octets, but had an FCS Error, a bad Frame Check Sequence (FCS) with an integral number of octets, or an Alignment Error, a bad FCS with a non-integral number of octets. |
| Oversize Packets | The total number of received packets that were longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed. |

*Table 6-4 Field Descriptions for the show rmon Command (continued)*

| Field | Description |
|---|---|
| Undersize Packets | The total number of received packets that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well-formed. |
| Fragments | The total number of received packets that were less than 64 octets in length (excluding framing bits but including FCS octets) and had an FCS Error, a bad Frame Check Sequence (FCS) with an integral number of octets, or an Alignment Error, a bad FCS with a non-integral number of octets.<br><br>It is normal for fragment statistics to increment because the CSS counts both runts (which are normal occurrences due to collisions) and noise hits. |
| Drop Events | The total number of events in which packets were dropped by the probe due to lack of resources. This number is not necessarily the number of packets dropped; it is the number of times this condition has been detected. |
| Slobbers | An internal counter. This field is always zero. |
| Jabbers | The total number of packets received that were longer than 1518 octets (excluding framing bits, but including FCS octets), and had an FCS Error, a bad Frame Check Sequence (FCS) with an integral number of octets, or Alignment Error, a bad FCS with a non-integral number of octets.<br><br>This definition of jabber is different than the definition in IEEE-802.3, section 8.2.1.5, 10BASE5, and section 10.3.1.4, 10BASE2. These documents define jabber as the condition where any packet exceeds 20 ms. The allowed range to detect jabber is between 20 and 150 ms. |

*Table 6-4 Field Descriptions for the show rmon Command (continued)*

| Field | Description |
|---|---|
| Collisions | The best estimate of the total number of collisions on this Ethernet segment.<br><br>The returned value depends on the location of the RMON probe. Section 8.2.1.3, 10BASE-5, and section 10.3.1.3, 10BASE-2, of IEEE standard 802.3 states that a station must detect a collision, in the receive mode, if three or more stations are transmitting simultaneously. A repeater port must detect a collision when two or more stations are transmitting simultaneously. Thus, a probe placed on a repeater port might record more collisions than would a probe connected to a station on the same segment.<br><br>Probe location plays a much smaller role when considering 10BASE-T. IEEE standard 802.3 14.2.1.4, 10BASE-T, defines a collision as the simultaneous presence of signals on the DO and RD circuits (transmitting and receiving at the same time). A 10BASE-T station can detect collisions only when it is transmitting. Probes placed on a station and a repeater should report the same number of collisions.<br><br>Ideally, an RMON probe inside a repeater should report collisions between the repeater and one or more other hosts (transmit collisions as defined by IEEE 802.3k), plus receiver collisions observed on any coaxial segments to which the repeater is connected. |
| Packets (0-64)<br><br>Packets (65-127)<br><br>Packets (128-255)<br><br>Packets (256-511)<br><br>Packets (512-1023)<br><br>Packets (1024-1518) | The total number of received packets (including bad packets) that were between the following octets in length inclusive (excluding framing bits but including FCS octets):<br>• 0 to 64<br>• 65 to 127<br>• 128 to 255<br>• 256 to 511<br>• 512 to 1023<br>• 1024 to 1518 |

### Clearing RMON Statistics

Use the **clear statistics** *port_name* command to reset the RMON statistics on a CSS Ethernet port to zero. The *port_name* variable is the name of the physical port (for example, ethernet-4). Enter the *port_name* variable as a case-sensitive, unquoted text string.

To clear the statistics for Ethernet port 1, enter:

```
# clear statistics Ethernet-1
```

To see a list of ports, enter:

```
# clear statistics ?
```

**Note** When you reset RMON statistics on a CSS Ethernet port to zero, the Ethernet errors and MIB-II statistics for the port are also reset to zero.

## Viewing History

You can display the default and configured RMON history information for a specific Ethernet port or all Ethernet ports in the CSS. For information on configuring an RMON history, see the "Configuring an RMON History" section.

By default, the CSS maintains two tables of history statistics for each port. One table contains the last 50 samples at 30-second intervals. The other table contains 50 samples at 30-minute intervals. You cannot modify the configuration for these histories.

- To view the RMON history for all ports in the CSS, enter:

  ```
  # show rmon-history
  ```

- To display the RMON history for a specified port, enter:

  ```
  # show rmon-history port_name
  ```

  To see a list of ports in the CSS, enter:

  ```
  # show rmon-history ?
  ```

- To display the RMON history for a specified port and history index, enter:

```
# show rmon-history port_name history_index
```

  To view the history 5 for the Ethernet-4 port, enter:

```
# show rmon-history ethernet-4 5
```

  To see a list of history indexes associated with a specified port, enter:

```
# show rmon-history port_name ?
```

  To see a list of histories for the Ethernet-4 port, enter:

```
# show rmon-history ethernet-4 ?
```

Table 6-5 lists and describes the fields in the **show rmon-history** command output.

*Table 6-5 Field Descriptions for the show rmon-history Command*

| Field | Description |
|---|---|
| Owner | The owner who configured the entry and is using the resources assigned to it. |
| Start Time | The time when the bucket sampling started. |
| Interval | The time interval, in seconds, when RMON takes a bucket sample. |
| Buckets | The number of discrete sampling intervals over which data is saved for the history. |
| Time | The time that the sample was taken. |
| Sample | The number of the sample. |

*Table 6-5 Field Descriptions for the show rmon-history Command (continued)*

| Field | Description |
|---|---|
| Octets | The total number of octets of data (including those in bad packets) received on the network, excluding framing bits but including FCS octets.<br><br>You can use this object as a reasonable estimate of Ethernet utilization. If greater precision is desired, sample the Ethernet statistic packet and octet objects before and after a common interval. The differences in the sampled values are packets (Pkts) and Octets, respectively, and the number of seconds in the Interval. These values are used to calculate the utilization of a 10 Mbps Ethernet port as follows:<br><br>$\text{Utilization} = \frac{\text{Pkts} * (9.6 + 6.4) + (\text{Octets} * .8)}{\text{Interval} * 10{,}000}$<br><br>The result of this equation is the utilization value, which is the utilization percentage of the Ethernet segment on a scale of 0 to 100 percent. |
| Packets | The total number of received packets (including bad packets, broadcast packets, and multicast packets). |
| Errors | The total number of errors that RMON received for this port. |
| Util% | The bandwidth utilization percentage of the Ethernet segment on a scale of 0 to 100 percent. |

## Viewing Events in a Log File

The CSS can send notifications of RMON alarm events to a traplog file or a configured log location, such as a log file on the CSS disk, a CSS session, a host syslog daemon, or an e-mail address. The notification itself displays the time that the event occurred, the event number, and its configured description in parentheses.

For example:

```
FEB 15 15:41:22 EVENT#4 FIRED: (Service Toys exceeded 30,000
connections).
```

For information on configuring an RMON event, see the "Configuring an RMON Event" section. For information on configuring an RMON alarm, see the "Configuring an RMON Alarm" section.

## Viewing a Traplog File

A traplog file is an ASCII file in the log directory containing generic and enterprise SNMP traps. No configuration is necessary for the traplog file. When an RMON alarm event occurs, a notification of its occurrence is automatically saved in the trap log file on the CSS. Even when traps are disabled, the CSS still produces a log message for any event that would normally generate a trap.

When a traplog file reaches its maximum size (50 MB for a hard disk-based CSS, 10 MB for a flash disk-based CSS), the CSS renames the traplog file to traplog.prev as a backup file and starts a new traplog file. The CSS overwrites the backup traplog file when it renames the traplog file. Each time the CSS reboots, it continues to use the existing traplog file until it reaches its maximum size.

The traps sent to the traplog file are the same traps sent to an SNMP network management station. Refer to Chapter 5, Configuring Simple Network Management Protocol (SNMP), for information on configuring SNMP.

To display all SNMP traps that have occurred on the CSS, enter:

```
# show log traplog
```

## Viewing a CSS Disk Log File

Before the CSS can send an event to a log location, you must:

- Configure the location by using the **logging disk**, **host**, **line**, or **sendmail** command.
- Enable logging for the network management subsystem. To do so, enter:

  ```
  (config)# logging subsystem netman level info-6
  ```

Refer to Chapter 4, Using the CSS Logging Features, for details on configuring logging for the CSS.

To view the events in a log file on the CSS disk, use the **show log** *log_filename* command. To view a log file named log1, enter:

```
# show log log1
```

# Using an XML Document to Configure the CSS

The CSS Content Application Program Interface (API) feature allows you to use a network management workstation to make web-based configuration changes to the CSS using Extensible Markup Language (XML) documents. XML is a powerful tool that can be used to automatically configure a CSS using all of the CLI commands included in the CSS software, such as to specify server weight and load, to configure load balancing across a group of servers, or to configure content rules to restrict access to a group of directories or files on the servers.

XML code loads a series of CLI commands into the CSS without the need to respond to the prompts, similar to operating in expert mode. As the CSS administrator, plan which type of changes you want to implement and the consequences of these changes as they are performed.

After you create the XML document, you publish (upload) the XML file to the Hypertext Transfer Protocol (HTTP) server embedded in the CSS using an HTTP PUT method.

This chapter contains the following major sections:

- Creating XML Code
- Allowing the Transfer of XML Configuration Files on the CSS
- Parsing the XML Code
- Publishing the XML Code to the CSS
- Testing the Output of the XML Code

## Creating XML Code

When developing XML code for the Content API to issue CLI commands, adhere to the following guidelines. You can use any text editor for creating the XML code. The maximum number of characters for each tag set is 300.

1. Include the following line as the first line in the XML file:

```
<?xml version="1.0" standalone="yes"?>
```

2. Enclose the CLI commands within the <action></action> tag set. For example:

```
<action>add service MyServiceName</action>
<action>vip address 10.2.3.4</action>
```

**Note** A nested **script play** command (to execute a script line by line from the CLI) is not allowed in an XML file. This restriction is enforced because the actual execution of the XML tag set is performed within a **script play** command

3. If special characters are required in an XML configuration, be aware of the following considerations:

   – The CSS supports the inclusion of the ~, !, @, #, $, ^, &, *, (, and ) characters in XML content. All other special characters (such as <, >, and %) are not supported in an XML configuration.

