Cisco Reader Comment Card

General Information

1 Years of networking experience: _____ Years of experience with Cisco products: _____

2 I have these network types: ☐ LAN ☐ Backbone ☐ WAN
☐ Other: _____

3 I have these Cisco products: ☐ Switches ☐ Routers
☐ Other (specify models): _____

4 I perform these types of tasks: ☐ H/W installation and/or maintenance ☐ S/W configuration
☐ Network management ☐ Other: _____

5 I use these types of documentation: ☐ H/W installation ☐ H/W configuration ☐ S/W configuration
☐ Command reference ☐ Quick reference ☐ Release notes ☐ Online help
☐ Other: _____

6 I access this information through: ____% Cisco.com (CCO) ____% CD-ROM
____% Printed docs ____% Other: _____

7 I prefer this access method: _____

8 I use the following three product features the most:

Document Information

Document Title: Cisco Application and Content Networking Software Caching Configuration Guide

Part Number: 78-13950-01 S/W Release: 4.1

On a scale of 1–5 (5 being the best), please let us know how we rate in the following areas:

_____ The document is written at my technical level of understanding.

_____ The information is accurate.

_____ The document is complete.

_____ The information I wanted was easy to find.

_____ The information is well organized.

_____ The information I found was useful to my job.

Please comment on our lowest scores:

Mailing Information

Company Name Date

Contact Name Job Title

Mailing Address

City State/Province ZIP/Postal Code

Country Phone ( ) Extension

Fax ( ) E-mail

Can we contact you further concerning our documentation? ☐ Yes ☐ No

You can also send us your comments by e-mail to **bug-doc@cisco.com**, or by fax to **408-527-8089**.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 4631 SAN JOSE CA

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DOCUMENT RESOURCE CONNECTION
CISCO SYSTEMS INC
170 WEST TASMAN DRIVE
SAN JOSE CA 95134-9883

Cisco Application and Content Networking System Software Caching Configuration Guide

Release 4.1

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

Customer Order Number: DOC-7813950=
Text Part Number: 78-13950-01

Cisco Application and Content Networking System Software Caching Configuration Guide

CONTENTS

Preface xi

Audience xi

Document Organization xi

Document Conventions xiii

Related Documentation xiii

Obtaining Documentation xiv

World Wide Web xiv

Documentation CD-ROM xiv

Ordering Documentation xiv

Documentation Feedback xiv

Obtaining Technical Assistance xv

Cisco.com xv

Technical Assistance Center xv

Cisco TAC Web Site xvi

Cisco TAC Escalation Center xvi

PART 1 **Content Engine Management Overview**

CHAPTER 1 **Cisco Cache Application Product Overview** 1-1

Cache Application Overview 1-1

Advanced Transparent Caching Service 1-1

Configuring Caching Management Features 1-2

Logging On to the Graphical User Interface 1-2

Administering Caching from the Cache Application Graphical User Interface 1-4

Secure Shell Version 1 Support for Login 1-5

System Logging 1-6

Mapping Syslog Priority Levels to RealProxy Error Codes 1-7

Network Time Protocol Time Synchronization 1-7

CHAPTER 2 **Cisco Cache Application Basic WCCP Configuration** 2-1

Basic Cache Application Router Configurations for HTTP Traffic and WCCP Version 2 2-2

Web Cache Service Configuration with Clients and Cache on Different Subnets 2-2

Configuring the Content Engine for Web Cache Service—Clients and Cache on Different Subnets 2-3

Configuring the Router for Web Cache Service—Clients and Cache on Different Subnets 2-3
Configuration Examples—Web Cache Service with Clients and Cache on Different Subnets 2-3
Web Cache Service Configuration with Clients and Cache on Same Subnet 2-4
Configuring the Content Engine for Web Cache Service—Clients and Cache on Same Subnet 2-5
Configuring the Router for Web Cache Service—Clients and Cache on Same Subnet 2-5
Configuration Examples—Web Cache Service with Clients and Cache on the Same Subnet 2-6
Configuring Reverse Proxy Service 2-7
Configuring the Content Engine for Reverse Proxy Service with Output Redirection 2-8
Configuring the Router for Reverse Proxy Service with Output Redirection 2-8
Configuring the Router for Reverse Proxy Service with Input Redirection 2-9
Configuration Examples—Reverse Proxy with Output Redirection 2-9
Disabling Transparent Caching Services on the Content Engine 2-10
Verifying the Software Configuration 2-11
Removing or Replacing a Content Engine 2-11

PART 2 **Content Engine Features**

CHAPTER 3 **Cisco Application and Content Networking Software Cache Application Features 3-1**
Cache Application Feature Set Table 3-1

CHAPTER 4 **Configuring Transparent Caching 4-1**
Transparent Caching Through WCCP 4-1
Transparent Mode Operation 4-2
Prerequisites 4-4
Configuration Tasks 4-4
Configuring the Content Engine 4-4
Enabling WCCP on the Router 4-4
Advanced Transparent Caching Features 4-5
Authentication Traffic Bypass 4-5
Dynamic Traffic Bypass 4-6
Scenario 1—Dynamic Bypass Upon Receiving a Web Server Error 4-6
Scenario 2—Dynamic Bypass Upon Receiving an Unsupported Protocol 4-6
Overload Bypass 4-7
Static Bypass 4-8
Examples 4-8
Multiport Transparent Redirection 4-9
Modifying Multiport Configurations 4-9

WCCP Flow Protection 4-9
Accelerated WCCP Layer 2 Support 4-10
Configuring Caching Services with the Cisco CSS 11000 Series Switch 4-11
CSS 11000 Switch Caching Configurations 4-11
Transparent Caching with the Cisco CSS 11000 Series Switch 4-11
Content Engine CLI Commands 4-12
Enabling Transparent Caching using the CSS 11000 4-12
Transparent Caching Sample Configuration Output 4-13

CHAPTER 5 **Configuring Proxy-Style Caching 5-1**
Proxy Mode Operation 5-1
HTTP Proxy Caching 5-2
FTP Proxy Caching 5-2
Examples 5-3
SSL Tunneling 5-3
HTTPS Proxy Features 5-4
Examples 5-5

CHAPTER 6 **Configuring Cache Parameter Settings 6-1**
Caching of Authenticated Content 6-1
Cache Freshness 6-1
Minimum and Maximum Time To Live Settings 6-2
Caching of Binary Content with Cookies 6-2
Maximum Object Size 6-3
Aborting Selected Objects 6-3
Caching of HTTP Range Requests 6-4

CHAPTER 7 **Configuring a Primary Proxy Server 7-1**
Overview 7-1
Parent Proxy Failover 7-2
Examples 7-3
Related Commands 7-4
Handling Proxy-Style Requests 7-5
Internet Cache Protocol 7-6

CHAPTER 8 **Configuring URL Filtering 8-1**
URL Filtering 8-1

Order of Precedence 8-1
URL Filtering with URL Lists 8-1
Custom Blocking Messages 8-3
URL Filtering with the N2H2 Server 8-4
N2H2 Features Supported 8-4
N2H2 CLI Commands 8-5
N2H2 Status and Statistics Commands 8-5
N2H2 Configuration Through the Content Engine GUI 8-6
N2H2 Configuration and Restrictions 8-6
URL Filtering with the Websense Enterprise Server 8-6
URL Filtering with SmartFilter Software 8-7

CHAPTER 9 **Configuring Transaction Logging** 9-1
Overview 9-1
Squid-Style Transaction Logging 9-1
Extended Squid-Style Transaction Logging 9-3
Apache-Style Transaction Logging 9-3
Sanitized Transaction Logs 9-4
Exporting Log Files 9-4
Exporting Transaction Logs to External FTP Servers 9-4
Restarting Export After Receiving a Permanent Error from the External FTP Server 9-6

CHAPTER 10 **Configuring Authentication** 10-1
User Authentication 10-1
TACACS+ Options for User Authentication 10-2
HTTP Request Authentication 10-3
NTLM Authentication 10-3
NTLM Authentication Transparency 10-4
RADIUS HTTP Request Authentication 10-4
RADIUS Authentication Redirection 10-4
LDAP HTTP Request Authentication 10-5
HTTP Request Considerations 10-5
Excluding Domains from HTTP Authentication Servers 10-6
Proxy Mode Server Authentication 10-6
Transparent Mode Authentication 10-6
Server Redundancy 10-7
Security Options 10-7
Hierarchical Caching 10-7

Hierarchical Caching in Transparent Mode 10-8
Hierarchical Caching, Content Engine in Transparent Mode with an Upstream Proxy 10-8
Authentication Cache Size Adjustments 10-9
Transaction Logging 10-9
End-to-End Authentication 10-9
Basic End-to-End Authentication 10-10
NTLM End-to-End Authentication 10-10

CHAPTER 11 **Configuring Network Management** 11-1
Simple Network Management Protocol Overview 11-1
Versions of SNMP 11-2
Key SNMP CLI Commands 11-3
Supported MIBs 11-4
SNMP Traps 11-4
CiscoWorks2000 11-5
Cisco Discovery Protocol 11-5

CHAPTER 12 **Configuring TCP Stack Parameters** 12-1
Overview 12-1
Configuring TCP Parameters Using the Content Engine GUI 12-2
TCP-Over-Satellite Extensions 12-3
TCP Configuration Examples 12-4

CHAPTER 13 **Configuring Windows Media Technologies 7.01 Streaming Media Caching** 13-1
Streaming Media Overview 13-1
Configuring Microsoft Windows Media Player 7.01 13-2
WMT Caching Proxy Details 13-2
Caching 13-3
Variable Bit Rate 13-3
Live Splitting 13-4
Proxy Authentication 13-4
Windows NTLM Authentication 13-5
Miscellaneous Features 13-5
Transaction Logging 13-5
Error Logging 13-5
Statistics 13-5
WMT Requirements 13-6
Enabling WMT on the Content Engine 13-6

Enabling Transparent WMT Service Using WCCP-Enabled Routers 13-6
Requirements 13-6
Procedure 13-7
Enabling Conventional WMT Proxy Service 13-8
Requirements 13-8
Procedure 13-9
WMT Multicasting 13-12
WMT Multicast and Broadcast CLI Commands 13-12
WMT Multicast 13-12
WMT Broadcast 13-13
Unicast-in Multicast-out 13-13
Multicast-in Multicast-out 13-14
Multicast-in Unicast-out 13-15
WMT Examples 13-15

CHAPTER 14 **Configuring RealProxy 8.01 Streaming Media Caching** 14-1
Configuring RealProxy 8.01 14-1
Configuring the RealProxy Software 14-3
Statistics 14-4
Enabling Transparent RTSP Proxy Service Using WCCP-Enabled Routers 14-4
Requirements 14-4
Procedure 14-4
Enabling Conventional RTSP Proxy Service 14-5
Requirements 14-5
Procedure 14-5
RealProxy Considerations 14-10
Disabling RealMedia Caching 14-11
Streaming On-Demand Clips and RealProxy 14-11
Unicasting, Splitting, Multicasting, and RealProxy 14-11
RealProxy and Access Control 14-11
RealProxy Examples 14-12

CHAPTER 15 **Configuring the Rules Template** 15-1
Rules Template 15-1
Actions and Patterns 15-2
Rules Template Processing Considerations 15-5
Examples 15-7

CHAPTER 16 **Miscellaneous Features** 16-1

Configuring the Content Engine as a Content Routing Agent 16-1
Examples 16-2
Browser Autoconfiguration 16-3
Configuring Healing Mode 16-3
Examples 16-4
Content Preloading 16-5

APPENDIX A **Web Cache Communication Protocol Version 1** A-1

Feature Overview A-1
Benefits A-2
Redirection Process A-2
Related Documents A-3
Prerequisites A-3
Configuration Tasks A-3
Configuring the Content Engine A-3
Configuring the Router A-3
Enabling WCCP on the Router A-4
Monitoring WCCP Version 1 A-4
Configuration Example A-4
Command Reference A-5
Debug Commands A-19

APPENDIX B **Web Cache Communication Protocol Version 2** B-1

Feature Overview B-1
Multirouter Support B-2
How Version 1 Works B-2
How Version 2 Works B-2
How Routers and Content Engines Communicate B-4
Improved Security B-4
Improved Throughput B-4
Redirection for Multiple TCP Port-Destined Traffic B-5
Web Cache Packet Return B-5
Load-Distributing Applications B-5
Client IP Address Transparency B-6
Restrictions B-6
Related Documents B-6
Prerequisites B-6

Configuration Tasks **B-7**
Configuring a Service Group Using WCCP Version 2 **B-7**
Running the Web Cache Service **B-7**
Running the Reverse Proxy Service **B-8**
Running a Custom Web Cache Service **B-8**
Running a Dynamic Web Cache Service **B-8**
Registering a Router to a Multicast Address **B-8**
Informing a Router of Valid IP Addresses **B-9**
Setting a Password for a Router and Content Engines **B-9**
Disabling Caching for Certain Clients **B-9**
Verifying WCCP Configuration Settings **B-9**
Monitoring and Maintaining WCCP Version 2 **B-11**
Configuration Examples **B-12**
Performing a General WCCP Version 2 Configuration **B-12**
Running the Web Cache Service **B-12**
Running the Reverse Proxy Service **B-12**
Running the Custom Web Cache Service **B-13**
Running a Generic Web Cache Service **B-13**
Registering a Router to a Multicast Address **B-13**
Informing a Router of Valid IP Addresses **B-13**
Setting a Password for a Router and Content Engines **B-13**
Bypassing the Cache with Router Access Lists **B-14**
Displaying WCCP Settings **B-14**
Command Reference **B-15**

INDEX

Preface

This preface describes who should read the *Cisco Application and Content Networking Software Caching Configuration Guide*, how it is organized, and its document conventions. This preface contains the following sections:

- Audience, page xi
- Document Organization, page xi
- Document Conventions, page xiii
- Related Documentation, page xiii
- Obtaining Documentation, page xiv
- Obtaining Technical Assistance, page xv

Audience

This guide is intended for network system administrators familiar with Cisco router and switch configuration. An understanding of caching concepts is necessary. This guide is not a tutorial.

Document Organization

This guide includes the following chapters:

Chapter	Title	Description
Chapter 1	Cisco Cache Application Product Overview	Provides a basic product overview of the Content Engine and procedures for managing the Content Engine using either the graphical user interface (GUI) or the command-line interface (CLI).
Chapter 2	Cisco Cache Application Basic WCCP Configuration	Contains basic configurations using WCCP for transparent, proxy, and reverse proxy caching.
Chapter 3	Cisco Application and Content Networking Software Cache Application Features	Lists the features supported by the Content Engine.

Chapter	Title	Description
Chapter 4	Configuring Transparent Caching	Explains transparent caching and shows configuration examples relevant to the Content Engine.
Chapter 5	Configuring Proxy-Style Caching	Explains proxy-style caching and presents configuration examples relevant to nontransparent (proxy-style) caching with the Content Engine.
Chapter 6	Configuring Cache Parameter Settings	Describes the caching parameter settings that need to be configured on the Content Engine.
Chapter 7	Configuring a Primary Proxy Server	Explains how to assign primary Content Engines and how to configure failover.
Chapter 8	Configuring URL Filtering	Lists the different methods used to filter content based on URL lists.
Chapter 9	Configuring Transaction Logging	Explains transaction logging and describes the different transaction logging formats available with the Content Engine.
Chapter 10	Configuring Authentication	Discusses authentication support on the Content Engine.
Chapter 11	Configuring Network Management	Explains how SNMP works and details additional network management agents available, including MIBs.
Chapter 12	Configuring TCP Stack Parameters	Describes TCP stack parameters and how best to optimize them for caching purposes.
Chapter 13	Configuring Windows Media Technologies 7.01 Streaming Media Caching	Describes how Windows Media Technology streaming media is supported on the Content Engine.
Chapter 14	Configuring RealProxy 8.01 Streaming Media Caching	Describes how RealProxy streaming media is supported on the Content Engine.
Chapter 15	Configuring the Rules Template	Describes how to use the Rules Template to allow for requests using various request parameters.
Chapter 16	Miscellaneous Features	Describes miscellaneous features associated with the Content Engine.
Appendix A	Web Cache Communication Protocol Version 1	Describes WCCP Version 1.
Appendix B	Web Cache Communication Protocol Version 2	Describes WCCP Version 2.

Document Conventions

This guide uses basic conventions to represent text and table information.

Convention	Description
boldface font	Commands and keywords are in **boldface**.
italic font	Variables for which you supply values are in *italics*.
`screen` font	Terminal sessions and information the system displays are printed in `screen` font.
`boldface screen` font	Information you must enter is in **`boldface screen`** font.
italic screen font	Variables you enter are printed in *`italic screen`* font.
{ }	Elements in braces are required input.
[ ]	Elements in square brackets are optional.
{**x** | **y** | **z**}	Required alternative keywords are grouped in braces and separated by vertical bars.
[**x** | **y** | **z**]	Optional keywords are grouped in brackets and separated by vertical bars.

Note Means *reader take note*. Notes contain helpful suggestions or references to materials not contained in the manual.

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Related Documentation

For additional information on the Cisco Content Engine, refer to the following documentation:

- *Cisco Content Networking Products Getting Started Guide, Release 4.1*
- *Cisco Application and Content Networking Software Command Reference, Release 4.1*
- *Cisco Application and Content Networking Software System Maintenance and Troubleshooting Guide*
- *Cisco Application and Content Networking Software E-CDN Administrator's Guide*
- *Cisco Content Networking Hardware Installation Guide for the Seven-Rack Unit Chassis*
- *Cisco Content Engine 500 Series Hardware Installation Guide*
- *Regulatory Compliance and Safety Information for the Cisco Content Networking Product Series*
- *Cisco Storage Array 6 Installation and Configuration Guide*
- *Cisco Storage Array 12 Installation and Configuration Guide*

Obtaining Documentation

The following sections explain how to obtain documentation from Cisco Systems.

World Wide Web

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

http://www.cisco.com

Translated documentation is available at the following URL:

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

Documentation CD-ROM

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

Ordering Documentation

Cisco documentation is available in the following ways:

- Registered Cisco Direct Customers can order Cisco product documentation from the Networking Products MarketPlace:

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

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

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

- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco corporate headquarters (California, USA) at 408 526-7208 or, elsewhere in North America, by calling 800 553-NETS (6387).

Documentation Feedback

If you are reading Cisco product documentation on Cisco.com, you can submit technical comments electronically. Click **Leave Feedback** at the bottom of the Cisco Documentation home page. After you complete the form, print it out and fax it to Cisco at 408 527-0730.

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

To submit your comments by mail, use the response card behind the front cover of your document, or write to the following address:

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

We appreciate your comments.

Obtaining Technical Assistance

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

Cisco.com

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

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

- Streamline business processes and improve productivity
- Resolve technical issues with online support
- Download and test software packages
- Order Cisco learning materials and merchandise
- Register for online skill assessment, training, and certification programs

You can self-register on Cisco.com to obtain customized information and service. To access Cisco.com, go to the following URL:

http://www.cisco.com

Technical Assistance Center

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

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

- Priority level 4 (P4)—You need information or assistance concerning Cisco product capabilities, product installation, or basic product configuration.
- Priority level 3 (P3)—Your network performance is degraded. Network functionality is noticeably impaired, but most business operations continue.

- Priority level 2 (P2)—Your production network is severely degraded, affecting significant aspects of business operations. No workaround is available.
- Priority level 1 (P1)—Your production network is down, and a critical impact to business operations will occur if service is not restored quickly. No workaround is available.

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

Cisco TAC Web Site

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

http://www.cisco.com/tac

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

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

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

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

If you have Internet access, it is recommended that you open P3 and P4 cases through the Cisco TAC Web Site.

Cisco TAC Escalation Center

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

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

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

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

PART 1

Content Engine Management Overview

CHAPTER 1

Cisco Cache Application Product Overview

Cache Application Overview

The Cache application portion of Cisco ACNS software deployed on Cisco Content Engines is one of the content delivery elements of the Content Delivery Network (CDN) solution from Cisco Systems. The CDN solution allows the proactive distribution of rich media files to Content Engines at the network edge for local access to e-business applications such as e-learning, e-commerce, knowledge sharing, and corporate communications. Designed for affordability and ease of installation, the CDN solution enables you to quickly deploy high-impact, high-bandwidth rich media, such as high-quality streaming video, with minimal administration.

Cisco Content Engines with Cache application software installed accelerate content delivery by caching frequently accessed content (transparently or proxy-style) and then locally fulfilling content requests rather than traversing the Internet or intranet to a distant server. This solution helps to protect your network from uncontrollable bottlenecks and accelerates the delivery of content, enabling service providers to offer higher service quality and enabling enterprise employees to be more productive. By caching content such as Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP) traffic, Cisco Content Engines minimize redundant network traffic that traverses WAN links. As a result, WAN bandwidth costs either decrease or grow less quickly. This bandwidth optimization increases network capacity for additional users or traffic and for new services, such as voice.

Advanced Transparent Caching Service

Cisco Content Engines offer advanced transparent caching technologies that include:

- Overload bypass—Prevents a Content Engine from becoming a bottleneck when traffic loads exceed the capacity of a Content Engine.
- Dynamic client bypass—Prevents source IP authentication problems by selectively allowing clients to directly connect to origin servers.
- Flow protection—Prevents existing flows from being broken when the Web Cache Communication Protocol (WCCP) cluster load distribution changes because of the addition or subtraction of a Content Engine into or from a cluster.
- WCCP slow start—Prevents cluster destabilization when a new Content Engine is added to a heavily loaded cluster.
- Rules Template—Enables flexible establishment of caching policies or rules, for example, “no-cache” policies, refresh policies, upstream proxy selection rules, and URL rewrite rules.

To integrate with existing proxy infrastructures, the Cisco ACNS software supports a number of proxied protocols, including FTP, Hypertext Transfer Protocol Secure (HTTPS), HTTP 1.0, and HTTP 1.1. With the Rules Template feature, administrators can establish proxy policies, providing control over how traffic is proxied.

The Cisco Content Engines can be deployed in front of a website (reverse proxy) to transparently cache inbound requests for content, significantly reducing the traffic and TCP connection maintenance performed by origin servers.

By supporting WCCP Version 2 or by interoperating with the Cisco CSS 11000 Series switches, a Content Engine can achieve a basic level of transparency that includes:

- Transparently receiving content traffic
- Fault tolerance
- Scalable clustering

Configuring Caching Management Features

You can configure the Cache application with the command-line interface (CLI) or with the Cache application graphical user interface (GUI). This guide contains mostly CLI configuration examples. However, GUI examples are shown whenever a feature requires them or to illustrate a feature through a screen capture.

For information on CLI commands, refer to the *Cisco Application and Content Networking Software Command Reference*.

Logging On to the Graphical User Interface

The graphical user interface has separate online help for those Cache application features supported with the graphical management interface.

To connect to the graphical user interface, perform the following steps.

Step 1 Start a web browser on a machine that has access to the network on which the Content Engine resides.

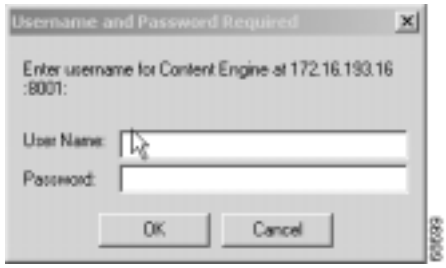
Note Be sure to enable Java, JavaScript, and Cascading Style Sheets on your Internet Explorer browser, or use Netscape 4.0 or later browser.

Step 2 Open the URL with the cache IP address specified in the initial Cache application configuration. Append the default port number 8001. For example:

```
http://172.16.13.8:8001
```

You are prompted for a username and password. (See Figure 1-1.)

Figure 1-1 Cache Application GUI—Authentication Challenge



Step 3 Enter a correct username and password. The Content Engine returns the graphical user interface home page, as shown in Figure 1-2.

If you forget your password, you must have another administrator reset your password. The password for the user **admin** is specified in the initial system configuration dialog.

Figure 1-2 Cache Application Graphical User Interface—Home Page

Administering Caching from the Cache Application Graphical User Interface

Graphical user interface (GUI) configuration pages exist for the following features:

- Authenticated cache
- Authentication
- Basic networking
- Bypass
- Cache on abort
- Content preload
- Disk configuration
- DNS
- FTP freshness
- FTP proxy
- HTTP freshness
- HTTP proxy
- HTTPS proxy
- Internet Cache Protocol (ICP) client
- ICP server
- Lightweight Directory Access Protocol (LDAP)
- NT LAN Manager (NTLM)
- Network Time Protocol (NTP)
- Persistent connection
- Proxy protocols
- RADIUS
- RealProxy
- Remote Authentication Dial-In User Service (RADIUS)
- Routing
- Rules Template
- SNMP
- Syslog
- TACACS+
- TCP
- Transaction logs
- URL filtering
- Users
- WCCP clustering
- WCCP custom web cache
- WCCP media caching

- WCCP reverse proxy
- WCCP services
- WCCP web caching
- WCCP WMT streaming
- WMT streaming

Access to the Cache application GUI can be controlled with multiple levels of username and password access, and access can be restricted to a subset of IP addresses (hosts). These access controls are configured with the **user** command and the **trusted-host** command, which are the same commands that you use to configure access to the CLI.

Note Be sure to enable Java, JavaScript, and Cascading Style Sheets on your Internet Explorer browser, or use Netscape 4.0 or later browser.

Secure Shell Version 1 Support for Login

Secure Shell (SSH) enables login access to the Content Engine through a secure and encrypted channel. SSH consists of a server and a client program. Like Telnet, you can use the client program to remotely log on to a machine that is running the SSH server, but unlike Telnet, messages transported between the client and the server are encrypted. The functionality of SSH includes user authentication, message encryption, and message authentication.

Before you enable the **sshd** command, use the **ssh-key-generate** command to generate a private and a public host key, which the client programs use to verify server's identity.

When a user runs an SSH client and logs in to the Content Engine, the public key for the SSH daemon running on the Content Engine is recorded in the client machine known_hosts file in the user's home directory. If the Content Engine administrator subsequently regenerates the host key by issuing the **ssh-key-generate** command, the user must delete the old public key entry associated with the Content Engine in the known_hosts file before running the SSH client program to log in to the Content Engine. When the user runs the SSH client program after deleting the old entry, the known_hosts file is updated with the new SSH public key for the Content Engine.

Note The Telnet daemon can still be used with the Content Engine. SSH does not replace Telnet.

This example generates an SSH public key and then enables the SSH service.

```
Console(config)# ssh-key-generate
Ssh host key generated successfully
Saving the host key to box ...
Host key saved successfully

Console(config)# sshd enable
Starting ssh daemon ...
Ssh daemon started successfully
```

System Logging

Use the **logging** command to set specific parameters for the system log file (syslog). This file contains authentication entries, settings of privilege levels and administrative details. System logging is always enabled internally. The system log file is located on the system file system (sysfs) partition as /local1/syslog.txt.

To configure the Content Engine to send varying levels of event messages to an external syslog host, use the **logging host** command. Logging can be configured to send various levels of messages to the console using the **logging console priority** option. (See Table 1-1.)

Table 1-1 Mapping of RealProxy Error Level to Syslog Priority Level

RealProxy Error Code	RealProxy Condition	RealProxy Usage	syslog Priority Level
0	Panic	Error potentially causing a system failure. RealSystem takes actions necessary to correct the problem.	Priority 0—LOG_EMERG, Emergency. System is unusable.
1	Severe	Error requiring immediate user intervention to prevent a problem.	Priority 1—LOG_ALERT, Alert. Immediate action needed.
2	Critical	Error that may require user intervention to correct.	Priority 2—LOG_CRI, Critical. Critical conditions.
3	General	Error that does not cause a significant problem with normal system operation.	Priority 3—LOG_ERR, Error. Error conditions.
4	Warning	Warning about a condition that does not cause system problems but may require attention.	Priority 4—LOG_WARNING Warning. Warning conditions.
5	Notice	Notice about a condition that does not cause system problems but should be noted.	5—LOG_NOTICE Notice. Normal but significant conditions.
6	Informational	Informational message only.	6—LOG_INFO Information. Informational messages.
7	Debug	Information of use only when debugging a program.	7—LOG_DEBUG Debug. Debugging messages.

Note In ACNS 4.1 software, syslog messages from the Content Engine to a remote host are sourced from port 10000 rather than port 514.

This example shows the last few lines of the syslog.txt file using the **type-tail** command, which only lists the last few lines of text in a file.

```
ContentEngine# type-tail syslog.txt
Jan 18 17:50:03 ContentEngine Host[3766]: authentication failure; (uid=0) -> aaHH for
content_engine_config service
Jan 18 17:50:05 ContentEngine login[3766]: Failed login session from 172.16.1.1 for user
aaHH:  Authentication service cannot retrieve authentication info.
```

```
Jan 18 18:39:05 ContentEngine Host[6787]: set privilege level to `0'
Jan 18 18:39:05 ContentEngine login: user login on 1 from 172.16.66.148
ContentEngine#
```

Mapping Syslog Priority Levels to RealProxy Error Codes

The RealProxy (See the "Configuring RealProxy 8.01" section on page 14-1) generates error messages and writes them to the RealProxy log file. These error messages are captured by the Cache software and passed to the system log file. There is a one-to-one mapping correspondence between the RealProxy error codes and the syslog priority levels, as shown in Table 1-1.

Network Time Protocol Time Synchronization

To configure the Network Time Protocol (NTP) and to allow the system clock to be synchronized by a time server, use the **ntp server** global configuration command. The Content Engine can remain within a particular time zone while it synchronizes to Coordinated Universal Time (UTC).

To disable this function, use the **no** form of this command.

ntp server {*hostname* | *ip-address*}

no ntp server {*hostname* | *ip-address*}

In this example the time of the Content Engine is synchronized to a time server with an IP address of 172.16.22.44.

```
ContentEngine(config)# ntp server 172.16.22.44
```

In this example the time-synchronization configuration to the time server at 172.16.22.44 is disabled.

```
ContentEngine(config)# no ntp server 172.16.22.44
```

CHAPTER 2

Cisco Cache Application Basic WCCP Configuration

The Web Cache Communication Protocol (WCCP) is a Cisco-developed content-routing technology that allows you to deploy transparent caching using the Content Engines in your network. Cisco IOS Release 12.1 and later allows the use of Version 1 (WCCPv1) or Version 2 (WCCPv2) of the WCCP.

This chapter describes how to configure a router or cluster of routers and Content Engines using WCCPv2. WCCPv2 offers many enhancements over WCCPv1, such as support for many routers, multicasting and support for re-direction of non-HTTP traffic. With WCCPv1 only a single router can be configured to service either a single Content Engine or a cluster of Content Engines. Further information on WCCP Version 1 commands and router configuration examples are in the Cisco IOS software online documentation as well as in Appendix A, "Web Cache Communication Protocol Version 1."

To make a Content Engine running the Cache application software operational on the network, you must perform these tasks:

- Use the *Cisco Content Delivery Networking Products Getting Started Guide* to configure your network settings.
- Configure the Content Engine for transparent or proxy-server operation.
- Configure the router or switch only if you are deploying a transparent caching operation.
- Verify your configuration.

The procedures for these tasks are in the following sections:

- Basic Cache Application Router Configurations for HTTP Traffic and WCCP Version 2, page 2-2
- Verifying the Software Configuration, page 2-11
- Removing or Replacing a Content Engine, page 2-11

For additional information on web caching concepts and network placement diagrams, refer to the Network Caching White Paper on the Cisco.com Content Networking website at the following URL:

http://www.cisco.com/warp/public/cc/pd/cxsr/500/tech/cds_wp.htm

To upgrade a Content Engine from earlier versions of the Cache software, use the upgrade procedures in the *Cisco ACNS Software System Maintenance and Troubleshooting Guide*.

See Chapter 3, "Cisco Application and Content Networking Software Cache Application Features," for a list of the feature described in this guide.

Basic Cache Application Router Configurations for HTTP Traffic and WCCP Version 2

A router or switch with the proper configuration is required for transparent caching services. The router or switch must be running a version of Cisco IOS software that supports the Web Cache Communication Protocol (WCCP) Version 2.

When cache support is enabled on the router or switch, and WCCP support is enabled on the Content Engines, the devices can communicate and deliver the services for which they are configured. To suspend caching services, you can disable cache support on the router or switch rather than powering off or otherwise disabling individual Content Engines. (For instance, use the **no ip wccp** router or switch command to disable caching.)

Many WCCP Version 2 features also require a configuration of the appropriate Cache application **wccp** global configuration command. Refer to the *Cisco Application and Content Networking Software Command Reference* and release notes for more details.

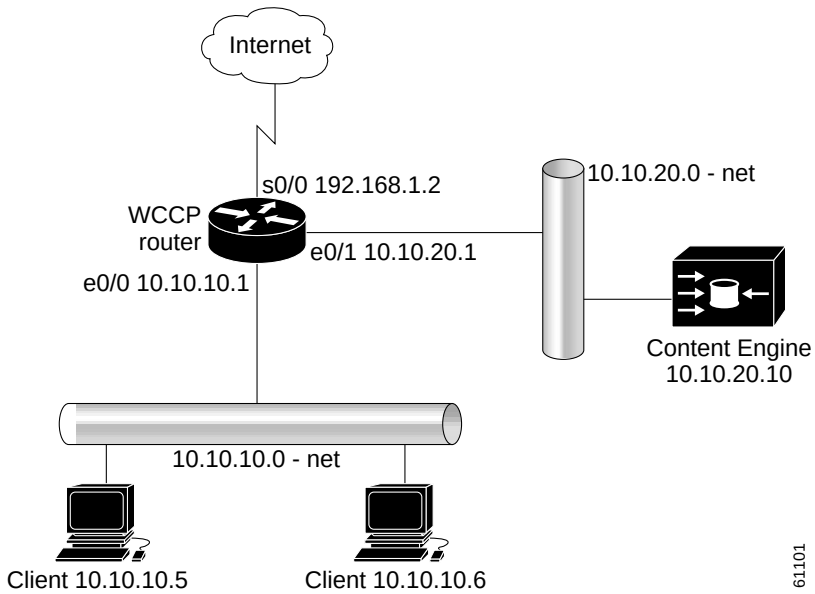
If you do not know how to configure a router or a switch, refer to the software documentation supplied with the devices. Further information on WCCP Version 2 commands and router configuration examples are in the Cisco IOS software online documentation as well as in Appendix B, "Web Cache Communication Protocol Version 2."

Web Cache Service Configuration with Clients and Cache on Different Subnets

In this scenario, the Content Engine and the requesting clients are on different subnets, as shown in Figure 2-1.

A router running WCCP Version 2 transparently redirects client HTTP traffic bound for router interface s0/0 to the Content Engine. The web cache service redirects HTTP traffic on port 80 only.

Figure 2-1 Web Cache Service with Content Engine and Clients on Different Subnets



Configuring the Content Engine for Web Cache Service—Clients and Cache on Different Subnets

To configure the Content Engine for the web cache service, perform the following steps while logged in to the Cache software in global configuration mode:

	Command	Purpose
Step 1	`contentengine(config)# wccp version 2`	Ensures that the Content Engine is running WCCP Version 2.
Step 2	`contentengine(config)# wccp router-list 1 10.10.20.1`	Configures a router list.
Step 3	`contentengine(config)# wccp web-cache router-list-num 1`	Informs the routers in the specified router list that the Content Engine is accepting web cache service.
Step 4	`contentengine(config)# exit`	Exits global configuration mode.
Step 5	`contentengine# write memory`	Writes running configurations to nonvolatile memory.

Configuring the Router for Web Cache Service—Clients and Cache on Different Subnets

To configure the router for web cache service, perform the following steps while logged in to the router global configuration mode:

	Command	Purpose
Step 1	`router(config)# ip wccp web-cache`	Instructs the router to run the web cache service.
Step 2	`router(config)# interface Serial0`	Specifies which router interface to configure. In this scenario, Serial0 is the router interface to the Internet.
Step 3	`router(config-if)# ip wccp web-cache redirect out`	Instructs the router to redirect web cache traffic bound for the specified interface to Content Engines that accept web cache service. In this scenario there is only one router. Web cache traffic is defined as HTTP packets on port 80.
Step 4	`router(config-if)# exit`	Exits global configuration mode.

Configuration Examples—Web Cache Service with Clients and Cache on Different Subnets

This example shows a Content Engine and router configured for the web cache service with the topology illustrated in Figure 2-1:

Content Engine Configuration

```
. . .
hostname Content_engine_4.1
!
clock timezone pst -8 0
!
ip domain-name cisco.com
!
exec-timeout 20
!
interface FastEthernet 0
ip address 10.10.20.10 255.255.255.0
no autosense
```

```
bandwidth 100
full-duplex
exit
interface FastEthernet 1
shutdown
exit
!
ip default-gateway 10.10.20.1
!
ip name-server 10.10.10.100
!
!
wccp router-list 1 10.10.20.1
wccp web-cache router-list-num 1
wccp version 2
!
!
!
...
```

WCCP-Enabled Router Configuration

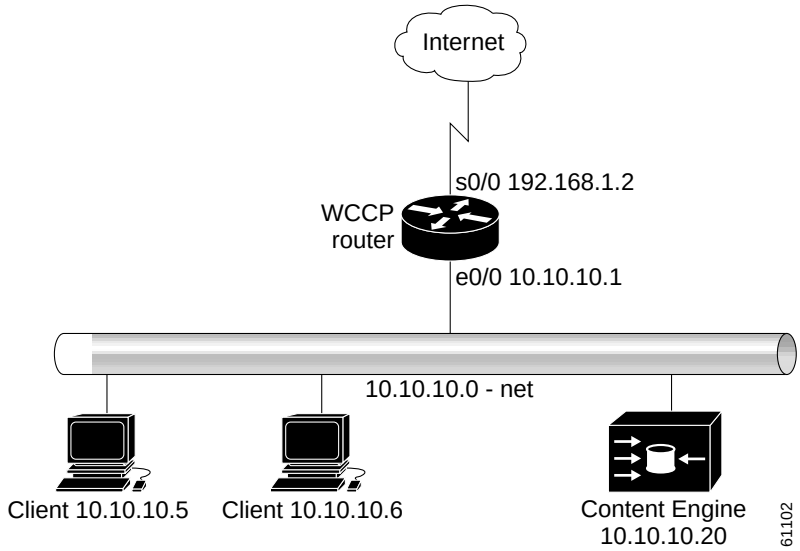
```
Building configuration...

Current configuration:
!
hostname WCCP-Router
!
!
ip subnet-zero
!
ip wccp web-cache
!
interface Ethernet0
 ip address 10.10.10.1 255.255.255.0
!
interface Ethernet1
 ip address 10.10.20.1 255.255.255.0
!
interface Serial0
 ip address 192.168.1.2 255.255.255.252
 no ip directed-broadcast
 no ip mroute-cache
ip wccp web-cache redirect out
 !
end
```

Web Cache Service Configuration with Clients and Cache on Same Subnet

In this scenario, the Content Engine and the requesting clients are on the same subnet, as shown in Figure 2-2.

A router running WCCP Version 2 transparently redirects client HTTP traffic bound for router interface s0/0 to the Content Engine. The web cache service redirects HTTP traffic on port 80 only.

Figure 2-2 Web Cache Service with Clients and Content Engine on Same Subnet



Configuring the Content Engine for Web Cache Service—Clients and Cache on Same Subnet

To configure the Content Engine for the web cache service, perform the following steps while logged in to the Cache software in global configuration mode:

	Command	Purpose
Step 1	`contentengine(config)# wccp version 2`	Ensures that Content Engine is running WCCP Version 2.
Step 2	`contentengine(config)# wccp router-list 1 10.10.10.1`	Configures a router list.
Step 3	`contentengine(config)# wccp web-cache router-list-num 1`	Informs the routers in the specified router list that the Content Engine is accepting web cache service.
Step 4	`contentengine(config)# exit`	Exits global configuration mode.
Step 5	`contentengine# write memory`	Writes running configurations to nonvolatile memory.

Configuring the Router for Web Cache Service—Clients and Cache on Same Subnet

To configure the router for the web cache service, perform the following steps while logged in to the Cache software global configuration mode:

	Command	Purpose
Step 1	`router(config)# ip wccp web-cache`	Instructs the router to run the web cache service.
Step 2	`router(config)# interface Ethernet0`	Specifies which router interface to configure. In this scenario, Ethernet0 is the router interface to which the Content Engine and clients are connected.

	Command	Purpose
Step 3	`router(config-if)# ip route-cache same-interface`	Enables fast switching of redirected packets back through the interface on which they were received. Without this command, the router does not use the high-speed switching cache, and the packets are process-switched, a much slower method.
Step 4	`router(config)# interface Serial0`	Specifies which router interface to configure. In this scenario, Serial0 is the router interface to the Internet.
Step 5	`router(config-if)# ip wccp web-cache redirect out`	Instructs the router to redirect web cache traffic bound for the specified interface to Content Engines that accept web cache service. In this scenario there is only one router. Web cache traffic is defined as TCP port 80 traffic.
Step 6	`router(config-if)# exit`	Exits global configuration mode.