   – Special characters can be included in a service, owner, or content name provided that the name is included in the content of the XML element and the name is enclosed within an <action></action> tag set. For example, `<action>service My@#Service</action>`.

   – If special characters do form part of an XML tag, specifically an attribute, these characters are not supported (for example, `<service name = My@#Service>`. In this case, the command request may be rejected or the special characters may be discarded. You must enclose the special characters within an <action></action> tag set, as described above.

4. Pay attention to mode hierarchy of the CLI commands in the XML file. Each mode has its own set of commands. Many of the modes have commands allowing you to access other related modes. If you enter a series of commands in the improper mode hierarchy, this will result in an XML file that fails to execute properly.

   As an example, the following commands configure an access control list (ACL):

```
<?xml version="1.0" standalone="yes" ?>
<config>
    <action>acl 98</action>
        <action>clause 10 permit any any dest any</action>
    <action>apply circuit-(VLAN3)</action>
</config>
```

   In another example, the following commands configure a CSS Ethernet interface:

```
<?xml version="1.0" standalone="yes" ?>
<config>
    <action>interface ethernet-6</action>
    <action>bridge vlan 3</action>
    <action>circuit VLAN3</action>
        <action>ip address 10.10.104.1/16</action>
</config>
```

5. Pay attention to the allowable CLI command conventions for syntax and variable argument in the XML file. If you enter an invalid or incomplete command, this will result in an XML file that fails to execute properly.

**Note** For overview information on the CLI commands you can use in global configuration mode and its subordinate modes, refer to the *Cisco Content Services Switch Command Reference*, Chapter 2, "CLI Commands."

## XML Document Example

The following example is a complete XML document. The XML document creates three services, an owner, and a content rule, and assigns one of the newly created services to the content rule.

```
<?xml version="1.0" standalone="yes"?>
<config>
    <service name="router">
        <ip_address>10.0.3.1</ip_address>
        <action>active</action>
    </service>
    <service name="sname2">
        <ip_address>10.0.3.2</ip_address>
        <weight>4</weight>
        <action>active</action>
    </service>
    <service name="sname3">
        <ip_address>10.0.3.3</ip_address>
        <weight>5</weight>
        <protocol>udp</protocol>
        <action>suspend</action>
    </service>
    <service name="nick">
        <ip_address>10.0.3.93</ip_address>
        <action>active</action>
    </service>
    <owner name="test">
    <content name="rule">
        <vip_address>10.0.3.100</vip_address>
        <protocol>udp</protocol>
        <port>8080</port>
        <add_service>nick</add_service>
        <action>active</action>
    </content>
    </owner>
</config>
```

## Allowing the Transfer of XML Configuration Files on the CSS

For client applications to publish XML configuration files on the CSS, they must upload the files to the CSS over HTTP connections. By default, the CSS denies HTTP connections for the transfer of XML configuration files. You can configure the CSS to allow the transfer of these files through either an unsecure HTTP connection or a secure HTTPS SSL connection. You cannot configure access for both secure and unsecure connections.

**Note** Because the CSS can process large configurations, the CSS processes only two concurrent XML configuration uploads over secure connections. If a third upload is attempted, it may not succeed. The CSS closes the connection and sends the following message to the client:

```
status 503 Service Unavailable
```

To allow the transfer of XML configuration files through:

- An unsecure connection, use the global configuration **no restrict xml** command. For example, enter:

  ```
  (config)# no restrict xml
  ```

  The CSS listens on port 80 for an unsecure connection request.

  To reset the default behavior to deny the transfer of XML configuration files through an unsecure connection, use the global configuration **restrict xml** command. For example, enter:

  ```
  (config)# restrict xml
  ```

- A secure SSL connection, use the global configuration **no restrict secure-xml** command. For example, enter:

  ```
  (config)# no restrict secure-xml
  ```

  The CSS listens on port 443 for a secure connection request. The client application can use SSL v2/3 or v3. However, the CSS performs all negotiations using SSL v3. The CSS requires a Secure Management license key to negotiate a secure connection using SSL strong encryption. Without the key, the CSS uses SSL weak encryption.

  To reset the default behavior to deny the transfer of XML configuration files through a secure connection, use the global configuration **restrict secure-xml** command. For example, enter:

  ```
  (config)# restrict secure-xml
  ```

## Parsing the XML Code

After you complete the XML file, parse the code to ensure that it is syntactically correct. One easy way to parse XML code is to open the XML file directly from Microsoft Internet Explorer. Syntax errors are flagged automatically when the file is loaded. If an error occurs, review your XML code and correct all syntax errors.

## Publishing the XML Code to the CSS

The completed XML file is remotely published (uploaded) to the HTTP server in the CSS from the external network management workstation by using an HTTP PUT method. The HTTP PUT method uses the IP address of the CSS as the destination URL where you want to publish the XML file.

**Note** When you configure TACACS+ on a CSS, the CSS does not authorize scripts through the TACACS+ server. Because the CSS transforms all XML commands into scripts, the CSS also does not authorize XML commands through the TACACS+ server.

When you publish XML files to the HTTP server in the CSS, the CSS requires a valid username and password as part of the user authentication process. The username must have SuperUser privileges to be able to add XML files to the CSS.

Ensure that the CLI commands in the XML document do not have an impact on the interface configuration through which the XML file transfer process occurs (for example, including the command **no ip addr**, which identifies the IP address of the CSS receiving the XML file). If this occurs, you will disconnect the workstation performing the XML file transfer.

Because the CSS can process large configurations, the CSS processes only two concurrent XML configuration uploads over secure connections. If a third upload is attempted, it may not succeed. The CSS closes the connection and sends the following message to the client:

```
status 503 Service Unavailable
```

Software is available to simplify the process of publishing XML files to the CSS HTTP server. These software packages offer a simple method to publish files to a web server. This software uses the unsecure HTTP or secure HTTPS protocol to publish files and requires no special software on the web server side of the connection.

**Note** An error code in the publishing process usually means that **no restrict secure-xml** or **no restrict xml** commands have not been issued on the CSS prior to publishing the XML file. See the "Allowing the Transfer of XML Configuration Files on the CSS" section for details.

## Testing the Output of the XML Code

Test the output of the XML code by reviewing the running configuration of the CSS. After the XML has been successfully published to the CSS, Telnet to the switch and enter the **show running-config** command to verify that the XML changes have properly occurred. If the XML changes are incorrect or missing, republish the XML code to the CSS as described in the "Publishing the XML Code to the CSS" section.

# 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 Content Load-Balancing Configuration Guide*.

**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 contains the following major sections:

- Script Considerations
- Playing a Script
- Using the Command Scheduler
- Using the echo Command
- Using Commented Lines
- Using Variables
- Using Logical and Relational Operators and Branch Commands
- Special Variables
- Using Arrays
- Capturing User Input

## Script Considerations

When you upgrade the CSS software, the old scripts remain in the old software version's /script directory. For details about 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

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.

Use the **script play** command to execute a script line-by-line from the CLI in SuperUser mode. You can also use this command to pass parameter values to a script.

For example, enter:

```
# script play MyScript "Argument1 Argument2"
```

You can display a list of available scripts using the **show script** command.

## Using the Command Scheduler

You can schedule the execution of CLI commands to schedule periodic content replication, the gathering of statistics, and scheduled configuration changes. Use the **cmd-sched** command to configure the scheduled execution of any CLI commands, including playing scripts. The commands that will be executed are referred to as the command string. To schedule commands, you must create a configuration record; the record includes the command string and a provision about when to execute the commands.

At the specified time, the command scheduler executes a command string by creating a pseudo-login shell where each string is executed. A **cmd-sched** record is scheduled for execution only upon completion of its shell. Use the **show lines** command to display information about active pseudo-shells.

**Note** To terminate the execution of a command string, use the **disconnect** command.

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

- **cmd-sched** - Enable command scheduling.
- **cmd-sched record** *name minute hour day month weekday* "*commands...*" {*logfile_name*} - Create a configuration record for the scheduled execution of any CLI commands, including the playing of scripts.

The variables are listed below. When entering minute, hour, day, month, and weekday variables, you may enter a single integer, a wildcard (*), a list separated by commas, or a range separated by a dash (-).

- *name* - The name of the configuration record. Enter an unquoted text string with a maximum of 16 characters.
- *minutes* - The minute of the hour to execute this command. Enter an integer from 0 to 59.
- *hour* - The hour of the day. Enter an integer from 0 to 23.
- *day* - The day of the month. Enter an integer from 0 to 31.
- *month* - The month of the year. Enter an integer from 1 to 12.
- *weekday* - The day of the week. Enter an integer from 1 to 7. Sunday is 1.

- *command* - The commands you want to execute. Enter a quoted text string with a maximum of 255 characters. Separate multiple commands with a semicolon (;) character. If the command string includes quoted characters, use a single quote character; any single quoted characters not preceded by a backslash (\) character is converted to double quotes when the command string is executed.
- *logfile_name* - An optional variable that defines the name of the log file. Enter a text string with a maximum of 32 characters.

Any of the time variables can contain one or some combination of the following values:

- A single number to define a single or exact value for the specified time variable.
- A wildcard (*) character matching any valid number for the specified time variable.
- A list of numbers separated by commas, with a maximum of 40 characters, to define multiple values for a time variable.
- Two numbers separated by a dash (-) character indicating a range of values for a time variable.

For example:

```
(config)# cmd-sched record periodic_shows 30 21 3 6 1 “show
history;show service;show rule;show system-resources”
```

To enable command scheduling, enter:

```
(config)# cmd-sched
```

To disable command scheduling, enter:

```
(config)# no cmd-sched
```

To delete a configuration record, enter:

```
(config)# no cmd-sched periodic_shows
```

## Showing Configured Command Scheduler Records

Use the **show cmd-sched** command to display the state of the command scheduler and information about the records for the scheduled CLI commands. The syntax and options are:

- **show cmd-sched** - Lists the state of the command scheduler and all scheduled CLI command records
- **show cmd-sched name** *record_name* - Lists information about the specified scheduled CLI command record

To view the command scheduler state and all scheduled CLI command records, enter:

```
(config)# show cmd-sched
```

Table 8-1 describes the fields in the **show cmd-sched** command output.

***Table 8-1 Field Descriptions for the show cmd-sched Command***

| Field | Description |
|---|---|
| Cmd Scheduler | State of the command scheduler (enabled or disabled) and the number of configured records. |
| Sched Rec | The name of the configuration record. |
| Id | The ID for the record. |
| Next exec | The date and time the record will be executed. |
| Executions | How many times the record has executed. |
| MinList | The configured minute of the hour to execute the command. |
| HourList | The configured hour of the day to execute the command. |
| DayList | The configured day of the month to execute the command. |
| monthList | The configured month of the year to execute the command. |
| WeekdayList | The configured day of the week to execute the command. Sunday is 1. |
| Cmd | The commands you want to execute. Separate multiple commands with a ; (semicolon) character. |

## 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

To disable the **echo** command without the text "no echo" appearing in the script output, use the commented **no echo** command as the first line in any script. 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
```

If you enter:

```
! Print Echo
echo "Hello"
```

The output is:

```
echo "Hello"Hello
```

This happens 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 Chapter 3, Configuring User Profiles.

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 and Setting Variables

To create a variable and set the variable with a value, use the **set** command. 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

To remove a variable from memory, use the **no set** command.

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

This section includes the following topics:

- Using the No Set and Set Commands
- Using Arithmetic Operators
- Using the Increment and Decrement Operators

### Using the No Set and Set Commands

To modify a variable, 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

To change the value of a variable with arithmetic operators (-, +, /, *, or modulus), use the **modify** command . 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 and Relational Operators and Branch Commands

To build structured command blocks, use the **if** and **while** branch commands. 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

To create a branch in a script based on the result of a previous command, use the **if** 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

To execute the same commands repeatedly based on the result of an associated expression, use the **while** branch command. 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

The informational variables are:

- **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

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 use the **socket** commands, the SOCKET variable is set automatically (see "Using socket Commands" later in this chapter). 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

To capture user input in a variable, use the **input** command. 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.

To modularize your scripts both for ease of reading and simplification, use the **function** command. 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.

## Passing Arguments to a Function

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

To call a script from another script, use the **SCRIPT_PLAY** function. 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 host name 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

To search for specified data and place the last line of the search results in a variable called UGREP, use the **grep** command with the -u option. 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

To assist you in building a structured protocol, use **socket** commands in a script keepalive. 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

To performs either a TCP connection handshake (SYN-SYNACK...) to a specific IP address/port or a UDP connection by reserving the host/port, use the **socket connect** command. 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***} {session}** {**nowait**}|**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 host name or IP address of the remote CSS.
- *ip_address* - The host name 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 integer 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.
- **nowait** - Keyword that tells the socket to send data immediately without waiting to aggregate the data first.

## socket send

To write data through a previously connected TCP connection, use the **socket send** command. Note that the **socket send** command clears all currently stored data in the 10-KB receive buffer.

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 website.

## socket receive

To fill up the socket’s 10-KB internal buffer with data from the remote host, use the **socket receive** command. 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

To fill up the socket's 10-KB internal buffer with data from the remote host, use the **socket waitfor** command. 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

To inspect the socket's internal data buffer for actual data, use the **socket inspect** command. If the software finds data, it displays to standard output the last 10 KB of data received. 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.."

**Note** If you use the **socket inspect** command in a keepalive script, you must use the "use-output" option in the command line when configuring the keepalive type for the service.

## socket disconnect

To close 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, use the **socket disconnect** command.

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 a FIN (finished) rather than RST to the remote host.

## Socket Administration

You can have a maximum of 64 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 8-2 describes the fields in the **show sockets** output.

***Table 8-2 Field Descriptions for the show sockets Command***

| Field | Description |
|---|---|
| Socket ID | Socket file descriptor |
| Remote Host:Port | The connected Host address and Port pair |
| Protocol | The protocol specified for the connection: either TCP or UDP. |
| 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** or **socket waitfor** 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.

**Note** The **socket send** command also clears all currently stored data in the 10-KB buffer. For details, see the "socket send" section.

## Displaying Scripts

To display a list of the scripts that reside in the CSS script directory or the contents of a specific script (with or without line numbers), use the **show script** command. This command is available in SuperUser mode and all configuration modes. The syntax of this command is:

**show script** {*filename* {**line-numbers**}}

The variables and option of this command are:

- *filename* - Name of a valid script file whose contents you want to display. Enter a case-sensitive unquoted text string with a maximum of 32 characters.
- **line-numbers** - Displays line numbers for each line in the script.

For example, to display a list of all scripts in the CSS script directory, enter:

```
# show script
```

To display the text of the ap-kal-dns keepalive script, including line numbers, enter:

```
# show script ap-kal-dns line-numbers
```

## 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 Chapter 1, Managing the CSS Software.
2. Upgrade your CSS software. For details on upgrading the CSS software version, refer to 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 Chapter 1, Managing the CSS Software.

## Using the showtech Script

To gather information designed to assist the Cisco Technical Assistance Center (TAC) in analyzing your CSS, use the **showtech** script. 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 CSS provides scripted keepalives to support the need for keepalives operations that cannot be handled using non-scripted keepalives. We recommend that you limit I/O operations in a scripted keepalive to socket operations used to probe network connectivity to a server and for determining application health on a server. Although the scripting language supports file I/O on the CSS hard drive or flash drive, we recommend that you do not use file I/O operations within scripted keepalives. Extensive file I/O operations within scripted keepalives may cause services to transition. File system access is allowed in scripts executed from the CLI or from the command scheduler.

The 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 (FIN)

Use the following script keepalive to open and gracefully close (using a FIN rather than a RST) a socket on user-specified TCP ports.

**Note** The service mode **keepalive tcp-close fin** command or global keepalive mode **tcp-close fin** command also gracefully closes (using a FIN rather than a RST) a socket.

```
!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
- LDAP Keepalive

## SMTP KEEPALIVE

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-smtp
! Parameters: HostName
!
! Description:
!  This script will log into an SMTP server and send a 'hello'
!  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 'hello'
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! 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}
81000044204545455045 4d454d464a434143414341434143414341434143414341
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 'hello' 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 hello 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
```

## LDAP Keepalive

```
!no echo
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Filename: ap-kal-ldap
! Parameters: HostName
!
! Description:    “Lightweight Directory Access Protocol v3”
!   This script will connect to an LDAP server and attempt to
!   “bind request” to the server.  Once the server gives a
!   positive response we will disconnect (RFC-2251).
!
! Bind Response Code we will search for is: 0x0a 0x01 0x00
!
! Failure Upon:
!   1. Not establishing a connection with the host.
!   2. Failure to receive the above response code.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Make sure the user has a qualified number of arguments
if ${ARGS}[#] “NEQ” “1”
    echo “Usage: ap-kal-ldap \'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 389 tcp 2000

set EXIT_MSG “Send: Failure”
! Send a Bind Request to the remote host.  This is simply a standard !
“capture” of a bind request in hex.  This should work for all standard
! version 3 LDAP servers. socket send ${SOCKET}
“300c020102600702010204008000” raw

set EXIT_MSG “Receive: Failure”
! Expect to receive a standard response from the host.  This should !
be equal to a SUCCESS response code: socket waitfor ${SOCKET} “0a0100”
2000 raw


set EXIT_MSG “Send: Failure”
! Send an exit “Unbind Request” to the remote host so that they ! are
not left hanging. socket send ${SOCKET} “30050201034200” raw