Configuration Examples—Web Cache Service with Clients and Cache on the Same Subnet

This example shows a Content Engine and router configured for the web cache service with the topology illustrated in Figure 2-2:

Content Engine

```
hostname Content_engine_4.1
!
clock timezone pst -8 0
!
ip domain-name cu.net
!
interface FastEthernet 0
ip address 10.10.20.10 255.255.255.0
no autosense
bandwidth 100
full-duplex
exit
interface FastEthernet 1
shutdown
exit
!
ip default-gateway 10.10.20.1
!
ip name-server 10.10.10.100
!
!
!
wccp router-list 1 10.10.20.1
wccp web-cache router-list-num 1
wccp version 2
!
!
!
. . .
```

WCCP-Enabled Router

```
Building configuration...

Current configuration:
!
version 12.0
```

```
!
hostname WCCP-Router
!
!
ip wccp web-cache
!
interface Ethernet0
 ip address 10.10.10.1 255.255.255.0
 ip route-cache same-interface
!
interface Serial0
ip address 192.168.1.2 255.255.255.252
no ip directed-broadcast
no ip mroute-cache
ip wccp web-cache redirect out
 !
end
```

Configuring Reverse Proxy Service

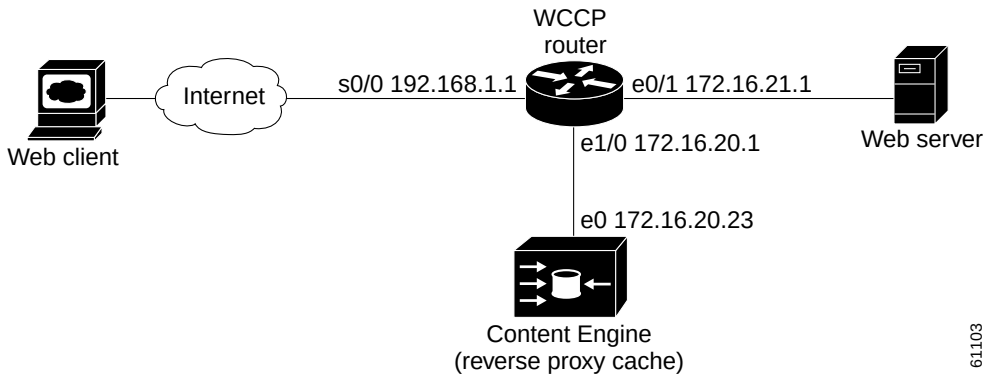
To ensure fast response times, maximized service availability, and the ability to withstand an excessive number of URL hits or an excess of bandwidth requested, Cisco Content Engines with Cache software can be deployed in front of a website server farm to offload traffic from busy firewalls and servers, helping to optimize the entire website infrastructure. This type of deployment is called web server acceleration, or reverse proxying.

The same Content Engine can provide a reverse proxy cache service while simultaneously providing a transparent network cache service.

In the deployment scenario shown in Figure 2-3, the Content Engine interoperates with a router running WCCP Version 2 to bring reverse proxy service within the web server environment. In this deployment, the router interface connected to the Internet has an IP address of 192.168.1.1. All HTTP requests destined for the web server are routed to the router interface of 172.16.21.1. Upon receiving the HTTP request at this interface, the router transparently intercepts and redirects the request to the Content Engine with an IP address of 172.16.20.23. Thus, the Content Engine is logically in front of the web server offloading web server HTTP traffic. The Content Engine sends a request to the web server only when it does not find the requested content in the cache.

Note A redirect list on the router or a static bypass list on the Content Engine can be used to allow flows to bypass interception. These lists use criteria based on source and destination IP addresses.

Figure 2-3 Reverse Proxy Service with WCCP-Enabled Router



Configuring the Content Engine for Reverse Proxy Service with Output Redirection

To configure reverse proxy on the Content Engine, perform the following steps starting from global configuration mode:

	Command	Purpose
Step 1	`contentengine(config)# wccp version 2`	Ensures that the Content Engine is running WCCP Version 2.
Step 2	`contentengine(config)# wccp router-list 1 172.16.20.1`	Configures a router list.
Step 3	`contentengine(config )# wccp reverse-proxy router-list-num 1`	Instructs the router to run the reverse proxy service.

Configuring the Router for Reverse Proxy Service with Output Redirection

To configure reverse proxy service on the router, perform the following steps starting in global configuration mode:

	Command	Purpose
Step 1	`router(config)# ip wccp 99`	Instructs the router to run the reverse proxy service. The reverse proxy service is indicated by a value of 99.
Step 2	`router(config)# interface Ethernet 0/1`	Specifies which router interface to configure. In this scenario, Ethernet 0/1 is the router interface to the web server.
Step 3	`router(config-if)# ip wccp 99 redirect out`	Instructs the router to redirect TCP port 80 traffic bound for the specified interface to Content Engines that accept reverse proxy service. In this scenario there is only one router.
Step 4	`router(config)# exit`	Exits global configuration mode.

Configuring the Router for Reverse Proxy Service with Input Redirection

	Command	Purpose
Step 1	`router(config)# ip wccp 99`	Instructs the router to run the reverse proxy service. The reverse proxy service is indicated by a value of 99.
Step 2	`router(config)# interface s0/0`	Specifies which router interface to configure. In this scenario, s0/0 is the router interface to the Internet.
Step 3	`router(config-if)# ip wccp 99 redirect in`	Instructs the router to redirect TCP port 80 traffic received on the specified interface to Content Engines that accept reverse proxy service.
Step 4	`router(config-if)# exit`	Exits interface configuration mode.

Note Input redirection is supported in Cisco IOS Release 12.1(3)T or 12.0(11)S and later versions of the same, or in Cisco IOS Release 12.2.

Configuration Examples—Reverse Proxy with Output Redirection

This section shows configurations of the Content Engine and the router for the deployment scenario in Figure 2-3. The reverse proxy related commands are in bold:

Content Engine

```
!
hostname ContentEngine
!
interface ethernet 0
 ip address 172.16.20.23 255.255.255.224
 ip broadcast-address 10.0.0.255
exit
!
interface ethernet 1
exit
!
ip default-gateway 172.16.20.1
ip name-server 172.16.20.238
ip domain-name cisco.com
ip route 0.0.0.0 0.0.0.0 172.16.20.1
!
wccp router-list 1 172.16.20.1
wccp 99 reverse-proxy router-list-num 1
wccp version 2
!
end
```

WCCP-Enabled Router

```
Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
```

```
!
hostname 2611-5
!
enable secret 5 $1$9VQ.$BTPbq8xlE2dsKwKPDDmpL/
!
!
ip subnet-zero
ip wccp 99
!
!
interface Ethernet0/0
 no ip address
 no ip redirects
 !
interface Serial0/0
 ip address 192.168.1.1 255.255.255.224
 !
interface Ethernet0/1
 ip address 172.16.21.1 255.255.255.224
 ip wccp 99 redirect out
!
interface Ethernet1/0
 ip address 172.16.20.1 255.255.255.224
 !
interface Ethernet1/1
 no ip address
 shutdown
 full-duplex
!
interface Ethernet1/2
 no ip address
 shutdown
!
. . .
ip classless
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

Disabling Transparent Caching Services on the Content Engine

To disable transparent caching on a Content Engine in a WCCP environment without powering it down, disable the running version of WCCP on the Content Engine by issuing the Cache software **no wccp version 2** global configuration command. The Content Engine will still service proxy-style requests, if so configured, and preserve its configuration settings.

Further information about the Content Engine commands used to configure the Content Engine for the deployment scenarios provided in this chapter can be found in the *Cisco Application and Content Networking Software Command Reference*. Refer to this document for more information about the Content Engine commands.

Verifying the Software Configuration

Once you have installed and configured the Content Engine and enabled WCCP caching services on the router, verify that the Cache application is working properly.

Step 1 Start a web browser and open various web pages on the Internet or your intranet. The web servers you connect to must be on a different subnet, so that the request goes through either a home router running WCCP Version 1 or Version 2, or a cluster of routers running WCCP Version 2. Request the same pages more than once, to ensure that pages you request are in the cache. For more information on WCCP, see Appendix A, "Web Cache Communication Protocol Version 1," and Appendix B, "Web Cache Communication Protocol Version 2."

Step 2 From the CLI, enter the following command to display the Content Engine HTTP caching saving statistics:

```
show statistics http savings
```

Step 3 Open a console or Telnet session on the home router or routers, and enter the **show ip wccp** commands to display statistics and status information about the Content Engine.

The statistics should show a number greater than 0 for packets redirected. Also, check for hash assignments, which indicate at the very least that the Content Engines are registered and communicating with the routers.

If the router shows that no packets are being redirected to the Content Engine, you must troubleshoot your setup.

Removing or Replacing a Content Engine

Refer to the Content Engine hardware documentation for instructions on physically removing a Content Engine from an active network.

The router and the Content Engine are in constant communication when WCCP is enabled; thus, when the router notices that the engine is no longer responding to it, the router stops sending requests to the engine. This is transparent to users. If other Content Engines are attached to the router, the router continues sending requests to the other engines.

When you remove a Content Engine, the pages that were cached on its hard disks are no longer available to the router or other Content Engines. Thus, you might see an increase in outgoing web traffic that might have otherwise been fulfilled by the Content Engine you are removing. However, after a time, the router and other Content Engines will redistribute the load of web traffic.

If you remove the last Content Engine from a router, you can also disable cache support on the router. However, this is not necessary; having cache support enabled when there are no Content Engines attached has no effect on the router's performance.

To replace a Content Engine, remove the old one from the network. Then, add the new Content Engine and configure it using the same startup configuration parameters that you used for the removed one.

PART 2

Content Engine Features

CHAPTER 3

Cisco Application and Content Networking Software Cache Application Features

Cache Application Feature Set Table

Table 3-1 lists the principal features of the ACNS software Cache application, with the associated command-line interface (CLI) commands. Release notes may contain updates to this information.

Table 3-1 Cisco Cache Application Feature Set

Cisco Cache Software, Release 4.1 Feature	Related CLI Commands	Section and Page
Transparent caching		
Transparency through WCCP	**wccp version 2** **wccp router-list**	Transparent Caching Through WCCP, page 4-1
Authentication bypass	**bypass auth-traffic** **bypass timer**	Authentication Traffic Bypass, page 4-5
Dynamic bypass	**bypass auth-traffic** **bypass timer**	Dynamic Traffic Bypass, page 4-6
Overload bypass	**bypass load**	Overload Bypass, page 4-7
Static bypass	**bypass static**	Static Bypass, page 4-8
Multiport transparent redirection	**proxy-protocols** **wccp port-list** **wccp service-number**	Multiport Transparent Redirection, page 4-9
WCCP flow protection	**wccp slow-start** **wccp flow-redirect**	WCCP Flow Protection, page 4-9
Accelerated WCCP Layer 2 support	**wccp custom-web-cache** **wccp media-cache** **wccp reverse-proxy** **wccp service-number** **wccp web-cache**	Accelerated WCCP Layer 2 Support, page 4-10

Table 3-1 Cisco Cache Application Feature Set (continued)

Cisco Cache Software, Release 4.1 Feature	Related CLI Commands	Section and Page
Transparent caching with the Cisco CSS11000 series switch	**http l4-switch enable**	Transparent Caching with the Cisco CSS 11000 Series Switch, page 4-11
Proxy-style caching (nontransparent operation)		
HTTP proxy caching	**http proxy incoming**	HTTP Proxy Caching, page 5-2
FTP proxy caching	**ftp proxy incoming**	FTP Proxy Caching, page 5-2
SSL tunneling	**https proxy incoming**	SSL Tunneling, page 5-3
Cache parameter settings		
Caching of authenticated content	**http cache-authenticated**	Caching of Authenticated Content, page 6-1
Cache freshness	**http min-ttl** **http max-ttl** **http age-multiplier** **http reval-each-request**	Cache Freshness, page 6-1
Caching of binary content with cookies	**http cache-cookies**	Caching of Binary Content with Cookies, page 6-2
Object size capping	**http object**	Maximum Object Size, page 6-3
Selective abort of object downloading on client-abort (also called “quick_abort”)	**http cache-on-abort**	Aborting Selected Objects, page 6-3
HTTP Range request caching	**http cache-on-abort**	Caching of HTTP Range Requests, page 6-4
Cache hierarchy		
Parent proxy failover	**http proxy outgoing**	Parent Proxy Failover, page 7-2
Handling proxy-style requests	**http proxy outgoing** **proxy-protocols**	Handling Proxy-Style Requests, page 7-5
ICP	**icp client** **icp server**	Internet Cache Protocol, page 7-6
Employee Internet management		
URL filtering	**url-filter** **url-filter bad-list-deny**	URL Filtering, page 8-1
N2H2 filtering	**url-filter N2H2 server** **url-filter N2H2 allowmode**	URL Filtering with the N2H2 Server, page 8-4
Websense enterprise server filtering	**url-filter websense enable** **url-filter websense server**	URL Filtering with the Websense Enterprise Server, page 8-6
SmartFilter filtering	**url-filter smartfilter**	URL Filtering with SmartFilter Software, page 8-7
Logging		
Squid-style transaction logging	**transaction-logs format squid**	Squid-Style Transaction Logging, page 9-1

Table 3-1 Cisco Cache Application Feature Set (continued)

Cisco Cache Software, Release 4.1 Feature	Related CLI Commands	Section and Page
Extended Squid transaction logging	**transaction-logs format extended-squid**	Extended Squid-Style Transaction Logging, page 9-3
Apache-style transaction logging	**transaction-logs format apache**	Apache-Style Transaction Logging, page 9-3
Sanitized transaction logs	**transaction-logs sanitize**	Sanitized Transaction Logs, page 9-4
Exporting log files	**transaction-logs export enable** **transaction-logs export ftp-server**	Exporting Log Files, page 9-4
User authentication		
User authentication configuration	**authentication login** **authentication configuration**	User Authentication, page 10-1
TACACS+ authentication	**authentication** **tacacs**	TACACS+ Options for User Authentication, page 10-2
Microsoft NT LAN Manager (NTLM) authentication	**http cache-authenticated** **ntlm server** **http authenticate-strip-ntlm**	NTLM Authentication, page 10-3
RADIUS authentication	**http authentication cache** **http authentication header** **radius-server**	RADIUS HTTP Request Authentication, page 10-4
LDAP authentication	**http authentication cache** **http authentication header** **ldap server**	LDAP HTTP Request Authentication, page 10-5
Network Management		
SNMP agent support	**snmp-server community**	Key SNMP CLI Commands, page 11-3
SNMP traps	**snmp-server enable traps** **snmp-server host**	SNMP Traps, page 11-4
CiscoWorks2000 syslog format	**logging cw2k**	CiscoWorks2000, page 11-5
Cisco Discovery Protocol	**interface FastEthernet 0/0 cdp enable**	Cisco Discovery Protocol, page 11-5
TCP stack parameters		
User-configurable TCP parameters	**tcp**	Configuring TCP Parameters Using the Content Engine GUI, page 12-2
TCP-over-satellite extensions	**tcp client-satellite** **tcp server-satellite**	TCP-Over-Satellite Extensions, page 12-3
Streaming media splitting and caching		
Microsoft Windows Media Technologies (WMT) 7.01	**disk config sysfs** *partitionsize* **disk config mediafs** *partitionsize* **wmt enable**	Configuring Microsoft Windows Media Player 7.01, page 13-2

Table 3-1 Cisco Cache Application Feature Set (continued)

Cisco Cache Software, Release 4.1 Feature	Related CLI Commands	Section and Page
RealProxy 8.01 support	**disk config sysfs** *partitionsize* **disk config mediafs** *partitionsize* **rtsp proxymedia-real enable**	Configuring RealProxy 8.01, page 14-1
Miscellaneous features		
Rules Template	**rule enable**	Rules Template, page 15-1
Boomerang agent	**boomerang dns enable**	Configuring the Content Engine as a Content Routing Agent, page 16-1
Browser autoconfiguration	**proxy-auto-config**	Browser Autoconfiguration, page 16-3
Healing mode	**http cluster** **http cluster misses** **http cluster max-delay** **http cluster http-port**	Configuring Healing Mode, page 16-3
Content preloading	**pre-load enable** **pre-load url-list-file**	Content Preloading, page 16-5

CHAPTER 4

Configuring Transparent Caching

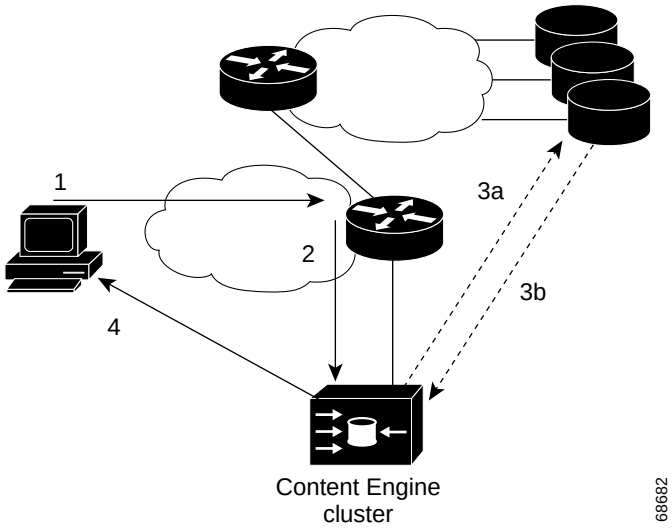
Transparent Caching Through WCCP

In transparent caching through WCCP, the user is unaware that the request made to an origin server is redirected to the Content Engine by a WCCP-enabled router. A request to a WCCP-enabled router allows for traffic interception on any port number for traffic that traverses the WCCP-enabled router or switch. WCCP contains many fail-safe mechanisms to ensure that the caching solution remains entirely transparent to the end user.

A Content Engine transparently caches as follows. (See Figure 4-1.)

1. A user requests a web page from a browser.
2. The WCCP-enabled router analyzes the request, and based on TCP destination port number, the router determines whether it should transparently redirect the request to a Content Engine. Access lists are used to control which requests can be redirected.
3. If the Content Engine does not have the requested content, it sets up a separate TCP connection to the end server to retrieve the content (3a). The content returns to, and is stored on, the Content Engine (3b).
4. The Content Engine sends the content to the client. Upon subsequent requests for the same content, the Content Engine transparently fulfills the request from its local storage.

Figure 4-1 Transparent Caching Through WCCP



WCCP can also handle asymmetric packet flows and always maintains a consistent mapping of web servers to caches regardless of the number of switches or routers used in a WCCP service group (up to 32 routers or switches communicating with up to 32 caches in a cluster).

There are some significant advantages to deploying caches in transparent mode:

- No end user configuration—The user does not have to point to the Content Engine.
- Fail-safe—Caches are automatically fault-tolerant and fail-safe. Any cache failure does not cause denial of service to the end user.
- Scalable—Cache service can be scaled by deploying multiple caches.
- Automatic bypass—Sites which depend on end user authentication or which fail to conform to HTTP standards will automatically bypass a transparent cache.

Transparent Mode Operation

A transparent request is a request redirected to the Content Engine from a router. Transparent requests have a different destination IP address than that of the Content Engine. The style of the URL within a transparent request is usually server-style but can be proxy-style when the Content Engine intercepts a request destined for another proxy server. Server-style requests do not include the protocol and host name, but Real-Time Streaming Protocol (RTSP) requests are the same for server-style and for proxy-style. If a server-style URL is received, only HTTP and RTSP are supported (if RTSP user agent criteria are met). If a proxy-style URL is received, HTTP, HTTP over Secure Socket Layer (HTTPS), FTP, and RTSP are supported when the respective proxy services are configured.

The **wccp service-number** global configuration command can enable up to eight dynamic WCCP redirection services (90 to 97) on a Content Engine, provided that the services are also configured on the router. Each **wccp service-number** command specifies a router list, single port list (containing up to eight ports), application type, hash parameters, password, and weight.

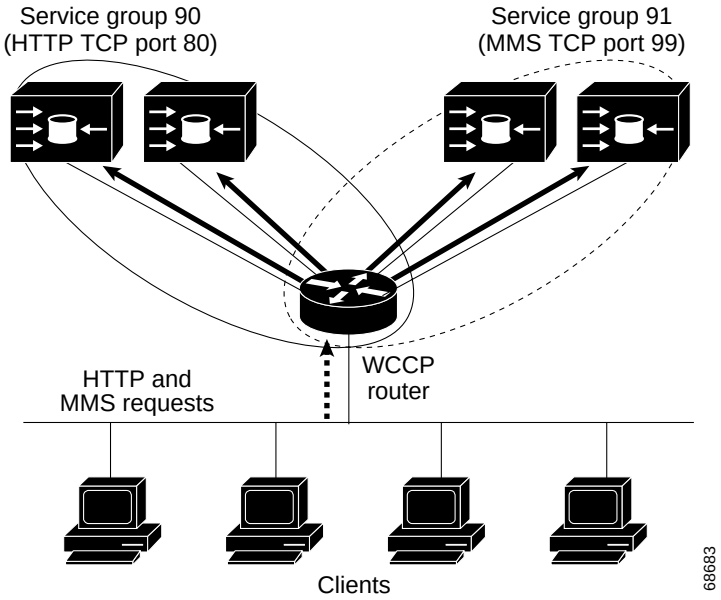
The hash parameters specify how traffic should be load-balanced among the different service groups. Specifically, hashing maps items from one item set to another, such as mapping destination IP addresses to different Content Engines in a service group from the WCCP-enabled router for load-balancing purposes.

The legacy custom web cache and reverse proxy services (service numbers 98 and 99) can be configured with only one port each. If only one legacy service is configured, the total maximum number of transparent redirection ports is 57. If both legacy services are configured, the maximum port total is 50.

Note The service group feature requires WCCP version 2.

With 8 dynamic services using a maximum number of 8 ports each, the maximum number of ports that can be specified for transparent redirection is 64. In Figure 4-2 the two Content Engines on the left handle only HTTP traffic through port 80 and are defined as members of Service Group 0. The two Content Engines on the right handle only Microsoft Media Server (MMS) requests and are defined as members of Service Group 1.

Figure 4-2 WCCP Service Group Feature



All ports receiving HTTP that are configured as members of the same WCCP service share the following characteristics:

- They have the same hash parameters, as configured with the **wccp service-number** command.
- The service on individual ports cannot be stopped or started individually (WCCP Version 2 restriction).

With Content Engines in a farm, the following restrictions apply:

- All Content Engines that use the same WCCP service are required to configure the same list of ports and the same hash parameters.
- A Content Engine that tries to join the farm with the same WCCP service using a different list of ports or different hash parameters is rejected by the router.
- To change the port list for a particular WCCP service, WCCP service must be stopped on all involved Content Engines, and then all must be restarted with the new parameters.

The Content Engine WCCP implementation currently allows global settings that apply to all WCCP services, such as healing parameters, slow start, and others. The multiple service model does not change that, and the settings in question remain global for the whole WCCP system.

Prerequisites

To use a WCCP-enabled router, an IP address must be configured on the interface connected to the Internet and the interface must be visible to the Content Engine on the network.

Configuration Tasks

Use the following configuration tasks to help you configure the Content Engine in transparent mode using a WCCP-enabled router. To learn about transparent caching with a Layer 4 switch, see the "Transparent Caching with the Cisco CSS 11000 Series Switch" section on page 4-11.

Configuring the Content Engine

To use WCCP, the Content Engine must be properly configured. Keep these important points in mind:

- The IP address of the router must be configured as the home router for the Content Engine. In this scenario, this router is the device that performs all the IP packet redirection for the Content Engine.
- Versions of software on the Content Engines must be compatible with those installed on the router.
- The Content Engines must not have their packets encrypted or compressed and should be part of the "inside" Network Address Translation if one is present.
- Placing a Content Engine beyond a web cache redirect-enabled interface and along the route to the server will not cause the IP route cache to be populated with an entry.

Enabling WCCP on the Router

To enable an interface to redirect web traffic to the Content Engine using WCCP Version 1, perform the following tasks beginning in global configuration mode:

	Command	Purpose
Step 1	**ip wccp enable**	Enables the router to use WCCP.
Step 2	**ip wccp redirect-list** {*number* | *name*}	(Optional.) Specifies the redirect access list. Only packets that match this access list are redirected. If you do not configure this command, all packets are redirected.
Step 3	**interface** *type number*	Enters interface configuration mode.
Step 4	**ip web-cache redirect**	Configures the interface connected to the Internet to redirect web traffic to the Content Engine.
Step 5	**ip route-cache same-interface**	(Optional.) If the client and a Content Engine are located on the same network, configures the router to use the fast switching path on the interface.
Step 6	**end**	Exits configuration mode.
Step 7	**copy running-config startup-config**	Saves the configuration.

For more information on how to configure a WCCP-enabled router for transparent caching see Appendix A, "Web Cache Communication Protocol Version 1" and Appendix B, "Web Cache Communication Protocol Version 2."

Advanced Transparent Caching Features

One of the fundamental principles of transparent network caching is that the cache must remain transparent to the end user at all times. A transparent caching solution must not introduce any possible failure conditions or side-effects in a network.

The Cisco caching solution uses a WCCP-enabled router and various advanced techniques to ensure that the cache remains transparent, even if web browsers are nonoperational or web servers that are not HTTP compliant.

The following features ensure that customers do not encounter unexpected problems while deploying Cisco caching solutions:

- Authentication Traffic Bypass
- Dynamic Traffic Bypass
- Overload Bypass
- Static Bypass
- Multiport Transparent Redirection
- WCCP Flow Protection

Note When bypass is enabled, the client itself tries to reach the origin server. You must make sure to disable all bypass options at this point to eliminate unnecessary burden on the network.

Authentication Traffic Bypass

Because of IP authentication, some web sites do not allow the Content Engine to connect directly on behalf of the client. In order to preserve cache transparency and avoid disruption of service, the Content Engine can use authentication traffic bypass to automatically generate a dynamic access list for the selected client/server pairs. Authentication bypass triggers are also propagated upstream and downstream in the case of hierarchical caching.

To enable transparent error handling and dynamic authentication bypass, and to configure static bypass lists, use the **bypass** global configuration command. To disable the bypass feature, use the **no** form of the command.

For example, when a client/server pair performs authentication bypass, it is bypassed for a configurable amount of time, which is set by the global configuration **bypass timer** command (10 minutes by default).

This example forces all authenticated HTTP traffic to bypass the Content Engine for 24 hours:

```
ContentEngine(config)# bypass auth-traffic enable
ContentEngine(config)# bypass timer 1440
```

Dynamic Traffic Bypass

The following two scenarios describe typical dynamic traffic bypass situations.

Scenario 1—Dynamic Bypass Upon Receiving a Web Server Error

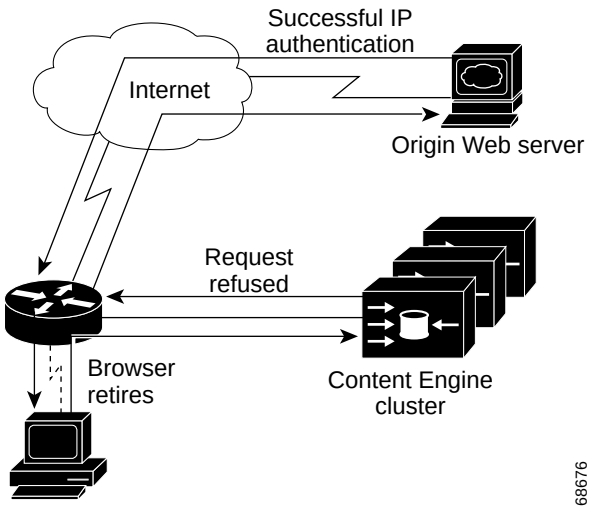
A user issues an HTTP request from a web browser. The request is transparently intercepted and redirected to the Content Engine. The Content Engine accepts the incoming TCP connection from the web browser, determines that the request is for an object not in storage (cache-miss), and issues a request for the object from the origin web server, but receives some kind of error (for instance, a protocol or authentication error) from the web server.

The Content Engine has already accepted the TCP connection from the web browser and the three-way TCP handshake has taken place. The Content Engine detects that the transaction with the web server is failed, but it does not know the cause (the origin web server is performing authentication based on user source IP address, incompatibility between the TCP stacks, and so forth).

Dynamic client bypass in this case means that the Content Engine returns an HTTP response code 200 (200 Temporarily Moved) to the browser with the exact same URL again. The Content Engine closes the TCP connection between the web browser and the Content Engine by issuing a “Connection: close” HTTP response header to the web browser. The browser then automatically retries the connection.

On the connection retry, the Content Engine does not accept the connection. It passes the request back to the WCCP-enabled router or switch unintercepted. The router then sends the flow toward the origin web server directly from the web browser, thereby bypassing the Content Engine. (See Figure 4-3.)

Figure 4-3 Dynamic Client Bypass



Scenario 2—Dynamic Bypass Upon Receiving an Unsupported Protocol

When the Content Engine receives non-HTTP requests over TCP port 80, the Content Engine issues a “retry” response, closes the connection, and does not accept subsequent connections in the same manner as in Scenario 1. A “retry” response is a normal HTTP response which states that the response needs a refresh or another try.

Note Non-HTTP includes nonconforming HTTP as well as different protocols such as Secure Shell (SSH), Simple Mail Transfer Protocol (SMTP), or Network News Transport Protocol (NNTP). An example of nonconforming HTTP is the failure of a web server to issue two carriage returns and line feeds at the end of the HTTP header section.

These two scenarios implement the WCCP return-path functionality, which is a mechanism whereby a Content Engine can return traffic to the WCCP-enabled router or switch, telling the router or switch to forward the packets as if the Content Engine were not present.

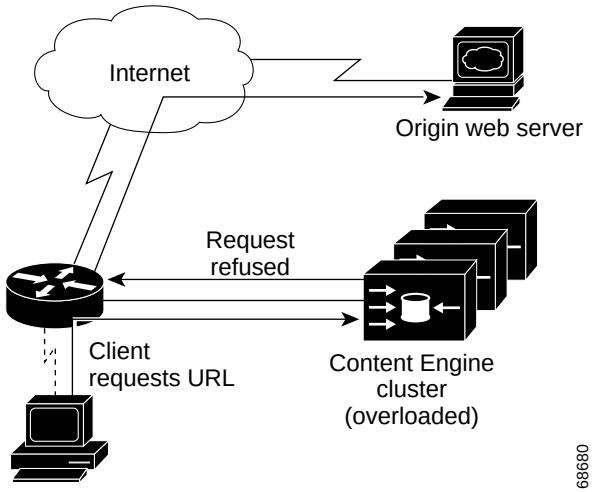
It is typical for about 3 percent of all HTTP traffic flows to fail. These failed flows are automatically retried using authentication bypass or dynamic client bypass, demonstrating that the failure conditions were preexisting and not due to the deployment of transparent caching.

Overload Bypass

If a Content Engine becomes overwhelmed with traffic, it can use the overload bypass feature to reroute the overload traffic.

When the Content Engine is overloaded and the **bypass load** command is enabled, the Content Engine refuses additional requests and forwards them to the origin servers. If the load remains too high, more traffic is bypassed to the servers, and so on until the Content Engine can handle the load. The time interval between one bucket being bypassed and the next, is set by the **out-interval** option. The default is 4 seconds. (See Figure 4-4.)

Figure 4-4 Overload Bypass



When the first bucket bypass occurs, a set interval must elapse before the Content Engine begins to again service the bypassed buckets. The duration of this interval is set by the **time-interval** option. The default is 10 minutes.

When the Content Engine begins to service the bypassed traffic again, it begins with a single bypassed bucket. If the load is serviceable, it picks up another bypassed bucket, and so on. The time interval between picking up one bucket and the next is set by the **in-interval** option. The default is 60 seconds.

Static Bypass

The **bypass static** command permits traffic from specified sources to bypass the Content Engine. The type of traffic sources are as follows:

- Specific web client to a specific web server
- Specific web client to any web server
- Any web client to a specific web server

Wildcards in either the source or the destination field are not supported.

To clear all static configuration lists, use the **no** form of the command.

Note You can also configure the router with access lists (ACLs) to increase static bypass performance.

Examples

This example forces HTTP traffic from a specified client to a specified server to bypass the Content Engine:

```
ContentEngine(config)# bypass static 10.1.17.1 172.16.7.52
```

This example forces all HTTP traffic destined to a specified server to bypass the Content Engine:

```
ContentEngine(config)# bypass static any-client 172.16.7.52
```

This example forces all HTTP traffic from a specified client to any web server to bypass the Content Engine:

```
ContentEngine(config)# bypass static 10.1.17.1 any-server
```

This example forces all authenticated HTTP traffic to bypass the Content Engine for 24 hours:

```
ContentEngine(config)# bypass auth-traffic enable
ContentEngine(config)# bypass timer 1440
```

A static list of source and destination addresses helps to isolate instances of problem-causing clients and servers.

To display static configuration list items, use the **show bypass list** command.

```
ContentEngine# show bypass list
Client              Server          Entry type
------              ------          ----------
10.1.17.1:0         172.16.7.52:0   static-config
any-client:0        172.16.7.52:0   static-config
10.1.17.2:0         any-server:0    static-config
```

The total number of entries in the bypass list is reported by the **show bypass summary** command.

```
Total number of HTTP connections bypassed = 0
        Connections bypassed due to system overload              = 0
        Connections bypassed due to authentication issues        = 0
        Connections bypassed due to facilitate error transparency = 0
        Connections bypassed due to static configuration         = 0

Total number of entries in the bypass list = 3
        Number of Authentication bypass entries = 0
        Number of Error bypass entries          = 0
        Number of Static Configuration entries  = 3
```

Multiport Transparent Redirection

The multiport feature can be summarized as follows:

- Up to eight incoming proxy ports are supported for each proxy protocol (FTP, HTTP, HTTPS, MMS, and RTSP).
- Proxy-style requests in HTTP, FTP, HTTPS, MMS, and RTSP protocols can be received on the same incoming proxy port.
- Both transparent and proxy-style requests can be serviced on the same port.
- Transparent traffic is disallowed on invalid ports.
- Nonsupported protocols are disallowed over incoming ports.

The **proxy incoming** option of the **http**, **https**, **ftp**, and **rtsp** global configuration commands now supports up to eight ports per protocol.

The multiport feature requires WCCP Version 2 and requires the configuration of the **wccp port-list** and the **wccp service-number** global configuration commands. The **application cache** option of the **wccp service-number** global configuration command redirects traffic to the Cache software conventional caching processes, whereas the **application streaming** option redirects traffic to the Content Engine media caching processes.

Note Domain Name System (DNS) must be configured in order to support all incoming proxy requests.

Modifying Multiport Configurations

To disable HTTP, HTTPS FTP, RTSP, and MMS incoming proxy services use the **no** *protocol* **proxy incoming** command. To add or remove ports in proxy mode, issue a new command that specifies all the ports to be used.

In transparent mode, to add or remove ports for a WCCP service, modify the port list or create a new port list for the WCCP service. A **no wccp service-number** command disables the specified service.

In the following example, ports 200, 3000, 110, 220, 330, 440, and 40000 are added to port list 3.

```
ContentEngine(config)# wccp port-list 3 10
ContentEngine(config)# wccp port-list 3 10 200 3000 110 220 330 440 40000
```

In this example, the Content Engine is configured to accept FTP, HTTP, and HTTPS proxy requests on ports 81, 8080, and 8081:

```
ContentEngine(config)# http proxy incoming 81 8080 8081
ContentEngine(config)# https proxy incoming 81 8080 8081
ContentEngine(config)# ftp proxy incoming 81 8080 8081
```

WCCP Flow Protection

WCCP flow protection is a mechanism that ensures that no existing flows are broken when a new Content Engine is brought online.

When transparent traffic interception or redirection first commences, WCCP flow protection ensures that no existing HTTP flows are broken by allowing those preexisting HTTP flows already established to continue on.

WCCP flow protection also ensures that when a new Content Engine joins an existing Content Engine cluster, existing flows serviced by preexisting caches in the cluster will continue to receive those existing flows.

The mechanisms used by WCCP flow protection result in all of the benefits of maintaining per flow state information in a centralized location but without the overhead, scaling issues, and redundancy or resiliency issues (for example, asymmetrical traffic flows) associated with keeping per flow state information in the switching layer.

Use the **wccp flow-redirect** command to implement WCCP flow protection.This command works with WCCP Version 2 only. This flow protection feature is designed to keep the TCP flow intact as well as to not overwhelm Content Engines when they come up or are reassigned new traffic. This feature also has a slow start mechanism whereby the Content Engines try to take a load appropriate for their capacity.

This example shows how to enable WCCP flow protection in a Content Engine.

```
ContentEngine(config)# wccp flow-redirect enable
```

Accelerated WCCP Layer 2 Support

Accelerated WCCP is a generic term for a deployment in which WCCP on a router or switch can take advantage of switching hardware that either partially or fully implements the traffic interception and redirection functions of WCCP in hardware at Layer 2. Accelerated WCCP is currently supported only with the Cisco Catalyst 6000 and 6500 Family switches.

The Content Engine must have a Layer 2 connection with the switch. Because there is no requirement for a generic routing encapsulation (GRE) tunnel between the switch and the Content Engine, the switch can use a cut-through method of forwarding packets. For the Catalyst 6000 and 6500 Family switches, this feature is called WCCP Layer 2 Policy Feature Card (PFC) Redirection and has been available since Cisco IOS Release 12.1(1)E. This method is intended to achieve forwarding performance of up to 3 gigabits per second using a combination of the Supervisor Engine 1A and the Multilayer Switch Feature Card 2 (MSFC2).

Related Cache software commands are **wccp custom-web-cache**, **wccp media-cache**, **wccp reverse-proxy**, **wccp service-number**, and **wccp web-cache**.

The following example configures a Content Engine to receive Layer 2 redirected traffic from a Catalyst 6500 Series switch with a Multilayer Switch Feature Card and Supervisory Engine 1 (MSFC/SUP 1). (Multiple caching services are configured for informational purposes only.)

```
ContentEngine(config)# wccp version 2
ContentEngine(config)# wccp router-list 1 172.16.55.1
ContentEngine(config)# wccp port-list 1 80
ContentEngine(config)# wccp web-cache router-list-num 1 l2-redirect
ContentEngine(config)# wccp media-cache router-list-num 1 l2-redirect
ContentEngine(config)# wccp web-cache router-list-num 1 l2-redirect
ContentEngine(config)# wccp reverse-proxy router-list-num 1 l2-redirect
ContentEngine(config)# wccp service-number 90 router-list-num 1 port-list-num 1
                       application cache l2-redirect
```

Configuring Caching Services with the Cisco CSS 11000 Series Switch

The Cisco CSS 11000 series switch can support transparent caching and conventional proxy caching. It also supports web acceleration through intelligent reverse proxy caching. The CSS switch provides several load-balancing methods depending how you want to distribute data over the caches (for example, entire URL, URL string, entire domain name, or domain string). The CSS switch also builds a list of known cacheable objects. The list may be modified, but much of the work is reduced by the Content Engine caching capabilities.

CSS 11000 Switch Caching Configurations

In a regular proxy cache configuration, the proxy server acts as a proxy for the client. In the reverse proxy configuration, the reverse proxy server acts as a proxy for the server. Also, a reverse proxy cache stores specific content, whereas proxy and transparent caches store frequently requested content. Reverse proxy caches serve two primary functions:

- Replication of content to geographically dispersed areas
- Replication of content for load balancing

In a reverse proxy cache configuration, the proxy server is configured with an Internet-routable IP address. Clients are directed to the proxy server based on a DNS resolution of a domain name. To a client, the reverse proxy server appears like a web server.

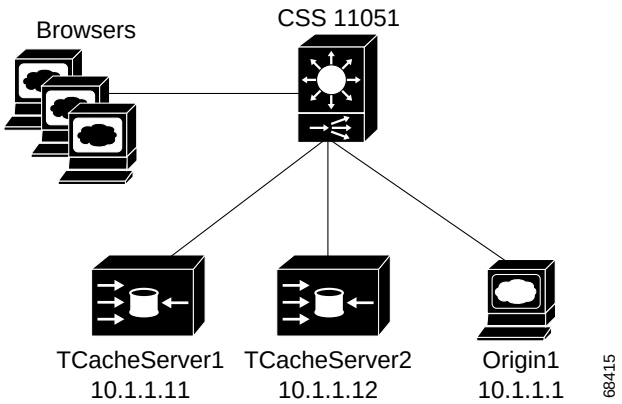
Transparent Caching with the Cisco CSS 11000 Series Switch

Transparent caching deploys cache servers that are transparent to the browsers. You do not have to configure browsers to point to a cache server. Cache servers duplicate and store inbound Internet data previously requested by clients.

When you configure transparent caching on the CSS switch, the switch intercepts and redirects outbound client requests for Internet data to the cache servers on your network. The cache either returns the requested content if it has a local copy or sends a new request to the origin server for the information. If all cache servers are unavailable in a transparent cache configuration, the CSS switch allows all client requests to progress to the origin servers.

A CSS switch is introduced between the user and the cache. (See Figure 4-5.) You can configure the CSS switch to dynamically analyze the content and determine if it is cacheable or not. If it is cacheable, the CSS switch directs it to the cache service. If it is not cacheable, the CSS switch sends it directly to the origin server.

Figure 4-5 Transparent Caching Network Diagram with the CSS 11000 Series Switch



The transparent cache servers will be stocked with static data (that is, HTML, Audio Video Interleaved [AVI], Joint Photographic Experts Group [JPEG], or Graphics Interchange Format [GIF]) files. Any files that are not cacheable (for example, ASP) will be passed directly to the server.

Requests for cacheable content are load-balanced over the two cache servers based on the URL. In a real-world scenario, they could also be balanced based on the domain name.

Content Engine CLI Commands

The **http l4-switch enable** command permits the Content Engine to transparently receive Layer 4 redirected traffic from Layer 4-enabled switches such as the Cisco CSS11000 series switches. See the "Transparent Caching Sample Configuration Output" section on page 4-13 for specific configuration information.

Enabling Transparent Caching using the CSS 11000

Use the following task and CLI commands to help you configure *serv1* as a transparent caching service using the CSS 11000 switch. Ensure that you have configured interfaces, services, owners, VLANs and content rules prior to configuring caching with the CSS 11000. Refer to the *Content Services Switch Basic Configuration Guide* for further information on how to configure these attributes.

For a complete description of each command, refer to the *Content Services Switch Command Reference*.

Step 1 Add service *serv1* reserved for transparent caching

```
CS150(config)# add service serv1
CS150(config-service[serv1])#
```

Step 2 Specify transparent caching service type for *serv1*.

```
CS150(config-service[serv1])# type transparent cache
```

Step 3 Create an extension qualifier list (EQL) in which you specify which content types the CSS 11000 is to cache.

```
CS150(config)# eql graphics
CS150(config-eql[graphics]#
```

Step 4 Describe the eql by entering a quoted text string with a maximum length of 64 characters.

```
CS150(config-eql[graphics]# description “This EQL specifies cacheable graphic files”
```

Step 5 Specify the extension for content you want the CSS 11000 to cache. Enter a text string from 1 to 8 characters.

```
CS150(config-eql[graphics]# extension jpeg
```

You may also provide here a description of the extension type. Enter a quoted text string with a maximum length of 64 characters.

```
CS150(config-eql[graphics]# extension jpeg “This is a graphics file”
CS150(config-eql[graphics]# exit
CS150(config)#
```

Step 6 Specify the EQL in a content rule to match all content requests with the desired extension.

```
CS150(config-owner-content[cisco.com-rule1])# url “/*” eql graphics
```

Step 7 Configure the load balancing method for the cache content rule. The default is roundrobin.

```
CS150(config-owner-content[cisco.com-rule1])# balance domain
```

Step 8 Specify a failover type to define how the CSS 11000 handles content requests when a service fails (bypass, linear, next). The default is linear.

```
CS150(config-owner-content[cisco.com-rule1])# failover bypass
```

Step 9 Display the EQL configuration.

```
CS150(config-owner-content[cisco.com-rule1])# show eql
```

Step 10 Display the content rule to show the cache configuration.

```
CS150(config-owner-content[cisco.com-rule1])# show rule
```

Transparent Caching Sample Configuration Output

The following is a very basic CSS 11000 switch configuration output for transparent caching using CLI commands. Use the **show running-config** CLI command to display the switch configuration parameters after provisioning the CSS 11000 switch.

```
!Generated MAY 19 15:36:34
!Active version: ap0310029s
prompt TransCache
configure
!************************ INTERFACE ************************
interface ethernet-12
  bridge vlan 2
!************************* CIRCUIT *************************
circuit VLAN1
  ip address 10.1.1.254 255.255.255.0
circuit VLAN2
  ip address 192.168.1.254 255.255.255.0
!************************* SERVICE *************************
service Origin1
  ip address 10.1.1.1
  keepalive frequency 60
service TCacheServer1
  ip address 10.1.1.11
  type transparent-cache
```

```
  port 80
  active
service TCacheServer2
  ip address 10.1.1.12
  type transparent-cache
  port 80
  active
!**************************** EQL ****************************
eql Cacheable
  description "This EQL contains extensions of cacheable content"
  extension pdf "Acrobat"
  extension fdf "Acrobat Forms Document"
  extension au "Sound audio/basic"
  extension bmp "Bitmap Image"
  extension z "Compressed data application/x-compress"
  extension gif "GIF Image image/gif"
  extension html "Hypertext Markup Language text/html"
  extension js "Java script application/x-javascript"
  extension mocha
  extension jpeg "JPEG image image/jpeg"
  extension jpg
  extension jpe
  extension jfif
  extension pjpeg
  extension pjp
  extension mp2 "MPEG Audio audio/x-mpeg"
  extension mpa
  extension abs
  extension mpeg "MPEG Video video/mpeg"
  extension mpg
  extension mpe
  extension mpv
  extension vbs
  extension m1v
  extension pcx "PCX Image"
  extension txt "Plain text text/plain"
  extension text
  extension mov "QuickTime video/quicktime"
  extension tiff "TIFF Image image/tiff"
  extension tar "Unix Tape Archive application/x-tar"
  extension pcx "PCX Image"
  extension txt "Plain text text/plain"
  extension text
  extension mov "QuickTime video/quicktime"
  extension tiff "TIFF Image image/tiff"
  extension tar "Unix Tape Archive application/x-tar"
  extension avi "Video for Windows video/x-msvideo"
  extension wav "Wave File audio/x-wav"
  extension gz "application/x-gzip"
  extension zip "ZIP file application/x-zip-compressed"

!************************** OWNER **************************
owner cisco.com
  content L5_Transparent
    protocol tcp
    port 80
    url "/*" eql Cacheable
    balance urlhash
    add service TCacheServer1
    add service TCacheServer2
    active
```

Note There is no virtual IP address because the switch is transparent (not visible to the client).

Note Anything that is cacheable should be load balanced using the URL and sent to the cache servers.

CHAPTER 5

Configuring Proxy-Style Caching

All of the features outlined in this chapter are associated with Content Engines operating in nontransparent mode. In this mode, end user web browsers need to be explicitly configured to the IP address or host name of the Cisco Content Engine products, and there is no need for additional hardware such as Layer 4 switches or WCCP-enabled routers to intercept user requests as in transparent caching. Customers are normally interested in deploying transparent network caching, but some customers may have a legacy requirement for a nontransparent cache. For more information on transparent caching configurations with the Content Engine, see the "Configuring Transparent Caching" section on page 4-1.

This chapter discusses the following nontransparent caching features:

- HTTP Proxy Caching
- FTP Proxy Caching
- SSL Tunneling

Proxy Mode Operation

A proxy-style request arrives with the same destination IP address as the Content Engine; it has been specifically routed to the Content Engine by the client. The Content Engine supports up to eight incoming ports each for FTP, HTTPS, HTTP, MMS, and RTSP proxy modes. The incoming proxy ports can be the same ports that are used by transparent mode services. The incoming proxy ports can be changed without stopping any WCCP services running on the Content Engine or on other Content Engines in the farm.

In proxy mode, the Content Engine services any protocols for which it has been configured. The supported protocols are HTTP, HTTPS, FTP, MMS, and RTSP. If the Content Engine is not configured to support a received protocol, the proxy server returns an error. For example, if port 8080 is configured to run an HTTP and HTTPS proxy service, an FTP request coming to this port is rejected.

TCP ports reserved for system or network services should not be used for proxying services in transparent mode or in proxy mode (for example, DNS or FTP). If more than eight ports for a protocol are required, the administrator can configure multiple dynamic WCCP services. Intercepted FTP, HTTP, and HTTPS requests addressed to other proxy servers (received on transparent mode ports) are serviced according to the **proxy-protocols transparent** command parameters.

HTTP Proxy Caching

The Content Engine accepts proxy-style requests in nontransparent mode from a web browser when the incoming proxy ports are configured with the **http proxy incoming** *ports* option. Up to eight incoming proxy ports can be specified on a single command line or on multiple command lines.

To configure the Content Engine to direct all HTTP miss traffic to a parent cache (without using ICP or WCCP), use the **http proxy outgoing host** *port* option, where **host** is the system name or IP address of the outgoing proxy server, and *port* is the port number designated by the outgoing (upstream) server to accept proxy requests. See the "Parent Proxy Failover" section on page 7-2 for more information on the **http proxy outgoing** command.

FTP Proxy Caching

FTP proxy caching in a Content Engine refers to the ability to service FTP-over-HTTP requests. The transport between the web browser and cache is over HTTP, and the cache initiates an FTP transfer to the origin FTP server.

Note The Content Engine caches FTP traffic only when the client uses the Content Engine as a proxy server for FTP requests. All FTP traffic that was sent directly from the web client to an FTP server, if transparently intercepted by the Content Engine, is treated as non-HTTP traffic. If a web browser is not explicitly configured for a proxy, then the browser will initiate an end-to-end FTP connection itself, and this will not be intercepted by the Content Engine.

The **ftp proxy** command enables the Content Engine to operate in environments where WCCP is not enabled, or where client browsers have previously been configured to use a legacy FTP proxy server.

The **ftp proxy incoming** *port* option specifies a port number used by the proxy server to receive requests. This number can range from 1 to 65535. A maximum of eight incoming proxy ports can be configured. You can share these incoming proxy ports with transparent-mode services and also with the other proxy-mode protocols supported by the Content Engine, such as HTTP and HTTPS. The **ftp proxy incoming** *port* option is disabled by default.

Note The Content Engine accepts and services FTP requests only on the ports configured for FTP proxy. All the FTP requests on other proxy mode ports are rejected in accordance with the error-handling settings on the Content Engine.

The Content Engine accepts FTP requests when URLs specify the FTP protocol (for example, GET ftp://ftp.cisco.com/pub/cao/READM). For these requests, the client uses HTTP as the transport protocol with the Content Engine, whereas the Content Engine uses FTP with the FTP server.

The Content Engine caches both the FTP file objects and directory listings in the cache file system (cfs). The Content Engine transforms the regular directory listings from the FTP server into HTML, with links that the client users can point to and click to download files.

When the Content Engine receives an FTP request from the web client, it first looks in its cache. If the object is not in its cache, it fetches the object from an upstream FTP proxy server (if one is configured), or directly from the origin FTP server.

The FTP proxy supports anonymous as well as authenticated FTP requests. Only base64 encoding is supported for authentication. The FTP proxy accepts all FTP URL schemes defined in RFC 1738. In the case of a URL in the form ftp://user@site/dir/file, the proxy sends back an authentication failure reply and the browser supplies a popup window for the user to enter login information.

The Content Engine can apply the Rules Template to FTP requests based on server name, domain name, server IP address and port, client IP address, and URL.

The Content Engine logs FTP transactions in the transaction log, in accordance with Squid syntax.

Examples

This example configures an incoming FTP proxy on ports 8080, 8081, and 9090. Up to eight incoming proxy ports can be configured on the same command line.

```
ContentEngine(config)# ftp proxy incoming 8080 8081 9090
```

This example removes one FTP proxy port from the list entered in the previous example. Ports 8080 and 9090 remain FTP proxy ports.

```
ContentEngine(config)# no ftp proxy incoming 8081
```

This example disables all the FTP proxy ports.

```
ContentEngine(config)# no ftp proxy incoming
```

This example configures an upstream FTP proxy with the IP address 172.31.76.76 on port 8888.

```
ContentEngine(config)# ftp proxy outgoing host 172.31.76.76 8888
```

This example specifies an anonymous password string for the Content Engine to use when contacting FTP servers. The default password string is anonymous@hostname.

```
ContentEngine(config)# ftp proxy anonymous-pswd newstring@hostname
```

This example configures the maximum size in kilobytes of an FTP object that the Content Engine will cache. By default, the maximum size of a cacheable object is not limited.

```
ContentEngine(config)# ftp object max-size 15000
```

This example forces the Content Engine to revalidate all objects for every FTP request.

```
ContentEngine(config)# ftp reval-each-request all
```

This example configures a maximum Time To Live of 3 days in cache for directory listing objects and file objects.

```
ContentEngine(config)# ftp max-ttl days directory-listing 3 file 3
```

SSL Tunneling

The SSL tunneling protocol allows a proxy server to act as a tunnel between the end user and the origin server. The client asks for an SSL tunnel through an HTTP request. This allows the Content Engine to service CONNECT method requests in the form https://*url* for tunneling SSL over HTTP.

Note Browsers only initiate HTTPS-over-HTTP requests when explicitly configured for a proxy. If a web browser is not explicitly configured for a proxy, then the browser will initiate an HTTP-over-SSL connection itself, and because this is on TCP port 443, it will not be intercepted by a Content Engine. Even if a Content Engine did intercept this request, it would not be able to do anything with it. SSL on port 443 uses end-to-end encryption and any transparent device in the middle will not see anything more than a stream of random bytes.

HTTPS Proxy Features

Cisco ACNS 4.1 software supports HTTPS in the following two scenarios:

- The Content Engine receives an HTTPS request sent by a web client configured to use the Content Engine as an HTTPS proxy server.
- The Content Engine in transparent mode intercepts a request sent by a web client to another HTTPS proxy server.

In both cases the Content Engine creates a connection to the origin server (directly or through another proxy server) and allows the web client and origin server to set up an SSL tunnel through the Content Engine.

Table 5-1 shows CLI commands associated with HTTPS proxy features. The order in which the CLI commands are entered is not important.

Table 5-1 HTTPS Proxy Features

HTTPS Proxy Features	Related CLI Commands (Abbreviated Syntax)
Supports up to eight incoming proxy ports.	**https proxy incoming** *port_1–65535, port, . . .*
Configures a WCCP service and an HTTPS incoming proxy on the same port. Shares proxy port with transparent services.	**https proxy incoming** *ports_1–65535* **wccp custom-web-cache** . . .
Configures outgoing HTTPS proxy server, using global exclude option for HTTPS proxy	**proxy-protocols outgoing-proxy exclude** *domain_name* **https proxy outgoing host** {*hostname* | *ip_address*} *port_1–65535*
Original versus default outgoing HTTPS proxy decision process	**proxy-protocols transparent** {**default-server** | **original-proxy**}
Handles in transparent mode an HTTPS request bound for another proxy host	**proxy-protocols transparent** {**default-server** | **original-proxy**}

HTTPS traffic is encrypted and cannot be interpreted by the Content Engine or any other device between the web client and the origin server. HTTPS objects are not cached.

The Content Engine acting as an HTTPS proxy server supports up to eight ports. It can share the ports with transparent-mode services and with HTTP. In proxy mode, the Content Engine accepts and services the HTTPS requests on the ports specified with the **https proxy incoming** command. All HTTPS requests on other proxy-mode ports are rejected in accordance with the error-handling settings on the Content Engine. In transparent mode, all HTTPS proxy-style requests intended for another HTTPS proxy server are accepted. The Content Engine acts on these transparently received requests in accordance with the **proxy-protocols transparent** command.

When the Content Engine is configured to use an HTTPS outgoing proxy with the **https proxy outgoing host** command, all incoming HTTPS requests are directed to this outgoing proxy. The **proxy-protocols outgoing-proxy exclude** command specifies a global proxy exclude domain effective for all proxy server protocols, including HTTPS. The Content Engine applies the following logic when an outgoing proxy server is configured:

- If the destination server is specified by the global exclude option, then go directly to the destination server.
- If the destination server is not specified by the global exclude option and the request is HTTP, go directly to the destination server.
- If the destination server is not specified by the global exclude option, then go to the outgoing proxy server.

When a Content Engine intercepts a proxy request intended for another proxy server and there is no outgoing proxy configured for HTTPS, and the **proxy-protocols transparent default-server** command is invoked, the Content Engine addresses the request to the destination server directly and not to the client's intended proxy server.

Statistics Reporting

Only connection statistics are reported. Because requests and responses are sent through the secure tunnel, the Content Engine is not able to identify the number of requests sent, or the number of bytes per request. Thus, the request and transaction per second (TPS) statistics are not available for HTTPS.

Transaction Logging

The Content Engine logs HTTPS transactions in the transaction log in accordance with Squid syntax. One log entry is made for each HTTPS connection, though many transactions are performed per connection. The Content Engine is not aware of objects conveyed through the SSL tunnel, only the HTTPS server name.

Examples

In this example, the Content Engine is configured as an HTTPS proxy server, and accepts HTTPS requests on port 8081. Only a single port is supported in the HTTPS protocol.

```
ContentEngine(config)# https proxy incoming 8081
```

In this example, the Content Engine is configured to forward HTTPS requests to an outgoing proxy server (10.1.1.1) on port 8880.

```
ContentEngine(config)# https proxy outgoing host 10.1.1.1 8880
```

In this example, a domain name is excluded from being forwarded to an outgoing proxy server.

```
ContentEngine(config)# proxy-protocols outgoing-proxy exclude cruzio.com
ContentEngine(config)# proxy-protocols transparent default-server
```

CHAPTER 6

Configuring Cache Parameter Settings

Caching of Authenticated Content

The authenticated data caching feature allows basic and NT LAN Manager (NTLM) authenticated content to be cached and served to more than one user, while maintaining security. If an authenticated object is cached, then subsequent requests for that object (from new users) require authentication. The cached object is revalidated with the origin server through the authorization header for the new user. If the user is not authorized, the server sends a 401 (Unauthorized) response. If the user is authorized and the object is not modified, the cached object is served to the client. See the "NTLM Authentication" section on page 10-3 for further information on NTLM authentication.

This example enables caching of basic and NTLM authenticated content:

```
Console(config)# http cache-authenticated all

Console(config)# show http cache-authenticated all
Basic authenticated objects are cached.
NTLM authenticated objects are cached.
```

Cache Freshness

The ACNS software Cache application can be tuned to balance HTTP and FTP object freshness with cache hit rate. The Cache application default parameters are weighted to ensure fresh content over maximizing the cache hit rate (to avoid the undesirable scenario of increasing the cache hit rate by serving stale content).

Table 6-1 explains the cache freshness features.

Table 6-1 Cache Freshness Settings

Configurable Cache Freshness Feature	Function
Minimum Time to Live Maximum Time to Live	Limits duration of objects in the cache.
Age multiplier	Determines at what percentage of an object's maximum Time To Live the Cache application issues an if-modified-since (IMS) request to the origin web server to validate the freshness of the object.

Table 6-1 Cache Freshness Settings (continued)

Configurable Cache Freshness Feature	Function
Revaluate each request	Forces the Cache application to revalidate all requests to the origin web server using an IMS request. The **reval-each-request** option enables the Content Engine to revalidate all objects requested from the cache, text objects only, or none at all.
Serve-IMS	Controls the ability of the cache to answer authoritatively on receiving an IMS request from the web browser.

Minimum and Maximum Time To Live Settings

The minimum and maximum Time To Live (TTL) settings permit the administrator to limit the duration of HTTP and FTP objects in the cache. By default, HTTP cacheable objects are kept for 5 minutes minimum and 3 to 7 days maximum (3 days for text-type objects, 7 days for binary). If an object is marked with expiry information from the web server different from the administrator settings, then the web server values take precedence.

For HTTP and FTP objects, use the **http min-ttl** and **ftp min-ttl** global configuration commands to set the minimum Time To Live, and the **http max-ttl** and **ftp max-ttl** command to set maximum Time To Live. The command syntax is as follows:

ftp max-ttl days directory-listing *dlmax_days* **file** *fmax_days*

ftp max-ttl hours directory-listing *dlmax_hours* **file** *fmax_hours*

ftp max-ttl minutes directory-listing *dlmax_ min* **file** *fmax_min*

ftp max-ttl seconds directory-listing *dlmax_ sec* **file** *fmax_sec*

ftp min-ttl *min_minutes*

http max-ttl days text *textdays* **binary** *bindays*

http max-ttl hours text *texthours* **binary** *binhours*

http max-ttl minutes text *textminutes* **binary** *binminutes*

http max-ttl seconds text *textseconds* **binary** *binseconds*

http min-ttl *minutes*

The following example configures the Cache application to keep HTTP and FTP objects in the cache for a minimum of 10 minutes, and a maximum of 24 hours (1 day).

```
ContentEngine(config)# http min-ttl 10
ContentEngine(config)# ftp min-ttl 10
ContentEngine(config)# http max-ttl days text 1 binary 1
ContentEngine(config)# ftp max-ttl hours directory-listing 24 file 24
```

Caching of Binary Content with Cookies

The Cache application can cache binary objects that were served with cookies attached. Use the **http cache-cookies** global configuration command to enable this feature.

The following example enables caching of binary content with cookies attached.

```
ContentEngine(config)# http cache-cookies
ContentEngine(config)#
```

Maximum Object Size

The Cache application can specify the maximum size of an FTP or HTTP object that can be stored in the cache. The maximum size limit for both an FTP object, and an HTTP object size is 204,799 kilobytes.

The following example sets the maximum size for an HTTP object as 500 kilobytes, and the maximum FTP object size as 2 megabytes.

```
ContentEngine(config)# ftp object max-size 2000
ContentEngine(config)# http object max-size 500
```

Aborting Selected Objects

The **http cache-on-abort** command provides user-defined thresholds to determine whether or not the Content Engine will complete the download of an object when the client has aborted the request. When the download of an object is aborted before it is completed, the object is not stored on the Content Engine or counted in the hit-rate statistics. Typically, a client aborts a download by clicking the Stop icon on the browser, or by closing the browser during a download.

If the **http cache-on-abort** command is enabled, the Content Engine will use a selective algorithm to determine whether to continue to cache an object if the client has aborted the request. If disabled, the Content Engine will always continue to download an object to the cache even if a client has aborted the request.

If the **http cache-on-abort min-threshold** command is used, the Content Engine will continue to cache an object if the number of kilobytes remaining to download from the server is less than or equal to the Minimum Threshold value. The default value is 32 KB.

If the **http cache-on-abort max-threshold** command is used, the Content Engine will not continue to cache an object if the number of kilobytes remaining to download from the server is greater than the Maximum Threshold value. The default value is 256 KB.

If the **http cache-on-abort percent** command is used, the Content Engine will continue to cache an object if the percentage of the object already downloaded is greater than the Percentage Threshold value. The default value is 80 percent.

Note Any combination of thresholds can be specified. They are used in the order shown above. If the **http cache-on-abort** command is enabled and all **http cache-on-abort** thresholds are disabled, then the Content Engine always aborts downloading an object to the cache. If the Content Engine determines that there is another client currently requesting the same object, downloading is not aborted. The Content Engine only applies those thresholds that have been enabled.

In this example, the Content Engine is configured to always continue downloading an object to the cache:

```
ContentEngine(config)# no http cache-on-abort
```

In this example, the Content Engine is configured to use the default minimum threshold when the **cache-on-abort** option has been enabled, and the threshold is set to 16 kilobytes:

```
ContentEngine(config)# http cache-on-abort min 16
```

In this example, the Content Engine is configured to not consider the minimum threshold:

```
ContentEngine(config)# no http cache-on-abort min
```

The **cache-on-abort max-threshold** and **percent** thresholds are configured like the minimum threshold shown in the examples.

Caching of HTTP Range Requests

The Content Engine serves HTTP Range requests that include a Range header requesting the specified range of the object instead of the whole object from the cache if the requested range exists in the Content Engine cache. Specifically, the Content Engine handles Range requests with the following logic:

```
lookup the object in the cache;
if object in the cache
{
            check whether the requested ranges are in the cache;
            if the requested ranges are in cache then serve the request from cache;
            else pipe through the request;
}
else pipe through the request;
```

If a client has a partial copy of an entity in its cache and wishes to have an up-to-date copy of the entire entity in its cache, it could use the Range request header with a conditional GET request (using either or both If-Unmodified-Since and If-Match.) However, if the condition fails because the entity has been modified, the client would then have to make a second request to obtain the entire current entity.

The If-Range header allows a client to short-circuit the second request. Informally, the meaning of this header is “If the entity is unchanged, send me the part(s) that I am missing; otherwise, send me the entire new entity.”

If-Range = "If-Range" ":" (entity-tag | HTTP-date)

An If-Range request from the client will be handled by the cache as follows:

```
lookup the object in the cache;
if (object is in the cache
    AND requested ranges are in the cache;
    AND entity tag given in the If-Range header
            matches cached object's entity-tag){
        serve partial request from the cache
} else {
        connect to remote server
        retrieve requested range, send data to client
}
```

If the If-Range header has a valid HTTP date instead of an entity tag, then the HTTP date is matched with the Last-Modified date of the cached object.

If the client has no entity tag for an entity but does have a Last-Modified date, it may use that date in an If-Range header. The If-Range header should only be used together with a Range header, and must be ignored if the request does not include a Range header, or if the server does not support subrange operation.

If the entity tag given in the If-Range header matches the current entity tag for the entity, then the server should provide the specified subrange of the entity using a 206 (Partial content) response. If the entity tag does not match, then the server should return the entire entity using a 200 (OK) response.

Note The **http cache-on-abort** feature must be disabled for the caching of HTTP range requests to occur. Some client applications close the server connection immediately after receiving the response header for the normal GET request (for example, to a PDF file). If the **http cache-on-abort** command is enabled, later Range requests to that object will not be cacheable.

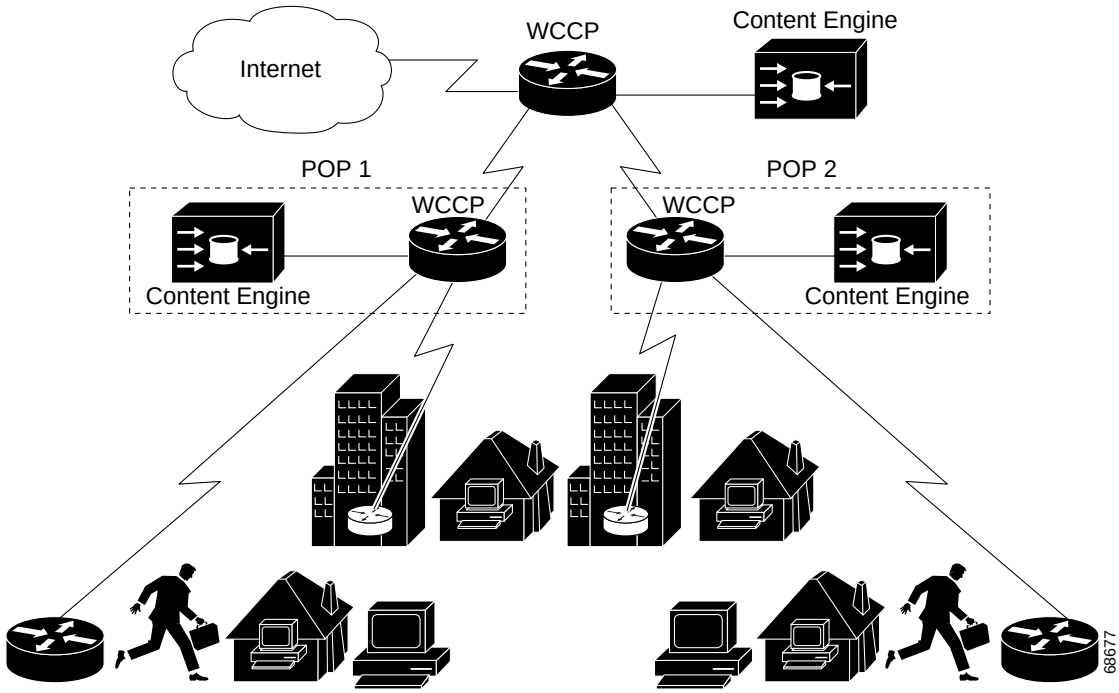
CHAPTER 7

Configuring a Primary Proxy Server

Overview

Because a Cisco Content Engine can be transparent to the client and to the network operation, customers can easily place Content Engines in several network locations in a hierarchical fashion. For example, if an Internet service provider (ISP) deploys a Content Engine at its main point of access to the Internet, all of its points of presence (POPs) benefit. Figure 7-1 depicts a typical caching hierarchy using Content Engines.

Figure 7-1 Caching Hierarchy

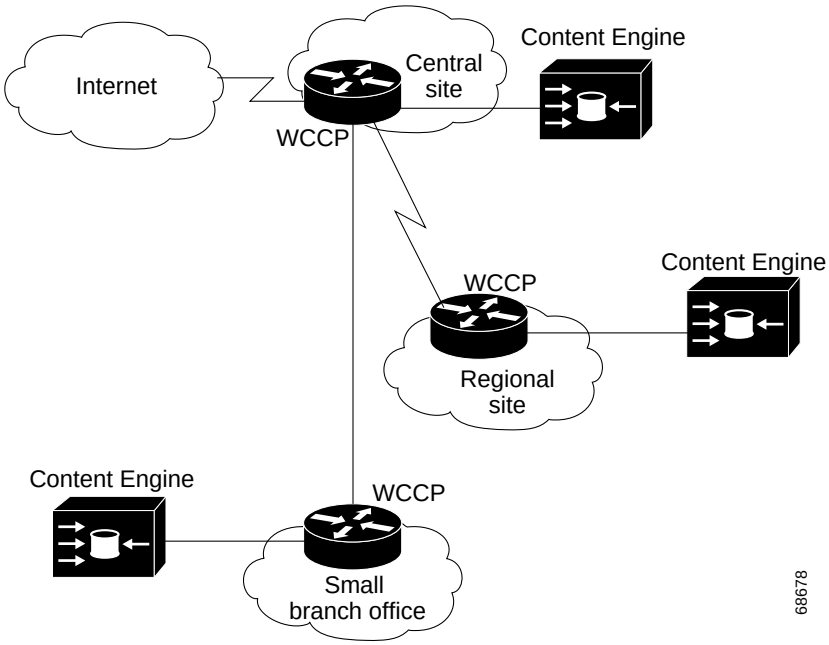


Client requests reach the Content Engine and are fulfilled from its storage. To further improve service to quality clients, ISPs can deploy Content Engines at each POP. Then, when a client accesses the Internet, the request is first redirected to the POP Content Engine. If the POP Content Engine is unable to fulfill the request from local storage, it makes a normal web request to the end server.

Upstream, this request is redirected to the Content Engine at the main Internet access point. If the request is fulfilled by the Content Engine, traffic on the main Internet access link is avoided, the origin web servers experience lower demand, and the client experiences better network response times.

Enterprise networks can apply this hierarchical transparent architecture in the same way. (See Figure 7-2.)

Figure 7-2 Caching Hierarchy in Enterprise Solutions



Parent Proxy Failover

Note The proxy failover feature supports HTTP only, not HTTPS or FTP.

The **http proxy outgoing** option can configure up to eight backup Content Engines or any standard proxy servers for the HTTP proxy failover feature. One outgoing proxy server functions as the primary server to receive and process all cache-miss traffic. If the primary outgoing proxy server fails to respond to the HTTP request, the server is noted as failed and the requests are redirected to the next outgoing proxy server until one of the proxies services the request. The **no http proxy outgoing connection-timeout** option causes the timeout to be set to the default value of 300 milliseconds.

To explicitly designate a proxy as primary, use the **http proxy outgoing host** *ip-address port* **primary** command. If several hosts are configured with the **primary** keyword, the last one configured becomes the primary failover host. Failover occurs in the order the proxy servers were configured. If all of the configured proxy servers fail, the Content Engine can optionally redirect HTTP requests to the origin server specified in the HTTP header with the **http proxy outgoing origin-server** command. If the **origin-server** option is not enabled, the client receives an error message. Response errors and read errors are returned to the client because it is not possible to detect whether these errors are generated at the origin server or at the proxy.

A background process monitors the state of the proxy servers. A monitoring interval is configured with the **http proxy outgoing monitor** command. This monitor interval is the interval of time over which the proxy servers are polled. If one of the proxy servers is unavailable, the polling mechanism waits for the connect timeout (300 milliseconds) before polling the next server. The state of the proxy servers can be viewed in syslog NOTICE messages and with the **show http proxy** command.

Note Only one of the outgoing proxy servers is available at a time. They cannot be used simultaneously.

Requests with a destination specified in the **proxy-protocols outgoing-proxy exclude** command bypass the primary outgoing proxy, and the failover proxies.

When an HTTP request intended for another proxy server is intercepted by the Content Engine in transparent mode, the Content Engine forwards the request to the intended proxy server if the **proxy-protocols transparent original-proxy** command was entered.

By default, the Content Engine strips the hop-hop 407 (Proxy Authentication Required) error code sent by Internet proxy. If the **http proxy outgoing preserve-407** command is invoked, the Content Engine sends the 407 error code to the client, and the Internet proxy authenticates the client.

Examples

In this example, the host 10.1.1.1 on port 8088 is designated the primary outgoing proxy server, and host 10.1.1.2 is a backup proxy server.

```
ContentEngine(config)# http proxy outgoing host 10.1.1.1 8088 primary
ContentEngine(config)# http proxy outgoing host 10.1.1.2 220
```

In this example, the Content Engine is configured to redirect requests directly to the origin server if all of the proxy servers fail.

```
ContentEngine(config)# http proxy outgoing origin-server
```

In this example, the Content Engine is configured to monitor the proxy servers every 120 seconds.

```
ContentEngine(config)# http proxy outgoing monitor 120
```

To disable any of the above commands, use the **no** version of the command.

Proxy Failover show Commands

```
ContentEngine# show http proxy
Incoming Proxy-Mode:
  Servicing Proxy mode HTTP connections on ports:   8080

Outgoing Proxy-Mode:
  Primary proxy server: 172.16.63.150   port 1 Failed
  Backup proxy servers: 172.16.236.151  port 8005
                        172.16.236.152  port 123
                        172.16.236.153  port 65535 Failed
                        172.16.236.154  port 10
Monitor Interval for Outgoing Proxy Servers is 60 seconds
Timeout period for probing Outgoing Proxy Servers is 300000 microseconds
Use of Origin Server upon Proxy Failures is disabled.
```

Statistics

```
ContentEngine# show statistics http requests
Statistics - Requests
                                                Total            % of Requests
```

```
                          ---------------------------------------------------
     Total Received Requests:                  49103                        -
              Forced Reloads:                    109                      0.2
               Client Errors:                     23                      0.0
               Server Errors:                    348                      0.7
                 URL Blocked:                      0                      0.0
      Sent to Outgoing Proxy:                      0                      0.0
Failures from Outgoing Proxy:                      0                      0.0
Excluded from Outgoing Proxy:                      0                      0.0
            ICP Client Hits:                       0                      0.0
            ICP Server Hits:                       0                      0.0
          HTTP 0.9 Requests:                       2                      0.0
          HTTP 1.0 Requests:                   49101                    100.0
          HTTP 1.1 Requests:                       0                      0.0
       HTTP Unknown Requests:                      0                      0.0
           Non HTTP Requests:                      0                      0.0
         Non HTTP Responses:                      46                      0.1
      Chunked HTTP Responses:                      0                      0.0
        Http Miss Due To DNS:                      0                      0.0
     Http Deletes Due To DNS:                      0                      0.0
  Objects cached for min ttl:                   2674                      5.

ContentEngine# show statistics http proxy outgoing

HTTP Outgoing Proxy Statistics
IP              PORT    ATTEMPTS   FAILURES
-------------------------------------------------
172.16.23.150    8000    0          0
172.16.23.151    8080    0          0
172.16.23.152    9000    0          0
172.16.23.153    9001    0          0
172.16.23.154    9005    0          0


 Requests when all proxies were failed: 0
```

Related Commands

proxy-protocols

rule no-proxy

rule use-proxy

show http

show http proxy

show statistics http requests

show statistics http proxy outgoing

Handling Proxy-Style Requests

When in transparent caching mode, the Content Engine can intercept requests sent to another proxy and send these requests to one of the following two destinations:

- Default server—This is the default option. The Content Engine retrieves the objects from the web server itself, or, if configured to use an outgoing proxy for this protocol, it forwards the request to its outgoing proxy. In this scenario, the client browser configuration is ignored and the Content Engine configuration is used to retrieve the object from the server.
- Original proxy—The Content Engine forwards the request to the original proxy that the client addressed the request to. This may be different than the Content Engine's own outgoing proxy for the specified protocol.

Use the **proxy-protocols** global configuration command to specify a domain name, host name, or IP address to be excluded from proxy forwarding. To selectively turn off outgoing-proxy exclude lists or to force transparently received proxy-style requests to be fulfilled by the Content Engine, use the **no** form of this command.

proxy-protocols outgoing-proxy exclude {**enable** | **list** *word*}

proxy-protocols transparent {**default-server** | **original-proxy| reset**}

no proxy-protocols {**outgoing-proxy exclude** {**enable** | **list** *word*} | **transparent** {**default-server** | **original-proxy**}}

The **proxy-protocols outgoing-proxy exclude** option allows the administrator to specify a single domain name, host name, or IP address to be globally excluded from proxy forwarding. Domains are entered as an ASCII string, separated by spaces. The wildcard character * (asterisk) can be used for IP addresses (for instance, 172.16.*.*). Only one exclusion can be entered per command line. Enter successive command lines to specify multiple exclusions. Requests with a destination specified in the **proxy-protocols outgoing-proxy exclude** command bypass the Content Engine proxy as well as the failover proxies.

When you enter the **proxy-protocols transparent default-server** global configuration command, the Content Engine forwards intercepted HTTP, HTTPS, and FTP proxy-style requests to the corresponding outgoing proxy server, if one is configured. If no outgoing proxy server is configured for the protocol, the request is serviced by the Content Engine and the origin server.

The **proxy-protocols transparent original-proxy** option specifies that requests sent by a web client to another proxy server, but intercepted by the Content Engine in transparent mode, be forwarded to the intended proxy server.

The following example configures the Content Engine to forward intercepted HTTPS proxy-style requests to an outgoing proxy server. The domain name cruzio.com is excluded from proxy forwarding. The **show proxy-protocols** command verifies the configuration.

```
ContentEngine(config)# https proxy outgoing host 172.16.10.10 266
ContentEngine(config)# proxy-protocols transparent default-server
ContentEngine(config)# proxy-protocols outgoing-proxy exclude cruzio.com

ContentEngine# show proxy-protocols all
Transparent mode forwarding policies: default-server
Outgoing exclude domain name: cruzio.com
```

The following example configures the Content Engine to forward intercepted HTTP proxy-style requests to the intended proxy server.

```
ContentEngine(config)# proxy-protocols transparent original-proxy
```

Internet Cache Protocol

Internet Cache Protocol (ICP) is a lightweight message format used for communicating among web caches and for supporting interoperability with older proxy protocols. ICP is used to exchange hints about the existence of URLs in neighboring caches. Caches exchange ICP queries and replies to gather information for use in selecting the most appropriate location from which to retrieve an object.

Although ICP has traditionally been a way to scale the overall size of a cluster of caches beyond a single unit, history has shown ICP to be a very poor way of scaling a cache clustering solution. In fact, because of the way that traffic is currently directed towards a transparent network cache cluster, the requirement for ICP is all but negated for the majority of cache deployments.

The ICPv2 protocol is documented in two standards documents:

- *RFC 2186: Internet Cache Protocol (ICP), version 2*
- *RFC 2187: Application of Internet Cache Protocol (ICP), version 2*

Note The ability to act as both an ICP server (servicing requests from neighboring caches) and an ICP client (sending requests to neighboring caches) is supported.

The following example restricts ICP parent and sibling to specific domain sets:

```
ContentEngine(config)# icp client add-remote-server 1.1.1.1 parent icp-port 3130 http-port
3128 domain_x.com domain_y.com domain_z.com
ContentEngine(config)# icp client add-remote-server 1.1.1.1 sibling icp-port 3130
http-port 3128 domain_a.com domain_b.com domain_c.com

ContentEngine(config)# icp client enable
ContentEngine(config)#
```

CHAPTER 8

Configuring URL Filtering

URL Filtering

Some enterprises have a requirement to monitor, manage, and restrict employee access to nonbusiness and objectionable content on the Internet. Employees or students can be allowed or denied access to websites, or can be coached with information about acceptable use of the Internet. By having a URL filtering scheme on the Content Engines, organizations realize an immediate return on investment as a result of increased productivity and recaptured network bandwidth, while reducing legal liability.

The URL filtering features presented in this chapter allow the Content Engine to control client access to websites in any of the following ways:

- Deny access to URLs specified in a list.
- Permit access only to URLs specified in a list.
- Direct traffic to a N2H2 server for filtering.
- Direct traffic to a Websense enterprise server for filtering.
- Filter traffic with Secure Computing Corporation SmartFilter™ Software, Release 3.0.2 for Cisco Content Engine ACNS Software, Release 4.1.x.

Only one form of URL filtering can be active at a time.

Note The URL filtering feature in ACNS 4.1 software differs from the URL feature in other releases as follows: There is now an **enable** command option for the **good-sites** and **bad-sites** options; the URL list filenames and the customized blocking message directory name are now specified in the command-line interface; you can now use the **url-filter local-list-reload** EXEC command to dynamically refresh a local URL list; and **bad-sites-block** has been changed to **bad-sites-deny**.

Order of Precedence

The **url-filter** command takes precedence over the **rule** command to the extent that even the **rule no-block** command is executed only if the **url-filter** command has *not* blocked the request.

URL Filtering with URL Lists

You can configure the Content Engine to deny client requests for URLs that are listed in a badurl.lst file, or configure it to fulfill only requests for URLs in a goodurl.lst file.

To deny requests for specific URLs, do the following:

Step 1 Create a plain text file named badurl.lst.

In this file, enter the URLs that you want to block. The list of URLs in the badurl.lst file must be written in the form http://www.domain.com/ and delimited with carriage returns.

Step 2 Copy the badurl.lst file to the /local1 system file system (sysfs) directory of the Content Engine.

Note We recommend creating a separate directory under local1 to hold the bad and good lists, for example, /local1/filtered_urls.

Step 3 Use the **url-filter bad-sites-deny** command to point to the bad URL list.

```
Console(config)# url-filter bad-sites-deny local/local1/badurl.lst
```

Step 4 Use the **url-filter bad-list enable** command to actively deny the URLs.

```
Console(config)# url-filter enable bad-list
```

To permit specific URLs to the exclusion of all other URLs, do the following:

Step 1 Create a plain text file named goodurl.lst.

In this file, enter the URLs that you want to exclusively allow. The list of URLs in the goodurl.lst file must be written in the form http://www.domain.com and delimited with carriage returns.

Step 2 Copy the goodurl.lst file to the /local1 sysfs directory of the Content Engine.

Note We recommend creating a separate directory under local1 to hold the bad and good lists, for example, /local1/filtered_urls.

Step 3 Use the **url-filter good-sites-allow** command to point to the goodurl.lst file.

```
Console(config)# url-filter good-sites-allow local/local1/goodurl.lst
```

Step 4 Use the **url-filter good-sites-allow enable** command to actively permit only the good URLs.