no set EXIT_MSG
socket disconnect ${SOCKET}

exit script 0
```

# Upgrading Your CSS Software

New software versions are periodically released for the CSS. This appendix provides information on how to upgrade your CSS with a new software release. This appendix contains the following major sections:

- Before You Begin
- Upgrading Software Considerations
- Upgrading Your CSS Software
- Updating MIBs

**Note** When syntax changes are made to existing CLI commands, the CSS updates your startup-config file automatically with most command syntax changes. For example, the CSS automatically updates the **dnsflow disable** command in the startup-config file to the **flow-state** command. If the CSS does not update a command syntax change in a startup-config file automatically, a startup error is displayed. Refer to the *Release Note for the Cisco Series Content Services Switch* for information on which command syntax changes display startup-config file errors.

## Before You Begin

Before you can upgrade your CSS, you must copy the new CSS software to your FTP server and configure an FTP server record so the CSS recognizes the server. Use the **show installed-software version-limit** command to display the maximum number of installed versions allowed on your hard disk or Flash disk.

### Copying the New CSS Software

ArrowPoint Distribution Images (ADIs) of the CSS software versions are located on the Cisco Systems website (www.cisco.com). Use your customer login and password to access this page. From this location, access the page listing the versions of GZIP-compressed software, then click an image to download it. Once the image is downloaded, place it on an FTP server that the CSS can access.

**Note** You do not need to uncompress the GZIP-compressed software. When you copy the software, or if the upgrade script copies the software to the CSS, the CSS automatically uncompresses it.

### Configuring an FTP Server Record on the CSS

Before you can copy the ADI from the FTP server to the CSS, you must create an FTP record file on the CSS identifying the ADI. The record contains the IP address, username, and password for the server. If you did not configure an FTP record before starting the upgrade script, the script prompts you to configure one.

To configure an FTP server record:

1. Log in to the CSS.
2. Access global configuration mode:

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

3. Use the **ftp-record** command to configure the default FTP server. The syntax is:

   **ftp-record** *ftp_record ip_or_host username* [**"***password***"**
   |**des-password** *des_pwd*] {**base_directory**}

   The options and variables for this command are:

   - *ftp_record* - Name for this FTP record file. Enter an unquoted text string with no spaces and a maximum of 32 characters.
   - *ip_or_host* - IP address or host name of the FTP server you want to access. Enter an IP address in dotted-decimal notation (for example, 192.168.11.1) or a mnemonic host name (for example, myhost.mydomain.com).
   - *username* - Valid login username on the FTP server. Enter a case-sensitive unquoted text string with no spaces and a maximum of 16 characters.
   - *password* - Password for the valid login username on the FTP server. Enter a case-sensitive, quoted text string with no spaces and a maximum of 16 characters. The CSS allows all special characters in a password except for the percent sign (%).
   - **des-password** *des_pwd* - Specifies the Data Encryption Standard (DES) encrypted password for the valid login username on the FTP server. Enter a case-sensitive, unquoted text string with no spaces and a maximum of 64 characters.
   - *base_directory* - (Optional) Base directory when using this record.

   For example:

   ```
   (config)# ftp-record DEFAULT_FTP 192.168.2.01 eng1 des-password
   serve
   ```

You can now upgrade your CSS.

## Upgrading Software Considerations

For the CSS software version 8.10 to support the SSL compression (SSL-C) module, before you upgrade to the CSS to version 8.10, the CSS *must* be at one of the following maintenance releases or higher:

- 7.50.1.03
- 7.40.2.02
- 7.30.4.02

If the CSS is not at one of these maintenance releases, you must perform the following upgrade sequence; first upgrade the CSS to the required maintenance release and then upgrade the CSS to software version 8.10.

If you are upgrading from a version of WebNS software earlier than version 7.40, be aware of the following Adaptive Session Redundancy (ASR) configuration restrictions in WebNS software versions 7.40 and higher:

- If your CSSs have mismatched chassis configurations (a different number of Session Processors (SPs) in each CSS), ASR will not function after the upgrade. Before you upgrade, ensure that both CSS chassis have the same number of SPs.
- If your CSSs meet the ASR requirement of having the same number of SPs in each chassis, you must upgrade both CSSs to WebNS Version 7.40.
- During the upgrade process, ASR does not function and you lose any sessions that are in progress.

## Upgrading Your CSS Software

When upgrading the CSS software, you can use the upgrade script or manually enter CLI commands. The upgrade script allows you to upgrade the CSS either automatically or interactively by responding to script prompts.

Either way, the tasks that the script performs include:

- Checking to see how many installed software versions are installed on the CSS, and if the CSS contains the maximum number of installed software versions, then deleting an older software version.
- Archiving the running configuration to the startup configuration.
- Copying the new ADI to the CSS boot-image directory.
- Unpacking the new ADI.
- Copying the scripts and user profiles from the older CSS software to the new software. The copied scripts do not include Cisco-supplied scripts except default-profile.
- Setting the primary boot file to the new ADI.
- Rebooting the CSS.

We recommend that you use the upgrade script because the upgrade script ensures that the existing scripts and user profiles are copied to the new software.

When manually upgrading the CSS software, you must enter CLI commands to perform the same tasks that the upgrade script would perform.

This section includes the following topics:

- Using the Upgrade Script
- Manually Upgrading the CSS Software

## Using the Upgrade Script

The upgrade script allows you to upgrade your CSS without having to enter any CLI commands. There are two ways to run the script:

- Automatically Running the Upgrade Script
- Interactively Using the Upgrade Script

If the upgrade script fails while upgrading the CSS to the same version of software that is currently running, the CSS software directory will be incomplete. To reinstall the software, you must upgrade the CSS manually (that is, use FTP to transfer the .adi to the CSS and perform a manual unpack).

## Automatically Running the Upgrade Script

You can run the upgrade script to perform the software upgrade without having to enter any information. The script automatically:

- Checks to see how many installed software versions are installed on the CSS. You can install a maximum of two software versions on the CSS. If the CSS contains the maximum number of installed software versions, the script deletes an older version.

  **Note** The script does not prompt you to delete a version of software that you configured as the primary or secondary boot file. On a disk-based system, you may need to quit and then deselect the primary or secondary boot file before continuing with the upgrade.

- Archives the running configuration to the startup configuration.
- Copies the new ADI to the CSS boot-image directory.
- Unpacks the new ADI.
- Copies the scripts and user profiles from the older CSS software to the new software. The copied scripts do not include Cisco-supplied scripts except default-profile.
- Sets the primary boot file to the new ADI.
- Reboots the CSS.

To upgrade your CSS software using the upgrade script:

1. Log in to the CSS.
2. Copy any changes in your current user profile to the scripts directory. During the upgrade, the upgrade script copies the contents of this directory including user profiles into the script directory of the new software image.

   To copy any changes to your current user profile to the scripts directory, use the **save_profile** alias command. For example, enter:

   ```
   # save_profile
   ```

3. Rename any Cisco-supplied scripts that you changed. The upgrade script does not overwrite the Cisco-supplied scripts in the script directory of the new software image with the changed versions of the scripts, except the default-profile script.

If the upgrade script detects differences between the old version of the default-profile script and the new version, it renames the new version default-profile.new and copies the old version into the script directory of the new software image.

**Note** If the default-profile.new script exists in the old script directory, the upgrade script does not copy this script to the script directory of the new software image.

**4.** Start the upgrade script and put the name of the ADI and its extension in quotes.

- If you are using a GZIP-compressed ADI from the FTP server, include the **.gz** file extension. For example:

  ```
  # upgrade "sg0730002.adi.gz"
  ```

- If you are using an uncompressed version of the ADI from the FTP server, include the **.adi** file extension. For example:

  ```
  # upgrade "sg0730002.adi"
  ```

If you did not configure a default FTP record before starting the upgrade script, the script prompts you to configure one. At the prompts, enter the FTP server information where you copied the upgrade ADI

When you configure a default FTP record, you see the following information during the upgrade:

```
Current Version:sg0730002

*** You must remove an installed version to upgrade. ***

Attempting to delete sg0720003

archive running-config startup-config

Attempting ftp of sg0740002.adi:
#     copy ftp DEFAULT_FTP ${new_version_adi} boot-image
Copying (-) 57,241,012
Completed successfully.
#(config-boot)#
unpack ${new_version_adi}
Unpacking(/) 99%
(config-boot)#
```

The script copies the scripts and user profiles from the old script directory to the script directory of the new software image. Note that *script_name* is the name of the current script being copied.

```
Copying scripts from c:/sg0730002/script/ to c:/sg0740002/script/
Working (/)
Copying script_name... Done
```

**Note** If the copy operation fails, the CSS displays the following messages and then exits the upgrade process:

```
Error: Copy of script_name failed!
Script copy failed, upgrade aborted!
```

If an old script has the same name as a Cisco-supplied script but its contents differ, the following message appears:

```
script_name differs between images
```

The upgrade script does not overwrite the Cisco-supplied scripts in the script directory of the new software image with the older versions of the scripts, except the default-profile script. If the upgrade script detects an older version of the default-profile script, the script renames the newer default-profile script as default-profile.new and then copies the older version of the default-profile script to the script directory of the new software image. For example:

```
Saving c:/sg0740002/script/default-profile.new... Done
Copying default-profile to new image... Done
```

The upgrade script sets the ADI as the primary boot file and reboots the CSS.

```
setting primary boot-file sg0740002

rebooting
```

The CSS automatically performs a Flash upgrade, if necessary, and then boots the new image.

5. After you upgrade and reboot a CSS 11506 that contains a passive SCM, use the **passive sync** command in boot-config mode (or the **passive sync** macro command) immediately after upgrading your CSS software to synchronize the boot configurations on the redundant SCMs. Refer to Chapter 2, Specifying the CSS Boot Configuration, for details on configuring a boot configuration record for a passive SCM.

6. If the CSS does not have a startup-config file, it uses the file in the archive directory. To restore the startup-config file on the CSS, use the **restore** *filename* **startup-config** command. For example, to restore the startup-config file in the archive directory as the startup-config file on the CSS, enter:

   ```
   # restore startup-config startup-config
   ```

## Interactively Using the Upgrade Script

The upgrade script allows you to enter information and make selections by responding to prompts as it runs. Before the script performs the upgrade, it prompts you to:

- Remove ADIs from the CSS if the script detects two installed versions on a hard disk-based system or on a Flash disk-based system
- Enter the version of the new ADI
- Set the primary boot-file to the new ADI
- Reboot the CSS with the ADI you are installing after the upgrade is done
- Archive the running configuration to the startup configuration

The script automatically copies the scripts and user profiles from the older CSS software to the new software. The copied scripts do not include Cisco-supplied scripts except default-profile.

To use the interactive version of the script:

1. Log in to the CSS.
2. Copy any changes in your current user profile to the scripts directory. During the upgrade, the upgrade script copies the contents of this directory including user profiles into the script directory of the new software image.

   To copy any changes to your current user profile to the scripts directory, use the **save_profile** alias command. For example, enter:

   ```
   # save_profile
   ```

3. Rename any Cisco-supplied scripts that you changed. The upgrade script does not overwrite the Cisco-supplied scripts in the script directory of the new software image with the changed versions of the scripts, except the default-profile script.

   If an old script has the same name as a Cisco-supplied script but its contents differ, the following message appears:

```
script_name differs between images
```

If the upgrade script detects differences between the old version of the default-profile script and the new version, it renames the new version as default-profile.new and copies the old version into the script directory of the new software image.

**Note** If the default-profile.new script exists in the old script directory, the upgrade script does not copy this script to the new script directory.

4. Start the upgrade script. For example:

```
# upgrade
```

5. If a default FTP record is configured on the CSS, respond to the following prompt:

```
Would you like to use this record? [y n q]? y
```

If you did not configure a default FTP record before starting the upgrade script, the script prompts you to configure one. At the prompts, enter the FTP server information where you copied the upgrade ADI

The script displays the current version of the ADI.

```
Current Version: sg0730002 (Official)
```

If the script detects the maximum number of ADIs, a message informs you to remove an ADI. Then the script prompts you to remove an older ADI. For example:

```
*** You must remove an installed version to upgrade.***

remove sg0720003[y n q]?
```

The script does not prompt you to delete a version of software that you configured as the primary or secondary boot file. On a Flash disk-based system, you may need to quit and then deselect the primary or secondary boot file before continuing with the upgrade.

6. Remove the ADI, if necessary.
   - Enter **y** to remove the displayed ADI version.
   - Enter **n** for the script to display another version to remove.

- Enter **q** to exit from the script.

```
remove sg0720003 [y n q]? y

Attempting to delete sg0720003
```

7. Enter the filename and extension of the GZIP-compressed ADI version to install at the prompt, and verify the information you entered. If you are using an uncompressed version of the ADI from the FTP server, include the **.adi** file extension (for example, sg0740002.adi). For example:

```
Please Enter Version to Install:sg0740002.adi.gz

Upgrade to Version sg0740002? [y n q] y
```

8. Determine whether to set the ADI as the primary boot file.
   - Enter **y** to set the ADI as the primary boot-file and change the CSS configuration.
   - Enter **n** to keep the same primary boot-file configuration.

```
Set primary boot-file to Version sg0740002? [y n q] y
```

9. Determine whether to have the CSS reboot with the ADI.
   - Enter **y** to reboot the CSS with this ADI after the upgrade is done.
   - Enter **n** to not reboot the CSS with the ADI after the upgrade is done.