```
Console(config)# url-filter good-sites-allow enable
```

Note When you update the badurl.lst or goodurl.lst file, use the **url-filter local-list-reload** EXEC command to recopy the URL list file to the Content Engine.

Use the **no** form of the command to disable blocking, Websense or N2H2 permission requests (for example, **no url-filter bad-sites-deny**).

Custom Blocking Messages

The Content Engine with ACNS 4.1 software can be configured to return a customized blocking message to the client. The custom message must be an administrator-created HTML page named block.html. Make sure to copy all embedded graphics associated with the custom message HTML page to the same directory that contains the block.html file. To enable the customized blocking message, use the **url-filter custom-message** command and specify the directory name.

To disable the custom message, use the **no url-filter custom-message** command.

The **url-filter custom-message** command can be enabled and disabled without affecting the **good-sites-allow** and **bad-sites-deny** configuration.

Note Do not use local1 or local2 as directories for custom blocking messages. Create a separate directory under local1 or local2 for holding the custom message file.

In the block.html file, objects (such as .gif, .jpeg, and so on) must be referenced within the custom message directory string /content/engine/blocking/url, as shown in the example below.

In the following example a block.html file displays a custom message *This page is blocked by the Content Engine* when the Content Engine intercepts a request to the blocked site.

Note Contact your administrator if you have any questions concerning access to the blocked site you requested.

```
<TITLE>Cisco Content Engine example customized message for url-filtering</TITLE>
<p>
<H1>
<CENTER><B><I><BLINK>
<FONT COLOR="#800000">P</FONT>
<FONT COLOR="#FF00FF">R</FONT>
<FONT COLOR="#00FFFF">A</FONT>
<FONT COLOR="#FFFF00">D</FONT>
<FONT COLOR="#800000">E</FONT>
<FONT COLOR="#FF00FF">E</FONT>
<FONT COLOR="#00FFFF">P</FONT>
<FONT COLOR="#FF8040">'</FONT>
<FONT COLOR="#FFFF00">S</FONT>
</BLINK>
<FONT COLOR ="#0080FF">Blocked Page</FONT>
</I></B></CENTER>
</H1>
<p>
<p>
<IMG src="/content/engine/blocking/url/my.gif">
<p>
This page is blocked by the Content Engine.
<p>
```

To disable the **custom-message** option without disabling URL filtering, enter the URL filtering command without the **custom-message** option (for example, **url-filter good-sites-allow**).

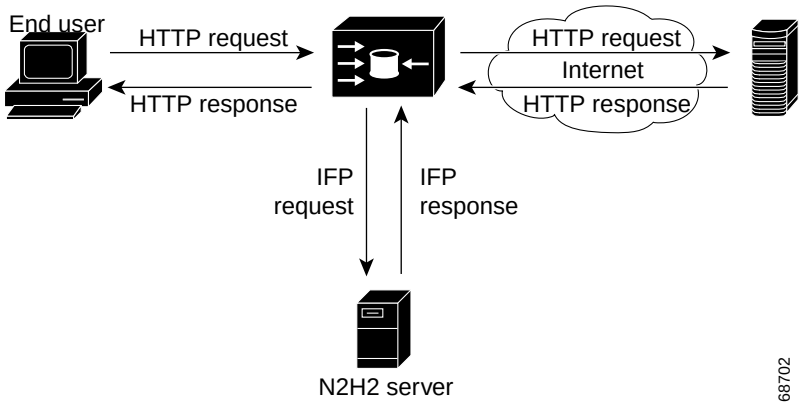
If a new block.html file is used it will automatically display its new message without having to renable the **url-filter custom-message** command.

URL Filtering with the N2H2 Server

N2H2 is a globally deployed URL-filtering software that can filter HTTP/HTTPS requests based on destination host name, destination IP address, and username and password. It relies on a sophisticated URL database exceeding 15 million sites and is organized into over 40 categories using both Internet technology and human review.

The Content Engine can perform URL filtering based on the N2H2 server. (See Figure 8-1.) The Content Engine and the N2H2 server use the Internet Filtering Protocol (IFP) Version 1 to communicate with each other. When the Content Engine receives a URL request, it sends an IFP request to the N2H2 server with the requested URL. The N2H2 server does some necessary lookups for the URL and sends back an IFP response. Based on the N2H2 server's IFP response, the Content Engine either blocks the HTTP request by redirecting the browser to a block page or proceeds with normal HTTP processing.

Figure 8-1 N2H2 Filtering



The filtering is applied to HTTP/HTTPS traffic before the Rules Template mechanism is applied, regardless of whether the requested object is in the cache or not. Filtering is applied to these traffic types:

- Proxy style or transparent style HTTP/HTTPS requests
- Proxy style and transparent redirect proxy style FTP over HTTP requests

N2H2 Features Supported

N2H2 supports three filtering methods. Table 8-1 lists the N2H2 features supported by the Content Engine. One N2H2 server can support multiple Content Engines simultaneously.

Table 8-1 N2H2 Features Supported

N2H2 Feature Name	Description
Global filtering	Applies filtering to all HTTP/HTTPS requests.
User-based filtering	Applies filtering to specific users or groups.
Client IP-based filtering	Applies filtering to specific client IP addresses.

N2H2 CLI Commands

The **url-filter N2H2 server** *IP address* [port *1-65535*] [timeout *1-120*] command configures the Content Engine to query the N2H2 server. The optional port field specifies the port on the N2H2 server to which the Content Engine should send IFP requests. The default port number is 4005. The optional timeout (in seconds) specifies how long the Content Engine should wait for an IFP response from the N2H2 server. The default timeout is 20 seconds.

This command does not verify whether or not an N2H2 server is accessible at the specified IP address in the current implementation. The configuration can be changed while N2H2 is enabled. The Content Engine will adopt the new configuration at run time.

The **url-filter enable N2H2** command enables the N2H2 server as the current URL filtering scheme. The command will not succeed if the server IP address is not configured, or if another URL filter is already enabled with N2H2 or other filtering schemes.

The **url-filter N2H2 allowmode enable** command allows HTTP/HTTPS requests to pass when the N2H2 server is enabled but the Content Engine has problems communicating with the N2H2 server. With **allowmode** enabled, if the Content Engine fails to receive responses from the N2H2 server successfully, the Content Engine still allows all traffic through (it proceeds with normal HTTP/HTTPS processing). With **allowmode** disabled, on the other hand, the Content Engine blocks all HTTP/HTTPS traffic through the Content Engine. By default, **allowmode** is disabled.

The **allowmode** option can be configured with or without N2H2 enabled and is independent of the N2H2 server configuration. The Content Engine will adopt the new configuration for **allowmode** if N2H2 is already running.

N2H2 Status and Statistics Commands

The **show url-filter** command shows the URL-filtering scheme enabled on the Content Engine and the configurations for each URL filtering scheme, such as the configuration data for N2H2.

In this example, the **show url-filter** command is used to display the URL-filtering schemes presently configured on the Content Engine:

```
ContentEngine# show url-filter
URL filter is DISABLED

Local list configurations
=================================
Good-list file name :
Bad-list file name :
Custom message directory :

Websense server configuration
=================================
Websense server IP     : <none>
Websense server port   : 15868
Websense server timeout: 20 (in seconds)
Websense allow mode is ENABLED

N2H2 server configuration
=============================
N2H2 server IP       : <none>
N2H2 server port     : 4005
N2H2 server timeout  : 20 (in seconds)
N2H2 allow mode is ENABLED
```

The **show statistics url-filter N2H2** command shows the request-reply statistics of the communication between the Content Engine and the N2H2 server. These statistics show the number of requests sent, replies received, pages blocked, pages allowed, and failure cases. More detailed URL-filtering statistics are available on the N2H2 server.

```
whh-590# show statistics url-filter N2H2
N2H2 URL Filtering Statistics:
             Lookup requests transmitted = 1021084
                Lookup response received = 1021080

                Requests BLOCKed by N2H2 = 0
                   Requests OKed by N2H2 = 1021080

Allow Mode Statistics:
                  No available connection = 1105
           Error sending lookup requests = 4
          Error recving lookup responses = 1
               Server error in responses = 0

Server Error in Responses:
                     Error in Filter Server = 0
                     Error in IFP server = 0
```

The statistics shown can be cleared using the **clear statistics url-filter N2H2**, **clear statistics urlfilter**, and **clear statistics all** commands.

The **clear stat url-filter N2H2** command resets the statistics counters for the N2H2 server. All the statistics counters are reset to 0.

N2H2 Configuration Through the Content Engine GUI

A user can choose N2H2 as the URL-filtering scheme through the Content Engine GUI and configure N2H2 server parameters such as the IP address, port number, timeout, and **allowmode** option.

N2H2 Configuration and Restrictions

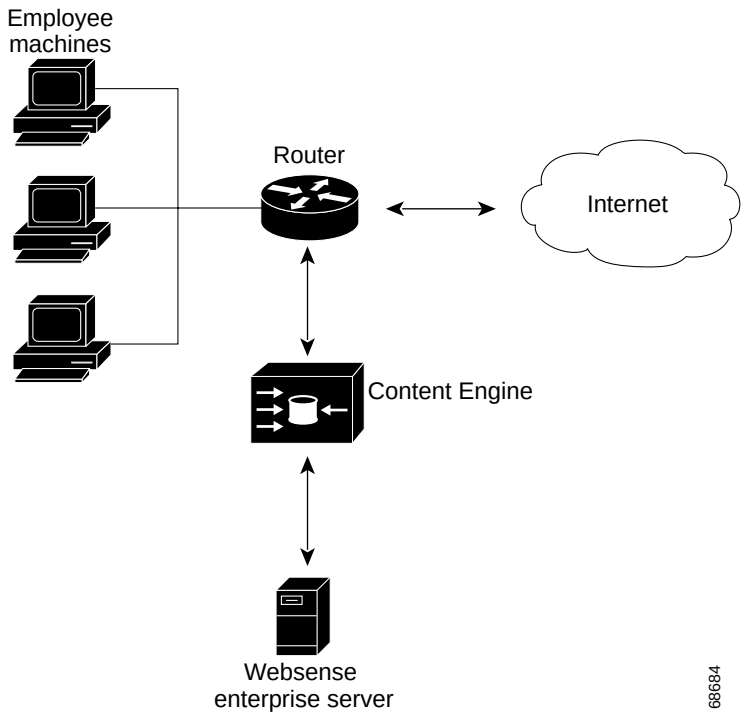
Only one URL-filtering scheme can be active at a time. In order to enable N2H2 URL filtering, you should first make sure that no other URL-filtering scheme is configured. You can then configure the server information for N2H2 using the **url-filter N2H2 server** *IP address* [**port** *1-65535*] [**timeout** *1-120*] command and enabling the N2H2 server.

The server IP address and port number configured in the Content Engine must match the IP address of the N2H2 server and the port that N2H2 server listens to for IFP requests. If the configuration on the Content Engine does not match the configurations on the N2H2 server, the Content Engine will time out all HTTP/HTTPS requests and either block or allow all HTTP traffic based on the **allowmode** option configuration.

URL Filtering with the Websense Enterprise Server

The Content Engine can use a Websense enterprise server as a filtering engine and enforce the filtering policy configured on the Websense server. (See Figure 8-2.) Refer to the Websense documentation for further information on Websense filtering policies.

Figure 8-2 URL Filtering with a Websense Server



To enable Websense URL filtering on the Content Engine, specify the Websense server IP address or host name using the **url-filter enable websense** command. The **timeout** option sets the maximum amount of time that the Content Engine will wait for a Websense response. The timeout default is 20 seconds. The **port** option specifies the port number on which the server will accept requests from the Content Engine (the default port is 15868). Use the **no url-filter websense server** command to disable Websense URL filtering.

The **url-filter websense allowmode enable** command permits the Content Engine to fulfill the client request after a Websense server timeout.

The Websense server returns its own blocking message.

To use Websense URL filtering with a cluster of Content Engines, make sure to configure the **url-filter websense server** command on each Content Engine in the cluster to ensure that all traffic is filtered.

URL Filtering with SmartFilter Software

SmartFilter™ software for the Cisco Content Engine provides employee Internet management (EIM) functionality with proxy servers, firewalls, and caching appliances. The integrated Content Engine and SmartFilter product preserves all functionality available in a regular Content Engine. The SmartFilter filtering capability is available as an add-on service on the Content Engine, and the service may be enabled or disabled as desired through the Content Engine CLI or GUI.

The integrated Content Engine and SmartFilter product provides a one-box solution for server functionality. The Content Engine uses a suite of plug-in APIs to allow the SmartFilter software to implement hooks at strategic points during an HTTP transaction and thus provide URL filtering.

The integrated Content Engine and SmartFilter product provides an end user management tool called the sfadmin console. This GUI component downloads configurations into the Content Engine for use by the SmartFilter process.

To use SmartFilter URL filtering with a cluster of Content Engines, make sure to enter the **url-filter smartfilter enable** command on each Content Engine in the cluster to ensure that all traffic is filtered. Refer to the *SmartFilter for Cisco Content Engine User's Guide* for more information on how to configure the SmartFilter software in the ACNS 4.1 release.

CHAPTER 9

Configuring Transaction Logging

Overview

Transaction logs allow administrators to view the traffic that has passed through the Content Engine. Typical fields in the transaction log are the date and time when a request was made, the URL that was requested, whether it was a cache-hit or a cache-miss, the type of request, the number of bytes transferred, and the source IP.

High-performance caching presents additional challenges other than how to quickly retrieve objects from storage, memory, or the web. Administrators of caches are often interested in what requests have been made of the cache and what the results of these requests were. This information is then used for such applications as:

- Problem identification and solving
- Load monitoring
- Billing
- Statistical analysis
- Security problems
- Cost analysis and provisioning

In ACNS 4.1 software, the user can choose between Squid, Extended Squid, and Apache log formats.

Squid-Style Transaction Logging

The Squid-style log format is the default format for transaction logging in the Content Engine. The Squid log file format used is the native log file format associated with the Squid-1.1 access.log file format. For details on the Squid-1.1 native log file format, refer to the Squid documentation “Frequently Asked Questions,” section 6.6, access.log heading at the following URL:
http://www.squid-cache.org/doc/FAQ/FAQ.html

The Squid log file format is:

time elapsed remotehost code/status bytes method URL rfc931 peerstatus/peerhost type

A Squid log format example looks like this:

1012429341.115 100 172.16.100.152 TCP_REFRESH_MISS/304 1100 GET
http://www.cisco.com/images/homepage/news.gif - DIRECT/www.cisco.com -

Squid logs are a valuable source of information about cache workloads and performance. The logs record not only access information but also system configuration errors and resource consumption, such as memory and disk space.

This example enables transaction logging in Squid-style format.

```
ContentEngine(config)# transaction-logs format squid
ContentEngine(config)#
```

Table 9-1 lists the fields associated with the Squid-style format.

Note Many public tools are available that can convert a Squid-style transaction log into reports. Visit the following website, http://www.squid-cache.org/Scripts for listings of such tools.

Table 9-1 Squid-Style Format Description

Field	Description
Time	UNIX time stamp as Coordinated Universal Time (UTC) seconds with a millisecond resolution.
Elapsed	Length of time in milliseconds that the cache was busy with the transaction. **Note** Entries are logged after the reply has been sent, not during the lifetime of the transaction.
Remote Host	IP address of the requesting instance.
Code/Status	Two entries separated by a slash. The first entry contains information on the result of the transaction: the kind of request, how it was satisfied, or in what way it failed. The second entry contains the HTTP result codes.
Bytes	Amount of data delivered to the client. This does not constitute the net object size, because headers are also counted. Also, failed requests may deliver an error page, the size of which is also logged here.
Method	Request method to obtain an object for example, GET.
URL	URL requested.
Rfc931	Contains the authentication server's identification or lookup names of the requesting client. This field will always be a "-" (dash).
Peerstatus/Peerhost	Two entries separated by a slash. The first entry represents a code that explains how the request was handled, for example, by forwarding it to a peer, or returning the request to the source. The second entry contains the name of the host from which the object was requested. This host may be the origin site, a parent, or any other peer. Also note that the host name may be numerical.
Type	Content type of the object as seen in the HTTP reply header. In the ACNS 4.1 software, this field will always contain a "-" (dash).

Extended Squid-Style Transaction Logging

The Extended Squid format logs the associated username for each record in the log file in addition to the fields logged by the Squid-style format, and is used for billing purposes. In this format the Rfc931 field associated with the Squid format (Table 9-1) is used to log the authorized user. This field always contains a "-" (dash) if no user information is available.

An Extended Squid-style log format example looks like this:

1012429341.115 100 172.16.100.152 TCP_MISS/302 184 GET http://www.cisco.com/cgi-bin/login myloginname DIRECT/www.cisco.com -

Use the **transaction-logs format extended-squid** command to enable transaction logging in Extended Squid format.

This example shows how to enable transaction logging in Extended Squid format.

```
ContentEngine(config)# transaction-logs format extended-squid
ContentEngine(config)#
```

Apache-Style Transaction Logging

This format is the Common Log File (CLF) format defined by the World Wide Web Consortium (W3C) working group. This format is compatible with many industry-standard log tools. For more information, see the W3C Common Log Format website at http://www.w3.org/Daemon/User/Config/Logging.html.

The Apache-style log file format is:

remotehost rfc931 authuser date request status bytes

An Apache-style log file format example looks like this:

172.16.100.152 - - [Wed Jan 30 15:26:26 2002]
"GET/http://www.cisco.com/images/homepage/support.gif HTTP/1.0" 200 632

Table 9-2 lists the fields associated with the Apache CLF format.

Table 9-2 Apache Common Log File Format Descriptions

Field	Description
Remotehost	Remote host name or IP address.
Rfc931	Contains the authentication server's identification or lookup names of the requesting client. This field will always contain a "-" (dash).
Authuser	Username that the user logged in for authentication purposes. This will be a "-" (dash) if no user information is available.
Date	Date and time of request.
Request	The first line of the request.
Status	The HTTP status code. Example: 200.
Bytes	The content-length of the document transferred.

This example shows how to enable transaction logging in Apache style format.

```
ContentEngine(config)# transaction-logs format apache
ContentEngine(config)#
```

Sanitized Transaction Logs

Use the **transaction-logs sanitized** command to disguise the IP address and usernames of clients in the transaction log file. The default is not sanitized. A sanitized transaction log disguises the network identity of a client by changing the IP address in the transaction logs to 0.0.0.0. The **no** form disables the sanitize feature.

This examples shows how to enable the sanitize feature in transaction logging.

```
ContentEngine(config)# transaction-logs sanitize
ContentEngine(config)#
```

Exporting Log Files

In order to facilitate the postprocessing of cache logfiles, it is possible to export transaction logs to an external host.

This feature allows for log files to be automatically exported by FTP to an external host at configurable intervals. The username and password used for FTP are configurable, as is the directory to which the log files are uploaded.

The log files automatically have a filename of:

<type>_<ipaddr>_yyyymmdd_hhmmss.txt

where

- *<type>* represents the type of log file with *celog* for cache logs such as HTTP, HTTPS, and FTP, and *mms_export* for Windows Media Technology (WMT) logs
- *<ipaddr>* represents the Content Engine IP address
- *yyyymmdd_hhmmss* represents the date and time when the log was archived for export

Note For MMS type logs there is no .txt extension in the filename.

Exporting Transaction Logs to External FTP Servers

To export transaction logs to an FTP server, you must first enable the feature and then configure the FTP server parameters. Use the **transaction-logs export enable** command to enable exporting of transaction logs to external FTP servers. You can then use the **transaction-logs export ftp-server** option to enter FTP server parameters. This option can support up to four FTP servers. The following information is required for each target FTP server:

- Server IP address or the host name

  The Content Engine translates the host name with a DNS lookup and then stores the IP address in the configuration.

- FTP user login and user password

- Path of the directory where transferred files are written

  Use a fully qualified path or a relative path for the user login. The user must have write permission to the directory.

Use the **transaction-logs archive compress** command to compress archived log files into gzip format prior to exporting them to external FTP servers. The compressed filename has a .gz extension in the filename. This feature uses less disk space for the archived files on both the Content Engine and the FTP export server and also requires less bandwidth during export.

In this example, two FTP servers are configured.

```
ContentEngine(config)# transaction-logs export ftp-server 10.1.1.1 mylogin mypasswd
/ftpdirectory
ContentEngine(config)# transaction-logs export ftp-server myhostname mylogin mypasswd
/ftpdirectory
```

To delete an FTP server, use the **no** form of the command.

```
ContentEngine(config)# no transaction-logs export ftp-server 10.1.1.1
ContentEngine(config)# no transaction-logs export ftp-server myhostname
```

Use the **no** form of the **transaction-logs export enable** command to disable the entire transaction log export feature while retaining the rest of the configuration.

```
ContentEngine(config)# no transaction-logs export enable
```

To change a username, password, or directory, reenter the entire line.

```
ContentEngine(config)# transaction-logs export ftp-server 10.1.1.1 mynewname mynewpass
/newftpdirectory
```

The **show transaction-logging** command displays information on the configuration of transaction logging in the Content Engine. Note that transaction log file information is displayed for HTTP and WMT Microsoft Media Server (MMS) caching proxy transactions.

```
ContentEngine# show transaction-logging
Transaction log configuration:
--------------------------------------
Logging is enabled.
Logging of ecdn internal communication is disabled.
End user identity is hidden. (sanitized)
File markers are disabled.
Archive interval: every-day every hour.
Maximum size of archive file: 2000000 KB
Log File format is extended-squid.

Exporting files to ftp servers is enabled.
File compression is disabled.
Export interval: every-day at 11:45  local time

ftp-server           username        directory
10.1.1.1           mylogin     /ftpdirectory
10.2.2.2           mylogin     /ftpdirectory
HTTP Caching Proxy Transaction Log File Info
  Working Log file - size : 83
                     age: 502845
  Archive Log file - celog_10.1.1.21_20020107_150300.txt        size: 1075
  Archive Log file - celog_10.1.1.21_20020117_150300.txt        size: 1199746
  Archive Log file - celog_10.1.1.21_20020118_000000.txt        size: 137583
  Archive Log file - celog_10.1.1.21_20020118_150300.txt        size: 12667
  Archive Log file - celog_10.1.1.21_20020123_150300.txt        size: 298
```

```
WMT MMS Caching Proxy/Server Transaction Log File Info
  Working Log file - size : 541
                     age: 54117
  Archive Log file - mms_export_10.1.1.21_20020107_225942       size: 541
  Archive Log file - mms_export_10.1.1.21_20020107_232156       size: 938
  Archive Log file - mms_export_10.1.1.21_20020117_193239       size: 541
  Archive Log file - mms_export_10.1.1.21_20020122_224556       size: 1993
  Archive Log file - mms_export_10.1.1.21_20020124_150334       size: 541
  Archive Log file - mms_export_10.1.1.21_20020131_025505       size: 541
ContentEngine#
```

Note For security reasons, passwords are never displayed.

Restarting Export After Receiving a Permanent Error from the External FTP Server

When an FTP server returns a permanent error to the Content Engine, the archive transaction logs are no longer exported to that server. You must reenter the Content Engine transaction log export parameters for the misconfigured server to clear the error condition. The **show statistics transaction-logs** command displays the current state of transaction log export readiness.

A permanent error (Permanent Negative Completion Reply, RFC 959) occurs when the FTP command to the server cannot be accepted, and the action does not take place. Permanent errors can be caused by invalid user logins, invalid user passwords, and attempts to access directories with insufficient permissions or directories that do not exist.

In the following example, an invalid user login parameter was included in the **transaction-logs export ftp-server** command. The **show statistics transaction-logs** command shows that the Content Engine failed to export archive files.

```
ContentEngine# show statistics transaction-logs
Transaction Log Export Statistics:

Server:172.16.10.5
      Initial Attempts:1
      Initial Successes:0
      Initial Open Failures:0
      Initial Put Failures:0
      Retry Attempts:0
      Retry Successes:0
      Retry Open Failures:0
      Retry Put Failures:0
      Authentication Failures:1
      Invalid Server Directory Failures:0
```

To restart the export of archive transaction logs, you must reenter the **transaction-logs export ftp-server** parameters.

```
ContentEngine(config)# transaction-logs export ftp-server 172.16.10.5 goodlogin pass
/ftpdirectory
```

CHAPTER 10

Configuring Authentication

User Authentication

Authentication, also referred to as "login," is the act of verifying usernames and passwords. Authorization refers to the setting of privileges to authenticated users in to a network.

The **authentication** command configures the authentication and authorization methods that govern login and configuration access to the Content Engine. ACNS 4.1 software supports local database and Terminal Access Controller Access Control System Plus (TACACS+) database authentication and authorization methods.

Login and configuration privileges can be obtained from both the local database or the TACACS+ remote database. If both databases are enabled, then both databases are queried; if the user data cannot be found in the first database queried, then the second database is tried.

The **authentication login** command specifies the method that determines whether the user has any level of permission to access the Content Engine. The **authentication configuration** command specifies the method that authorizes the user with privileged access (configuration access) to the Content Engine.

The **authentication login local** and the **authentication configuration local** commands use a local database for authentication and authorization.

The **authentication login tacacs** and **authentication configuration tacacs** commands use a remote TACACS+ server to determine the level of user access. The Content Engine **tacacs** global configuration command and a TACACS+ server must be configured to use the TACACS+ authentication and authorization method.

When the **primary** keyword is entered for TACACS+ login or configuration authentication, the TACACS+ database is queried first, and the local database is queried second. If the TACACS+ database is not designated as primary, and both the local and the TACACS+ database are enabled, the local database is queried first. If both the local and the TACACS+ databases are disabled (**no authentication**), the Content Engine verifies that both are disabled and if so, sets the Content Engine to the default state.

By default, the local method is enabled and TACACS+ is disabled for both login and configuration. Whenever TACACS+ is disabled, local is automatically enabled. Both TACACS+ and local methods can be enabled at the same time. The **primary** option specifies the first method to attempt; the **secondary** option specifies the method to use if the primary method fails. If both methods of an **authentication login** or **authentication configuration** command are configured as primary, or both as secondary, local is attempted first, then TACACS+.

The following example enables local and TACACS+ authentication and authorization, setting TACACS+ as the first method used and local as the secondary method to use if TACACS+ fails.

```
Console(config)# authentication login tacacs enable primary
Console(config)# authentication login local enable secondary
```

```
Console(config)# authentication configuration local enable secondary
Console(config)# authentication configuration tacacs enable primary
```

This is an example of the **show authentication user** command:

```
Console# show authentication user
Login Authentication:         Console/Telnet Session
----------------------------- ----------------------
local                         enabled
tacacs                        enabled (primary)

Configuration Authentication: Console/Telnet Session
----------------------------- ----------------------
local                         enabled
tacacs                        enabled
```

TACACS+ Options for User Authentication

The TACACS database validates users before they gain access to a Content Engine. TACACS+ is derived from the United States Department of Defense (RFC 1492) and is used by Cisco Systems as an additional control of nonprivileged and privileged mode access. This release supports TACACS+ only and not TACACS or Extended TACACS.

TACACS+ provides both authentication and authorization options. Authentication or login is the action of identifying and validating a user. It verifies a username with the password. Authorization or configuration is the action of determining what a user is allowed to do. To configure TACACS+, use the **authentication** and **tacacs** commands.

The Users GUI page or the **user** global configuration commands provide a way to add, delete, or modify usernames, passwords, and access privileges in the local database. The TACACS+ remote database can also be used to maintain login and configuration privileges for administrative users. The **tacacs** command or the TACACS+ GUI page allows you to configure the network parameters required to access the remote database.

Use the **tacacs key** command to specify the TACACS+ key, used to encrypt the packets transmitted to the server. This key must be the same as the one specified on the server daemon. The maximum number of characters in the key should not exceed 99 printable ASCII characters (except tabs). An empty key string is the default. All leading spaces are ignored; spaces within and at the end of the key string are not ignored. Double quotes are not required even if there are spaces in the key, unless the quotes themselves are part of the key.

One primary and two backup TACACS+ servers can be configured; authentication is attempted on the primary server first, then on the others in the order in which they were configured. The primary server is the first server configured unless another is explicitly specified as primary with the **tacacs server** *hostname* **primary** command.

The **tacacs timeout** is the number of seconds the Content Engine waits before declaring a timeout on a request to a particular TACACS+ server. The range is from 1 to 20 seconds with 5 seconds as the default. The number of times the Content Engine repeats a retry-timeout cycle before trying the next TACACS+ server is specified by the **tacacs retransmit** command. The default is two retry attempts.

Three unsuccessful login attempts are permitted. TACACS+ logins may appear to take more time than local logins depending on the number of TACACS+ servers and the configured timeout and retry values.

This example configures the key used in encrypting packets:

```
Console(config)# tacacs key human789
```

This example configures the host named spearhead as the primary TACACS+ server:

```
Console(config)# tacacs server spearhead primary
```

This example sets the timeout interval for the TACACS+ server:

```
Console(config)# tacacs timeout 10
```

This example sets the number of times authentication requests are retried (retransmitted) after a timeout:

```
Console(config)# tacacs retransmit 5
```

HTTP Request Authentication

The ACNS 4.1 software Cache application supports Microsoft NT LAN Manager (NTLM), Lightweight Directory Access Protocol (LDAP), and RADIUS server HTTP request authentication. HTTP request authentication authenticates a user's domain, username, and password with a preconfigured primary domain controller (PDC) before allowing requests from the user to be served by the Content Engine.

NTLM Authentication

The NTLM protocol can be used to authenticate and block user access to the Internet. When a user logs in to a Windows NT or a Windows 2000 domain, the information is stored by the browser and later used as NTLM credentials to access the Internet. The browser sends the NTLM credentials with the domain name to the ACNS cache, which in turns sends a request to the Windows NT domain controller to check the validity of the user in the domain. If the user is not a valid user in the domain, then the request to access the Internet is denied. If authentication succeeds, the source IP address is entered in the authentication cache. Future requests from this IP address are not challenged until the authentication cache entry expires, or is cleared.

Use the **ntlm server** command to enable NTLM authentication and configure the NTLM server domain name, NT primary domain controller (PDC) name or IP address, and optionally set the host name or address as primary or secondary.

Before invoking an NTLM authentication request, make sure that the following conditions exist.

- The NTLM primary domain controller has an entry in the Domain Name System (DNS) that matches its NetBIOS-named computer account.
- The primary domain controller is both forward and reverse DNS-resolvable.
- The domain name configured on the Content Engine matches the domain of which the primary domain controller is a part.

In the following example, server1 must be in the cisco.com domain and must have an entry in DNS that matches its NetBIOS-named computer account.

```
ip domain-name cisco.com
ntlm server host server1
```

NTLM Authentication Transparency

For clients within the domain using the Internet Explorer browser in proxy mode, authentication is "popless"; this is, the user is not prompted with a dialog box to enter a username and password. In transparent mode, authentication is transparent only if the Internet options security settings are customized and set to **User Authentication > Logon > Automatic logon with current username and password**.

This example configures a Content Engine for NTLM request authentication and blocking.

```
Console(config)# ntlm server enable
Console(config)# ntlm server domain cisco_abc
Console(config)# ntlm server host 172.16.10.10 primary
Console(config)# ntlm server host 172.16.10.12 secondary

Console# show ntlm
Primary:       172.16.10.10
Secondary:     172.16.10.12
State:          Enabled
Domain name: cisco_abc
```

RADIUS HTTP Request Authentication

RADIUS authentication clients reside on the Content Engine running ACNS 4.1 software. When enabled, these clients send authentication requests to a central RADIUS server, which contains user authentication and network service access information.

To configure RADIUS parameters, use the **radius-server** command in global configuration mode. To disable RADIUS authentication parameters, use the **no** form of this command. For more information on the **radius-server** command, refer to the *Cisco Application and Content Networking Software Command Reference*.

RADIUS Authentication Redirection

The **redirect** option of the **radius-server** command redirects an authentication response to a different authentication server if an authentication request using the RADIUS server fails.

Note The **rule** command is relevant to RADIUS only if **redirect** has been configured.

The following example enables the RADIUS client, specifies a RADIUS server, specifies the RADIUS key, accepts retransmit defaults, and excludes the domain name, mydomain.net, from RADIUS authentication. The configuration is verified with the **show radius-server** and **show rule all** commands.

```
Console(config)# radius-server enable
Console(config)# radius-server host 172.16.90.121
Console(config)# radius-server key myradiuskey
Console(config)# rule enable
Console(config)# rule no-auth domain mydomain.net

Console(config)# show radius-server
Radius Configuration:
---------------------
Radius Authentication is on
    Timeout       = 5
    Retransmit    = 3
```

```
    Key           = ****
    Servers
    -------
    IP 172.16.90.121 Port =   1645    State: ENABLED

Console# show rule all
Rules Template Configuration
----------------------------
Rule Processing Enabled
rule no-auth domain mydomain.net
```

The following example disables RADIUS authentication on the Content Engine.

```
Console(config)# no radius-server enable
```

LDAP HTTP Request Authentication

System administrators can now use the Content Engine to restrict user Internet access using a Lightweight Directory Access Protocol (LDAP) server for autentication purposes, which provides most of the services of the X.500 protocol with less complexity and overhead.

Use the **ldap** global configuration command to enable LDAP authentication. Use the **no** form of the command to disable LDAP functions. An LDAP-enabled Content Engine authenticates users with an LDAP server. With an HTTP query, the Content Engine obtains a set of credentials from the user (user ID and password) and compares them against those in an LDAP server.

ACNS 4.1 software supports LDAP version 2 and version 3 and supports all LDAP features except for Secure Authentication and Security Layer (SASL).

Note The HTTP authentication featuring RADIUS and LDAP existed in Cache software 2.x releases and were configured through the **radius-server** and **ldap** commands, respectively. For ACNS 4.1 software, the **radius-server authtimeout** option and the **ldap authcache max-entries** and **ldap authcache auth-timeout** options have been removed and are now configurable through the **http authentication cache max-entries** and **timeout** commands, respectively. The **ldap client auth-header** option has been removed and is now configurable through the **http authentication header** command. The **multi-user-prompt** has been removed and replaced by the **http avoid-multiple-user-prompts** option. In addition, the **radius-server** command options **exclude** has been removed. The **rule no-auth domain** command replaces **radius-server exclude**; however, there is no replacement available for the **multi-user-prompt** option. The **ldap server** command has the following added options: **enable** and **version**.

HTTP Request Considerations

When the Content Engine authenticates a user through an NTLM, RADIUS, or LDAP server, a record of that authentication is stored locally in the Content Engine RAM (authentication cache). As long as the authentication entry is kept, subsequent attempts to access restricted Internet content by that user do not require server lookups.

The **http authentication cache timeout** command specifies how long an inactive entry can remain in the authentication cache before it is purged. Once a record has been purged, any subsequent access attempt to restricted Internet content requires reauthentication.

LDAP authentication can be used with Websense URL filtering, but not with RADIUS authentication. Both LDAP and RADIUS rely on different servers, which may require different user IDs and passwords, making LDAP and RADIUS authentication schemes mutually exclusive. Should both RADIUS and LDAP be configured on the Content Engine at the same time, LDAP authentication is executed, not RADIUS authentication.

Excluding Domains from HTTP Authentication Servers

To exclude domains from HTTP authentication servers, use the **rule no-auth domain** command. LDAP, NTLM, or RADIUS authentication takes place only if the site requested does not match the specified pattern.

Proxy Mode Server Authentication

The events listed below occur when the Content Engine is configured for HTTP request authentication and one of the following two scenarios is true:

- The Content Engine receives a proxy-style request from a client.
- The Content Engine receives a transparent (WCCP-style) request from a client and the Content Engine **http authentication header** command parameter is set to 407 (because there is an upstream proxy).

1. The Content Engine examines the HTTP headers of the client request to find user information (contained in the Proxy-Authorization header).
2. If no user information is provided, the Content Engine returns a 407 (Proxy Authorization Required) message to the client.
3. The client resends the request, including the user information.
4. The Content Engine searches its authentication cache (based on user ID and password) to see whether the client has been previously authenticated.
5. If a match is found, the request is serviced normally.
6. If no match is found, the Content Engine sends a request to the authentication server to find an entry for this client.
7. If the server finds a match, the Content Engine allows the request to be serviced normally and stores the client user ID and password in the authentication cache.
8. If no match is found, the Content Engine again returns a 407 (Proxy Authorization Required) message to the client.

Transparent Mode Authentication

The events listed below occur when the Content Engine is configured for authentication and both of the following are true:

- The Content Engine receives a redirected request from a client.
- The **http authentication header** command parameter is set to 401 (because there is no upstream proxy).

1. The Content Engine searches its authentication cache to see whether the user's IP address has been previously authenticated.
2. If a match is found, the Content Engine allows the request to be serviced normally.

3. If no match is found in the first step, the Content Engine examines the HTTP headers to find user information (contained in the Authorization header).
4. If no user information is provided, the Content Engine returns a 401 (Unauthorized) message to the client.
5. The client resends the request, including the user information.
6. The Content Engine sends a request to the authentication server to find an entry for this user.
7. If the server finds a match, the Content Engine allows the request to be serviced normally and stores the client IP address in the authentication cache.
8. If no match is found, the Content Engine again returns a 401 (Unauthorized) message to the client.

In transparent mode, the Content Engine uses the client IP address as a key for the authentication database.

If you are using user authentication in transparent mode, we recommend that the AuthTimeout interval configured with the **http authentication cache timeout** command be short. IP addresses can be reallocated, or different users can access the Internet through an already authenticated device (PC, workstation, and the like). Shorter AuthTimeout values help reduce the possibility that individuals can gain access using previously authenticated devices. When the Content Engine operates in proxy mode, it can authenticate the user with the user ID and password.

Server Redundancy

Two authentication servers can be specified with the **server host command option** to provide redundancy and improved throughput. Content Engine load-balancing schemes distribute the requests to the servers. If the Content Engine cannot connect to either server, no authentication can take place, and users who have not been previously authenticated are denied access.

Security Options

The Content Engine uses simple (nonencrypted) authentication to communicate with the LDAP server. Future expansion may allow for more security options based on Secure Socket Layer (SSL), SASL, or certificate-based authentication.

Hierarchical Caching

In some cases, users are located at branch offices. A Content Engine (CE1) can reside with them in the branch office. Another Content Engine (CE2) or another HTTP-compatible proxy device can reside upstream, with an NTLM, RADIUS, or LDAP server available to both Content Engines or devices for user authentication.

Note The **http append proxy-auth-header** global configuration command must be configured on the downstream Content Engines to ensure that proxy authorization information, required by upstream Content Engines, is not stripped from the HTTP request by the downstream Content Engines. Up to 8 upstream IP addresses can be configured on each downstream Content Engine.

If branch office user 1 accesses the Internet, and content is cached at CE1, then this content cannot be served to any other branch office user unless that user is authenticated. CE1 must authenticate the local users.

Assuming that both CE1 and CE2 are connected to the server and authenticate the users, when branch office user 2 firsts requests Internet content, CE1 responds to the request with an authentication failure response (either HTTP 407 if in proxy mode, or HTTP 401 if in transparent mode). User 2 enters the user ID and password, and the original request is repeated with the credentials included. CE1 contacts the HTTP request authentication server to authenticate user 2.

Assuming authentication success, and a cache miss, the request along with the credentials is forwarded to CE2. CE2 also contacts the authentication server to authenticate user 2. Assuming success, CE2 either serves the request out of its cache or forwards the request to the origin server.

User 2 authentication information is now stored in the authentication cache in both CE1 and CE2. Neither CE1 nor CE2 needs to contact the authentication server for user 2's subsequent requests (unless user 2's entry expires and is removed from the authentication cache).

This scenario assumes that CE1 and CE2 use the same method for authenticating users. Specifically, both Content Engines must expect the user credentials (user ID and password) to be encoded in the same way.

Note If you wish to avoid authentication on an upstream Content Engine after authentication is performed downstream, you can use the **rule no-auth** command to exclude the downstream Content Engine IP address.

Hierarchical Caching in Transparent Mode

When the Content Engine operates in transparent mode, the user IP address is used as a key to the authentication cache. When user 2 sends a request transparently to CE1, after authentication, CE1 inserts its own IP address as the source for the request. Therefore, CE2 cannot use the source IP address as a key for the authentication cache.

When CE1 inserts its own IP address as the source, it must also insert an X-Forwarded-For header in the request (**http append x-forwarded-for-header** command). CE2 must first look for an X-Forwarded-For header. If one exists, that IP address must be used to search the authentication cache. Assuming the user is authenticated at CE2, then CE2 must not change the X-Forwarded-For header, just in case there is a transparent CE3 upstream.

In this scenario, if CE1 does not create an X-Forwarded-For header (for example, if it is not a Cisco Content Engine and does not support this header), then authentication on CE2 will not work.

Hierarchical Caching, Content Engine in Transparent Mode with an Upstream Proxy

In a topology with two Content Engines, assume that CE1 is operating in transparent mode and CE2 is operating in proxy mode, with the browsers of all users pointing to CE2 as a proxy.

Because the browsers are set up to send requests to a proxy, an HTTP 407 message is sent from CE1 back to each user to prompt for credentials. By using the 407 message, the problem of authenticating based on source IP address is avoided. The username and password can be used instead.

This mode provides better security than using the HTTP 401 message. The Content Engine examines the style of the address to determine whether there is an upstream proxy. If there is, the Content Engine uses an HTTP 407 message to prompt the user for credentials even when operating in transparent mode.

Authentication Cache Size Adjustments

If the authentication cache is not large enough to accommodate all authenticated users at the same time, the Content Engine purges older entries that have not yet timed out. The Content Engine has a timeout value range from 30 to 1440 seconds. Its default timeout value is 480 seconds.Use the **http autentication cache timeout** command to configure this parameter if necessary.

The maximum number of entries that are maintained in authentication cache is 32000. The minimum number is 500. The default value is 16000. Use the **http authentication max-entries** command to configure this parameter if necessary.

The **http authentication** command has a **header** option that can be set to display a message to the client when authorization has failed. In this scenario you can choose **http authentication header 401** or **http authentication header 407**. By default, the Content Engine authenticates cache loads based on the URL syntax of the incoming request.

Use the **show http authentication** command in EXEC mode to display the authentication cache parameters.

Transaction Logging

Once a user has been authenticated through LDAP, NTLM, or a RADIUS server, all transaction logs generated by the Content Engine for that user contain user information. If the Content Engine is acting in proxy mode, the user ID is included in the transaction logs. If the Content Engine is acting in transparent mode, the user IP address is included instead.

If the **transaction-logs sanitize** command is invoked, the user information is suppressed.

In this example, the host for the LDAP server daemon is configured:

```
Console(config)# ldap server host www.someDomain.com port 390
```

To delete an LDAP server, use the **no ldap server** command.

```
Console(config)# no ldap server host 1.1.1.1
```

In this example, the host for the RADIUS server is configured:

```
Console(config)# radius-server 172.16.90.121
```

In this example, the length of time that entries are valid in the authentication cache is set:

```
Console(config)# http authentication cache timeout 1000
```

The following example specifies that the Content Engine should use header 407 when asking the end user for authentication credentials (user ID and password).

```
Console(config)# http authentication header 407
```

End-to-End Authentication

The ACNS 4.1 software caching application supports both basic and NTLM end-to-end authentication. End-to-end NTLM authentication includes pass-through servicing and the caching of web objects that require NTLM authentication. HTTP request authentication authenticates a user's domain, username, and password with a preconfigured NTLM domain controller before allowing requests from the user to be served by the Content Engine. NTLM authentication works only in a Microsoft environment (for instance, Microsoft Internet Explorer clients accessing Microsoft Internet Information Servers).

Note End-to-end NTLM authentication is supported with WCCP Version 2 transparent caching only. For HTTP request authentication, if NTLM authentication is used but the browser does not support NTLM authentication, the username and password information is passed to the Content Engine in clear text with a Basic authentication header. The Content Engine then uses this information to authenticate the user against the preconfigured Windows NT domain controller.

Basic End-to-End Authentication

The ACNS software Cache application can strip NTLM authentication headers to allow fallback to a basic-style authentication challenge against Microsoft Internet Information System (IIS) servers.

This feature is designed to allow browsers to authenticate against a Microsoft IIS web server that issues an NTLM-based challenge. NTLM is proprietary and undocumented. Removing the NTLM headers allows the browser to fall back on the basic authentication method. If IIS is configured to still accept basic authentication, IIS authentication credentials can proceed through a Content Engine, but with reduced security. Use the **http authenticate-strip-ntlm** global configuration command to enable this feature.

NTLM End-to-End Authentication

The two levels of NTLM end-to-end support can be summarized as follows:

- NTLM pass-through service

  If NTLM pass-through level is set on the server, the Content Engine sets up a secure persistent connection between the client and the server through the Content Engine. NTLM authentication messages pass through this virtual persistent connection. The Content Engine does not cache any object transferred on the virtual connection. All the client requests are served by the origin server.

- NTLM object caching

  The ACNS 4.1 software Cache application can be configured to cache objects that require NTLM authentication. The server puts a “no-store” flag on a reply object to prevent the reply from being cached. If no such flag is present, the object is cacheable. When the Content Engine receives a request from a client already connected with the intended NTLM server, the ACNS software searches the cache. For a cache miss, the request is forwarded to the origin server. The reply object is then sent to the client and a copy is cached. On a cache hit, the Content Engine checks for a secured connection between this client and the server. If the object requires NTLM authentication and there is no virtual persistent connection set up between the client and the server, the Content Engine establishes the secured connection between client and server and forwards the request to the server. If there is a virtual persistent connection between the client and the server, an If-Modified-Since (IMS) message is sent to the server to verify the validity of the object and the user’s access rights to this object before the cached copy is served to the client.

This example configures a Content Engine for end-to-end NTLM authentication. By default, basic and NTLM authenticated objects are not cached.

```
Console(config)# no http authenticate-strip-ntlm
Console(config)# http cache-authenticated ntlm
Console# show http cache-authenticated ntlm
Basic authenticated objects are not cached.
NTLM authenticated objects are cached.
```

CHAPTER 11

Configuring Network Management

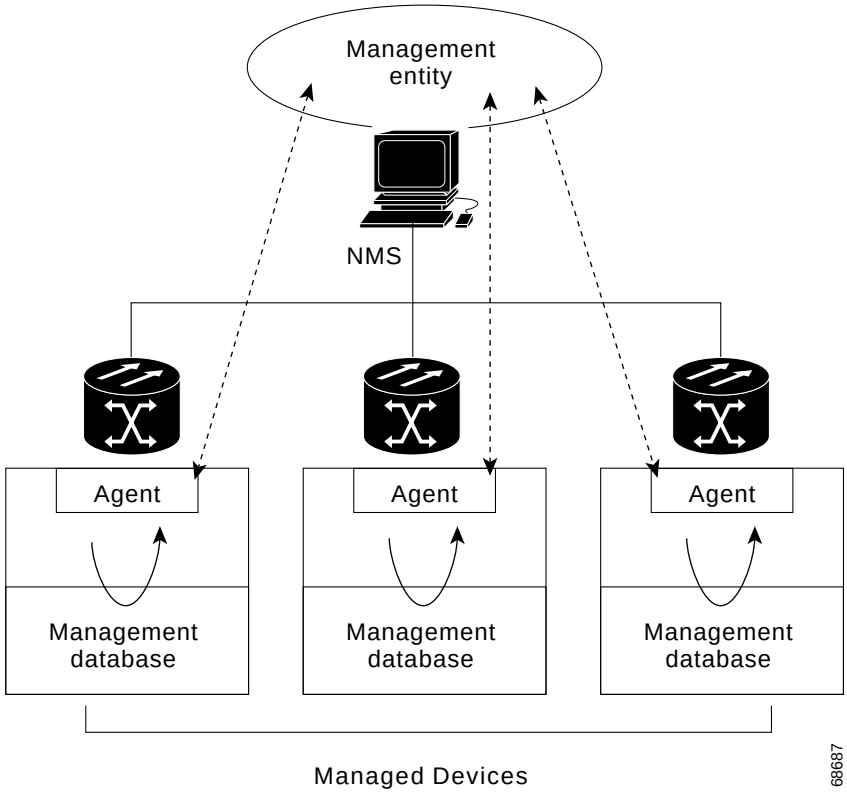
Simple Network Management Protocol Overview

Simple Network Management Protocol (SNMP) is an interoperable standards-based protocol that allows for external monitoring of the Content Engine through an SNMP agent.

An SNMP-managed network consists of three primary components: managed devices, agents, and management systems. A managed device is a network node that contains an SNMP agent and resides on a managed network. Managed devices collect and store management information and use SNMP to make this information available to management systems that use SNMP. Managed devices include routers, access servers, switches, bridges, hubs, computer hosts, and printers.

An agent is a software module that resides in a managed device. An agent has local knowledge of management information and translates that information into a form compatible with SNMP. The SNMP agent gathers data from the management information base (MIB), which is the repository for information about device parameters and network data. The agent can also send traps, or notification of certain events, to the manager. Figure 11-1 illustrates these SNMP operations.

Figure 11-1 SNMP Components



Versions of SNMP

ACNS 4.1 software supports the following versions of SNMP:

- Version 1 (SNMPv1)—This is the initial implementation of SNMP. Refer to RFC 1157 for a full description of its functionality.
- Version 2 (SNMPv2c)—This is the second release of SNMP, described in RFC 1902. It provides additions to data types, counter size, and protocol operations.
- Version 3 (SNMPv3)—This is the most recent version of SNMP defined in RFC 2271 through RFC 2275.

SNMPv1 and SNMPv2C do not have any security (that is, authentication or privacy) mechanisms to keep SNMP packet traffic on the wire confidential. As a result, packets on the wire can be detected and SNMP community strings compromised.

To solve the security shortcomings of SNMPv1 and SNMPv2c, SNMPv3 provides secure access to Content Engines by authenticating and encrypting packets over the network. In ACNS 4.1 software, SNMPv3 features are added to the SNMP agent in addition to SNMPv1 and SNMPv2c.

The following security features are provided in SNMPv3:

- Message integrity—Ensures that nothing has interfered with a packet during transmission.
- Authentication—Determines that the message is from a valid source.
- Encryption—Scrambles the contents of a packet to prevent it from being seen by an unauthorized source.

SNMPv3 provides both security models and security levels. A security model is an authentication process that is set up for a user and the group in which the user resides. A security level is the permitted level of security within a security model. A combination of a security model and a security level determines which security process is used when handling an SNMP packet. Three security models are available: SNMPv1, SNMPv2c, and SNMPv3. Table 11-1 describes the combinations of security models and security levels.

Table 11-1 SNMP Security Models and Security Levels

Model	Level	Authentication	Encryption	Process
v1	noAuthNoPriv	Community string	No	Uses a community string match for user authentication.
v2c	noAuthNoPriv	Community string	No	Uses a community string match for user authentication.
v3	noAuthNoPriv	Username	No	Uses a username match for user authentication.
v3	AuthNoPriv	Message Digest 5 (MD5) or Secure Hash Algorithm (SHA)	No	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms.
v3	AuthPriv	MD5 or SHA	Yes	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Provides DES 56-bit encryption (packet authentication) based on the CBC-DES (DES-56) standard.

The SNMPv3 agent can be used in the following modes:

- noAuthnoPriv mode (that is, no security mechanisms turned on for packets)
- AuthNoPriv mode (for packets that do not need to be encrypted using the privacy algorithm [DES 56])
- AuthPriv mode (for packets that must be encrypted; privacy requires that authentication to be performed on the packet).

Using SNMPv3, users can securely collect management information from their SNMP agents without fear that the data has been tampered with. Also, confidential information, such as SNMP set packets that change a Content Engine's configuration, can be encrypted to prevent their contents from being exposed on the wire. Also, the group-based administrative model allows different users to access the same SNMP agent with varying access privileges.

Key SNMP CLI Commands

Use the **snmp-server group** global configuration command to select one of three SNMP versions, SNMPv1, SNMPv2c, or SNMPv3. Use the **no** form of this command to remove the specified version. Refer to the *Cisco Application and Content Networking Software Command Reference* for more information on how to use this and other SNMP commands.

The **snmp-server community string** command provides view-based access control for SNMPv1, SNMPv2c, and SNMPv3 but also continues to provide backward compatibility between different versions. The previous version of this command did not have an option to create a community string that allows SNMP messages to execute a set operation on a MIB object. A **rw** option has been introduced for this purpose. Also, the previous version of the SNMP agent did not provide selective access control to MIB objects. Access to any MIB object was denied or granted based on authentication of the SNMP community string. With the introduction of view-based access control, it is now possible to configure a community string that grants access to only part of the MIB subtree. To provide backward compatibility

with the previous version of this command, a default read group or default write group (if the **rw** option is specified on the command line) is associated with the community string if no group name is specified. Both of these default groups are hidden from users and not displayed in the configuration file or in the **show snmp group** command, but are created during initialization of the SNMP agent.

Note The SNMP agent is disabled by default and a community string is not configured.

The following example enables the SNMP agent and assigns the community string *comaccess* to SNMP:

```
507-1(config)# snmp-server community comaccess
```

The preceding example defines a community string *comaccess* used as a password for authentication when you access the SNMP agent of the Content Engine. Any SNMP message sent to the Content Engine must have the "Community Name" field of the message match the community string defined here in order to be authenticated. Entering a community string enables the SNMP agent.

The following example disables the SNMP agent and removes the previously defined community string.

```
507-1(config)# no snmp-server community
```

Supported MIBs

The ACNS 4.1 software implementation of SNMP supports the following MIBs:

- MIB-II
- ENTITY-MIB
- CISCO-CONTENT-ENGINE-MIB
- CISCO-ENTITY-ASSET-MIB
- CISCO-CONFIG-MAN-MIB
- CISCO-CDP-MIB

Use the following link to access these MIBs:

ftp://ftp.cisco.com/pub/mibs/v2/.

SNMP Traps

To enable the Content Engine to send SNMP traps, use the **snmp-server enable traps** global configuration command. If you do not enter the **snmp-server enable traps** command, no traps are sent. Use the **no** form of this command to disable all SNMP traps or only SNMP authentication traps.

The **snmp-server enable traps** command is used in conjunction with the **snmp-server host** command. Use the **snmp-server host** command to specify which hosts or hosts receive SNMP traps. To send traps, you must configure at least one host.

For a host to receive a trap, both the **snmp-server enable traps** command and the **snmp-server host** command for that host must be enabled.

In addition, SNMP must be enabled with the **snmp-server community** command.

To disable the sending of the MIB-II SNMP authentication trap, you must enter the command **no snmp-server enable traps snmp authentication**.

The following example enables the Content Engine to send all traps to the host 172.31.2.160 using the community string *public*:

```
ContentEngine(config)# snmp-server enable traps
ContentEngine(config)# snmp-server host 172.31.2.160 public
```

The following example disables all traps:

```
Content Engine (config)# no snmp-server enable traps
```

CiscoWorks2000

CiscoWorks2000 (CW2K) is a Cisco product that provides a suite of management applications used to manage most Cisco devices. The Content Engine can interoperate with CiscoWorks2000 without any modification in the following ways:

- CW2K can list the Content Engine under “Generic SNMP” devices.
- The CW2K inventory module lists the Content Engine with the device name, system name, description (including the software version), uptime, and network interface information.
- The CW2K syslog module can understand Content Engine syslogs.
- The CW2K availability module can track the Content Engine.

You can enable or disable syslog message generation in CiscoWorks2000- compliant format through either the command-line interface (CLI) or the graphical user interface (GUI).

Use the **logging cw2k** command to enable CiscoWorks2000 syslog message generation. Use the **no logging cw2k** command to disable CiscoWorks2000 syslog message generation.

Cisco Discovery Protocol

Cisco Discovery Protocol (CDP) is a device discovery protocol that runs on all Cisco manufactured devices. With CDP, each device within a network sends periodic messages to all devices within the network. These devices listen to periodic messages sent by others in order to learn about neighboring devices and determine the status of their interfaces.

With CDP, network management applications can learn the device type and the SNMP agent address of neighboring devices. Applications are then able to send SNMP queries within the network.

Also, CiscoWorks 2000 discovers the Content Engine by noticing the CDP packets that are sent by the Content Engine after booting.

Content Engine-related tasks require that the Content Engine platform support CDP in order to be able to notify the system manager of the existence, type, and version of the Content Engine platform.

The following example enables CDP implementation with a single CLI command:

```
ContentEngine(config)# interface FastEthernet 0/0 cdp enable
```

CHAPTER 12

Configuring TCP Stack Parameters

Overview

Caches are typically deployed by customers for any of the following reasons:

- To save bandwidth
- To accelerate the delivery of content
- To apply policies on what content is viewed (content filtering)
- To increase the throughput of HTTP streams over TCP end to end

The fourth reason is an often overlooked and much less understood property of deploying caching, and it can often have a huge benefit on the performance of TCP end to end.

Queries sent between a server and a client and the replies generated are defined as transactions. For data transactions between client and servers, the size of windows and buffers is important, and fine-tuning the TCP stack parameters therefore becomes the key to maximizing this benefit.

The relevant TCP parameters to maximize cache performance and throughput include the ability to tune timeout periods, client and server receive/send buffer sizes, and TCP window scaling behavior.

Note Because of the complexities involved in TCP parameters, care is advised in tuning these parameters. In nearly all environments, the default TCP settings are adequate. Fine tuning of TCP settings is for network administrators with adequate experience and full understanding of TCP operation details.

Configuring TCP Parameters Using the Content Engine GUI

To view the TCP configuration parameters in the Content Engine, start the Content Engine GUI, click **Systems**, and then choose **TCP**. The TCP Configuration window is displayed. (See Figure 12-1.) See Table 12-1 for a description of the TCP configuration parameters and the corresponding CLI commands.

Figure 12-1 TCP Configuration Window

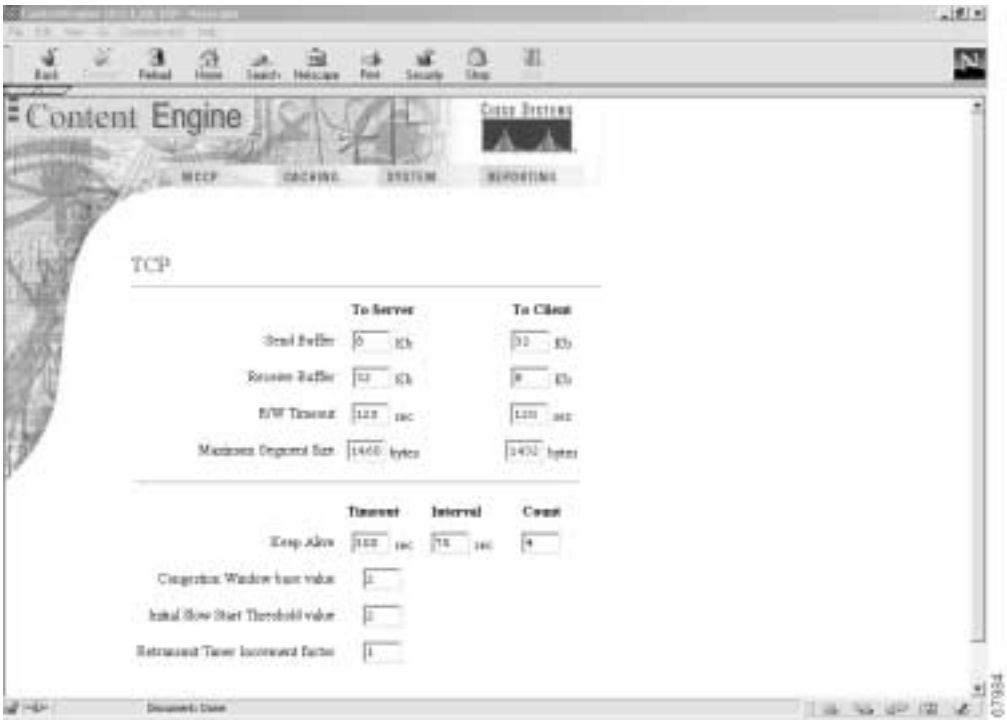


Table 12-1 TCP CLI Configuration Parameters

Fields	TCP CLI Commands	Descriptions
Send Buffer		
To Server	**tcp server-send-buffer** *kbytes*	TCP outgoing window size in kilobytes (1–128). The default is 8 KB.
To Client	**tcp client-send-buffer** *kbytes*	TCP outgoing window size in kilobytes (1–128). The default is 32 KB.
Receive Buffer		
To Server	**tcp server-receive-buffer** *kbytes*	TCP incoming window size in kilobytes (1–128). The default is 32 KB.
To Client	**tcp client-receive-buffer** *kbytes*	TCP incoming window size in kilobytes (1–128). The default is 8 KB.
R/W Timeout		
To Server	**tcp server-rw-timeout** *seconds*	Interval after which the Content Engine will time out trying to read or write to the network (1–3600). The default is 120 seconds.
To Client	**tcp client-rw-timeout** *seconds*	Interval after which the Content Engine will time out trying to read or write to the network (1–3600). The default is 120 seconds.

Table 12-1 TCP CLI Configuration Parameters (continued)

Fields	TCP CLI Commands	Descriptions
Keepalive		
Keepalive	**tcp keepalive-probe-interval**	Length of time that the Content Engine will keep a connection open before disconnecting.
Timeout	**tcp keepalive-interval** *seconds*	Length of time that the Content Engine will keep an idle connection open (1–3600 seconds). The default is 300 seconds.
Count	**tcp keepalive-probe-cnt** *count*	Number of times the Content Engine will retry a connection (1–10 attempts). The default is 4 attempts.
Congestion Window Base	**tcp cwnd-base**	Initial congestion window value (1–10 segments). The default is 2 segments.
Initial slow start threshold value	**tcp init-ss-threshold**	Low watermark threshold for slow start (2–10 segments). The default is 2 segments.
Retransmitter Timer Increment factor	**tcp increase-xmit-timer-value**	Factor (1–3) used to modify the length of the retransmit timer by 1 to 3 times the base value determined by the TCP algorithm. The default is 1, which leaves the times unchanged. **Note** Modify this factor with caution. It can improve throughput when TCP is used over slow reliable connections but should never be changed in an unreliable packet delivery environment.
Maximum Segment Size		
Server	**tcp server-mss** *maxsegsize*	Maximum packet size sent to servers. The default is 1460 bytes.
Client	**tcp client-mss**	Maximum packet size sent to clients. The default is 1432 bytes
Others		
Satellite	**tcp server-satellite** **tcp client-satellite**	Server and client TCP compliance with RFC 1323. See the "TCP-Over-Satellite Extensions" section on page 12-3.
Type of Service	**type-of-service**	TCP Type of Service. The default is disabled.
Ecn	**ecn**	TCP explicit congestion notification.

TCP-Over-Satellite Extensions

The Content Engine has the ability to turn on TCP-over-satellite extensions (as documented in RFC 1323) to maximize performance and end-to-end throughput over satellite-type connections.

The large number of satellites available to network infrastructures has increased the amount of bandwidth available in the air. Taking advantage of these connections through satellite-type connections has created new challenges in the use of TCP transactions and acknowledgments:

- Latency—Round trip times to satellites orbiting 24,000 miles above the earth are 550 milliseconds for a single satellite hop. Window size must be set to prevent low-throughput connections.
- Bit errors—Packet loss can occur in a land-based- device-to-satellite connection in addition to the losses caused by regular network congestion.

- Asymmetric bandwidth—Return bandwidth from satellites can be narrower than receiving bandwidth, thereby affecting performance.

Use the **tcp server-satellite** and **tcp client-satellite** commands to set the TCP connection so that it complies with RFC 1323.

TCP Configuration Examples

To display current TCP configuration information, use the **show tcp** EXEC command. Note that the default 8 KB incoming window size for the client buffer is used.

```
ContentEngine# show tcp
        ==TCP Configuration==
TCP keepalive timeout 300 sec
TCP keepalive probe count 4
TCP keepalive probe interval 75 sec
TCP server R/W timeout 120 sec
TCP client R/W timeout 120 sec
TCP server send buffer 8 k
TCP server receive buffer 32 k
TCP client send buffer 32 k
TCP client receive buffer 8 k
TCP server max segment size 1460
TCP satellite (RFC1323) disabled
TCP client max segment size 1432
TCP explicit congestion notification disabled
TCP type of service disabled
TCP cwnd base value 2
TCP initial slowstart threshold value 2
TCP increase(multiply) retransmit timer by 1
ContentEngine#
```

In this example, the **tcp client-receive-buffer** command is used to change the TCP incoming window size to 100 KB.

```
ContentEngine(config)# tcp client-receive-buffer 100
ContentEngine(config)
```

You can now verify the configuration change with the **show tcp** command.

```
ContentEngine# show tcp
        ==TCP Configuration==
TCP keepalive timeout 300 sec
TCP keepalive probe count 4
TCP keepalive probe interval 75 sec
TCP server R/W timeout 120 sec
TCP client R/W timeout 120 sec
TCP server send buffer 8 k
TCP server receive buffer 32 k
TCP client send buffer 32 k
TCP client receive buffer 100 k
TCP server max segment size 1460
TCP satellite (RFC1323) disabled
TCP client max segment size 1432
TCP explicit congestion notification disabled
TCP type of service disabled
TCP cwnd base value 2
TCP initial slowstart threshold value 2
TCP increase(multiply) retransmit timer by 1
ContentEngine#
```

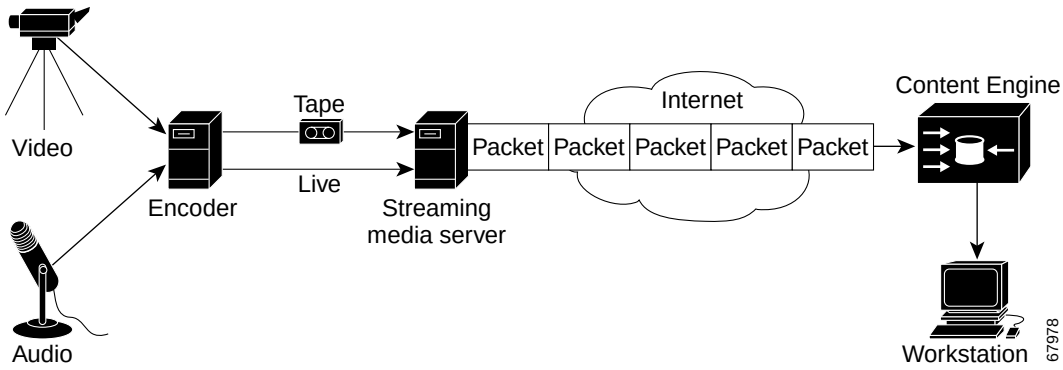
CHAPTER 13

Configuring Windows Media Technologies 7.01 Streaming Media Caching

Streaming Media Overview

Streaming media files are files that are sent to the user and played on the user's media player as the files are received from the network. Streaming media files avoid a waiting period for viewing these files because they are immediately available as a continuous stream of data packets. (See Figure 13-1.) This eliminates the need to store large media files for viewing purposes or the need to allocate storage space for these files before playing them.

Figure 13-1 Streaming Media Model with Content Engine as Manual Proxy



In Figure 13-1 both audio and video are being recorded from an event to be distributed later either as video on demand (VOD) or live to a network of users. The encoder software and hardware compress the signal into streamable files, which are then sent to a media file server. This media server in turns delivers the media files on a live or on-demand basis to users with the particular media software on their personal computers or other electronic devices. Note that in this figure the Content Engine is manually configured to be the caching proxy for the users in that particular network.

Table 13-1 lists the different types of streaming media protocols, control channels, the corresponding data format, and transport types supported by ACNS 4.1 software.

Table 13-1 Streaming Media Protocols

Streaming Media Protocol	Control Channel	Data Format	Transport Protocol
Windows Media format	TCP	MMS[1]	UDP[2], TCP, HTTP, IP multicast
RealNetworks media format	TCP	RTSP, PNA[3]	UDP, TCP, HTTP, IP multicast

1. MMS = Microsoft Media Server.
2. UDP = User Datagram Protocol.
3. PNA = Progressive Networks Audio.

Configuring Microsoft Windows Media Player 7.01

Microsoft Windows Media Technologies (WMT) is a set of streaming solutions for creating, distributing, and playing back digital media files on the Internet. WMT includes the end user application (Windows Media Player) Version 7.01 and the server and distribution application (Windows Media Server). To disseminate live and pre-positioned Windows Media content on a Content Delivery Network (CDN), you need WMT caching proxy and server capabilities on the Content Engine.

Note The WMT product is licensed software. To enable the WMT proxy, you must have a license keyword. If you ordered WMT with the CE, a license certificate with the keyword is shipped in the box. If you are downloading the ACNS 4.1 software, you can purchase a WMT license though the Cisco.com website.

WMT Caching Proxy Details

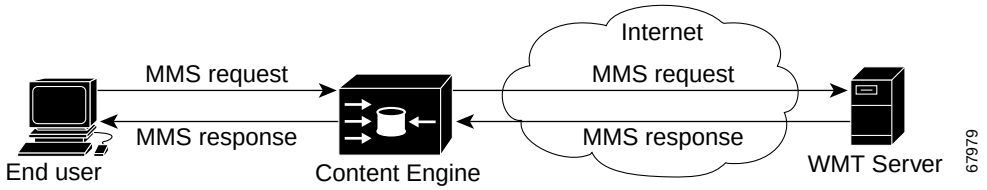
The Content Engine acting as a WMT caching proxy supports a basic proxy feature—it accepts incoming WMT streaming requests from clients and acts on behalf of the clients communicating with the origin server. The WMT caching proxy accepts and serves the streaming requests over Microsoft Media Server (MMS) protocol as well as the HTTP protocol. (See Figure 13-2.) MMS is the protocol that WMT uses for communication between players and servers.

When possible, the WMT caching proxy also caches the streaming content and serves the request from the cache instead of the origin server. It accepts transparently intercepted requests (through WCCP or L4 redirect) as well as manual proxy requests (clients configured to use an upstream proxy).

The WMT caching proxy also supports splitting of live streams (that is, it splits a single inbound feed to multiple clients) and multicasting.

Note If a firewall is positioned between a Content Engine acting as a proxy, and a requesting client, make sure that you assign the external IP address of the Content Engine in its configuration.

Figure 13-2 Content Engine as Conventional WMT Caching Proxy



Note For MMS over TCP and UDP, the proxy always fetches the stream from a server using TCP (MMST) so that the item cached is reliable and complete for future delivery. Streams requested using HTTP will be fetched using HTTP.

Caching

Caching frequently accessed content closer to the user provides better-quality streams and saves network bandwidth. When the proxy fetches the content from the server to be served to the client, it will store the cacheable stream on the media file system (mediafs). The proxy can cache partial streams. When a client requests a part of the stream that has not been cached, the proxy fetches the missing portions from the origin server. Even when there is a cache hit, a connection is maintained with the origin server in order to report client usage statistics, which can be used for accounting and other purposes by the origin server.

Streams served from the cache are guaranteed to be fresh because the Content Engine always checks with the origin server, using the appropriate cache parameters settings. When the storage limit is reached, older objects are replaced using an appropriate replacement algorithm to ensure a proper hit ratio. See the "Cache Freshness" section on page 6-1 for more information regarding the appropriate cache parameter settings.

The proxy can cache authenticated streams, and on a cache hit, the authentication credentials of the requesting client are revalidated against the origin server before the content is served.

Caching is supported in nontransparent (manual) proxy as well as transparent proxy modes.

Note Caching of Moving Picture Experts Group Layer-3 audio (MP3), waveform audio (WAV), and Audio Video Interleaved (AVI) files over MMS is supported. Windows Media content fetched by a nonstreaming HTTP server (at the request of a Windows Media Player) is not cached. Live streams are not recorded or cached.

Variable Bit Rate

A content provider can create streaming media files at different bit rates to ensure that different clients who have different connections—for example, modem, DSL or LAN—can handle a particular bit rate of their choice. The WMT caching proxy can cache multiple bit rate or variable bit rate (VBR) files, and based on the bit rate specified by the client, it serves the appropriate stream.

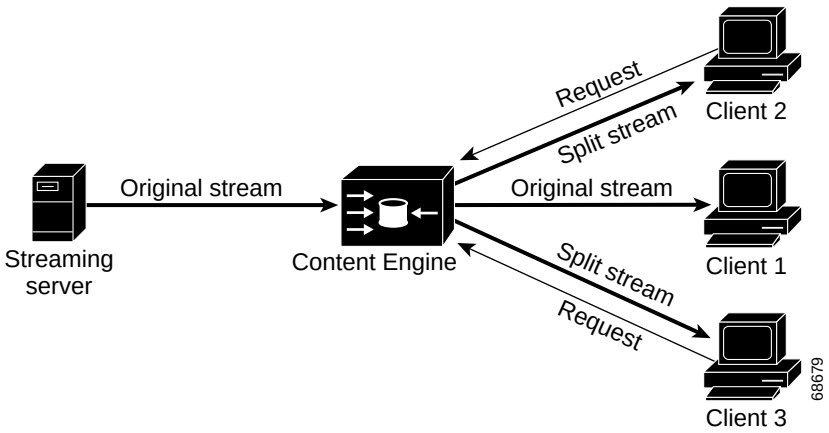
Another advantage of creating variable bit rate files is that a single URL is all that must be specified for the delivery of streaming media.

Use the **wmt max-bitrate** command to configure the maximum bit rate for streaming media delivery.

Live Splitting

The Content Engine splits requests for live streams. That is, a single stream from the origin streaming server is split to serve each client that requested the stream. (See Figure 13-3.) In the case of the WMT caching proxy, when the client requests a publishing point on a server (without specifying an ASF file), then the WMT caching proxy dynamically creates an alias file that references the remote server. All further requests to that station are served by splitting the stream. Note that when the first client (Client 1) that requested the original stream disconnects from the network, the proxy continues to serve the other clients (Client 2 and Client 3), until all clients disconnect from the network.

Figure 13-3 Live Splitting



Live splitting at the proxy is better than at the server, because the proxy is closer to the clients, thereby potentially saving considerable network bandwidth between the client and the origin server.

Note Live splitting is supported for different data packet transport protocols (HTTP, MMS TCP [MMST], MMS UDP [MMSU], and IP multicast).

Proxy Authentication

The WMT proxy supports both basic and NTLM authentication by the origin server. When a client requests content that needs user authentication, the proxy acts as an agent, conveying the authentication information to and from the client and server to authenticate the client. Once the client is authenticated, the content is streamed as usual. The authentication is performed for both cached content as well as noncached VOD content.

In the case of basic authentication, the server requests the client's identification in the form of an encoded username and password. If the authentication fails, the client is informed accordingly, in which case the client retries or disconnects. If the authentication is successful, then the streaming media is served to the client. This is supported in manual as well as transparent proxy mode, over both HTTP and MMS.

Windows NTLM Authentication

Windows NTLM is a connection-based challenge-response authentication scheme. In the basic authentication scheme, the proxy transfers the authentication information to and from the client and server, as if the request were coming from the client, until the client is authenticated. On the other hand, because the NTLM protocol authenticates every connection, the proxy cannot arbitrarily create new connections with the origin server. The proxy must reuse connections initiated by the client. Hence, a file is served from the cache only if it is a complete cache hit, that is, the complete file is present on disk. If the file is not a complete hit, then the entire file is fetched from the origin server in the case of NTLM.

This is supported in manual as well as transparent proxy mode, over both HTTP and MMS.

The proxy supports caching and delivery of Digital Rights Management (DRM)-protected Windows Media files. Access control lists enforced by the origin server are automatically enforced by the proxy.

Note Filtering based on user identification is also supported. The proxy only supports authentication by the origin server. Proxy authorization, or authentication of the user to use the proxy, will be supported in a future release. Live streams that are split to clients are also authenticated with the origin server in the ACNS 4.1 software.

Miscellaneous Features

This section lists features that are supported by the WMT caching proxy in the ACNS 4.1 software.

Transaction Logging

The logging format is consistent with that of the Windows Media server and the W3C-compliant log format. A log line is written for every stream accessed by the client. The location of the log is not configurable. These logs can be exported using FTP. All client information in the transaction logs is sent to the origin server by default and cannot be disabled.

Error Logging

The error log consists of the debugging logs written by the application. Error logs are in the same format and location as syslogs.

Statistics

The WMT proxy needs to maintain key statistics to provide byte savings, number of cache hits and cache misses, number of concurrent streams (according to bandwidth rates), percentage of cache or disk space used, and so forth.

You can view WMT statistics by choosing **Reporting > WMT Streaming** from the Content Engine GUI. (See Figure 13-4.) You can also use the **show statistics wmt all** command to view WMT statistics using CLI. (See the "WMT Examples" section on page 13-15.)

WMT Requirements

Interoperability is the most important requirement for WMT software components. The WMT caching proxy is required to work with all versions of Microsoft Windows Media Player, Windows Media Server, Windows Media Encoder and third-party Windows Media applications.

In order to support transparent proxy mode, you need to have WCCP running on the Content Engine. See the "Enabling Transparent WMT Service Using WCCP-Enabled Routers" section on page 13-6.

Enabling WMT on the Content Engine

To enable WMT on the Content Engine, follow these steps.

Note You can only use CLI commands to enable WMT on the Content Engine.

Step 1 Enter the **wmt license-key** command to be able to use WMT capabilities on the Content Engine.

```
507-1(config)# wmt license-key WORD
```

where **WORD** is the WMT license shipped with the Content Engine.

Step 2 Accept the license agreement.

```
507-1(config)# wmt accept-license-agreement
```

Step 3 Enable the Content Engine with the **wmt enable** command.

```
507-1(config)# wmt enable
```

You are now able to configure the WMT parameters required for streaming media connections through WMT technologies by transparent caching or conventional caching.

Note You must configure disk space to include mediafs storage with the **disk config** command before you can run cache streaming media using WMT.

Enabling Transparent WMT Service Using WCCP-Enabled Routers

During transparent caching, the user's network traffic flows through the WCCP-enabled router rather than the Content Engine to access streaming media.

Requirements

- Content Engine running ACNS 4.1 or later software
- WMT proxy software installed with mediafs partitions mounted on the Content Engine
- WMT license key
- IP address of the router or routers

Procedure

To enable transparent redirection of WMT traffic to the Content Engine acting as a WMT proxy, follow these steps.

Step 1 On the routers running WCCP Version 2, turn the WCCP feature on for the specified service groups used to redirect WMT traffic to the Content Engine. For more information on router commands, see Appendix B, "Web Cache Communication Protocol Version 2."

```
router(config)# ip wccp 81
router(config)# ip wccp 82
```

Note MMS works on two transport protocols: TCP and UDP. To perform a WCCP redirect of MMS traffic, the router has to redirect both TCP and UDP traffic. With a router running WCCP Version 2, that means you must enable two WCCP service groups on the router. The standard service group is 81 for TCP and 82 for UDP.

Step 2 Configure the outbound interfaces to the Internet and enter interface configuration mode. In the following example, the outbound interface is the Ethernet 0 device.

```
router(config)# interface Ethernet 0
```

Note Although typical router configuration in a branch office scenario involves configuring the outgoing interface, you can also configure the incoming interface on the router for traffic redirection. This depends primarily on the client's network topology.

Step 3 Enable WCCP redirection to service groups 81 and 82 on the specified interface.

```
router(interface)# ip wccp 81 redirect out
router(interface)# ip wccp 82 redirect out
```

Step 4 Set the WCCP Version 2 parameters on the Content Engine.

```
ContentEngine(config)# wccp version 2
ContentEngine(config)# wccp wmt router-list-num 1
```

Step 5 Enter the numbered router list that you wish to associate with this service. In the following example, the WCCP Version 2-enabled routers have the IP addresses 172.16.25.25 and 172.16.24.24.

```
ContentEngine(config)# wccp router-list 1 172.16.25.25 172.16.24.24
```

Step 6 Enable WMT on the Content Engine. See the "Enabling WMT on the Content Engine" section on page 13-6.

Step 7 Save the new configuration.

```
ContentEngine# copy running-config startup-config
```

Step 8 Configure WMT parameters as needed using CLI commands.

Note Refer to the *Cisco Application and Content Networking Software Command Reference, Release 4.1* for more detailed information on these parameters. Alternatively, you can click the **WMT Config** button to configure the WMT parameters as needed with the Content Engine GUI. (See Figure 13-4.)

Figure 13-4 WMT Administration Page in Content Engine GUI

Step 9 Use the following CLI command to display all WMT statistics once you have started the WMT player:

```
ContentEngine# show statistics wmt all
```

Note The WMT statistics relate only to objects transported over MMS that were requested by a WMT client. Objects transported over HTTP are counted in the HTTP statistics. Streaming objects requested by other clients or transported over other protocols other than RTSP (for RealPlayer streaming media) bypass the Content Engine. See the "Configuring RealProxy 8.01" section on page 14-1 for more information about RTSP streaming and caching with the Content Engine.

Enabling Conventional WMT Proxy Service

During conventional proxy caching, the user media player is pointed to the Content Engine rather than a WCCP-enabled router to access streaming media.

Requirements

- Content Engine running ACNS 4.1 or later software
- WMT Proxy software installed with mediafs partitions mounted on the Content Engine
- Microsoft WMT license key

- IP address of the Content Engine

Procedure

To configure the Content Engine to service WMT clients with the WMT proxy, follow these steps:

Step 1 Enable WMT on the Content Engine. See the "Enabling WMT on the Content Engine" section on page 13-6.

Step 2 Configure the Content Engine to listen for WMT traffic on a specified port. The standard WMT port is 1755.

```
ContentEngine(config)# wmt incoming 1755
```

Step 3 Configure WMT media players to use the Content Engine as the WMT proxy. (See Figure 13-5.)

a. Open the Windows Media Player.

b. Choose **Tools > Options**.

c. Click the **Network** tab.

d. Click the **Multicast**, **UDP**, **TCP** and **HTTP** radio buttons if not selected.

e. Click **MMS** under Proxy Settings and click **Configure.** The Configure Protocol Page in WMT Media Player page appears. (See Figure 13-6.)

f. Click **Use the following proxy server**.

g. Enter the IP address of the Content Engine in the Address field.

h. In the port field, enter the port number that you entered in Step 2.

i. Click **OK**.

Figure 13-5 WMT Media Player Network Options

Figure 13-6 Configure Protocol Page in WMT Media Player

Step 4 Save the new configuration.

```
ContentEngine# copy running-config startup-config
```

Step 5 Configure WMT parameters as needed using the CLI commands. Use the **wmt ?** command to see available configuration options.