```
Reboot with Version sg0740002? [y n q] y
```

10. Determine whether to have the CSS archive the contents of the running-config file to the startup-config file.
    - Enter **y** to archive the contents of the running-config file to the startup-config file.
    - Enter **n** to keep the same startup configuration.

```
Archive running-config to startup-config? [y n q] y

archive running-config startup-config
```

The upgrade script copies the ADI from the FTP server, then unpacks and installs it.

```
Attempting ftp of sg0740002.adi.gz:

#     copy ftp DEFAULT_FTP ${new_version_adi} boot-image
```

```
Copying (-) 57,241,012

Completed successfully.
#
(config-boot)# unpack ${new_version_adi}
unpacking(/) 99%

(config-boot)#
```

The upgrade script copies the scripts and user profiles from the old script directory to the directory of the new software image. Note that *script_name* is the name of the current script being copied.

```
Copying scripts from c:/sg0730002/script/ to c:/sg0740002/script/
Working (/)
Copying script_name... Done
```

**Note** If the copy operation fails, the CSS displays the following messages and then exits the upgrade process:

```
Error: Copy of script_name failed!
Script copy failed, upgrade aborted!
```

If an old script has the same name as a Cisco-supplied script but its contents differ, the following message appears:

```
script_name differs between images
```

The upgrade script does not overwrite the Cisco-supplied scripts in the script directory of the new software image with the older versions of the scripts, except the default-profile script. If the upgrade process detects an older version of the default-profile script, the script renames the newer default-profile script as default-profile.new and then copies the older version of the default-profile script to the script directory of the new software image. For example:

```
Saving c:/sg0740002/script/default-profile.new... Done
Copying default-profile to new image... Done
```

Then the upgrade script sets the ADI as the primary boot file.

```
setting primary boot-file sg0740002
```

11. If you decided to reboot the CSS with the installed ADI in Step 8, the CSS reboots automatically. If you made the ADI the primary boot file and archived the contents of the running-config file to the startup-config file, the CSS automatically performs a Flash upgrade, if necessary, and then boots the new image.

    To manually reboot the system, enter the following commands:

    ```
    (config)# boot
    (config-boot)# reboot
    ```

12. After you upgrade and reboot the software in a CSS 11506 that contains a passive SCM, use the **passive sync** command in boot-config mode (or the **passive sync** macro command) immediately after upgrading your CSS software to synchronize the boot configurations on the redundant SCMs. For details on configuring a boot configuration record for a passive SCM, see Chapter 2, Specifying the CSS Boot Configuration.

13. If the CSS does not have a startup-config file, it uses the file in the archive directory. To restore the startup-config file, use the **restore** *filename* **startup-config** command. For example, to restore the startup-config file in the archive directory as the startup-config file on the CSS, enter:

    ```
    # restore startup-config startup-config
    ```

## Manually Upgrading the CSS Software

If desired, you can manually enter CLI commands to upgrade the CSS. Make sure you configure a default FTP server, as described in the "Before You Begin" section.

To manually upgrade the software version on your CSS:

1. Log in to the CSS.
2. Remove an older version of the ADI from the CSS if there are two installed versions.

**Caution** Do not remove the ADI currently running on the CSS. Use the **version** command to see the currently running software version.

To remove an ADI:

a. List the ADIs on the CSS.

```
(config)# show installed-software
sg0720104
sg0730003
```

b. Access boot mode:

```
(config)# boot
(config-boot)#
```

c. Use the **remove** command to remove the ADI.

```
(config-boot)# remove sg0720104
```

3. Archive your running configuration to the startup config file. For example:

```
(config-boot)# <Ctl-z>
# archive running-config startup-config
```

You can also use the **save_config** alias to archive your startup-config file. To view all available aliases, use the **show aliases** command.

4. Archive your custom scripts and user-profile files from the CSS scripts directory to the archive directory. Since the upgrade process installs the Cisco-supplied scripts with the new software but does not overwrite the files in the archive directory, archiving your custom script and user-profile files allows you to save them for use with the new software. After the upgrade, you can restore these files to the scripts directory.

To archive each file to the archive directory, use the **archive script** command. The syntax for this command is:

**archive script** *script_filename* {*archive_filename*}

The variables are as follows:

- *script_filename* - The filename of the script to archive. To see a list of scripts, enter:

```
# archive script ?
```

- *archive_filename* - (Optional) The name you want to assign to the archive file. Enter an unquoted text string with a maximum of 32 characters.

For example, to archive the admin-profile file from the scripts directory to the archive directory, enter:

```
# archive script admin-profile
```

To copy any changes to your current user profile to the scripts directory and then archive the profile to the archive directory, use the **save_profile** alias command. For example, enter:

```
# save_profile
```

5. Copy the new ADI to the CSS as the boot-image. If you are copying an uncompressed version of the ADI from the FTP server, include the .adi file extension (for example, sg0740002.adi).

```
(config-boot)# <Ctl-z>
# copy ftp DEFAULT_FTP sg0740002.adi.gz boot-image
```

DEFAULT_FTP is the FTP record file defined in the "Configuring an FTP Server Record on the CSS" section.

**Note** When you copy a GZIP-compressed ADI onto the CSS, the CSS automatically uncompresses it.

6. Unpack the ADI.

```
(config)# boot
(config-boot)# unpack sg0740002.adi
```

7. Set the new ADI as the primary boot file and install it. For example:

```
(config-boot)# primary boot-file sg0730003
```

8. Reboot the system.

```
(config)# boot
(config-boot)# reboot
```

The CSS automatically performs a Flash upgrade, if necessary, and then boots the new image.

9. After you upgrade and reboot the software in a CSS 11506 that contains a passive SCM, use the **passive sync** command in boot-config mode (or the **passive sync** macro command) immediately after upgrading your CSS software to synchronize the boot configurations on the redundant SCMs. Refer to Chapter 2, Specifying the CSS Boot Configuration, for details on configuring a boot configuration record for a passive SCM.

10. Use the **restore** command to restore the startup-config file, custom scripts, and user-profile files previously archived in the CSS archive directory. To see a list of files in the archive directory, enter:

    ```
    restore ?
    ```

    If the CSS does not have a startup-config file, it uses the file in the archive directory. To restore the startup-config file, use the **restore** *filename* **startup-config** command. For example, to restore the startup-config file in the archive directory as the startup-config file on the CSS, enter:

    ```
    # restore startup-config startup-config
    ```

    To restore each custom script and user profile file to the script directory, use the **restore** *filename* **script** command. For example, to restore the admin-profile filename to the CSS script directory, enter:

    ```
    # restore admin-profile script
    ```

## Updating MIBs

Typically, MIBs do not change for maintenance or sustaining releases. However, CSS Enterprise MIBs are included in the CSS GZIP file. During the software upgrade, the MIBs are loaded into the CSS /mibs directory.

We recommend that you always update the MIBs after a software upgrade. To update the CSS MIBs on your management station after you upgrade the CSS:

1. Use FTP to transfer the MIBs from the CSS MIBs (/v1 or /v2) directory to your management station.
2. Unload the MIBs from the management application.
3. Load the MIBs into the management application.

Refer to Chapter 5, Configuring Simple Network Management Protocol (SNMP), for information on CSS Enterprise MIBs.

# Using the Offline Diagnostic Monitor Menu

During the boot process, you can access the Offline Diagnostic Monitor (Offline DM) Main menu. The Offline DM Main menu allows you to perform the following functions:

- Set the boot configuration:
  - Configure a primary and secondary location from which the CSS accesses the boot image
  - Configure an IP address for the CSS
  - Configure a subnet mask
  - Configure a default gateway
- Show the boot configuration
- Select Advanced Options to:
  - Delete a software version from the disk
  - Set a password for the Offline DM Main menu
  - Set an administrative username and password
  - Reformat the disk and perform a check disk
  - Configure disks
- Reboot the CSS

This appendix contains the following major sections:

## Accessing the Offline DM Main Menu

To access the Offline DM Main menu:

1. Connect a console to the console port on the CSS. Configure the console with the following default values: 9600 baud, no parity, 8 data bits, 1 stop bit, and flow control set to none.
2. If accessing the Offline DM Main menu from a terminal server, configure the client application to display 24 lines to enable the Offline DM Main menu to display properly.
3. Power on the CSS. After the CSS begins to boot (approximately 15 seconds), it displays the following message (for approximately 5 seconds):

   ```
   Would you like to access the Offline Diagnostic Monitor
   menu?(y<cr>)
   ```

   At this point in the boot sequence, you may either:

   - Take no action (or press **n**) and let the CSS continue booting automatically with the default boot configuration.
   - Press **y**, then press **Enter** to halt the boot process and display the Offline DM Main menu.

The Offline DM Main menu appears:

```
MAIN MENU

Enter a menu number:

1* Set Boot Configuration
2. Show Boot Configuration
3* Advanced Options
4. Reboot System
```

An asterisk (*) next to a menu option indicates that the option contains a submenu.

**Note** If the 5-second time period elapses before you press **y** to halt the boot process, power down the CSS and then power it up again to access the Offline DM menu.

Table B-1 describes each menu item.

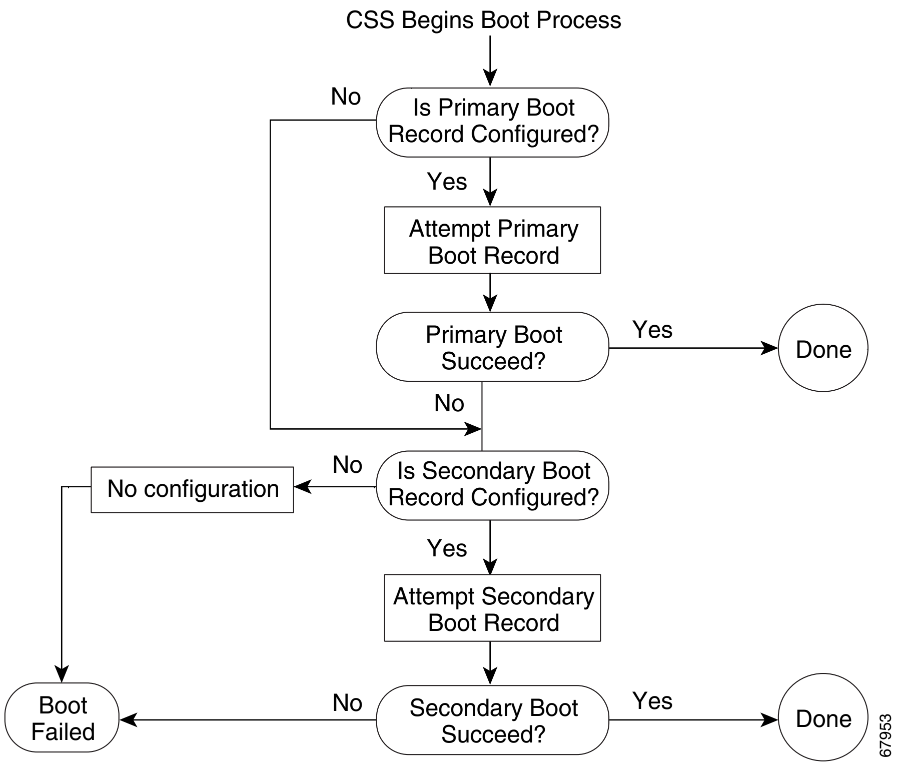
*Table B-1 Offline Diagnostic Monitor Menu Options*

| Menu Option | Function |
|---|---|
| 1. Set Boot Configuration | 1. Set Primary Boot Configuration<br>2. Set Secondary Boot Configuration<br>3. Set IP Address and Subnet Mask<br>r. Return to previous menu |
| 2. Show Boot Configuration | Display boot configurations (including primary and secondary boot configurations, records, and IP information) |
| 3. Advanced Options | 1. Delete a software version<br>2. Security Options<br>3. Disk Options<br>4. Set MSD Mapping<br>r. Return to previous menu |
| 4. Reboot System | Reboot the CSS. The CSS displays the following message before rebooting:<br>`Are you sure you want to reboot? (Y/N)`<br>Enter:<br>• **Y** to reboot the CSS<br>• **N** to continue using the Offline DM Main menu |

## Using the Boot Configuration Menu

The flowchart in Figure B-1 illustrates how the CSS uses the boot configuration information to complete the boot process.

***Figure B-1 Boot Configuration Flowchart***



The Boot Configuration menu is shown below:

```
BOOT CONFIGURATION MENU

Enter the number of a menu selection:

1. Set Primary Boot Configuration
2. Set Secondary Boot Configuration
3. Set IP Address and Subnet Mask
r. Return to previous menu
```

The Boot Configuration menu enables you to perform the tasks described in Table B-2.

***Table B-2 Boot Configuration Options***

| Menu Option | Function |
|---|---|
| 1. Set Primary Boot Configuration | Specifies the primary location (Network, FTP, Disk, or Clear) from which the CSS accesses the boot image. The default location is Disk. |
| 2. Set Secondary Boot Configuration | Specifies the secondary location (Network, FTP, Disk, or Clear) from which the CSS accesses the boot image. The default location is Clear. |
| 3. Set IP Address and Subnet Mask | Configures an IP address for the Ethernet management port and configure a subnet mask. |
| r. Return to previous menu | Displays the Offline DM main menu. |

## Setting Primary Boot Configuration

The information you provide for the primary boot configuration specifies the location from which the CSS accesses the primary boot image upon system reboot or when you download new software. When you select **Set Primary Boot Configuration** from the Boot Configuration menu, the CSS displays the following information. If you have previously entered information, the CSS displays the existing information and default values in square brackets [ ].

```
Configuring PRIMARY Boot Record
Boot via [N]etwork, [F]TP, [D]isk, or [C]lear: [D]
Press <Enter> to continue...
```

- Boot via Network - Allows you to boot the CSS using FTP from CSS software on a network-mounted file system on a remote system
- Boot via FTP - Allows you to download an ADI file containing CSS software that you want to install on the CSS disk (hard or Flash disk)
- Boot via Disk - Allows you to boot the CSS from software currently on the CSS disk (hard or Flash disk)
- Boot via Clear - Instructs the CSS to boot the CSS from the secondary boot record

This section includes the following topics:

- Specifying a Network-Mounted File System as the Primary Boot Record
- Specifying FTP as the Primary Boot Record
- Specifying Disk as the Primary Boot Record
- Specifying Clear as the Primary Boot Record

## Specifying a Network-Mounted File System as the Primary Boot Record

Set **Network** as the primary boot record when you want to boot the CSS from a network-mounted file system on a remote system using FTP. Instead of the CSS disk, the network file system contains the CSS software. The CSS boots from this file system and loads the configuration into memory.

Perform a network boot when:

- You want multiple CSSs to use the same boot image while keeping their own configuration information. You provide an alternate path for the location of the configuration information. However, this information must exist on the same network file system with the boot image.

  **Note** When using an alternate configuration path, make sure the path leads to a directory containing the script, log, and info subdirectories. These subdirectories must contain the files in the corresponding subdirectories in the boot image. First, create these subdirectories on the FTP server, then copy the files from the boot image to the subdirectories.

- The CSS has a disk failure. A network boot allows the CSS to boot independently from its disk and to load the configuration into memory.

Before the CSS can boot from the network:

- Locate the remote system on the network where you want to copy the CSS software.
  - Make sure the CSS can access the system using FTP.
  - Copy the CSS software .zip file from www.cisco.com onto the designated network server.
  - Create a directory and unzip the file into it. This directory will contain all of the boot files and directories.
- On the CSS, create an FTP record to the directory containing the CSS software on the network drive.

- Make sure you properly connect the Ethernet management port on the CSS to the network. Note the locations of the Ethernet management port on the CSS as listed below.
  - CSS 11503 and CSS 11506 - SCM 10 Mbps-Ethernet management port
  - CSS 11501 - Front panel 10 Mbps-Ethernet management port
- Be aware of the following network boot restrictions:
  - A network boot is not supported on UNIX workstations.
  - The War-FTP daemon is not supported for network-booting the system software.

When you select **Network**, the CSS prompts you for the FTP kernel information.

1. Enter the FTP kernel path information. This path is the FTP daemon-addressable location where the boot image has been unpacked. You must also include the FTP server IP address and the username and password to access the boot image. For example:

```
Enter the FTP Kernel path:[] k:/sg0730002/hdd
Enter FTP Server IP address:[] 10.3.6.58
Enter FTP Server authentication username:[] mandy
Enter FTP Server authentication password:[] fred
```

2. Enter an alternate path to the configuration files, including the startup configuration and script files, if the configuration information is not in the same directory as the boot image.

**Note** The CSS must be able to access the configuration path through the previously configured FTP server IP address, login username, and password.

For example:

```
Enter the FTP Config Path? [] k:/atlanta-config/
Press <Enter> to continue...
```

3. Press **Enter** to display the Boot Configuration menu.
4. Enter **r** to display the Offline DM Main menu.

5. Select option **4** to reboot the CSS.

When the CSS completes the current boot process, it:

- Accesses the network file system containing the boot image
- Boots the CSS using the boot image you specified

## Specifying FTP as the Primary Boot Record

Select **FTP** as the primary boot record when you want to upgrade the CSS software on the CSS disk. The CSS accesses the ADI or GZIP file containing the CSS software from an FTP server, copies it to the disk, and unpacks the software. Then the CSS boots from disk (hard or Flash disk).

Make sure that you properly connect the Ethernet management port on the CSS to the network. Note the locations of the Ethernet management port on the CSS as listed below.

- CSS 11503 and CSS 11506 - SCM 10 Mbps-Ethernet management port
- CSS 11501 - Front panel 10 Mbps-Ethernet management port

When you select **FTP**, the CSS prompts you for the boot image filename and FTP information.

**Note** The CSS FTP server supports only the active (normal) FTP mode of operation. The FTP server does not support the passive FTP mode of operation.

1. Enter a valid FTP pathname, if required. For example:

```
Enter the boot image filename: /ftpimages/sg0730003
Enter FTP Server IP address: 10.3.6.58
Enter FTP Server authentication user name: mandy
Enter FTP Server authentication password: fred
```

The CSS queries if you want to access the boot image directly from the disk at the next reboot (that is, the next time you reboot the CSS after completing the current boot process).

```
Boot from Disk at next reboot? y/n
Press <Enter> to continue...
```

2. Enter one of the following:
   - **y** to copy the boot image from the FTP server to the disk. The CSS accesses the boot image directly from the disk at the next reboot. The CSS also changes the information in the primary boot record to Disk.
   - **n** to FTP the boot image from the FTP server at the next reboot.
3. Press **Enter** to display the Boot Configuration menu.
4. Enter **r** to display the Offline DM Main menu.
5. Select option **4** to reboot the CSS.

When the CSS completes the current boot process, it:

- Accesses the ADI file from the FTP server and unpacks (uncompresses) the file
- Boots the CSS using the boot image you specified

### Specifying Disk as the Primary Boot Record

When you select **Disk** as the primary boot record, the CSS displays all boot image versions that reside on the disk. For example:

```
sg0730002
sg0720104
```

1. Enter the boot image filename you wish to use at the prompt.

   ```
   Enter the boot image filename: sg0730002
   ```

2. Press **Enter** to display the Boot Configuration menu.

   ```
   Press <Enter> to continue...
   ```

3. Press **r** to display the Offline DM Main menu.
4. Select option **4** to reboot the CSS. Upon reboot, the CSS boots up using the boot image you specified.

### Specifying Clear as the Primary Boot Record

To use the secondary boot record information instead of the primary boot record to boot the CSS:

1. Select **Clear** as the primary boot record.
2. Press **Enter** to display the Boot Configuration menu.

3. Press **r** to display the Offline DM Main menu.
4. Select option **4** to reboot the CSS. Upon reboot, the CSS uses the secondary boot record.

## Setting Secondary Boot Configuration

The information you provide for the secondary boot configuration specifies the location from which the CSS accesses the boot image if you specified **Clear** as a primary boot record or if the primary boot record fails.

When you select **Set Secondary Boot Configuration** from the Boot Configuration menu, the CSS displays the following information. If you have previously entered information, the CSS displays the existing information and default values in square brackets [ ].