```
ContentEngine(config)#wmt ?
  accept-license-agreement     Accept license; View by 'show wmt license-agreement'
  broadcast                    Broadcast live configuration.
  cache                        WMT cache config
  disallowed-client-protocols  Specify disallowed WMT client protocols
  enable                       Enable WMT
  evaluate                     Start/continue 60-day evaluation of WMT.
  incoming                     Configuration for incoming WMT requests
  l4-switch                    Configure layer 4 switch interoperability for WMT.
  license-key                  Required license key for WMT
  max-bandwidth                Maximum aggregate bandwidth limitation.
  max-bitrate                  Maximum stream bit rate that can be served to a client.
  max-concurrent-sessions      Maximum number of unicast clients that can be served concurrently.
  multicast                    Multicast configuration and scheduling.
ContentEngine(config)
```

Note Alternatively, you can click the **WMT Config** button to configure the WMT parameters as needed with the Content Engine GUI. (See Figure 13-4.)

Step 6 Use the following CLI command to display all WMT statistics once you have started the WMT player:

```
ContentEngine# show statistics wmt all
```

Note The WMT statistics relate only to objects transported over MMS that were requested by a WMT client. Objects transported over HTTP are counted in the HTTP statistics. Streaming objects requested by other clients or transported over other protocols other than RTSP (for RealPlayer streaming media) bypass the Content Engine.

WMT Multicasting

Based on the capabilities and limitations of the network, a Content Engine can receive and deliver WMT streaming content through IP multicast in the following three scenarios:

- Unicast-in Multicast-out
- Multicast-in Multicast-out
- Multicast-in Unicast-out

The Unicast-in Multicast-out multicast delivery feature enables you to distribute streaming media efficiently by allowing different devices on the IP multicast to receive a single stream of media content from the Content Engine simultaneously. This can save significant network bandwidth consumption, because a single stream is sent to many devices, rather than sending a single stream to a single device every time that this stream is requested.This multicast delivery feature is enabled by setting up a multicast address on the Content Engine to which different devices, configured to receive content from the same channel, can subscribe. The delivering device sends content to the multicast address set up at the Content Engine, from which it becomes available to all subscribed receiving devices.

The Multicast-in Multicast-out multicast receive feature enables you to receive multicast WMT streams delivered through IP multicasting, then relay them to end users through another delivery channel (unicast or multicast).

The two WMT multicast-out features combined enable you to receive and deliver WMT streaming media content through IP multicasting, and to do conversions with unicast and relays in all fashions.

The Multicast-in Unicast-out scenario enables you to create a broadcasting publishing point to deliver an incoming stream live to requesting clients using multicast as the source of the streaming media.

WMT Multicast and Broadcast CLI Commands

Two global configuration CLI commands are needed to configure the Content Engine for the WMT multicasting scenarios described:

- **wmt multicast**—Unicast-in Multicast-out
- **wmt multicast**—Multicast-in Multicast-out
- **wmt broadcast**—Multicast-in Unicast-out

WMT Multicast

Use the **wmt multicast** {**schedule-start** *name minute hour day month* | **station-configuration** *name dest_addr dest_port media_source* [**play-forever**]} command to enable WMT multicasting for the Unicast-in Multicast-out and Multicast-in Multicast-out scenarios on the Content Engine. The **schedule-start** *name minute hour day month* option creates a scheduling option to allow the Content Engine to start a multicast at a specified time. This option only works if you have configured a multicast station first.

Note You must enable WMT on the Content Engine before you can use the **wmt multicast** and **wmt broadcast** commands. See the "Enabling WMT on the Content Engine" section on page 13-6 for information on the steps needed to enable this feature.

The **station-configuration** *name dest_addr dest_port media_source* option specifies a multicast station name, an IP multicast address, port number and media source for the multicast station created. Each station needs a multicast IP address. You must enter a valid class D IP address multicast address in the range 224.0.0.0 to 239.255.255.255, except for the reserved IP ranges based on RFC 1700 and related documents as follows:

- 224.0.0.0–224.0.6.255
- 224.0.13.0–224.0.13.255
- 224.1.0.0–224.2.255.255
- 232.0.0.0–232.255.255.255

Note You must choose a multicast IP address that does not conflict internally within the same multicast-enabled network configuration. This multicast IP address is not related to the IP address of the Content Engine.

The allowed multicast port range defined by the *dest_port* option is 1 through 65535. However, the multicast-enabled network may impose certain restrictions on your choice of port. Normally, port numbers below 1024 should be avoided, but the Content Engine does not enforce any restrictions.

The *media_source* option determines the source of the multicast. The source can be any valid WMT URL. In other words, if you can play the URL on your Windows Media player, then you can make this URL the source of your multicast.

WMT Broadcast

Use the **wmt broadcast** {**alias-name** *name* **source** *url*} command to configure the Multicast-in Unicast-out broadcast scenario on the Content Engine. With this command, you create a broadcasting alias to deliver an incoming stream live to requesting clients using multicast as the source of the streaming media.

Note You can also configure WMT multicasting parameters with the Content Engine GUI. Click the **WMT Config** button shown in Figure 13-4 to access these parameters.

Unicast-in Multicast-out

The unicast input can be from a video on demand (VOD) publishing point, a live unicast publishing point, an encoder, or a streaming media source from a local disk. The ASF header obtained from the unicast input and the parameters used to configure the multicast station are used by the Content Engine to automatically create the multicast description .nsc file. The clients use this easily accessible file to subscribe to the multicast.

To enable WMT multicasting in this scenario, follow these steps:

Step 1 Enable WMT multicasting and configure a multicast station on the Content Engine in global configuration mode with the **wmt multicast station-configuration** command:

```
ContentEngine(config)# wmt multicast station-configuration test1 233.33.33.33 3333
mms://sourceIPaddress/source.asf play-forever
ContentEngine(config)#
```

In this example a multicast station named *test1* is configured and used by the Content Engine as the multicast source file. Its class D IP address is 233.33.33.33 and the multicast port is 3333. The **play-forever** option is used. When the input source.asf is a VOD file, this option automatically restarts from the beginning of the source.asf file once the end of this file has been reached.

Note This source file *source.asf* can be located on any WMT server, including a Windows server, or the Content Engine. In the case of the Content Engine, pre-positioned media files should be stored in the /local1/wmt_vod directory. In this scenario, the media source is represented by mms://CEIPaddress/wmt_vod/source.asf.

Step 2 Start the multicast using the **wmt multicast-station** command in EXEC mode.

```
ContentEngine# wmt multicast-station start test1
ContentEngine#
```

Step 3 Open your WMT player and choose **File** > **Open URL**. Enter the followingURL:

```
http://CEIPaddress/test1.nsc
```

Click **OK**. The WMT player should retrieve the multicast description .nsc file and join the multicast station that is specified in Step 1.

Note The use of port 80 is implied in the URL for WMT multicasting. An equivalent URL is http://CEIPaddress:80/test1.nsc.

Multicast-in Multicast-out

In this multicasting scenario, a description file *.nsc is created that is accessible through multicast-out to clients. This is similar to the Unicast-in Multicast-out scenario except that the input source is multicast. The clients use this file to subscribe to the multicast.

To enable WMT multicasting in this scenario with CLI commands follow these steps:

Step 1 Enable WMT multicasting and configure a multicast station on the Content Engine in global configuration mode with the **wmt multicast station-configuration** command:

```
ContentEngine(config)# wmt multicast station-configuration test2 233.33.33.34 6667
mms://172.16.30.31/source.nsc
ContentEngine(config)#
```

In this example a multicast station named *test2* is configured and used by the Content Engine as the multicast source file. Its class D IP address is 233.33.33.34 and the multicast port is 6667. The **play-once** default option is used. This option stops the multicast once the end of the source.nsc file has been reached.

Step 2 Start the multicast with the **wmt multicast-station** command in EXEC mode.

```
ContentEngine# wmt multicast-station start test2
ContentEngine#
```

Step 3 Open your WMT player and choose **File** > **Open URL**. Enter the following URL:

```
http://CEIPaddress/test2.nsc
```

Click **OK**. The WMT player should receive the MMS media source specified in Step 1.

Multicast-in Unicast-out

In this scenario a unicast-out publishing point is created to deliver the incoming stream live to requesting clients.

To enable WMT multicasting in this scenario with CLI commands follow these steps:

Step 1 Enable WMT multicasting and configure a broadcasting alias on the Content Engine in global configuration mode with the **wmt broadcast** command:

```
ContentEngine(config)# wmt broadcast alias-name myunicast source
http://172.16.30.31/station.nsc
ContentEngine(config)#
```

In this step a unicast publishing point with the alias name *myunicast* is configured with a multicast source station.nsc file. This source is a server sending out WMT multicast streams. A source of an alias in the format of http://server/file.nsc signals the Content Engine to treat this source as a multicast input source.

Step 2 Open your WMT player and choose **File > Open URL**. Enter the following URL:

```
mms://CEIPaddress/myunicast
```

Click **OK**. The WMT player should receive the MMS media source specified in Step 1. Note that in this scenario an MMS URL is used to access the streaming media, and that only the *alias-name* is specified instead of the **.nsc* files in the Multicast-out scenarios.

This converts the multicast stream to unicast and sends it to the requesting client.

WMT Examples

In the following example, the **show statistics wmt ?** command is used to show the CLI options for displaying statistics monitored by the Content Engine.

```
ContentEngine(config)# show statistics wmt ?
  all        Display all Windows Media statistics
  bytes      Display unicast bytes statistics
  errors     Display errors statistics
  multicast  Display multicast statistics
  requests   Display unicast request statistics
  savings    Display savings statistics
  usage      Display current usage statistics
  statistics Display statistics by module
```

The following example displays request statistics. In this example, the statistics reported are the total number of requests served, the type of content (live or VOD), source of content, transport protocol, and the source of content.

```
ContentEngine# show statistics wmt requests
Unicast Requests Statistics
===========================
Total unicast requests received: 4
```

```
-----------------------------------

                              Total           % of Total
                                            Unicast Requests
                            ------------------------------------------

    Total Requests served:          4              100.00%

                              Total        % of Total Requests
                            ------------------------------------------

By Type of Content
------------------
             Live content:          2               50.00%
        On-Demand Content:          2               50.00%

By Transport Protocol
---------------------
                     MMSU:          4              100.00%
                     MMST:          0                0.00%
                     HTTP:          0                0.00%

By Source of Content
--------------------
                    Local:          0                0.00%
               Remote MMS:          4              100.00%
              Remote HTTP:          0                0.00%
                Multicast:          0                0.00%

ContentEngine#
```

In the following example, all WMT statistics are displayed.

```
ContentEngine# show statistics wmt all
Unicast Requests Statistics
===========================
Total unicast requests received: 4
-----------------------------------

                              Total           % of Total
                                            Unicast Requests
                            ------------------------------------------

    Total Requests served:          4              100.00%

                              Total        % of Total Requests
                            ------------------------------------------

By Type of Content
------------------
             Live content:          2               50.00%
        On-Demand Content:          2               50.00%

By Transport Protocol
---------------------
                     MMSU:          4              100.00%
                     MMST:          0                0.00%
                     HTTP:          0                0.00%

By Source of Content
--------------------
                    Local:          0                0.00%
               Remote MMS:          4              100.00%
              Remote HTTP:          0                0.00%
```

```
               Multicast:          0               0.00%

Unicast Bytes Statistics
========================
Total unicast outgoing bytes: 51089546
--------------------------------
                                        Total     % of Total Unicast
                                                   Outgoing Bytes
                                    ------------------------------------------

By Type of Content
------------------
            Live content:             50721284          99.28%
       On-Demand Content:               368262           0.72%

By Transport Protocol
---------------------
                    MMSU:             51089546         100.00%
                    MMST:                    0           0.00%
                    HTTP:                    0           0.00%

Unicast Savings Statistics
==========================
Total bytes saved: 353256
--------------------------
                                      Total       % of Total Bytes
                                                        Saved
                                    ------------------------------------------
By Pre-positioned content:                0           0.00%
        By Live-splitting:                0           0.00%
             By Cache-hit:              353256         100.00%

                                      Total         % of Total
                                                Incoming Live Bytes
                                    ------------------------------------------
Live Splitting
--------------
          Incoming bytes:             50750993         100.00%
          Outgoing bytes:             50721284         100.00%
             Bytes saved:                    0           0.00%

                                      Total       % of Bytes Cache
                                                        Total
                                    ------------------------------------------
Caching
-------
         Bytes cache-miss:               15006           4.07%
          Bytes cache-hit:              353256          95.93%
        Bytes cache-total:              368262         100.00%

     Bytes cache-bypassed:                   0

                                      Total         % of Req Cache
                                                        Total
                                    ------------------------------------------

Cacheable requests
------------------
           Req cache-miss:                   0           0.00%
            Req cache-hit:                   2         100.00%
    Req cache-partial-hit:                   0           0.00%
          Req cache-total:                   2         100.00%

       Req cache-bypassed:                   0
```

```
Objects not cached
------------------
           Cache bypassed:                    0
          Exceed max-size:                    0

Multicast statistics
====================

Total Multicast Outgoing Bytes: 0

Aggregate Multicast Out Bandwidth (Kbps)
----------------------------------------------
        Current:     0.000
            Max:     0.000

Number of Concurrent Active Multicast Sessions
----------------------------------------------
        Current:         0
            Max:         0

List of All Configured Multicast Stations
-----------------------------------------------
Total Number of Configured Multicast Stations: 0


Usage Summary
=============
Concurrent Unicast Client Sessions
----------------------------------
        Current:         1
            Max:         1

Concurrent Active Multicast Sessions
------------------------------------
        Current:         0
            Max:         0

Concurrent Remote Server Sessions
---------------------------------
        Current:         1
            Max:         1

Concurrent Unicast Bandwidth (Kbps)
-----------------------------------
        Current:   107.125
            Max:   107.125

Concurrent Multicast Out Bandwidth (Kbps)
-----------------------------------------
        Current:     0.000
            Max:     0.000

Concurrent Bandwidth to Remote Servers (Kbps)
---------------------------------------------
        Current:    50.343
            Max:   107.125

Error Statistics
================
                   Total request errors:         0

  Errors generated by this box
                  Reach MAX connections:         0
```

```
                      Reach MAX bandwidth:         0
                       Reach MAX bit rate:         0
                          MMSU under wccp:         0
                         MMSU not allowed:         0
                         MMST not allowed:         0
                        MMSU/T not allowed:         0
                          HTTP not allowed:         0
   1st tcp pkt error, possible port scan:         0
                                Illegal url:         0
                                  No socket:         0
                              Cannot connect:         0
                         Authentication fail:         0
                         Remote server error:         0
                                Client error:         0
                              Internal error:         0
                    Local vod file not found:         0
           Local vod file header corrupted:         0
             Local vod file data corrupted:         0
                               Unknown error:         0
Errors generated by remote servers
                        Reach MAX connections:         0
                          Reach MAX bandwidth:         0
                           Reach MAX bit rate:         0
                                  Illegal url:         0
                             Invalid request:         0
                                  No socket:         0
                              Cannot connect:         0
                           Connection refused:         0
                                 Access deny:         0
                         Invalid stream type:         0
                         Remote server error:         0
                              Remote timeout:         0
                          Remote proxy error:         0
                              File not found:         0
                        File header corrupted:         0
                          File data corrupted:         0
                         Remote unknown error:         0

  Authentication Retries from Clients:                0
```

CHAPTER 14

Configuring RealProxy 8.01 Streaming Media Caching

Configuring RealProxy 8.01

The RealProxy software from RealNetworks, Inc., included as a software option to ACNS 4.1 software, supports both stream splitting (distributing live feeds) and streaming media caching (on-demand content) in the Real-Time Streaming Protocol (RTSP)-based format.

When performing stream splitting, the RealProxy accepts a live stream from a RealServer and re-serves the stream to multiple requesting RealPlayer clients, thus eliminating multiple connections to the RealServer. The RealServer is preconfigured to act as a RealMedia transmitter and the RealProxy is preconfigured to act as a RealMedia receiver.

Note The RealNetworks, Inc. RealProxy product is licensed software. To activate the RealProxy, you must have a license keyword, which is supplied on a certificate shipped with the Content Engine. If you are downloading the ACNS 4.1 software, you can purchase a RealProxy license though the Cisco.com website.

Streaming media caching provides content on demand. If one user has viewed a cached streaming media file, it can be served to subsequent users without the requirement to connect with the origin server. Live broadcasts are not files and are not cached.

Table 14-1 lists the different types of streaming media protocols, control channels, the corresponding data format, and transport types supported by ACNS 4.1 software.

Table 14-1 Streaming Media Protocols

Streaming Media Protocol	Control Channel	Data Format	Transport Protocol
Windows Media format	TCP	MMS[1]	UDP[2], TCP, HTTP, IP multicast
RealNetworks media format	TCP	RTSP, PNA[3]	UDP, TCP, HTTP, IP multicast

1. MMS = Microsoft Media Server.
2. UDP = User Datagram Protocol.
3. PNA = Progressive Networks Audio.

Table 14-2 describes the features and benefits of RealProxy software.

Table 14-2 RealProxy Features and Benefits

RealProxy Feature	Description	Benefits
Proxy for RealPlayer 8.01	The RealProxy makes requests for content on behalf of client RealPlayer users.	• Manages traffic inside the firewall by coordinating requests for similar content. • Masks end user IP addresses.
Splitting support for live broadcasts	The RealProxy "splits" a single inbound live broadcast feed to multiple client RealPlayers. See the "Live Splitting" section on page 13-4 of the Configuring Windows Media Technologies 7.01 Streaming Media Caching chapter for more information on live-splitting.	• Reduces inbound bandwidth usage to a single stream of content during a live event. • Improves RealPlayer quality of experience.
Caching of RealSystem G2 and PNA (progressive network Audio) content.	The RealProxy caches all proxied streaming media traffic from RealNetworks servers. RealProxy caches content locally after authentication with origin RealNetworks server.	Significantly reduces inbound bandwidth usage by eliminating redundant file transmissions across the network.
Authentication/accounting	The RealProxy authenticates every content request with the origin RealNetworks server before delivering the cached content to the client.	• Broadcaster retains access to general usage data. • Users are appropriately authenticated. • End users are guaranteed the freshest content.
Aggregate bandwidth thresholds	Setting thresholds caps inbound and outbound bandwidth to the RealProxy.	Provides control over aggregate bandwidth usage within the network and prevents stress on mission-critical applications.
Proxy routing	Ability to tier proxies and manage bandwidth at lower nodes in the network. "Parent" proxies can be chosen based on logical sets of rules on the downstream proxy.	Allows network administrators to proxy route requests, providing an additional level of control.

Note For an overview of streaming media technology, see the "Streaming Media Overview" section on page 13-1. For information on how to enable Windows Media Technologies (WMT) streaming on the Content Engine, see the "Configuring Microsoft Windows Media Player 7.01" section on page 13-2.

Configuring the RealProxy Software

The Content Engine can be configured to accept transparently redirected RTSP requests as well as traditional proxy-style RTSP requests from RealPlayer client software. The redirection of RTSP traffic to the Content Engine media cache is enabled with the Content Engine CLI. The RealProxy software is configured with the RealAdministrator GUI, accessed from the RealProxy page of the Content Engine management GUI.

Detailed configuration, statistics, and reporting of RealProxy status are obtained through the RealAdministrator GUI. Table 14-3 lists RealProxy-related CLI commands.

A RealProxy page has been added to the Content Engine management GUI. To access the RealSystem administrator, click the **Admin** button on the RealProxy page. (See Figure 14-2.) The **Admin** button is active when the RealProxy software is installed and enabled. You will be provided with a default user and password to access this administration page from the Content Engine GUI.

Note You must configure disk space to include mediafs storage with the **disk config** command before you can run RTSP traffic through the Content Engine.

Table 14-3 RealProxy-Related CLI Commands

CLI Command	Function
rtsp proxy incoming *port*	Specifies the port on which to listen for RTSP proxy-style requests.
rtsp proxy l4-switch enable	Enables redirection with a Layer 4 switch, such as a Cisco CSS 11000 series switch.
rtsp proxy media-real enable	Enables the RealProxy.
rtsp proxy media-real ip-address *ipaddress*	Specifies the IP address of the RealProxy.
rtsp proxy media-real license-key *keyword*	Specifies the license keyword to unlock the RealProxy software.
wccp media-cache . . .	Registers the Content Engine for WCCP RTSP redirection services with the router.

Use the **rtsp proxy** global configuration command to configure the Content Engine to accept redirected RTSP traffic from either a Layer 4-enabled switch or a WCCP-enabled router, or to configure the Content Engine as a media proxy to receive RTSP proxy-style requests from RealPlayer clients. RTSP requests not from RealPlayer clients are directed to the specified origin server.

The **wccp media-cache** global configuration command registers the Content Engine with WCCP Version 2-enabled routers that can transparently redirect RTSP traffic to the Content Engine.

The **rtsp proxy l4-switch** global configuration command enables the configuration of Layer 4 switch interoperability for media caching using RTSP.

The RTSP proxy redirector listens to port 554 traffic or any other port configured to listen to this traffic, and if the player is a RealPlayer, it redirects the RTSP request to use the RealProxy for RealMedia traffic. Traffic that is not supported (for instance, QuickTime), is bypassed by the Content Engine.

Statistics

Use the following CLI commands to display RealProxy statistics. In ACNS 4.1 software, only requests and savings statistics are supported. See the "RealProxy Examples" section on page 14-12 for sample outputs of the **show statistics** command.

```
ContentEngine# show statistics mediacache real requests
ContentEngine# show statistics mediacache real savings
```

Note The **mediacache real** statistics relate only to objects transported over RTSP that were requested by a RealPlayer client. Objects transported over HTTP are counted in the HTTP statistics. Streaming objects requested by other clients or transported over other protocols other than MMS bypass the Content Engine.

Enabling Transparent RTSP Proxy Service Using WCCP-Enabled Routers

During transparent caching, the user's network traffic flows through the WCCP-enable router rather than the Content Engine to access streaming media.

Requirements

- Content Engine running ACNS 4.1 or later software
- RealProxy software installed with mediafs partitions mounted
- RealMedia Networks, Inc. license key
- IP addresses of the RealProxy and routers

Procedure

To enable transparent redirection of RTSP traffic to the RealProxy, follow these steps:

Step 1 On the routers running WCCP Version 2, turn the WCCP feature on for the specified service group used to redirect RTSP traffic to the Content Engine. For more information on router commands, see Appendix B, "Web Cache Communication Protocol Version 2."

```
router(config)# ip wccp 80
```

Step 2 Configure the outbound interfaces to the Internet and enter interface configuration mode. In the following example, the outbound interface is the Ethernet 0 device.

```
router(config)# interface Ethernet 0
```

Step 3 Enable WCCP redirection to service group 80 on the interface specified in Step 2.

```
router(interface)# ip wccp 80 redirect out
```

Step 4 Set the WCCP Version 2 parameters on the Content Engine.

In the following example, the WCCP Version 2-enabled routers have the IP addresses 172.16.25.25 and 172.16.24.24.

```
ContentEngine(config)# wccp version 2
ContentEngine(config)# wccp router-list 1 172.16.25.25 172.16.24.24
```

```
ContentEngine(config)# wccp media-cache router-list-num 1
```

Step 5 Set the IP address for the RealProxy. Make sure that the IP address of the RealProxy is visible to the RealPlayers that use it. This step is required before you can enable RealProxy media cache.

```
ContentEngine(config)# rtsp proxy media-real ip-address 172.16.16.16
```

Note This IP address is the external address of the Content Engine in case the Content Engine and the RealPlayer requesting content are separated by a firewall.

Step 6 Enter the RealProxy license number.

```
ContentEngine(config)# rtsp proxy media-real license-key mylicense
```

Step 7 Accept the license agreement.

```
ContentEngine(config)# rtsp proxy media-real accept-license-agreement
```

Step 8 Enable the RealProxy.

```
ContentEngine(config)# rtsp proxy media-real enable
```

Step 9 Save the new configuration.

```
ContentEngine# copy running-config startup-config
```

Step 10 Configure the RealProxy parameters as needed with the RealSystem administrator GUI, shown in Figure 14-3.

Enabling Conventional RTSP Proxy Service

During conventional proxy caching, the user media player is pointed to the Content Engine rather than a WCCP-enabled router to access streaming media.

Requirements

- Content Engine running ACNS 4.1 or later software
- RealProxy software installed with mediafs partitions mounted
- RealMedia Networks, Inc. license key
- IP address of the RealProxy

Procedure

To configure the Content Engine to service RealPlayer clients with the RealProxy on the Content Engine, perform the following steps:

Step 1 Set the IP address for the RealProxy. Make sure that the IP address of the RealProxy is visible to the RealPlayers that use it.

```
ContentEngine(config)# rtsp proxy media-real ip-address 172.16.16.16
```

Step 2 Enter the RealProxy license number shipped with the Content Engine.

```
ContentEngine(config)# rtsp proxy media-real license-key mylicense
```

Step 3 Accept the license agreement.

```
ContentEngine(config)# rtsp proxy media-real accept-license-agreement
```

Step 4 Enable the RealProxy.

```
ContentEngine(config)# rtsp proxy media-real enable
```

Step 5 Configure the Content Engine to listen for RTSP traffic on a specified port. The standard RTSP port is 554.

```
ContentEngine# rtsp proxy incoming 554
```

Step 6 Configure RealPlayer 8.01 clients to use the RealProxy on the Content Engine. (See Figure 14-1.)

a. Open RealPlayer.

b. Choose **View** > **Preferences**.

c. Click the **Proxy** tab.

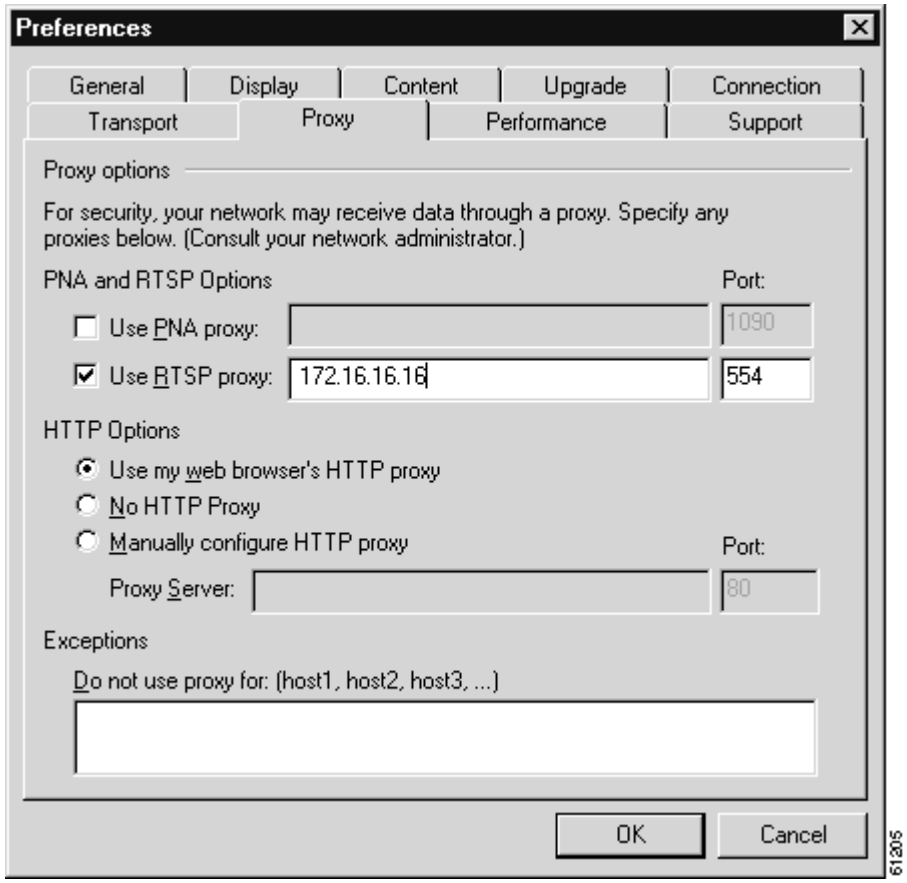
d. Check the **Use RTSP proxy** check box.

e. Enter the IP address of the Content Engine in the Use RTSP proxy field.

f. In the port field, enter the port number that you entered in Step 5.

g. Click **OK**.

Figure 14-1 RealPlayer 8.01 Configured to use Content Engine as Conventional Proxy for RTSP Traffic



Step 7 Configure the RealProxy parameters as needed with the RealSystem administrator GUI.

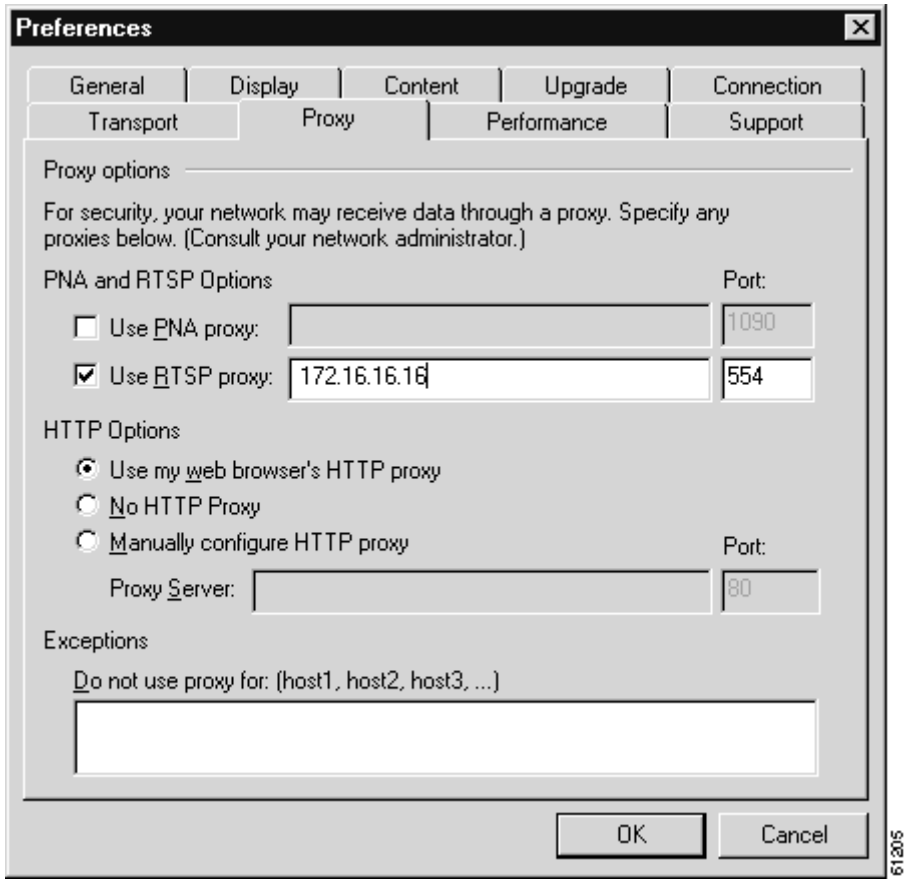
To access the RealSystem administrator GUI, click the **Admin** button on the RealProxy page in the Content Engine Management GUI. (See Figure 14-2.) The **Admin** button is active when the RealProxy software is installed and enabled. Use *admin* as a default user and *diamond* as password to access this administration page from the Content Engine GUI.

Figure 14-2 Content Engine Management GUI—RealProxy Page

The RealSystem Administrator Configuration page is shown in Figure 14-3. For appropriate documentation regarding the RealSystem administrator GUI, click **Documentation** on this page.

Figure 14-3 Content Engine Management GUI—RealSystem 8.01 Administrator Page

Figure 14-4 RealPlayer Configured to Use Content Engine as Conventional Proxy for RTSP Traffic



RealPlayer is now able to use the Content Engine as a RealProxy to fetch streaming objects.

For more information on setting up the RealSystem components used with the RealProxy, refer to the readme “Setting Up RealSystem Server” and “Setting Up RealSystem Player” sections at the following URL:

http://service.real.com/help/library/guides/proxy/readme.htm#5

Step 8 Save the Content Engine configuration to Flash memory.

```
ContentEngine# copy running-config startup-config
```

RealProxy Considerations

Use the following sections to help you configure additional features on the RealProxy.

Disabling RealMedia Caching

The RealProxy player comes with a streaming media cache of its own for the replication of on-demand content. However an administrator may wish to disable caching due to a variety of reasons:

- To collect more accurate data regarding cache hits or misses.
- To prevent delivery of stale data.
- To prevent personalized content by users.

To prevent caching of all material from all servers and the RealProxy, complete the following procedure:

Note The administrator, usernames, and all associated passwords configured on the Content Engine are not synchronized with the RealProxy authentication database. Use *admin* as a default user and *diamond* as password to access the RealSystem administration page from the Content Engine GUI. See Figure 14-2. Use this administrator authentication to configure other RealProxy users.

Step 1 Access the RealSystem Administrator GUI page by clicking the **Admin** button on the RealProxy page of the Content Engine management GUI. (See Figure 14-2. You must enable the RealProxy before you can access the **Admin** button on this page.)

Step 2 Choose **Configure** > **Cache**.

Step 3 In the Enable Caching field, choose **No**.

Step 4 Click **Apply**.

Step 5 Restart the RealProxy by restarting the Content Engine GUI.

Streaming On-Demand Clips and RealProxy

All on-demand clips are automatically available to the Content Engine. If there is content served by your RealServer that you do not want to be cached, see the "Disabling RealMedia Caching" section on page 14-11.

Unicasting, Splitting, Multicasting, and RealProxy

Live clips are not available to caching software; the RealProxy will still proxy the live broadcasts for clients. RealServer acts as a source for live splitting, and the RealProxy acts as a splitter.

RealProxy and Access Control

If a client requests a cached stream, the RealProxy sends the request to the source RealServer for permission before allowing the client to play the stream. If RealServer denies the request, the RealProxy does not allow the client to receive the stream.

You can block a single RealProxy from caching the material served by your RealServer by creating an access control rule from the RealSystem Administrator GUI that prohibits the IP address of that RealProxy from connecting to your RealServer. You can also restrict access to content based on the number of players and bandwidth.

The RealProxy cannot cache live broadcasts, because no actual downloadable file is there to cache. However, the RealProxy includes an ability to “share” live streams among clients and thus reduce the bandwidth required from a transmitter. RealServer and the RealProxy communicate through live splitting; RealServer is preconfigured to act as a transmitter, and the RealProxy is automatically set up to act as a receiver.

A description of the Real-Time Streaming Protocol (RTSP) is available as IETF RFC 2326.

RealProxy Examples

The following example displays request statistics. The statistics reported are the total number of requests served, the total number of cache hits and misses, the total demand pass-through, and the total number of live connections, either by splitting or by live pass-through.

```
ContentEngine# show statistics mediacache real requests
                  Media Cache Statistics - Requests
                                              Total             % of Requests
                              -------------------------------------------------
     Total Received Requests:                    17                        -
            Demand Cache Hit:                    11                     64.7
           Demand Cache Miss:                     6                     35.3
         Demand Pass-Through:                     0                      0.0
                  Live Split:                     0                      0.0
           Live Pass-Through:                     0                      0.0
```

The following example displays savings statistics. In this example, the statistics reported are the total number of requests served, the total number of cache hits and misses, the bytes delivered, and the savings incurred by caching.

```
ContentEngine# show statistics mediacache real savings
              Media Cache Statistics - Savings
                           Requests                          Bytes
          ------------------------------------------------------------
  Total:                      17                        16666028
   Hits:                      11                         3656524
   Miss:                       6                        13009504
Savings:                   64.7 %                            21.9 %
ContentEngine#
```

CHAPTER 15

Configuring the Rules Template

Rules Template

The Rules Template feature allows for requests to be *matched* using an arbitrary number of parameters with an arbitrary number of *policies* applied against the matches. Requests can be matched against regular expressions symbolizing domain names, source IP addresses and network masks, destination IP addresses and network masks, destination port numbers, MIME types, or regular expressions symbolizing a URL.

Policies that can be applied include:

- Blocking the request
- Bypassing authentication for the request
- Resetting the request
- Using a specific object freshness calculation factor
- Not caching an object
- Bypassing an upstream proxy for the request
- Redirecting the request to a different URL
- Revalidating the object with the origin server
- Rewriting the URL
- Selectively caching the object
- Using a specific upstream proxy
- Using a specific server for the request
- Setting TOS/DSCP in response sent to client
- Setting TOS/DSCP in response sent to server

The Rules Template feature is applicable only for HTTP, FTP, and HTTPS traffic and is not applicable for streaming protocols (RTSP, Progressive Networks Audio [PNA], and MMS) implemented in ACNS 4.1 software.

Note To enter a question mark (?) character in a rule regular expression configuration from the command-line interface, use the escape character (\) followed by a question mark (?) character. This prevents the command-line interface from displaying context-sensitive help.

Actions and Patterns

A rule is an action and a pattern. An action is performed on an HTTP request if this request matches the pattern specified in the **rule** command.

An action is something that the Content Engine performs when processing an HTTP request, for instance, blocking the request, using an alternative proxy, and so forth.

A pattern defines the limits of an HTTP request; for instance, a pattern may specify that the source IP address fall in the subnet range 172.16.*.*.

Rules can be dynamically added, displayed, or deleted from the Content Engine. The rules are preserved across reboots because they are written into persistent storage such as NVRAM using the appropriate CLI commands. Only the system resources limit the number of rules that the Content Engine can support. Because rules consume resources, the more rules there are defined, the more Content Engine performance may be affected.

Actions

The Rules Template feature supports the following types of actions:

- **Block**—Blocks this request.
- **DSCP—**Configures the IP ToS/DSCP codepoint field.
  - **client cache-hit**—Configures the IP ToS/DSCP codepoint field for **cache-hit** responses to the client.
  - **client cache-miss**—Configures the IP ToS/DSCP codepoint field for **cache-miss** responses to the client.

  Setting the Type of Service (ToS) or differentiated services code point (DSCP) is called packet marking, allowing you to partition network data into multiple priority levels or types of service. You can set the ToS or DSCP values in IP packets based on a URL match, a file type, a domain, a destination IP address, a source IP address, or a destination port.

  You can set specific ToS or DSCP values for the following:

  - Requests from the Content Engine to the server
  - Responses to the client on cache hit
  - Responses to the client on cache miss

  The ToS or DSCP may be set based on any of the policies matching the **src-ip-address**, **dst-ip-address**, **dst-port-number**, **domain regex**, **url-regex**, or **mime-type regex** options. In addition, you can now configure global ToS or DSCP settings with the **ip dscp** command.

  Note The Rules Template configuration takes precedence over the **ip dscp** command, and the **url-filter** command takes precedence over the **rule** command to the extent that even the rule **no-block** command is executed only if the **url-filter** command has not blocked the request.

- **DSCP server**—Configures the IP ToS/DSCP codepoint field for requests to the origin server.
- **Freshness-factor**—Determines the Time To Live if the request URL matches a specified regular expression. The **refresh** configuration takes priority over **freshness-factor** configurations.
- **No-auth**—Does not authenticate.

Note that the **no-auth** rules result in the display of multiple authentication windows in the following scenario:

- When the main page (for example, index.htm) is excluded from proxy authentication by using **no-auth** rules
- When the user entry is not already included in the Content Engine authentication cache
- When the index.htm page contains objects belonging to different domains

To avoid multiple authentication windows, configure the **http avoid-multiple-auth-prompts** command in global configuration mode. Once it is configured, check the configuration with the **show http avoid-multiple-auth-prompts** command as shown the following example.

```
ContentEngine# show http avoid-multiple-auth-prompts
Avoiding multiple authentication prompts due to no-auth rules is enabled
```

Note The command in the example is hidden, because it is applicable only to this specific scenario.

- **No-cache**—Does not cache this object. If both **no-cache** and **selective-cache** actions are matched, **no-cache** takes precedence.
- **No-proxy**—For a cache miss, does not use the configured upstream proxy but rather contacts the server directly.
- **Redirect**—Redirects the original request to a specified URL. Redirect is relevant to the RADIUS server only if the RADIUS server has been configured for **redirect**.
- **Refresh**—For a cache hit, forces an object freshness check with the server.
- **Reset**—Issues a TCP RST. This reset request in useful when resetting Code Red or Nimda virus requests.
- **Rewrite**—Rewrites the original request as a specified URL. The Content Engine searches for the rewritten URL in cache, and then on a cache miss, fetches the rewritten URL and returns the object transparently to the client. It is preferable to use a **redirect** rule rather than **rewrite** because of possible performance impacts.

  The URL rewrite could change the domain name of the URL, which necessitates a DNS lookup to find the destination (dst) IP address of the new rewritten server to which the request must be sent. The original dst IP address derived from the WCCP redirect packet cannot be used.
- **Selective-cache**—Caches this object only if it is a match and is allowed to be cached by HTTP. If one or more rules specify this action, an object is cached if and only if it matches at least one of the **selective-cache** rules and passes every other caching restriction such as the object-size check and the no-cache-on-authenticated-object check. If the object does not match any of the **selective-cache** rules, the object is *not* cached.
- **Use-proxy**—For a cache miss, uses a specific upstream proxy. Specify the upstream proxy IP address (or domain name) and port number. If both **no-proxy** and **use-proxy** are matched, **no-proxy** takes precedence.
- **Use-proxy-failover**—Supports failover capability. The **use-proxy-failover** rule is similar to the **use-proxy** rule, except that if the connection attempt on the configured outgoing proxy fails, the requests fail over to the outgoing proxies configured with the HTTP proxy outgoing configuration. The rule requests use the HTTP proxy outgoing **origin-server** option, if it is configured. The **use-proxy-failover** rule takes precedence over the **use-proxy** rule. If both **no-proxy** and **use-proxy-failover** are matched, **no-proxy** takes precedence.

The HTTP failover does not apply if the destination is on the exclude list. When in transparent mode, the setting for the original proxy takes precedence.

- **Use-server**—Sends server-style HTTP requests from the Content Engine to the specified IP address and port on a cache miss.

  Among **use-server**, **no-proxy**, and **use-proxy** rules, the **use-server** rule is the first one to be checked. If it results in a rule miss, **no-proxy** and **use-proxy** rules are executed in succession (**use-proxy** is not checked if a **no-proxy** rule matches).

  If a rule is configured with a fully qualified domain name (FQDN) and a request is received with the partial domain name in transparent mode, the rule fails to be executed, as the FQDN is not in the request URL. In transparent mode, if a request is destined for a particular domain (for which a domain rule is configured) and does not contain the Host header, the rule pattern match fails.

Patterns

The Rules Template feature supports the following types of patterns.

- **Domain**—Matches the domain name in the URL or the Host header against a regular expression. For example, ".*ibm.*" matches any domain name that contains the "ibm" substring. "\.foo\.com$" matches any domain name that ends with the ".foo.com" substring.

  Note In regular expression syntax, the dollar sign "$" metacharacter directs that a match is made only when the pattern is found at the end of a line.

- **Dst-ip**—Matches the request's destination IP address and netmask. Specify an IP address and a netmask. In proxy mode, the Content Engine does a DNS lookup to resolve the destination IP address of the HTTP request, making the response time longer, and possibly negating the benefit of setting a **dst-ip** rule. When an outgoing proxy is configured, cache miss requests are forwarded by the Content Engine to the outgoing proxy without examination of the destination server IP address, making the **dst-ip** rule unenforceable on the first Content Engine.
- **Dst-port**—Matches the request's destination port number. Specify a port number.
- **Mime-type**—Matches the MIME type of the response. Specify a MIME type string, for example, "image/gif," as defined in RFC 2046 (http://www.faqs.org/rfcs/rfc2046.html). The administrator can specify a substring, for example, "java" and have it apply to all MIME types with the "java" substring, such as "application/x-javascript."
- **Src-ip**—Matches the request's source IP address and netmask. Specify an IP address and a netmask.
- **URL-regex**—Matches the URL against a regular expression. The match is case insensitive. Specify a regular expression whose syntax can be found at the following URL:

  http://yenta.www.media.mit.edu/projects/Yenta/Releases/Documentation/regex-0.12/.

- **URL-regsub**—For the **rewrite** and **redirect** actions, matches the URL against a regular expression to form a new URL per pattern substitution specification. The match is case insensitive. The valid substitution index range is from 1 to 9.
- **Header-field**—Requests header field pattern.

  Request header field patterns **referer**, **request-line**, and **user-agent** are supported for actions **block**, **reset**, **redirect**, and **rewrite**. The **referer** pattern matches against the Referer header in the request, **request-line** pattern matches against the first line of the request, and **user-agent** pattern matches against the User-Agent header in the request.

Rules Template Processing Considerations

There is a predefined order of execution among the actions and the patterns. In other words, a group of rules with the same action will always be executed either before or after another group of rules with a different action. See the "Rule Action Execution Order" section on page 15-5 for the order of rule action execution.This order is not affected by the order in which the rules are entered using CLI commands.

Among the rules of the same action, there is a predefined execution order among the rules pattern. This means that within a group of rules of the same action, one group of rules with the same pattern will always be executed either before or after another group of rules with a different pattern. See the "Rule Pattern Execution Order" section on page 15-6 for the order of rule pattern execution. This order is not affected by the order in which the rules are entered using CLI commands.

Rule Action Execution Order

The order of rule action execution is as follows:

1. **No-Auth**—Before authentication using RADIUS/LDAP/NTLM
2. **Reset**—Before cache lookup
3. **Block**—Before cache lookup
4. **Redirect**—Before cache lookup
5. **Rewrite**—Before cache lookup
6. **Refresh**—On cache hit
7. **Freshness-factor**—On cache hit
8. **Use-server**—On cache miss
9. **No-proxy**—On cache miss
10. **Use-proxy-failover**—On cache miss
11. **Use-proxy**—On cache miss
12. **TOS/DSCP server**—On cache miss
13. **TOS/DSCP client**
14. **No-cache**—On cache miss
15. **Selective-cache**—On cache miss

Note The commands **rule no-proxy**, **rule use-proxy-failover**, and **rule use-proxy** take precedence over **https proxy outgoing**, **http proxy outgoing**, and **ftp proxy outgoing** commands.

During a request using the rules template CLI commands, rule actions 1–4 use the original URL request for pattern matches. After a URl rewrite (rule action 5), rule actions 6–-15 use the transformed URL for rule executions.

The commands **rule reset**, **rule block**, **rule rewrite**, and **rule redirect** support the following additional patterns for rule templates request:

- request-line—Matches first line.
- referer—Matches referer header.
- user-agent—Matches user-agent header.

Rule Pattern Execution Order

The order of rule pattern execution is as follows:

1. **Dst-port**—Destination port check.
2. **Src-ip**—Source IP address check.
3. **URL-regex**—URL regex check.
4. **Domain**—Domain rule check.
5. **Dst-ip**—Destination IP address check.
6. **MIME-type**—Mime-type regex check.

Note Because the MIME type exists only in the response, only the actions **freshness-factor**, **refresh**, **no-cache**, and **selective-cache** apply to a rule of MIME type.

A search for a rule match with the remaining pattern will not be performed if a match has already been found. For instance, if a match for the **rule block** action is found with a **URL-regex** request, then the remaining patterns **Domain**, **Dst-ip**, or **MIME-type** are not searched.

Rules are ORed together. Multiple rules may all match a request; then all actions are taken, with precedence among conflicting actions. Each rule contains one pattern; patterns cannot be ANDed together. In future releases, ANDed patterns may be supported.

It is possible to circumvent some rules. For example, to circumvent a rule with the **domain** pattern, enter the web server IP address instead of the domain name in the browser. A rule may have unintended effects. For instance, a rule with the **domain** pattern specified as “ibm” that is intended to match “www.ibm.com” can also match domain names like www.ribman.com.

A **src-ip** rule may not apply as intended to requests that are received by a Content Engine from another proxy or Content Engine because the original client IP address is in an X-forwarded-for header. This means that the original request source IP address is transparently replaced with the sending Content Engine IP address to another proxy or Content Engine and then to the origin server.

If a rule pattern match occurs, then the rest of the patterns are not searched. If the server has already marked an object as non-cacheable, **no-cache** rules are not checked at all, since the server already recognizes that this object is not cached. Any **no-cache** rule checks are performed only for cacheable requests.

Order of Execution Among Rules of Same Action and Same Pattern

Among the rules of the same action and the same pattern, the order of execution of rules is in the reverse order in which the rules are entered. For instance, if the **use-proxy** commands are entered in the following order:

use-proxy 1.2.3.4 abc.abc.com

use-proxy 2.3.4.5 *.abc.com

then a request to abc.abc.com is sent to proxy 2.3.4.5 because the **use-proxy 2.3.4.5 *.abc.com** command is entered last and evaluated first. However, if the same commands are entered in a reverse order as follows:

use-proxy 2.3.4.5 *.abc.com

use-proxy 1.2.3.4 abc.abc.com

then a request to abc.abc.com is sent to proxy 1.2.3.4, as the **use-proxy 1.2.3.4 abc.abc.com** command is entered last and evaluated first.

Examples

In the following example the **rule block** command (action) blocks all domains that contain .foo.com in the URL request using the domain \.foo.com pattern.

```
ContentEngine(config)# rule block domain \.foo.com ?
 LINE    <cr>
```

Multiple patterns can be entered on the same line. If any of them matches the incoming HTTP request, the corresponding action is taken. In the following example the **rule block** command (action) block all domains that contain .foo.com and bar.com in the URL request pattern.

```
ContentEngine(config)# rule block domain \.foo.com bar.com
ContentEngine(config)#
```

The following example prevents caching of requests that match a URL request which contains the *cgi-bin* string.

```
ContentEngine(config)# rule no-cache url-regex \.*cgi-bin.*
ContentEngine(config)#
```

Most actions do not have any parameters, as in the preceding examples. Exceptions to this are **use-server**, **freshness-factor**, and **use-proxy**, as in the following example.

```
ContentEngine(config)# rule use-proxy CE.foo.com 8080 url-regex .*\.jpg$ .*\.gif$ .*\.pdf$
ContentEngine(config)#
```

To delete rules, use **no** in front of the rule creation command.

```
ContentEngine(config)# no rule block url-regex .*\.jpg$ .*\.gif$ .*\.pdf$
```

The following example sets the freshness factor for MIME-type images.

```
ContentEngine(config)# rule freshness-factor 75 mime-type image/.*
```

The following example redirects a request for *old-domain-name*, which has been changed to *new-domain-name*.

```
ContentEngine(config)# rule redirect url-regsub http://old-domain-name/
http://new-domain-name/
```

The following example redirects requests from an IETF site to one that is locally mirrored:

```
ContentEngine(config)# rule redirect url-regsub http://www.ietf.org/rfc/(.*)
http://wwwin-eng.cisco.com/RFC/RFC/\1
```

For the preceding example, if the request URL is http://www.ietf.org/rfc/rfc1111.txt, the Content Engine rewrites the URL as http://wwwin-eng.cisco.com/RFC/RFC/rfc1111.txt and sends a 302 Temporary Redirect response with the rewritten URL in the Location header to the client. The browser automatically initiates a request to the rewritten URL.

The following example redirects all requests for linux.org to a local server in India that is closer to where the Content Engine is located:

```
ContentEngine(config)# rule redirect url-regsub http://linux.org/(.*)
http://linux.org.in/\1
```

The following example rewrites requests from an IETF site to one that is locally mirrored:

```
ContentEngine(config)# rule rewrite url-regsub http://www.ietf.org/rfc/.*
http://wwwin-eng.cisco.com/RFC/$1
```

The following example replaces the string internal.domain.com in the URL request to dummy while forwarding the request to the server.

```
ContentEngine(config)# rule rewrite header-field referer internal.domain.com dummy
ContentEngine(config)#
```

If an empty string is given as a replacement pattern, then the Referer header is stripped. The same usage applies to the user-agent pattern.

CHAPTER 16

Miscellaneous Features

Configuring the Content Engine as a Content Routing Agent

ACNS 4.1 software adds support that allows you to configure a Content Engine as a content routing agent. A content routing agent is used in conjunction with the Cisco Content Router 4430 (CR-4430) running Content Routing software. The Content Router redirects a user request to the "closest" (best) replicated-content site, based on network delay and other parameters, using a technology called boomerang.

Before you configure the Content Engine as a content routing agent, configure the Content Router as described in the *Cisco Content Routing Software Configuration Guide and Command Reference.*

Use the **boomerang dns enable** command to enable content routing software on a Content Engine that you want to configure as a content routing agent. Use the **boomerang dns domain** command to configure the Content Engine as a content routing agent for a specified domain and to enter the domain configuration mode to establish operating parameters for the specified domain name.

Boomerang agents support multiple domains, where each agent domain may be associated with a different boomerang server. Other than memory limits, there are no limits to the number of domains supported on the agent. For more information on the boomerang agent, refer to the current release of the *Cisco Content Routing Software Configuration and Command Reference.*

Use the **boomerang dns domain** *domain-name* **alias** *alias-name* command to set alternate boomerang domain names that share the same operating parameters. If you are using the Content Engine as a content routing agent, use this command on both the Content Router and the Content Engine to establish an alternative name for a domain.

Note Corresponding alias domain names must also be configured on the boomerang server. Each client domain can be associated with a different boomerang server.

If the Content Engine is not used to serve web pages, use the **content-server** *ip-address* **file** *filename* option to specify the address of the cache to be used. If it wins the race, the content server is the local web cache or mirror cache that serves content for the requesting web client that initiated the DNS race. The boomerang agent probes the content server periodically to ensure that it is running and able to serve web pages. The probe consists of an HTTP GET request for the configured filename. A response of 200 OK indicates that the content server is running. If a filename is not given, attempts to connect are made only through port 80.

If you are using the Content Engine as a content routing agent, use the **boomerang dns domain** *domain-name* **dns-ttl** command to specify the DNS TTL value contained in the DNS response generated by the agent. In general, a lower DNS TTL value ensures more recent content, whereas a higher DNS

TTL value reduces the Content Router load. The higher the DNS TTL value, the less the load on the Content Router. A lower value means an increased Content Router load, but also means that the addresses of Content Engines that won DNS races are used for a shorter length of time in the annealing process. For example, if the DNS TTL is set at 60 seconds, a DNS server returns to the Content Router to look up a domain name no more than once a minute. In other words, the name server uses the winning Content Engine address for 60 seconds before consulting the Content Router again. Use **no dns-ttl** to reset the delay to its default value.

Note A **dns-ttl** command entered on a Content Engine boomerang agent overrides a **dns-ttl** command entered on the Content Router.

The number of hops to live value of the DNS response is generated by the client. The value specified by the **hops** option overrides the value specified by the boomerang server.

The **key** shared secret string specifies the secret that is matched against the secret contained in the packets sent by the server. The shared secret configured on the client domain needs to be the same as the secret configured on the server.

Use the **origin-server** *ip-address* subcommand if the Content Engine is used as a cache rather than mirror site. If the web cache does not have the requested content, or there is a cache miss, it must get the content from the origin server and cache it for future requests. Because the Content Engine web cache does not have the IP address of the origin server, this subcommand must be set to provide the IP address to get content from the origin server.

Caution A Content Engine can be used simultaneously for transparent caching and as a content routing agent. Make sure that you use the **boomerang origin-server** command to configure the Content Engine to cache requests.

The **boomerang log-races enable** command enables logging of the DNS IP address resolution timing results between the boomerang server and the agent. To disable the command, enter the **no boomerang log-races** command.

To delete a domain, enter the **no boomerang** command. It is not necessary to enter arguments and variables before deleting the current domain name.

Examples

In the following example, assume that a domain is named www.foobar.com. It is given the domain name www.foobar.com on the Content Router.

```
Console(config)# boomerang dns enable
Console(config)# boomerang dns domain www.foobar.com
```

In the following example, assume that a domain is named www.foobar.com. It is given the alias www.foobar.net on the Content Router.

```
Console(config-boomerang)# alias www.foobar.net
```

When configuring www.foobar.com on the agent, enter the alias on the Content Engine as follows.

```
Console(config-bomerang)# alias www.foobar.net
```

Browser Autoconfiguration

ACNS 4.1 software provides support for proxy automatic configuration files (.pac files). A browser obtains proxy IP address and port configuration information from the proxy automatic configuration file when the browser's autoconfiguration URL field is configured with the Content Engine IP address, incoming port number, file directory, and .pac filename.

The **proxy-auto-config download** EXEC command downloads an automatic configuration file from an FTP server to the present working directory of the Content Engine. To enable the proxy automatic configuration file feature, enter the **proxy-auto-config enable** global configuration command. Each time you download a new automatic configuration file to the Content Engine, enter a **no proxy-auto-config enable** and **proxy-auto-config enable** command.

Note You must configure disks /local1 or /local2 as a sysfs volume before downloading the autoconfiguration file to either of these two disk locations.

The autoconfiguration feature is supported by the Microsoft Internet Explorer and Netscape browsers. The browser must be manually configured for automatic proxy configuration.

This example demonstrates how to download an automatic configuration file from an FTP server to the Content Engine:

```
ContentEngine# proxy-auto-config download 172.16.10.10 remotedirname theproxyfile.pac
```

This example enables browser autoconfiguration on the Content Engine:

```
ContentEngine(config)# proxy-auto-config enable
```

This example demonstrates the URL syntax to enter in the browser's automatic proxy configuration URL field:

http://ContentEngine-IPaddress:portnumber/theproxyfile.pac

Note Use the port number specified by the **http proxy incoming *portnumber*** global configuration command for configuring proxy incoming ports. For instance, if port 8080 is specified with the **http proxy incoming 8080** command, then use 8080 as your port number in the example shown.

Configuring Healing Mode

When a Content Engine is added to an existing WCCP Version 2 cache group (cluster), it can receive requests for content that was formerly served by another cache in the cluster. This event is termed a "near-miss," because if the request had been sent to the former Content Engine, it would have been a cache hit. A near-miss lowers the overall cache hit rate of the Content Engine cluster.

The Content Engine in healing mode is called a healing client. The caches in the cluster that respond to healing client requests are called healing servers. Healing mode allows the newly added Content Engine to query and obtain cache objects from all other caches in the cluster on a cache miss event. If the object is found in the cluster, one of the healing servers sends back a response saying that it has the object in its cache and that the healing client can request an object from it. If the object is not found in the cluster, the Content Engine processes the request through the outgoing proxy or origin server.

Note Healing mode is only invoked on a healing client when the request is transparently redirected to the Content Engine. Healing mode is not invoked when the request is sent to the Content Engine in proxy mode.

The **http cluster** command modifies the healing mode parameters. The **http cluster http-port** command specifies the port number over which requests from the healing Content Engine are sent to other Content Engines in the cluster.

Note The default port number is 80. If you choose to configure a port other than the default, you must ensure that the port configured matches the port specified in the **http proxy incoming** command on healing servers in the farm. Otherwise, the healing client is not able to retrieve objects from the healing servers.

The **http cluster misses** command specifies the maximum number of misses that the healing Content Engine can receive from the cluster after the last healing mode hit response until the healing process is disabled. The **http cluster max-delay** command specifies the maximum time interval in seconds for which a healing Content Engine waits for a healing response from the cluster before considering the healing request a miss.

To enable the healing client, you should, at the least, configure the **max-delay** and **misses** options. The default port number for **http-port** is 80; therefore, if you use the default port, you do not have to configure **http-port**.

To disable the healing client, you should, at the least, configure either **misses** or **max-delay** to 0, or you can use the **no** form of the command:

- **http cluster misses 0**
- **no http cluster misses**
- **http max-delay 0**
- **no http cluster max-delay**

Examples

This example enables the healing mode feature by setting the HTTP port for forwarding HTTP requests to a healing server, setting the maximum delay to wait for a response from the cluster in seconds before considering the healing request a miss, and setting the maximum number of misses that the healing Content Engine can receive from the cluster before healing mode is disabled at the healing client.

```
Console(config)# http cluster http-port 8080
Console(config)# http cluster max-delay 5
Console(config)# http cluster misses 5
```

In this example, the **show statistics http cluster** command displays the statistics of the healing client and the healing server. The **clear statistics http cluster** command resets the healing mode statistics:

```
Console(config)# show statistics http cluster
Healing mode max attempts              = 0
Healing mode max latency               = 0
Healing mode current cumulative misses = 0

Healing mode client statistics
------------------------------
Client Requests  Sent     = 0
```

```
Client Responses Received = 0
Client Responses Hit      = 0
Client Responses Miss     = 0
Client Responses Error    = 0
Client Responses Timeout  = 0

Healing mode server statistics
------------------------------
Server Requests Received  = 0
Server Responses Sent     = 0
Server Responses Hit      = 0
Server Responses Miss     = 0
Server Responses Error    = 0
```

The **show http cluster** command displays **max-delay**, **misses**, and HTTP port values. In the first example, the values are set to 0 and the healing client is disabled.

```
Console(config)# show http cluster
Healing client is disabled
Timeout for responses = 0 seconds
Max number of misses allowed before stop healing mode = 0
Http-port to forward http request to healing server is not configured
```

Content Preloading

ACNS 4.1 software can read a file of URLs and download the specified content to the Content Engine. This preloading can be scheduled with the **pre-load schedule** global configuration command, or triggered immediately with the **pre-load force** EXEC command.

A list file of URLs to be preloaded (URL list) is maintained by the administrator. If the administrator wants to preload authenticated content to the Content Engine, the URL list file entry must be written as follows:

```
http://username:passwrod@www.authenticatedsite.com/ <depth level>
```

The URL list must be created on a remote system. This list is accessed from this remote system with a frequency set by the **pre-load schedule** command. In other words, whenever a pre-loading is scheduled to run, this URL list is fetched from the remote system.

Note The URL list is not cached by the Content Engine. It must be fetched from the remote system every time content preloading is scheduled.

The ftp/http URL location of the URL list file can be specified in the **pre-load url-list-file** option. The URL list can also be transferred to a sysfs volume on the Content Engine, in which case, the path of the URL list is specified by the **pre-load url-list-file** option. The URL list is a list of URLs, each with an optional depth parameter. The depth parameter specifies how many levels down the preloading will be performed. For example, http://www.espn.com 3 means download http://www.espn.com and all content three levels deep. If the depth level is not specified, then the pre-load depth level default is used. The URLs are delimited with a carriage return as follows:

```
<cr>
. . .
http://www.cnn.com  3 <cr>
ftp://ftp.lehigh.edu/  2 <cr>
http://www.yahoo.com <cr>
. . .
```

```
<cr>
```

Use the **pre-load schedule** command to specify the time intervals at which the preload event executes, or use the **pre-load force** EXEC command to launch a preload event at any time.

Note If content preloading does not complete before the scheduled end time you must restart the preloading process to capture content intended.

If the content to be preloaded is already in the Content Engine, then the Content Engine revalidates the freshness of the stored copy.

A preload request process (wget) is spawned for every URL in the list. These processes operate concurrently. Use the **pre-load concurrent-requests** option to configure the maximum number of preload processes to run at the same time. If the number of URLs in the URL list file is less than the number of specified concurrent requests, then the lesser number is active.

All configured HTTP parameters and rules apply to the preloaded objects.

This example enables the preload feature:

```
ContentEngine(config)# pre-load enable
```

This example specifies the path name of the preload URL list file:

```
ContentEngine(config)# pre-load url-list-file /local1/myurllist
```

This example specifies the depth level for URL retrieval at 4:

```
ContentEngine(config)# pre-load depth-level-default 4
```

This example creates a filter for the objects to be excluded:

```
ContentEngine(config)# pre-load no-fetch suffix .mil .su .ca
```

This example specifies a filter for the domain to be fetched:

```
ContentEngine(config)# pre-load fetch domain cisco.com
```

This example specifies that other domains in an HTML page should be traversed (by default, other domains in an HTML page are not traversed):

```
ContentEngine(config)# pre-load traverse-other-domains
```

This example specifies the maximum number of concurrent connections:

```
ContentEngine(config)# pre-load concurrent-requests 5
```

This example specifies a daily interval for scheduling the preload:

```
ContentEngine(config)# pre-load schedule every-day start-time 01:00 end-time 02:00
```

This example specifies an hourly interval for scheduling the preload:

```
ContentEngine(config)# pre-load schedule every-hour start-time 8 end-time 20
```

The **pre-load schedule every-week** option permits configuring a preload on more than one day of the week. This example specifies a biweekly interval for scheduling the preload:

```
ContentEngine#(config)# pre-load schedule every-week Sun Wed start-time 01:00 end-time
06:00
```

The default start time for the preloading operation is 00:00 (that is, the start of the day). If the end time is not specified, the preload operation is completed after all the objects have been downloaded.

The following are examples of preload-related **show** commands:

```
ContentEngine# show statistics pre-load
Statistics of last Preloading operation
---------------------------------------

Preloading was initiated by cron.
Preloading started at Sat Feb 10 21:00:01 2001
Preloading ended   at Sun Feb 11 00:45:25 2001

Number of invalid entries in URL list file  =          0
Total number of preloaded objects           =      44178
Total number of preloaded bytes             =  895723727


ContentEngine# show pre-load
Preloading is enabled
Number of concurrent sessions: 10
Depth level: 3
URL List File: /local1/preload/preload.txt
Preload will not traverse other domains.
Fetch Domains:
Fetch Suffix:
Fetch Directory:
No-fetch Domain:
No-Fetch Suffix:
No-Fetch Directory:
Scheduling on:
Sunday
        Start Time: 00:00
        End Time  : Till completion
```

Web Cache Communication Protocol Version 1

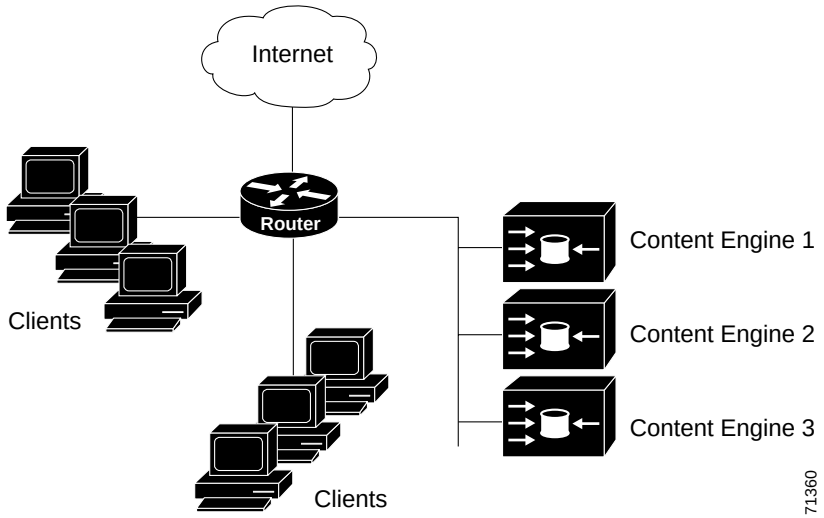
This appendix describes the Web Cache Communication Protocol (WCCP) Version 1 feature. (WCCP is also known as Web Cache Control Protocol and Web Cache Coordination Protocol.) This appendix includes information on the benefits of this feature, and other information you may need to work with WCCP Version 1. This appendix includes the following sections:

- Feature Overview, page A-1
- Related Documents, page A-3
- Prerequisites, page A-3
- Configuration Tasks, page A-3
- Configuration Example, page A-4
- Command Reference, page A-5
- Debug Commands, page A-19

Feature Overview

The WCCP feature allows you to use a Content Engine to handle web traffic, thus reducing transmission costs and downloading time. This traffic includes user requests to view pages and graphics on World Wide Web servers, whether internal or external to your network, and the replies to those requests. Figure A-1 shows a sample WCCP network configuration.

Figure A-1 Content Engine Network Configuration Using WCCP Version 1



When a user (client) requests a page from a web server (located on the Internet, in this case), the router sends the request to a Content Engine (Content Engine 1, Content Engine 2, or Content Engine 3). If the Content Engine has a copy of the requested page in storage, the engine sends the user that page. Otherwise, the engine gets the requested page and the objects on that page from the web server, stores a copy of the page and its objects (caches them), and forwards the page and objects to the user.

WCCP transparently redirects HTTP requests from the intended server to a Content Engine. End users do not know that the page came from the Content Engine rather than from the originally requested web server.

Benefits

Web caches reduce transmissions costs and the amount of time required to download web files. If a client requests a web page that is already cached, the request and data only have to travel between the Content Engine and the client. Without a web cache, the request and reply must travel over the Internet or wide-area network. Cached pages can be loaded faster than noncached pages and do not have to be transmitted from the Internet to your network.

Cisco IOS support of WCCP provides a transparent web cache solution. Users can benefit from web proxy caches without having to configure clients to contact a specific proxy server in order to access web resources. Many web proxy caches require clients to access web resources through a specific proxy web server rather than using the originally requested web server URL. With WCCP, the clients send web requests to the desired web server URL. Cisco IOS routers intelligently intercept HTTP requests and transparently redirect them to a Content Engine.

Redirection Process

When a WCCP-enabled router receives an IP packet, the router determines whether the packet is a request that should be directed to a Content Engine. The router looks for TCP as the protocol field in the IP header and for 80 as the destination port in the TCP header. If the packet meets these criteria, it is redirected to a Content Engine.

Through communication with the Content Engines, the routers running WCCP are aware of available Content Engines.

Related Documents

Refer to the following Cisco IOS documentation for further information on WCCP.

- *Cisco IOS Configuration Fundamentals Configuration Guide*
- *Cisco IOS Configuration Fundamentals Command Reference*

Prerequisites

To use a WCCP-enabled router, an IP address must be configured on the interface connected to the Internet, and the interface must be connected to the Content Engine.

Note A Content Engine and a WCCP-enabled router cannot be separated by a firewall. The firewall handles only packet traffic toward the origin web server and does not handle packet traffic sent to the client by the Content Engine on behalf of the server.

Configuration Tasks

Configuring the Content Engine

To use WCCP, the Content Engine must be properly configured. Keep these important points in mind:

- The IP address of the router must be configured as the home router for the Content Engine.
- Versions of software on the Content Engines must be compatible with those installed on the router.
- The Content Engines must not have their packets encrypted or compressed and should be part of the “inside” Network Address Translation if one is present.
- Placing a Content Engine beyond a web cache redirect-enabled interface and along the route to the server will not cause the IP route cache to be populated with an entry.

Configuring the Router

To configure WCCP on the router, you must perform the following tasks. The first task is required, whereas the second is optional.

- Enabling WCCP on the Router
- Monitoring WCCP

Enabling WCCP on the Router

To enable an interface to redirect web traffic to the Content Engine using WCCP Version 1, perform the following tasks beginning in global configuration mode:

	Command	Purpose
Step 1	**ip wccp version 1**	Enables the router to use WCCP version 1.
Step 2	**ip wccp redirect-list** {*number* | *name*}	(Optional) Specifies the redirect access list. Only packets that match this access list are redirected. If you do not configure this command, all web-based packets are redirected.
Step 3	**interface** *type number*	Enters interface configuration mode.
Step 4	**ip web-cache redirect**	Configures the interface connected to the Internet to redirect web traffic to the Content Engine.
Step 5	**ip route-cache same-interface**	(Optional) If the client and a Content Engine are located on the same network, configures the router to use the fast switching path on the interface.
Step 6	**end**	Exits configuration mode.
Step 7	**copy running-config startup-config**	Saves the configuration.

Monitoring WCCP Version 1

To monitor WCCP, perform any of the following tasks in EXEC mode:

Command	Purpose
show ip wccp	Displays global WCCP statistics.
show ip wccp web-caches	Displays information about all known Content Engines.
show ip interface	Shows whether web cache redirecting is enabled on an interface.

The **show ip wccp** and **show ip wccp web-caches** commands display a count of the number of packets redirected. Use the **clear ip wccp** EXEC command to clear this counter.

Configuration Example

The following example configures a router to support WCCP Version 1 and to redirect web-related packets from Ethernet interface 0 to the Content Engine:

```
Router# configure terminal
Router(config)# ip wccp enable
Router(config)# interface Ethernet 0
Router(config-if)# ip web-cache redirect
Router(config-if)# end
```

```
Router#
%SYS-5-CONFIG_I: Configured from console by console.
Router# copy running-config startup-config
```

After the router has been configured, use the **show ip wccp web-caches** command to verify that WCCP is enabled and aware of Content Engines. In this example, the **show ip wccp web-caches** command is entered immediately after the router has been configured. After a few seconds, the Content Engine becomes usable, as seen in the second output.

```
Router# show ip wccp web-caches

WCCP Web-Cache information:
        IP Address:             192.168.51.102
        Protocol Version:       1.0
        State:                  NOT Usable
        Initial Hash Info:      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                                FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
        Assigned Hash Info:     00000000000000000000000000000000
                                00000000000000000000000000000000
        Hash Allotment:         0 (0.00%)
        Packets Redirected:     0
        Connect Time:           00:00:06

Router# show ip wccp web-caches

WCCP Web-Cache information:
        IP Address              192.168.51.102
        Protocol Version:       0.3
        State:                  Usable
        Initial Hash Info:      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                                FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
        Assigned Hash Info:     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                                FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
        Hash Allotment:         256 (100.00%)
        Packets Redirected:     0
        Connect Time:           00:00:31
```

Command Reference

This section documents new or modified commands. All other commands used with this feature are documented in the Cisco IOS Release 11.1 or Release 11.2 command references.

- **clear ip wccp**
- **ip wccp enable**
- **ip wccp redirect-list**
- **ip web-cache redirect**
- **show ip interface**
- **show ip wccp**
- **show ip wccp web-caches**

clear ip wccp

To clear the counter for packets redirected by the Web Cache Communication Protocol, use the **clear ip wccp** EXEC command.

clear ip wccp

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

EXEC

Command History

This command first appeared in Cisco IOS Release 11.2 P and IOS Release 11.1 CA.

Usage Guidelines

The "Packets Redirected" count is displayed by the **show ip wccp** and **show ip wccp web-caches** commands.

Examples

The following example shows output from the **show ip wccp web-caches** command before and after the **clear ip wccp** command is used:

```
Router# show ip wccp web-caches

WCCP Web-Cache information:
        IP Address:             192.168.88.11
        Protocol Version:       1.0
        State:                  Usable
        Initial Hash Info:      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
                                AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        Assigned Hash Info:     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                                FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
        Hash Allotment:         256 (100.00%)
        Packets Redirected:     21345
        Connect Time:           00:13:46

Router# clear ip wccp
Router# show ip wccp web-caches

WCCP Web-Cache information:
        IP Address:             192.168.88.11
        Protocol Version:       1.0
        State:                  Usable
        Initial Hash Info:      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
                                AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        Assigned Hash Info:     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                                FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
        Hash Allotment:         256 (100.00%)
        Packets Redirected:     0
        Connect Time:           00:13:46
```

Related Commands **show ip wccp**

show ip wccp web-caches

ip wccp enable

To enable the router to support WCCP, use the **ip wccp enable** global configuration command. The **no** form of this command disables support for WCCP.

ip wccp enable

no ip wccp enable

Syntax Description

This command has no arguments or keywords.

Defaults

WCCP is disabled on the router.

Command Modes

Global configuration

Command History

This command first appeared in Cisco IOS Release 11.2 P and Release 11.1 CA.

Usage Guidelines

This command and the **ip web-cache redirect** interface configuration command are the only commands required to start redirecting requests to the Content Engine using WCCP. To see whether WCCP is enabled on the router, use the **show ip wccp** command.

When this command is enabled but the **ip web-cache redirect** command is disabled, the router is aware of Content Engines but does not use them.

Use the **ip wccp redirect-list** command to limit the redirection of packets to those matching an access list.

Examples

The following example configures a router to support WCCP and redirects web-related packets from Ethernet interface 0 to the Content Engine:

```
Router# configure terminal
Router(config)# ip wccp enable
Router(config)# interface Ethernet 0
Router(config-if)# ip web-cache redirect
Router(config-if)# end
Router#
%SYS-5-CONFIG_I: Configured from console by console.
```

Related Commands

ip wccp redirect-list

ip web-cache redirect

show ip wccp

show ip wccp web-caches

ip wccp redirect-list

To specify which packets are redirected to a Content Engine, use the **ip wccp redirect-list** global configuration command. The **no** form of this command enables redirection of all packets.

ip wccp redirect-list {*number* | *name*}

Syntax Description

number	Standard or extended IP access list number from 1 to 199.
name	Standard or extended IP access list name. This argument is only available in Cisco IOS Release 11.2 P.

Defaults

All HTTP packets are redirected to the Content Engine.

Command Modes

Global configuration

Command History

This command first appeared in Cisco IOS Release 11.2 P and Release 11.1 CA.

Usage Guidelines

Use this command to specify which packets should be redirected to the Content Engine. When WCCP is enabled but this command is not configured, all web-related packets are redirected to the Content Engine. When you enter this command, only packets that match the access list are redirected.

Some web sites use the source IP address of packets for authentication. The Content Engine uses its own IP address when sending requests to web sites. Thus, the requests from the Content Engine may not be authenticated. Use this command to bypass the Content Engine in these cases.

Use the **ip wccp enable** and **ip web-cache redirect** commands to configure WCCP.

Examples

The following example configures a router to redirect web-related packets without a destination of 192.168.196.51 to the Content Engine:

```
Router# configure terminal
Router(config)# access-list 100 deny ip any host 192.168.196.51
Router(config)# access-list 100 permit ip any any
Router(config)# ip wccp enable
Router(config)# ip wccp redirect-list 100
Router(config)# interface Ethernet 0
Router(config-if)# ip web-cache redirect
Router(config-if)# end
Router#
%SYS-5-CONFIG_I: Configured from console by console.
```

Related Commands

clear ip wccp

ip wccp enable

ip web-cache redirect

show ip wccp

ip web-cache redirect

To instruct an interface to check for appropriate outgoing packets and redirect them to a Content Engine, use the **ip web-cache redirect** interface configuration command. The **no** form of this command disables the redirection of messages to the Content Engine.

ip web-cache redirect

no ip web-cache redirect

Syntax Description

This command has no arguments or keywords.

Defaults

The interface does not redirect messages to the Content Engine.

Command Modes

Interface configuration

Command History

This command first appeared in Cisco IOS Release 11.2 P and Release 11.1 CA.

Usage Guidelines

This command and the **ip wccp enable** interface command are the only commands required to start redirecting requests to the Content Engine using WCCP.

Examples

The following example configures a router to support WCCP and redirects web-related packets from Ethernet interface 0 to the Content Engine:

```
Router# configure terminal
Router(config)# ip wccp enable
Router(config)# interface Ethernet 0
Router(config-if)# ip web-cache redirect
Router(config-if)# end
Router#
%SYS-5-CONFIG_I: Configured from console by console.
```

Related Commands

clear ip wccp

ip wccp enable

ip wccp redirect-list

show ip interface

show ip wccp

show ip wccp web-caches

show ip interface

To display the usability status of interfaces configured for IP, use the **show ip interface** EXEC command.

show ip interface [*type number*]

Syntax Description

type	(Optional) Interface type.
number	(Optional) Interface number.

Defaults

No default behavior or values

Command Modes

EXEC

Command History

This command first appeared in Cisco IOS Release 10.0.

Usage Guidelines

The Cisco IOS software automatically enters a directly connected route in the routing table if the interface is usable. A usable interface is one through which the software can send and receive packets. If the software determines that an interface is not usable, it removes the directly connected routing entry from the routing table. Removing the entry allows the software to use dynamic routing protocols to determine backup routes to the network (if any).

If the interface can provide two-way communication, the line protocol is marked "up." If the interface hardware is usable, the interface is marked "up."

If you specify an optional interface type, you will see information about that specific interface only.

If you specify no optional arguments, you will see information about all of the interfaces.

When an asynchronous interface is encapsulated with Point-to-Point Protocol (PPP) or Serial Line Internet Protocol (SLIP), IP fast switching is enabled. A **show ip interface** command on an asynchronous interface encapsulated with PPP or SLIP displays a message indicating that IP fast switching is enabled.

Examples

The following is sample output from the **show ip interface** command:

```
Router# show ip interface

Ethernet0 is up, line protocol is up
  Internet address is 172.168.78.24, subnet mask is 255.255.255.240
  Broadcast address is 255.255.255.255
  Address determined by non-volatile memory
  MTU is 1500 bytes
  Helper address is not set
  Secondary address 172.31.255.255, subnet mask 255.255.255.0
  Directed broadcast forwarding is enabled
  Multicast groups joined: 224.0.0.1 224.0.0.2
  Outgoing access list is not set
  Inbound  access list is not set
```

```
Proxy ARP is enabled
Security level is default
Split horizon is enabled
ICMP redirects are always sent
ICMP unreachables are always sent
ICMP mask replies are never sent
IP fast switching is enabled
IP fast switching on the same interface is disabled
IP SSE switching is disabled
Router Discovery is disabled
IP output packet accounting is disabled
IP access violation accounting is disabled
TCP/IP header compression is disabled
Probe proxy name replies are disabled
Web Cache Redirect is enabled
```

Table A-1 describes the fields shown in the display.

Table A-1 Field Descriptions—show ip interface Command

Field	Description
Ethernet0 is up	If the interface hardware is usable, the interface is marked “up.” For an interface to be usable, both the interface hardware and the line protocol must be up.
Line protocol is up	If the interface can provide two-way communication, the line protocol is marked “up.” For an interface to be usable, both the interface hardware and the line protocol must be up.
Internet address	Shows the IP address.
Subnet mask	Shows the subnet mask address.
Broadcast address	Shows the broadcast address.
Address determined by...	Indicates how the IP address of the interface was determined.
MTU	Shows the maximum transmission unit (MTU) value set on the interface.
Helper address	Shows a helper address, if one has been set.
Secondary address	Shows a secondary address, if one has been set.
Directed broadcast forwarding	Indicates whether directed broadcast forwarding is enabled.
Multicast groups joined	Indicates the multicast groups this interface is a member of.
Outgoing access list	Indicates whether the interface has an outgoing access list set.
Inbound access list	Indicates whether the interface has an incoming access list set.
Proxy ARP	Indicates whether proxy Address Resolution Protocol (ARP) is enabled for the interface.
Security level	Specifies the IP Security Option (IPSO) security level set for this interface.
ICMP redirects	Specifies whether redirects will be sent on this interface.
ICMP unreachables	Specifies whether unreachable messages will be sent on this interface.
ICMP mask replies	Specifies whether mask replies will be sent on this interface.

Table A-1 Field Descriptions—show ip interface Command (continued)

Field	Description
IP fast switching	Specifies whether fast switching has been enabled for this interface. It is generally enabled on serial interfaces, such as this one.
IP SSE switching	Specifies whether IP silicon switching engine (SSE) switching is enabled.
Router Discovery	Specifies whether the discovery process has been enabled for this interface. It is generally disabled on serial interfaces.
IP output packet accounting	Specifies whether IP accounting is enabled for this interface and what the threshold (maximum number of entries) is.
TCP/IP header compression	Indicates whether compression is enabled or disabled.
Probe proxy name	Indicates whether HP probe proxy name replies are generated.
Web Cache Redirect	Indicates whether HTTP packets are redirected to a Content Engine.

show ip wccp

To display global statistics related to WCCP, use the **show ip wccp** EXEC command.

show ip wccp

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values

Command Modes EXEC

Command History This command first appeared in Cisco IOS Release 11.2 P and Release 11.1 CA.

Usage Guidelines Use the **clear ip wccp** command to reset the counter for the "Total Packets Redirected" information.

Examples The following example shows sample **show ip wccp** output:

```
Router# show ip wccp

Global WCCP information:
         Number of web-caches:               2
         Total Packets Redirected:           101
         Redirect access-list:               no_linux
         Total Packets Denied Redirect:      88
         Total Packets Unassigned:           0
```

Table A-2 describes the fields shown in this example.

Table A-2 Field Descriptions—show ip wccp Command

Field	Description
Number of web-caches	Number of Content Engines using the router as their home router.
Total Packets Redirected	Total number of packets redirected by the router.
Redirect access-list	Name or number of the redirect access list. Only packets matching the access list are redirected.
Total Packets Denied Redirect	Total number of packets that were not redirected because they did not match the access list.
Total Packets Unassigned	Number of packets that were not redirected because they were not assigned to any web cache. Packets are sometimes not assigned during initial discovery of Content Engines or when a Content Engine is dropped from a farm.

Related Commands

clear ip wccp

ip wccp enable

ip wccp redirect-list

ip web-cache redirect

show ip interface

show ip wccp web-caches

show ip wccp web-caches

To display information about the router's known Content Engines, use the **show ip wccp web-caches** EXEC command.

show ip wccp web-caches

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

EXEC

Command History

This command first appeared in Cisco IOS Release 11.2 P and Release 11.1 CA.

Usage Guidelines

Use the **clear ip wccp** command to reset the counter for the "Packets Redirected" information.

Examples

The following example shows sample **show ip wccp web-caches** output:

```
Router# show ip wccp web-caches

WCCP Web-Cache information:
        IP Address:             172.168.88.11
        Protocol Version:       1.0
        State:                  Usable
        Initial Hash Info:      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
                                AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        Assigned Hash Info:     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
                                FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
        Hash Allotment:         256 (100.00%)
        Packets Redirected:     21345
        Connect Time:           00:13:46
```

Table A-3 explains the fields shown in this display.

Table A-3 Field Descriptions—show ip wccp web-caches Command

Field	Description
IP Address	IP address of the Content Engine.
Protocol Version	Version of the Web Cache Communication Protocol the Content Engine is running.
State	State of the Content Engine. Possible values are "Usable" and "NOT Usable."
Initial Hash Info	Initial contents of the hash field. The hash field contains information about how the router intends to use the Content Engine.

Table A-3 Field Descriptions—show ip wccp web-caches Command (continued)

Field	Description
Assigned Hash Info	Current hash information of the Content Engine. The hash information field contains information about how the router intends to use the Content Engine.
Hash Allotment	Percentage of all possible web servers for which the router redirects HTTP requests to this web cache. In this example, there is only one Content Engine, so all HTTP requests are redirected to it.
Packets Redirected	Number of packets redirected to this Content Engine.
Connect Time	Indicates how long the Content Engine has used this router as its home router.

Related Commands

clear ip wccp

ip wccp enable

ip web-cache redirect

show ip interface

show ip wccp

Debug Commands

This section documents new or modified **debug** commands related to WCCP.

- **debug ip wccp events**
- **debug ip wccp packets**

debug ip wccp events

To display information about significant WCCP events, use the **debug ip wccp events** command. To disable debugging output, use the **no** form of this command.

debug ip wccp events

no debug ip wccp events

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Examples

The following example is sample output of the **debug ip wccp events** command when a Content Engine is added to the list of available web caches:

```
Router# debug ip wccp events

WCCP-EVNT: Built I_See_You msg body w/1 usable web caches, change # 0000000A
WCCP-EVNT: Web Cache 192.168.25.3 added
WCCP-EVNT: Built I_See_You msg body w/2 usable web caches, change # 0000000B
WCCP-EVNT: Built I_See_You msg body w/2 usable web caches, change # 0000000C
```

debug ip wccp packets

To display information about every WCCP packet received or sent by the router, use the **debug ip wccp packets** command. To disable debugging output, use the **no** form of this command.

debug ip wccp packets

no debug ip wccp packets

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Examples

The following example is sample output of the **debug ip wccp packets** command. The router is sending keepalive packets to the Content Engines at 192.168.25.4 and 192.168.25.3. Each keepalive packet has an identification number associated with it. When the Content Engine receives a keepalive packet from the router, it sends a reply with the identification number back to the router.