```
Configuring SECONDARY Boot Record
Boot via [N]etwork, [F]TP, [D]isk, or [C]lear: [D]
Press <Enter> to continue...
```

- Boot via Network - Allows you to boot the CSS using FTP from CSS software on a network-mounted file system on a remote system
- Boot via FTP - Allows you to download an ADI file containing CSS software that you want to install on the CSS disk
- Boot via Disk - Allows you to boot the CSS from software currently on the CSS disk
- Boot via Clear - Instructs the CSS to boot the CSS from the primary boot record

This section includes the following topics:

- Specifying a Network-Mounted File System as the Secondary Boot Record
- Specifying FTP as the Secondary Boot Record
- Specifying Disk as the Secondary Boot Record
- Specifying Clear as the Secondary Boot Record

## Specifying a Network-Mounted File System as the Secondary Boot Record

Select **Network** as the secondary boot record when you want to boot the system from a network-mounted file system on a remote system using FTP. Instead of the CSS disk, the network file system contains the CSS software. The CSS boots from this file system and loads the configuration into memory. Perform a network boot when:

- You want multiple CSSs to use the same boot image while keeping their own configuration information. You provide an alternate path for the location of the configuration information. However, this information must exist on the same network file system with the boot image.

**Note** When using an alternate configuration path, make sure the path leads to a directory containing the script, log, and info subdirectories. These subdirectories must contain the files in the corresponding subdirectories in the boot image. First, create these subdirectories on the FTP server, then copy the files from the boot image to the subdirectories.

- The CSS has a disk failure. A network boot allows the CSS to boot independently from its disk and to load the configuration into memory.

Before the CSS can boot from the network:

- Locate the remote system on the network where you want to copy the CSS software.
  - Make sure the CSS can access the system using FTP.
  - Copy the CSS software .zip file from www.cisco.com onto the designated network server.
  - Create a directory and unzip the file into it. This directory will contain all of the boot files and directories.
- On the CSS, create an FTP record to the directory containing the CSS software on the network drive.

- Make sure you properly connect the Ethernet management port on the CSS to the network:
  - CSS 11503 or CSS 11506 - SCM 10 Mbps-Ethernet management port
  - CSS 11501 - Front panel 10 Mbps-Ethernet management port
- Be aware of the following network boot restrictions:
  - A network boot is not supported on UNIX workstations.
  - The War-FTP daemon is not supported for network-booting the system software.

When you select **Network**, the CSS prompts you for the FTP kernel information.

1. Enter the FTP kernel path information. This path is the FTP daemon addressable location where the boot image has been unpacked. You must also include the FTP server IP address and the username and password to access the boot image. For example:

```
Enter the FTP Kernel path:[] k:/sg0720104/hdd
Enter FTP Server IP address:[] 10.3.6.58
Enter FTP Server authentication username:[] mandy
Enter FTP Server authentication password:[] fred
```

2. Enter an alternate path to the configuration files, including the startup-config and script files, if the configuration information is not in the same directory as the boot image.

**Note** The CSS must be able to access the configuration path through the previously configured FTP server IP address, login username, and password.

For example:

```
Enter the FTP Config Path? [] k:/atlanta-config/
Press <Enter> to continue...
```

3. Press **Enter** to display the Boot Configuration menu.
4. Enter **r** to display the Offline DM Main menu.
5. Select option **4** to reboot the CSS.

When the CSS completes the current boot process, it:

- Accesses the network file system containing the boot image
- Boots the CSS using the boot image you specified

### Specifying FTP as the Secondary Boot Record

Select **FTP** as the secondary boot record value when you want to upgrade the CSS software on the CSS disk. The CSS accesses the ADI or GZIP file containing the CSS software from an FTP server, copies it to the disk, and unpacks the software. Then the CSS boots from disk (hard or Flash disk).

Make sure you properly connect the Ethernet management port on the CSS to the network:

- CSS 11503 or CSS 11506 - SCM 10 Mbps-Ethernet management port
- CSS 11501 - Front panel 10 Mbps-Ethernet management port

When you select **FTP**, the CSS prompts you for the boot image filename and FTP information.

1. Enter a valid FTP pathname, if required. For example:

   ```
   Enter the boot image filename: /ftpimages/sg0720104
   Enter FTP Server IP address: 10.3.6.58
   Enter FTP Server authentication user name: mandy
   Enter FTP Server authentication password: fred
   ```

   The CSS queries if you want to access the boot image directly from the disk at the next reboot (that is, the next time you reboot the CSS after completing the current boot process).

   ```
   Boot from Disk at next reboot? y/n
   ```

2. Enter one of the following:
   - **y** to copy the boot image from the FTP server to the disk. The CSS accesses the boot image directly from the disk at next reboot. The CSS also changes the information in the secondary boot record to Disk.
   - **n** to FTP the boot image from the FTP server at next reboot.
3. Press **Enter** to display the Boot Configuration menu.

   ```
   Press <Enter> to continue...
   ```

4. Enter **r** to display the Offline DM Main menu.

5. Select option **4** to reboot the CSS.

   When the CSS uses the secondary boot record to reboot, it:

   - Accesses the ADI file from the FTP server and unpacks (uncompresses) the file
   - Boots the CSS using the boot image you specified

### Specifying Disk as the Secondary Boot Record

When you select **Disk** as the secondary boot record, the CSS displays all boot image versions that reside on the disk and prompts you to enter a boot image.

1. Enter a boot image filename.

   ```
   Boot via [N]etwork, [F]TP, [D]isk, or [C]lear: [D]

   ap0730002
   ap0720104

   Enter the boot image filename: sg0720104
   ```

2. Press **Enter** to display the Boot Configuration menu.

   ```
   Press <Enter> to continue...
   ```

3. Enter **r** to display the Offline DM Main menu.
4. Select option **4** to reboot the CSS. Upon reboot, the CSS boots up using the boot image you specified.

### Specifying Clear as the Secondary Boot Record

If you do not wish to specify a secondary boot record:

1. Select **Clear** as the secondary boot record.
2. Press **Enter** to display the Boot Configuration menu.
3. Enter **r** to display the Offline DM Main menu.
4. Select option **4** to reboot the CSS. Upon reboot, the CSS uses the primary boot record.

## Setting IP Address, Subnet Mask, and Default Gateway

When you select **Set IP Address and Subnet Mask** from the Boot Configuration menu, the CSS prompts you to:

1. Enter an IP address for the Ethernet management port.

   An IP address of 0.0.0.0 for the Ethernet management port is a legal setting and disables the management port upon reboot. If you enter 0.0.0.0, and you attempt to use the **subnet mask** command, the following message appears:

   ```
   The mask cannot be set because the IP address is 0.0.0.0.
   ```

   To enable the Ethernet management port, use the Boot Configuration menu to enter an IP address for the management port.

   The Ethernet management port IP address must be a different subnet from any other CSS VLAN circuit IP subnet. If you do not make the Ethernet management port IP address unique, you cannot access the port.

   The first time that you configure an IP address for the Ethernet management port, the CSS automatically assigns a default subnet mask of 255.255.255.0. If necessary, you can overwrite the default subnet mask with a mask that is more appropriate for your application.

   ```
   Enter IP Address or 0.0.0.0 to disable [<current value>]:
   10.3.6.58
   ```

2. Enter a subnet mask.

   ```
   Enter Subnet Mask: 255.0.0.0
   ```

3. Enter a default gateway address for the Ethernet management port.

   ```
   Enter Default Gateway: 172.16.11.1
   ```

4. Press **Enter** to display the Boot Configuration menu.

   ```
   Press <Enter> to continue...
   ```

5. Enter **r** to display the Offline DM Main menu.
6. Select option **4** to reboot the CSS.

## Displaying the Boot Configuration

When you select **Show Boot Configuration** from the Offline DM Main menu, the CSS displays the following boot information. The Miscellaneous information appears only if you set password-protection on the Offline DM Main menu.

```
***************** Miscellaneous ********************
Offline Diagnostic Monitor menu is password-protected
***************** IP/MAC Information ***************
IP Address:10.3.6.58
Subnet Mask:255.0.0.0
Gateway Address:172.16.11.1
MAC Address00-10-58-00-12-ca
***************** PRIMARY **************************
Boot Type:DISK
Boot File:sg0730002
***************** SECONDARY ************************
Boot Type: DISK
Boot File: sg0720104
```

1. Press **Enter** to display the Offline DM Main menu.

   ```
   Press <Enter> to continue...
   ```

2. Press **r** to display the Offline DM Main menu.

## Using the Advanced Options

The CSS hard disk or Flash disk enables you to store two versions of software (including the version you are currently running). If you are storing the maximum number of software versions on the CSS and wish to download a new version to the disk, you must delete a version before the CSS allows the download to begin.

When you select **Advanced Options** from the Offline DM Main menu, the CSS displays the Advanced Options menu:

```
A D V A N C E D  O P T I O N S

Enter the number of a menu selection:

1. Delete a Software Version
2* Security Options
3* Disk Options
4. Set MSD Mapping
r. Return to previous menu
```

## Deleting a Software Version

To delete a software version from the disk:

1. Select option **1** to display the software versions currently stored on the disk. The CSS prompts you to enter the software version to delete. For example:

   ```
   sg0730002
   sg0720104

   Enter the software version to delete: sg0720104
   ```

**Warning** **Ensure you do not delete the software version that you are currently running.**

2. Press **Enter** to display the Advanced Options menu.

   ```
   Press <Enter> to continue...
   ```

3. Enter **r** to display the Offline DM main menu.
4. Select option **4** to reboot the CSS.

## Using the Security Options

The Security Options menu enables you to:

- Set password protection on the Offline DM menu
- Change the administrative username and password

The Security Options menu is as follows:

```
S E C U R I T Y  O P T I O N S

Enter the number of a menu selection:

1. Set Password Protection for Offline Diagnostic Monitor
2. Set Administrative Username and Password
r. Return to previous menu
```

### Setting Password Protection

The CSS allows you to password-protect the Offline DM Main menu against unauthorized access. The default is disabled; no password is required to access the Offline DM Main menu.

**Caution** Use care when password protecting the Offline DM Main menu and ensure you take adequate steps to record the password. If you lose the new password, it cannot be recovered and you cannot access the Offline DM Main menu. At that point, the only solution would be to contact the Cisco Technical Assistance Center (TAC). Refer to the "Obtaining Technical Assistance" section in the Preface.

To access the Offline DM Main menu password protection option:

1. Select option **1** from the Security Options menu.

   ```
   Password protect Offline Diagnostic Monitor menu (yes,no):
   The administrative username and password are required to access
   the Offline Diagnostic Monitor menu.
   ```

   - If you enter **yes**, the CSS prompts you to enter a username and password when you access the Offline DM Main menu.
   - If you enter **no**, the CSS does not prompt you for a username and password when you access the Offline DM Main menu.

2. Press **Enter** to display the Security Options menu.

   ```
   Press <Enter> to continue...
   ```

3. Enter **r** to return to the Advanced Options menu.
4. Enter **r** to return to the Offline DM Main menu.
5. Enter **4** to reboot the CSS, or select another option to continue using the Offline DM Main menu.

## Changing the Administrative Username and Password

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

**Note** You may also use the **username-offdm** CLI command to change the default administrative username and password (refer to the *Cisco Content Services Switch Getting Started Guide*).

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

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

To change the administrative username and password through the Offline DM Main menu:

1. Select option **2** from the Security Options menu.
2. Enter a username. The CSS prompts you for this username when you log in. The CSS also prompts you for this username and password if you set password protection on the Offline DM Main menu.

   ```
   Enter <administrator> username (minimum 4 characters):
   ```

3. Enter a password. Note that the CSS does not display passwords.

   ```
   Enter <administrator> password:
   ```

4. Reenter the password for confirmation.

   ```
   Confirm <administrator> password:
   ```

   The CSS displays the Security Options menu.
5. Enter **r** to return to the Advanced Options menu.
6. Enter **r** to return to the Offline DM Main menu.
7. Enter **4** to reboot the CSS, or select another option to continue using the Offline DM Main menu.

## Using the Disk Options

The Disk Options menu includes the following menu items:

- **Format Disk** - Reformats the disk. This option permanently erases all data on the disk. If you wish to retain the startup configuration, be sure to move it off the CSS before reformatting the disk. Also make sure you have a copy of the CSS software ADI file to reinstall on the CSS.
- **Check Disk** - Runs a quick check disk or a complete check of the disk.
- **Check Disk Disable** - Disables running a check of the disk at boot time or enable it again. By default, check disk is enabled.

**Note** We do not recommend running a Flash disk with the **Check Disk Disable** option selected.

The Disk Options menu is as follows:

```
D I S K   O P T I O N S

Enter the number of a menu selection:

1. Format Disk
2. Check Disk
3. Check Disk Disable
r. Return to previous menu
```

### Reformatting the Disk

If the CSS detects unrecoverable errors when performing a disk check, you must reformat the disk. Reformatting the disk erases all data from the disk permanently.

To reformat the disk:

1. Select option **1** from the Disk Options menu.

   If the CSS contains two disks, you see the following prompt:

   ```
   Format volume in which PCMCIA slot? (0,1):
   ```

2. Enter **0** (for slot 0) or **1** (for slot 1).

3. Press **Enter** to continue.

   The CSS prompts you to format the disk.

   ```
   Formatting the disk results in all disk data being permanently
   erased.
   Are you sure you want to continue? (yes,no):
   ```

4. Enter one of the following:
   - **yes** to reformat the disk.
   - **no** to stop the reformat function. If the disk has unrecoverable errors and you do not reformat it, be aware that the file system may be corrupted and functionality compromised.

   The CSS prompts you to perform a quick format or a complete format.

   ```
   Quick format? (yes,no):
   ```

5. Enter one of the following:
   - **yes** to reformat the disk using the quick format (does not perform cluster verification). Use the quick format only when you are certain of the disk integrity.
   - **no** to reformat the disk including cluster verification.

   After the CSS reformats the disk, it displays:

   ```
   Operation completed successfully.
   ```

6. Enter **r** to return to the Advanced Options menu.
7. Enter **r** to return to the Offline DM Main menu.

   Because the disk is empty, you must configure a primary boot record to instruct the CSS where to locate the new ADI file containing the CSS software.

8. Select option **1** to set the primary boot configuration (see the "Setting Primary Boot Configuration" section).

   If you do not set the primary boot configuration before booting the CSS, the boot process halts at the prompt:

   ```
   Would you like to access the Offline Diagnostic Monitor
   menu?(y<cr>)
   ```

   You must enter the Offline DM Main menu to set the primary boot configuration.

## Performing a Check Disk

During the check disk process, the CSS detects and recovers from the following error conditions:

- File Allocation Tables (FATs) are out of synchronization
- Sector write truncation revitalization (may occur from a power loss at the time the CSS is writing to the disk)
- Bad cluster identification (of all in-use cluster) and mapping in the FAT when reformatting the disk
- Crosslinked FAT entries
- Disk entry validation, name, size, cluster assignment, cluster chaining
- Recovery of lost clusters

The amount of time the CSS requires to perform a disk check is proportional to the number of installed files and directories on the disk. The greater the number of installed files and directories, the longer the CSS takes to complete the disk check.

**Note** The CSS cannot recover from sector failures located within the first 754 sectors on the disk (for example, boot, primary/secondary FAT, or root directory entries). If a sector failure occurs, use the Format Disk option of the Disk Options menu (see the "Reformatting the Disk" section). If the CSS is unable to properly format the CSS disk after using the Format Disk option, contact TAC.

To perform a disk check:

1. Select option **2** from the Disk Options menu. The CSS prompts you about correcting errors if discovered.

```
Correct errors if discovered (yes,no):
```

2. Choose whether you want the CSS to correct errors it detects. Enter one of the following:
   - **yes** to enable the CSS to correct recoverable errors it detects. When the CSS completes the disk check, it displays a summary of what was fixed.
   - **no** to prevent the CSS from correcting recoverable errors it detects. The CSS displays a summary of what would have been fixed if you had run the disk check.

   The CSS prompts you to perform a quick check of the disk.

   ```
   Quick check disk? (yes,no):
   ```

3. Choose whether you want the CSS to perform a quick disk check or a complete disk check. Enter either:
   - **yes** to instruct the CSS to perform a quick disk check (does not include cluster verification). Use quick disk check only when you are certain of the disk integrity.
   - **no** to instruct the CSS to perform a complete disk check, including cluster verification.

   The CSS performs the disk check. When completed the CSS displays:

   ```
       c:\ - Volume is OK (\)
       Press <Enter> to continue...
   ```

4. Enter **r** to return to the Advanced Options menu.
5. Enter **r** to return to the Offline DM Main menu.
6. Select option **4** to reboot the CSS.

## Disabling or Enabling Disk Check

The Disk Options menu provides an option to disable or enable the running of check disk. When you select this option, it toggles to disable check disk if the option is currently enabled, or to enable check disk if the option is currently disabled.

**Note** We do not recommend running a Flash disk with the **Check Disk Disable** option selected.

For example, if check disk is currently enabled, to disable it:

1. Select option **3** from the Disk Options menu.
2. Enter **r** to return to the Advanced Options menu.
3. Enter **r** to return to the Offline DM Main menu.
4. Select option **2** to display the boot configuration.

   When check disk is disabled, the CSS displays the following:

```
****************** Miscellaneous **********************
Check Disk is disabled
***************** IP/MAC Information ***************
IP Address:    10.3.6.58
Subnet Mask:   255.0.0.0
Gateway Address:172.16.11.1
MAC Address:   00-10-58-00-12-ca
***************** PRIMARY **************************
Boot Type:     DISK
Boot File:     sg0730002
***************** SECONDARY ***********************
Boot Type:     DISK
Boot File:     sg0720104
Press <Enter> to continue...
```

If check disk is currently disabled, to reenable it:

1. Select option **3** from the Disk Options menu.
2. Enter **r** to return to the Advanced Options menu.
3. Enter **r** to return to the Offline DM Main menu.

4. Select option **2** to display the boot configuration.

   When check disk is enabled, no state information appears in the Miscellaneous field of the boot configuration:

```
***************** IP/MAC Information ***************
IP Address:        10.3.6.58
Subnet Mask:       255.0.0.0
MAC Address:       00-10-58-00-12-ca
***************** PRIMARY **************************
Boot Type:         DISK
Boot File:         sg0730002
***************** SECONDARY ************************
Boot Type:         DISK
Boot File:         sg0720104
Press <Enter> to continue...
```

## Configuring Disks in a Two-Disk CSS

The CSS 11501 and the SCM in the CSS 11503 and CSS 11506 contain two PCMCIA slots for a hard disk or Flash disk. These disks contain the CSS system software and are also used for logging and storing offline system files. The two disks are identified by the PCMCIA slots (slot 0 and slot 1) in which they are installed. Disk 0 is the default storage location for the primary and secondary boot records in the CSS. The default storage location for log files and core dumps in the CSS is the specified disk from which the CSS boots (disk 0 or disk 1).

To specify the CSS disk that is to be the storage location for booting (primary and secondary boot disk), for logging output, and for core dumps:

1. Select option **4** from the Advanced Options menu to configure the disks in the active SCM.

   The CSS prompts you to specify the disk for the primary boot record.

```
Set Primary-Boot to which PCMCIA slot? (0,1):
```

2. Enter either **0** for the disk in slot 0 (the default setting) or **1** for the disk in slot 1.

   The CSS prompts you to specify the disk for the secondary boot record.

```
Set Secondary-Boot to which PCMCIA slot? (0,1):
```

3. Enter either **0** for the disk in slot 0 (the default setting) or **1** for the disk in slot 1.

   The CSS prompts you to specify the disk for core dump files when the CSS experiences a fatal error.

   ```
   Set Core to which PCMCIA slot? (0,1):
   ```

4. Enter either **0** for the disk in slot 0 or **1** for the disk in slot 1.

**Note** Core dump information is intended for Cisco Technical Assistance Center (TAC) use only.

   The CSS prompts you to specify the disk for writing information to log files.

   ```
   Set Log to which PCMCIA slot? (0,1):
   ```

5. Enter either **0** for the disk in slot 0 or **1** for the disk in slot 1.
6. Enter **r** to return to the Advanced Options menu.
7. Enter **r** to return to the Offline DM Main menu.
8. Select option **4** to reboot the CSS.

## Rebooting the CSS

To reboot the CSS from the Offline DM Main menu:

1. Select option **4** to reboot the system. The following reboot confirmation is displayed:

   ```
   Are you sure you want to reboot? (y/n)
   ```

2. Enter either **y** to reboot or **n** to continue using the Offline DM Main menu.

When the CSS completes the current boot process, it:

- Accesses the network file system containing the boot image
- Boots the CSS using the boot image you last specified

# INDEX