```
Router# debug ip wccp packets

WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.4 w/rcvd_id 00003532
WCCP-PKT: Sending I_See_You packet to 192.168.25.4 w/ rcvd_id 00003534
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.3 w/rcvd_id 00003533
WCCP-PKT: Sending I_See_You packet to 192.168.25.3 w/ rcvd_id 00003535
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.4 w/rcvd_id 00003534
WCCP-PKT: Sending I_See_You packet to 192.168.25.4 w/ rcvd_id 00003536
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.3 w/rcvd_id 00003535
WCCP-PKT: Sending I_See_You packet to 192.168.25.3 w/ rcvd_id 00003537
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.4 w/rcvd_id 00003536
WCCP-PKT: Sending I_See_You packet to 192.168.25.4 w/ rcvd_id 00003538
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.3 w/rcvd_id 00003537
WCCP-PKT: Sending I_See_You packet to 192.168.25.3 w/ rcvd_id 00003539
```

Web Cache Communication Protocol Version 2

This appendix describes the Web Cache Communication Protocol (WCCP) Version 2 feature. (WCCP is also known as Web Cache Control Protocol and Web Cache Coordination Protocol.) This appendix includes information on the benefits of this feature, and other information you may need to work with WCCP Version 2. This appendix includes the following sections:

- Feature Overview, page B-1
- Prerequisites, page B-6
- Configuration Tasks, page B-7
- Monitoring and Maintaining WCCP Version 2, page B-11
- Configuration Examples, page B-12
- Command Reference, page B-15

Feature Overview

Cisco developed the Web Cache Communication Protocol (WCCP) within Cisco IOS software to enable routers or switches to transparently redirect packets to network caches. It does not interfere with normal router or switch operations. Using WCCP, the router redirects requests on configured TCP ports to network caches rather than to intended host sites. It also balances traffic load across a cache cluster and ensures fault-tolerant and fail-safe operation. As Content Engines are added to or deleted from a cache cluster, the WCCP-aware router or switch dynamically adjusts its redirection map to reflect the currently available caches, resulting in maximized performance and content availability.

WCCP Version 2 contains the following features:

- Multiple router support
- Improved security
- Faster throughput
- Redirection of multiple TCP port-destined traffic
- Load-distributing applications capability
- Client IP addressing transparency

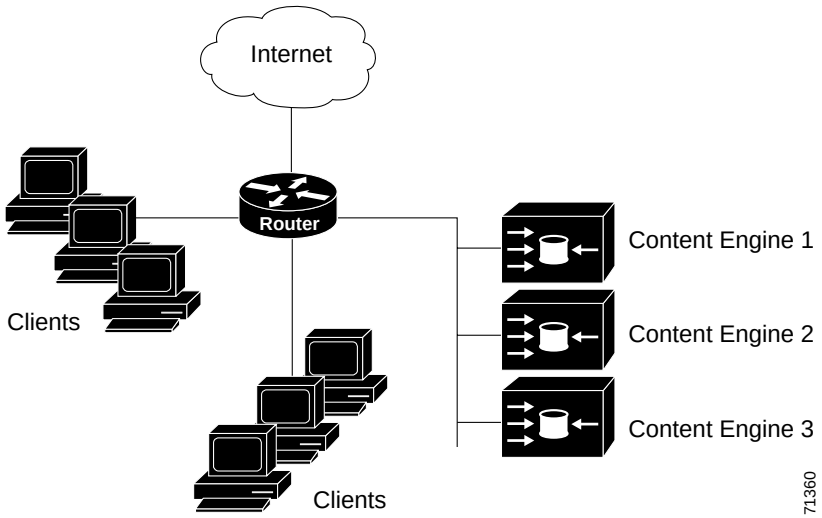
Multirouter Support

WCCP Version 2 enables a series of Content Engines, called a *Content Engine cluster*, to connect to multiple routers. This feature provides redundancy and a more distributed architecture for instances when a Content Engine needs to connect to a large number of interfaces. This strategy also has the benefit of keeping all the Content Engines in a single cluster, avoiding unnecessary duplication of web pages across several clusters.

How Version 1 Works

With WCCP Version 1, only a single router services a cluster, becoming the default home router for the cluster. In this scenario, this router is the device that performs all the IP packet redirection. Figure B-1 illustrates how this configuration appears.

Figure B-1 Content Engine Network Configuration Using WCCP Version 1



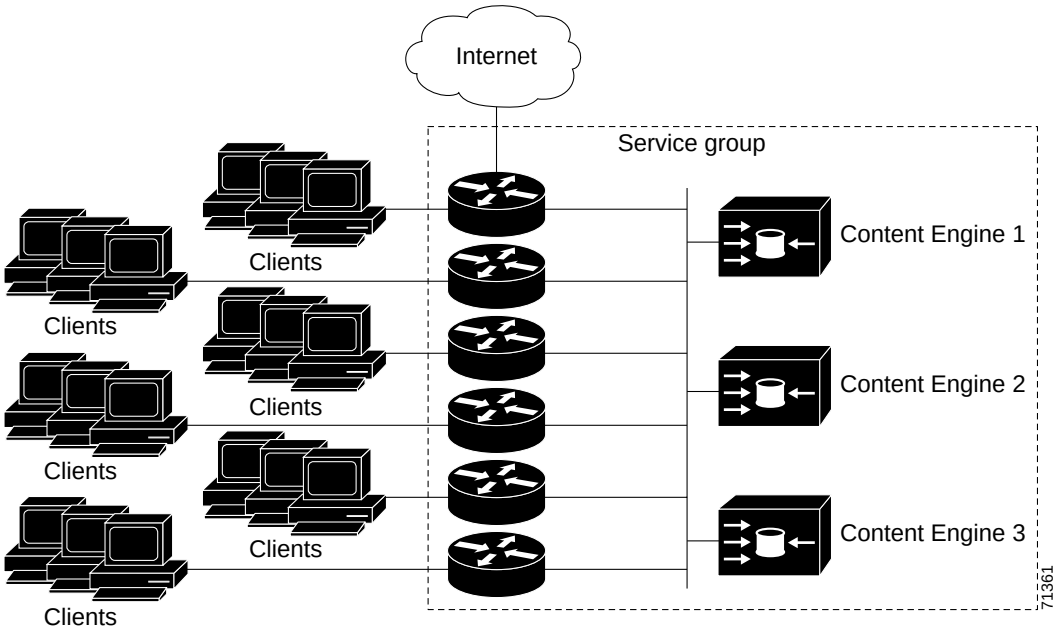
The following sequence of events details how this process works:

1. Each Content Engine records the IP address of the router servicing the cluster.
2. The Content Engines then transmit their IP addresses to the router, indicating their presence to one another in the cluster.
3. The router then replies to the Content Engines, establishing that each can connect to others in the cluster, and providing a *view* (a list) of Content Engine addresses in the cluster, indicating that all can recognize one another.
4. Once the view has been established, one Content Engine is designated the lead and indicates to the router how IP packet redirection should be performed. The lead Content Engine is defined as one seen by all the routers in the service group and that has the lowest IP address.

How Version 2 Works

With WCCP Version 2, multiple routers can service a cluster. This allows any of the available routers in a service group to redirect packets to each of the Content Engines in the cluster. Figure B-2 illustrates how this configuration appears.

Figure B-2 Content Engine Network Configuration Using WCCP Version 2



You can configure the router to run one of the following cache-related services:

- Web cache—A global service that sends content to a large collection of destination World Wide Web servers
- Reverse proxy—A local service that sends content to a small number of destination World Wide Web servers
- Custom web cache—A service in which service group number 98 permits routers to redirect HTTP traffic to the Content Engine
- Dynamic web service—Service in which service group numbers 90 to 97 can be configured to redirect traffic to the Content Engine

The subset of Content Engines within a cluster and routers connected to the cluster that are running the same service is known as a *service group*.

Available services include TCP and User Datagram Protocol (UDP) redirection for streaming media applications.

Using WCCP Version 1, the Content Engines were configured with the address of the single router. WCCP Version 2 requires that each Content Engine be aware of all the routers in the service group. To specify the addresses of all the routers in a service group, you must choose one of the following methods:

- Unicast—A list of router addresses for each of the routers in the group is configured on each Content Engine. In this case, the address of each router in the group must be explicitly specified for each Content Engine during configuration.
- Multicast—A single multicast address is configured on each Content Engine. In the multicast address method, the Content Engine sends a single-address notification that provides coverage for all routers in the service group. For example, a Content Engine could indicate packets should be sent to a multicast address of 224.0.0.100, which in turn would send a multicast packet to all routers in the service group configured for group listening using WCCP. (See the **ip wccp group-listen** command for details.)

The multicast option is easier to configure because you have to specify only a single address on each Content Engine. This option also allows you to add and remove routers from a service group dynamically, without having to reconfigure the Content Engines with a different list of addresses each time.

The following sequence of events details how WCCP Version 2 configuration works:

1. Each Content Engine is configured with a list of routers.
2. Each Content Engine announces its presence and a list of all routers with which it has established communications. The routers reply with their view (list) of Content Engines in the group.
3. Once the view is consistent across all Content Engines in the cluster, one Content Engine is designated the lead and sets the policy that the routers need to deploy in redirecting packets.

You must also perform these tasks to configure the routers that will become members of the service group:

- Configure an IP multicast address for use by the cluster.
- Enable the **ip wccp** command.

For network configurations in which the Content Engine sends to a target router a packet that needs to traverse an intervening router, the router being traversed must be configured to perform IP multicast routing. You must configure two components to enable traversal over an intervening router:

- Enable IP multicast routing using the **ip multicast routing** command.
- Enable the interfaces to which the Content Engines will connect to receive multicast transmissions using the **ip pim** command.

How Routers and Content Engines Communicate

Routers and Content Engines become aware of one another and form a service group using a management protocol. Once the service group has been established, one of the Content Engines is designated to determine load assignments among the Content Engines. The Content Engines also send periodic "Here I am" messages to the routers which allows them to rediscover the Content Engines.

If there is a group of Content Engines, the one seen by all routers and the one that has the lowest IP address becomes the lead Content Engine. The role of this Content Engine is to determine how traffic should be allocated across Content Engines. The assignment information is passed to the entire service group from the designated Content Engine so that the routers of the group can redirect the packets properly and the Content Engines of the group can manage their load better.

Improved Security

WCCP Version 2 provides authentication that enables you to control which routers and Content Engines become part of the service group. You use passwords and the HMAC MD5 standard set by the **ip wccp** [**password** [*0-7*] *password*] command to control service group membership.

Improved Throughput

Cisco Express Forwarding (CEF) has been integrated into WCCP Version 2 to achieve optimal performance during packet redirection.

Redirection for Multiple TCP Port-Destined Traffic

WCCP Version 2 enables more TCP ports to have traffic redirected to the Content Engine. Previously, web-cached information could be redirected only if it was destined for TCP port 80. Many applications require packets intended for other ports to be redirected, for example, proxy web cache handling, FTP proxy caching, web caching for ports other than 80, and RealAudio, and video.

Packets that the Content Engines do not service are tunneled back to the same router from which they were received. When a router receives a formerly redirected packet, it knows not to redirect it again.

The criteria for determining whether to redirect the traffic are:

- IP protocol
- Ports
- Priority
- Distribution scheme
- Default handling

Note that service information has been added to the protocol to indicate which service the WCCP messages refer to. This information is used to help verify that service group members are all using or providing the same service.

Web Cache Packet Return

WCCP Version 2 filters packets to determine which redirected packets have been returned from the Content Engine and which ones have not. It does not redirect the ones that have been returned, because the Content Engine has determined that the packets should not be processed. WCCP Version 2 returns packets that the Content Engine does not service to the same router from which they were transmitted. Typical reasons why a Content Engine would reject packets and initiate packet return are:

- The Content Engine is overloaded and has no resources to service the packets.
- The Content Engine activates the automatic bypass feature as a result of server error or authentication failure. In this scenario the Content Engine switches itself off the session to allow the client reach the server directly, and assures that the Content Engine is not the reason for failure.
- The Content Engine is filtering certain conditions that make processing packets counterproductive, for example, when IP authentication has been turned on.
- The Content Engine is configured with a static bypass list by the administrator. See the "Static Bypass" section on page 4-8 for more information on how to configure a static bypass list.

Note The packets are redirected to the source of the connection between the router and the Content Engine. Depending on the IOS version used, this could be either the address of the outgoing interface or the router IP address. In the latter case, it is important that the Content Engine has the router IP address stored in the router list.

Load-Distributing Applications

WCCP Version 2 has the capability to adjust the load being offered to individual Content Engines to provide more effective use of the resources available and at the same time help to ensure high quality of service to the clients. It uses two techniques to perform this task:

- Load balancing allows the set of hash buckets assigned to a Content Engine to be adjusted so that the load can be shifted from an overwhelmed Content Engine to other Content Engines that have available capacity.
- Load shedding enables the router to selectively redirect the load to avoid exceeding the capacity of the Content Engines.

Client IP Address Transparency

The Content Engine accepts traffic and establishes the connection with the client, acting as if it were the original destination server. Once the connection is established, if the object being requested is not available on the Content Engine, the engine then establishes its own connection out to the original destination server.

Restrictions

The following limitations apply to WCCP Version 2:

- The Time To Live (TTL) value of routers servicing a cluster must be 15 hops or less. The TTL indicates how many hops or times a request is allowed to travel back and forth between the router and the Content Engines.
- The protocol needs to include the list of routers in the service group as part of its messages to properly depict the view.
- Service groups can comprise up to 32 Content Engines and 32 routers.
- All Content Engines in a cluster must include all routers servicing the cluster in its configuration. If a Content Engine within a cluster does not include one or more of the routers in its configuration, the service group will detect the inconsistency and the Content Engine will not be allowed to operate within the service group.
- Multicast addresses must be between 224.0.0.0 and 239.255.255.255.
- WCCP works with IP networks only.

Note A Content Engine and a WCCP-enabled router cannot be separated by a firewall. The firewall handles only packet traffic toward the origin web server and does not handle packet traffic sent to the client by the Content Engine on behalf of the server.

Related Documents

- *Cisco IOS Configuration Fundamentals Configuration Guide*
- *Cisco IOS Configuration Fundamentals Command Reference*

Prerequisites

Before you use WCCP Version 2, you must complete the following tasks:

- Properly install and configure a cache cluster connected to one or more routers.

- Configure IP on the interface connected to the Internet and the interface connected to the Content Engine. The interface connected to the Content Engine must be an Ethernet or Fast Ethernet interface.

Configuration Tasks

You can configure a router to run the web cache, custom web cache, and reverse proxy services associated with WCCP Version 2. The services can be configured simultaneously. Perform the following tasks to configure a cluster with multiple routers.

- Configuring a Service Group Using WCCP Version 2, page B-7
- Running the Web Cache Service, page B-7
- Running the Reverse Proxy Service, page B-8
- Running a Custom Web Cache Service, page B-8
- Running a Dynamic Web Cache Service, page B-8
- Registering a Router to a Multicast Address, page B-8
- Informing a Router of Valid IP Addresses, page B-9
- Setting a Password for a Router and Content Engines, page B-9
- Disabling Caching for Certain Clients, page B-9

Configuring a Service Group Using WCCP Version 2

	Command	Purpose
Step 1	`Router(config)# ip wccp {web-cache | service-number} [group-address groupaddress] [redirect-list access-list] [group-list access-list] [password [0-7] password]`	Turns the WCCP feature on or off for the specified service.
Step 2	`Router(config)# interface type number`	Specifies an interface to configure and enters interface configuration mode.
Step 3	`Router(config-if)# ip wccp {web-cache | service-number} redirect {out | in}`	Enables WCCP redirection on the specified interface.
Step 4	`Router(config-if)# ip wccp redirect exclude in`	Allows inbound packets on this interface to be excluded from redirection.

Running the Web Cache Service

	Command	Purpose
Step 1	`Router(config)# ip wccp web-cache`	Turns on the protocol for web caching.
Step 2	`Router(config)# interface type number`	Specifies an interface for web caching.
Step 3	`Router(config-if)# ip wccp web-cache redirect out`	Enables the check on packets to determine whether they need to be redirected to a web cache.

Running the Reverse Proxy Service

	Command	Purpose
Step 1	`Router(config)# ip wccp 99`	Turns the WCCP feature on or off for the reverse proxy service. The service number for reverse proxy is 99.
Step 2	`Router(config)# interface type number`	Specifies an interface for the reverse proxy service.
Step 3	`Router(config-if)# ip wccp 99 redirect out`	Specifies “out” for the reverse proxy service.

Running a Custom Web Cache Service

	Command	Purpose
Step 1	`Router(config)# ip wccp 98`	Turns the WCCP feature on or off for the custom web cache service. The service number for custom web cache is 98.
Step 2	`Router(config)# interface type number`	Specifies an interface on which the custom web cache service will run.
Step 3	`Router(config-if)# ip wccp 98 redirect out`	Specifies “out” for the custom web cache service.

Running a Dynamic Web Cache Service

	Command	Purpose
Step 1	`Router(config)# ip wccp 90`	Turns the WCCP feature on or off for the generic web cache service. The service numbers for generic web cache services are 90 to 97.
Step 2	`Router(config)# interface type number`	Specifies an interface on which the reverse proxy service will run.
Step 3	`Router(config-if)# ip wccp 90 redirect out`	Specifies “out” for the custom web cache service.

Registering a Router to a Multicast Address

	Command	Purpose
Step 1	`Router(config)# ip wccp web-cache group-address groupipaddress`	Configures the group address for the service group.
Step 2	`Router(config)# interface type number`	Specifies an interface that will listen for the multicast address.
Step 3	`Router(config-if)# ip wccp web-cache group-listen`	Configures an interface on a router to enable or disable the reception of IP multicast packets for WCCP.

Informing a Router of Valid IP Addresses

	Command	Purpose
Step 1	`Router(config)# ip wccp web-cache group-list access-list`	Indicates to the router which Content Engine IP addresses to allow packets from.
Step 2	`Router(config)# access-list access-list number permit host host-address`	Creates an access list that enables or disables traffic redirection to the Content Engine.

Setting a Password for a Router and Content Engines

	Command	Purpose
Step 1	`Router(config)# ip wccp web-cache password [0-7] password`	Sets a password for the Content Engine that the router is trying to access.

Disabling Caching for Certain Clients

	Command	Purpose
Step 1	`Router(config)# ip wccp web-cache redirect-list access-list number`	Sets the access list used to enable redirection.
Step 2	`Router(config)# access-list access-list number deny host host-address`	Creates an access list that enables or disables traffic redirection to the Content Engine.
Step 3	`Router(config)# access-list access-list number permit any any`	Sets the access list to enable access to any host.

Verifying WCCP Configuration Settings

Step 1 To view the configuration, enter the **show running-config** command.

A sample configuration follows:

```
Console# show running-config

Building configuration...
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname router4
!
enable secret 5 $1$nSVy$faliJsVQXVPW.KuCxZNTh1
enable password alabama1
```

```
!
ip subnet-zero
ip wccp web-cache
ip wccp 99
ip domain-name cisco.com
ip name-server 10.1.1.1
ip name-server 10.1.1.2
ip name-server 10.1.1.3
!
!
!
interface Ethernet0
ip address 10.3.1.2 255.255.255.0
no ip directed-broadcast
ip wccp web-cache redirect out
ip wccp 99 redirect out
no ip route-cache
no ip mroute-cache
!
interface Ethernet1
ip address 10.4.1.1 255.255.255.0
no ip directed-broadcast
ip wccp 99 redirect out
no ip route-cache
no ip mroute-cache
!
interface Serial0
no ip address
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
shutdown
!
interface Serial1
no ip address
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
shutdown
!
ip default-gateway 10.3.1.1
ip classless
ip route 0.0.0.0 0.0.0.0 10.3.1.1
no ip http server
!
!
!
line con 0
transport input none
line aux 0
transport input all
line vty 0 4
password alaska1
login
!
end
```

Step 2 To view values associated with WCCP variables, enter the **show ip wccp** command. Output similar to the following is displayed:

```
Console# show ip wccp

Global WCCP Information:
Service Name: web-cache:
```

```
Number of Content Engines:1
Number of Routers:1
Total Packets Redirected:213
Redirect access-list:no_linux
Total Packets Denied Redirect:88
Total Packets Unassigned:-none-
Group access-list:0
Total Messages Denied to Group:0
Total Authentication failures:0

Service Name: 99
Number of Content Engines:1
Number of Routers:2
Total Packets Redirected:198
Redirect access-list:-none-
Total Packets Denied Redirect:0
Total Packets Unassigned:0
Group access-list:11
Total Messages Denied to Group:0
Total Authentication failures:0
```

Monitoring and Maintaining WCCP Version 2

Command	Purpose

`Router# show ip wccp`
 	
or
 	
`Router# show ip wccp {web-cache | 90-99}`
	Displays global statistics related to WCCP.

`Router# show ip wccp {web-cache | 90-99} detail`	Queries the router for information about which Content Engines the router has detected in a specific service group. The information can be displayed for service groups ranging in value from 90–99.
`Router# show ip interface`	Shows whether any **ip wccp direct** commands are configured on an interface.
`Router# show ip wccp {web-cache | 90-99} view`	Displays which devices in a particular service group have been detected and which Content Engines are not visible to all other routers to which the current router is connected. The information can be displayed for service groups ranging in value from 90–99.

Configuration Examples

This section provides the following configuration examples:

- Performing a General WCCP Version 2 Configuration, page B-12
- Running the Web Cache Service, page B-12
- Running the Reverse Proxy Service, page B-12
- Running the Custom Web Cache Service, page B-13
- Running a Generic Web Cache Service, page B-13
- Registering a Router to a Multicast Address, page B-13
- Informing a Router of Valid IP Addresses, page B-13
- Setting a Password for a Router and Content Engines, page B-13
- Bypassing the Cache with Router Access Lists, page B-14
- Displaying WCCP Settings, page B-14

Performing a General WCCP Version 2 Configuration

The following example shows a general WCCP Version 2 configuration session:

Note You must enter the **ip wccp version 2** command in all Version 2 configurations to enable redirection using WCCP Version 2.

```
ip wccp web-cache group-address 224.1.1.100 password alabama1
interface ethernet0
ip wccp web-cache redirect out
```

Running the Web Cache Service

The following example shows a web cache service configuration session:

```
configure terminal
ip wccp web-cache
interface ethernet 0
ip wccp web-cache redirect out
```

Running the Reverse Proxy Service

The following example shows a reverse proxy service configuration session:

```
configure terminal
ip wccp 99
interface ethernet 0
ip wccp 99 redirect out
```

Running the Custom Web Cache Service

The following example shows a custom web cache configuration session:

```
configure terminal
ip wccp 98
interface ethernet 0
ip wccp 98 redirect out
```

Running a Generic Web Cache Service

The following example shows a generic web cache configuration session:

```
configure terminal
ip wccp 91
interface ethernet 0
ip wccp 91 redirect out
```

Registering a Router to a Multicast Address

The following example shows how to register a router to a multicast address of 192.168.0.0:

```
configure terminal
ip wccp web-cache group-address 172.168.0.0
interface ethernet 0
ip wccp web cache group-listen
```

Informing a Router of Valid IP Addresses

To achieve better security, you can use a standard access list to notify the router which IP addresses are valid addresses for a Content Engine attempting to register with the current router. The following example shows a standard access list configuration session in which the access list number is 10 for some sample hosts:

```
configure terminal
access-list 10 permit host 10.1.1.1
access-list 10 permit host 10.1.1.2
access-list 10 permit host 10.1.1.3
ip wccp web-cache group-list 10
```

Setting a Password for a Router and Content Engines

The following example shows a WCCP Version 2 password configuration session in which the password is alabama2:

```
configure terminal
ip wccp web-cache password alabama2
```

Bypassing the Cache with Router Access Lists

The router can be configured with access lists to permit or deny redirection of traffic to the Content Engine. In the following example, traffic conforming to the following criteria is not redirected by the router to the Content Engine:

- Originating from the host 10.1.1.1 destined for any other host
- Originating from any host destined for the host 10.255.1.1

```
router# configure terminal
router(config)# ip wccp web-cache redirect-list 120
router(config)# access-list 120 deny ip host 10.1.1.1 any
router(config)# access-list 120 deny ip any host 10.255.1.1
router(config)# access-list 120 permit ip any any
```

Traffic not explicitly permitted is implicitly denied redirection. The **access-list 120 permit ip any any** command explicitly permits all traffic (from any source en route to any destination) to be redirected to the Content Engine. Because criteria matching occurs in the order by which the commands are entered, the global permit command is the last command entered. For further information on access lists, refer to Cisco IOS software documentation.

Displaying WCCP Settings

The following example displays WCCP settings, using the **show running-config** command:

```
Console# show running-config

Building configuration...
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service udp-small-servers
service tcp-small-servers
!
hostname router4
!
enable secret 5 $1$nSVy$faliJsVQXVPW.KuCxZNTh1
enable password alabama1
!
ip subnet-zero
ip wccp web-cache
ip wccp 99
ip domain-name cisco.com
ip name-server 10.1.1.1
ip name-server 10.1.1.2
ip name-server 10.1.1.3
!
!
!

interface Ethernet0
ip address 10.3.1.2 255.255.255.0
no ip directed-broadcast
ip wccp web-cache redirect out
ip wccp 99 redirect out
no ip route-cache
```

```
no ip mroute-cache
!
interface Ethernet1
ip address 10.4.1.1 255.255.255.0
no ip directed-broadcast
ip wccp 99 redirect out
no ip route-cache
no ip mroute-cache
!
interface Serial0
no ip address
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
shutdown
!
interface Serial1
no ip address
no ip directed-broadcast
no ip route-cache
no ip mroute-cache
shutdown
!
ip default-gateway 10.3.1.1
ip classless
ip route 0.0.0.0 0.0.0.0 10.3.1.1
no ip http server
!
!
!
line con 0
transport input none
line aux 0
transport input all
line vty 0 4
password alaska1
login
!
end
```

Command Reference

This section documents new or modified commands that configure the WCCP Version 2 feature.

- **clear ip wccp**
- **ip wccp**
- **ip wccp group-listen**
- **ip wccp redirect exclude in**
- **ip wccp redirect**
- **ip wccp version**
- **show ip interface**
- **show ip wccp**

In Cisco IOS Release 12.0(1)T or a later version of Release 12.0 T, you can search and filter the output for **show** and **more** commands. This functionality is useful when you need to sort through large amounts of output, or if you want to exclude output that you do not need to see.

To use this functionality, enter a **show** or **more** command followed by the "pipe" character (|), one of the keywords **begin**, **include**, or **exclude**, and an expression that you want to search or filter on:

command | {**begin** | **include** | **exclude**} *regular-expression*

Following is an example of the **show atm vc** command in which you want the command output to begin with the first line where the expression "PeakRate" appears:

show atm vc | **begin PeakRate**

For more information on the search and filter functionality, refer to the Cisco IOS Release 12.0(1)T feature module titled *CLI String Search*.

clear ip wccp

To remove Web Cache Communication Protocol (WCCP) statistics maintained on the router either for a particular service or for all the services, use the **clear ip wccp** EXEC command.

clear ip wccp {web-cache | *service-number*}

Syntax Description

web-cache	Directs the router to remove statistics for the web cache service.
service-number	Directs the router to remove statistics for a specified web cache service. The number can be from 0 to 99. The reverse proxy service is indicated by a value of 99.

Defaults

No default behavior or values

Command Modes

EXEC

Command History

Release	Modification
11.1 CA	This command was introduced.
11.2 P	This command was introduced.
12.0(3)T	This command has been expanded to be explicit about service.

Usage Guidelines

Use the **show ip wccp** and **show ip wccp detail** commands to display WCCP statistics.

Examples

```
clear ip wccp web cache
```

Related Commands

Command	Description
ip wccp	Directs a router to enable or disable the support for a service group.
show ip wccp	Displays global statistics related to the WCCP feature.

ip wccp

To direct a router to enable or disable the support for a Content Engine service group, use the **ip wccp** global configuration command. To remove the ability of a router to control support for a service group, use the **no** form of this command.

ip wccp {**web-cache** | *service-number*} [**group-address** *groupaddress*] [**redirect-list** *access-list*] [**group-list** *access-list*] [**password** [*0-7*] *password*]

no ip wccp {**web-cache** | *service-number*} [**group-address** *groupaddress*] [**redirect-list** *access-list*] [**group-list** *access-list*] [**password** [*0-7*] *password*]

Syntax Description

web-cache	Enables the web cache service.
service-number	Identification number of the Web Cache Communication Protocol (WCCP) service being controlled by a router. The number can be from 0 to 99. The reverse proxy service is indicated by a value of 99.
group-address	(Optional) Directs the router to use a specified multicast IP address for communication with the WCCP service group.
groupaddress	(Optional) Multicast address used by the router to determine which Content Engine should receive redirected messages.
redirect-list	(Optional) Directs the router to use an access list to control traffic redirected to this service group.
access-list	(Optional) String (not to exceed 64 characters) that is the name of the access list that determines which traffic is redirected to a Content Engine.
group-list	(Optional) Directs the router to use an access list to determine which Content Engines are allowed to participate in the service group.
access-list	(Optional) String (not to exceed 64 characters) that is the name of the access list that determines which Content Engines are allowed to participate in the service group.
password	(Optional) String that directs the router to apply MD5 authentication to messages received from the service group specified by the service name given. Messages that are not accepted by the authentication are discarded.
0-7	(Optional) Value that indicates the HMAC MD5 algorithm used to encrypt the password. This value is generated when an encrypted password is created for the Content Engine.
password	(Optional) Password name that is combined with the HMAC MD5 value to create security for the connection between the router and the Content Engine.

Defaults

This command is disabled by default.

Command Modes

Global configuration

Command History

Release	Modification
12.0(3)T	This command was introduced.

Examples

The following example shows a router configured to run WCCP reverse proxy service, using (listening to) the multicast address 172.31.0.0:

```
ip wccp 99 group-address 172.31.0.0
```

Related Commands

Command	Description
ip wccp group-listen	Configures an interface on a router to enable or disable the reception of IP multicast packets for the WCCP feature.

ip wccp group-listen

To configure an interface on a router to enable or disable the reception of IP multicast packets for the Web Cache Communication Protocol (WCCP) feature, use the **ip wccp group-listen** interface configuration command. To remove control of the reception of IP multicast packets for the WCCP feature, use the **no** form of this command.

ip wccp {**web-cache** | *service-number*} **group-listen**

no ip wccp {web-cache | *service-number***} group-listen**

Syntax Description

web-cache	Directs the router to transmit packets to the web cache service.
service-number	Identification number of the Content Engine service group being controlled by a router. The number can be from 0 to 99. The reverse proxy service is indicated by a value of 99.

Defaults

This command is disabled by default.

Command Modes

Interface configuration

Command History

Release	Modification
12.0(3)T	This command was introduced.

Examples

The following example shows that multicast packets have been enabled for a web cache with an address of 192.168.0.0.

```
configure terminal
ip wccp web-cache group-address 192.168.0.0
interface ethernet 0
ip wccp web cache group-listen
```

Related Commands

Command	Description
ip wccp	Directs a router to enable or disable the support for a service group.
ip wccp redirect out	Configures an interface to enable or disable the exclusion of a redirection check for packets that were received on the interface.

ip wccp redirect exclude in

To configure an interface to enable or disable exclusion of packets received on an interface from being redirected to a Content Engine, use the **ip wccp redirect exclude in** interface configuration command. To disable a router's ability to verify that only appropriate packets are being redirected to a Content Engine, use the **no** form of this command.

ip wccp redirect exclude in

no ip wccp redirect exclude in

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values

Command Modes

Interface configuration

Command History

Release	Modification
12.0(3)T	This command was introduced.

Usage Guidelines

Note that the command is global to all the services and should be applied to any inbound interface that has been configured to be excluded from redirection on an outbound interface that the traffic will traverse.

Examples

```
configure terminal
ip wccp 99
interface ethernet0
ip wccp redirect exclude in
```

Related Commands

Command	Description
ip wccp	Directs a router to enable or disable the support for a service group.
ip wccp redirect out	Configures an interface to enable or disable the exclusion of a redirection check for packets that were received on the interface.

ip wccp redirect

To enable packet redirection on an outbound or inbound interface using Web Cache Communication Protocol (WCCP), use the **ip wccp redirect** interface configuration command. To disable WCCP redirection, use the **no** form of this command.

ip wccp {**web-cache** | *service-number*} **redirect** {**out** | **in**}

no ip wccp {**web-cache** | *service-number*} **redirect** {**out** | **in**}

Syntax Description

web-cache	Enables the web cache service.
service-number	Identification number of the Content Engine service group being controlled by a router. The number can be from 0 to 99. The reverse proxy service is indicated by a value of 99.
redirect	Enables packet redirection checking on an outbound or inbound interface.
out	Specifies packet redirection on an outbound interface.
in	Specifies packet redirection on an inbound interface.

Defaults

Redirection checking on the interface is disabled.

Command Types

Interface configuration

Usage Guidelines

Redirection can be specified for outbound interfaces or inbound interfaces. Inbound traffic can be configured to use Cisco Express Forwarding (CEF), distributed Cisco Express Forwarding (dCEF), fast forwarding, or process forwarding.

Configuring WCCP for redirection for inbound traffic on interfaces allows you to avoid the overhead associated with CEF forwarding for outbound traffic. Setting an output feature on any interface results in the slower switching path of the feature being taken by all packets arriving at all interfaces. Setting an input feature on an interface results in only those packets arriving at that interface taking the configured feature path; packets arriving at other interfaces will use the faster default path.

Configuring WCCP for inbound traffic also allows packets to be classified before the routing table lookup, which translates into faster redirection of packets.

Note This command has the potential to affect the **ip wccp redirect exclude in** command. If you have **ip wccp redirect exclude in** set on an interface and you subsequently configure the **ip wccp redirect in** command, the **exclude in** command is overridden. The opposite is also true: configuring the **exclude in** command overrides the **redirect in** command.

Command History

Release	Modification
12.0(3)T	This command was introduced.

Release	Modification
12.0(11)S	The **in** keyword was added to the 12.0 S release train.
12.1(3)T	The **in** keyword was added to the 12.1 T release train.

Examples

The following example shows a configuration session in which reverse proxy packets on Ethernet interface 0 are being checked for redirection and redirected to a Cisco Content Engine:

```
Router# configure terminal
Router(config)# ip wccp 99
Router(config)# interface ethernet 0
Router(config-if)# ip wccp 99 redirect out
```

The following example shows a configuration session in which HTTP traffic arriving on interface 0/1 is redirected to a Cisco Cache Engine:

```
Router# configure terminal
Router(config)# ip wccp web-cache
Router(config)# interface ethernet 0/1
Router(config-if)# ip wccp web-cache redirect in
```

Related Commands

Command	Description
ip wccp redirect exclude in	Configures an interface to enable or disable redirection of packets received on an interface.

ip wccp version

To configure the WCCP version number, use the **ip wccp version** global configuration command. The default WCCP version is Version 2. Use this command to override the default.

ip wccp version {**1** | **2**}

Syntax Description

1	Enables WCCP Version 1.
2	Enables WCCP Version 2.

Defaults

The default is Version 2.

Command Modes

Global configuration

Command History

Release	Modification
12.0(5)T	This command was introduced.

Examples

```
ip wccp version 1
```

show ip interface

To display status about any **ip wccp direct** commands configured on an interface, use the **show ip interface** EXEC command.

show ip interface [*type-number*]

Syntax Description

type-number	(Optional) The interface number whose status is being displayed.

Defaults

No default behavior or values

Command Modes

EXEC

Command History

Release	Modification
10.0	This command was introduced.
12.0	This command was enhanced.
12.0(3)T	This command was enhanced to include the status of **ip wccp redirect out** and **ip wccp redirect exclude in** commands.

Usage Guidelines

The Cisco IOS software automatically enters a directly connected route in the routing table if the interface is usable. A usable interface is one through which the software can send and receive packets. If the software determines that an interface is not usable, it removes the directly connected routing entry from the routing table. Removing the entry allows the software to use dynamic routing protocols to determine backup routes to the network (if any).

If the interface can provide two-way communication, the line protocol is indicated to be up. If the interface hardware is usable, the interface is indicated to be up.

If you specify an optional interface type, you will see information on that specific interface only.

If you specify no optional arguments, you will see information on all the interfaces.

When an asynchronous interface is encapsulated with Point-to-Point Protocol (PPP) or Serial Line Internet Protocol (SLIP), IP fast switching is enabled. A **show ip interface** command on an asynchronous interface encapsulated with PPP or SLIP displays a message indicating that IP fast switching is enabled.

Examples

The following example displays output from the **show ip interface** command, using the interface e3/0:

```
show ip interface e3/0

Ethernet3/0 is up
Internet address is 17.1.1.38/24
Broadcast address is 255.255.255.255
Address determined by non-volatile memory
MTU is 1500 bytes
Helper address is not set
```

```
Directed broadcast forwarding is enabled
Outgoing access list is not set
Inbound access list is not set
Proxy ARP is enabled
Security level is default
Split horizon is enabled
ICMP redirects are always sent
ICMP unreachables are always sent
ICMP mask replies are never sent
IP fast switching is enabled
IP fast switching on the same interface is disabled
IP Optimum switching is enabled
IP multicast fast switching is enabled
Router Discovery is disabled
IP output packet accounting is disabled
IP access violation accounting is disabled
TCP/IP header compression is disabled
Probe proxy name replies are disabled
Gateway Discovery is disabled
Policy routing is disabled
Network address translation is disabled
WCCP Redirect outbound is enabled
WCCP Redirect exclude is disabled
```

Table B-1 describes the fields shown in the display.

Table B-1 Field Descriptions—show ip interface Command

Field	Description
Ethernet 3/0 *is up*	Indicates the status of an interface. If the interface hardware is usable, the interface is marked “up.” For an interface to be usable, both the interface hardware and line protocol must be up.
Internet address	Shows the IP address of the interface.
Broadcast address	Shows the broadcast address.
Address determined by	Indicates how the IP address of the interface was determined.
MTU	Shows the maximum transmission unit (MTU), or the maximum size of packets allowed to be transmitted from the router to a Content Engine.
Helper address	Shows a helper address, if one has been set.
Directed broadcast forwarding	Shows a secondary address, if one has been set.
Outgoing access list	Indicates whether the interface has an outgoing access list set.
Inbound access list	Indicates whether the interface has an incoming access list set.
Proxy ARP	Indicates whether proxy Address Resolution Protocol (ARP) is enabled for the interface.
Security level	Specifies the default IPSO security level for this interface.
Split horizon	Specifies that routing updates sent to a particular neighbor router should not contain information about routes that were learned from that neighbor.
ICMP redirects	Indicates whether Internet Control Message Protocol (ICMP) redirects will be sent on this interface.
ICMP unreachables	Indicates whether unreachable messages will be sent on this interface.

Table B-1 Field Descriptions—show ip interface Command (continued)

Field	Description
ICMP mask replies	Specifies whether mask replies will be sent on this interface.
IP fast switching	Indicates whether fast switching has been enabled for this interface. It is generally enabled on serial interfaces, such as this one.
IP fast switching on the same interface	Indicates whether fast switching has been disabled for this interface. It is generally enabled on serial interfaces, such as this one.
IP Optimum switching	Indicates whether the IP optimum switching feature has been turned on.
IP multicast fast switching	Indicates whether the IP multicast fast switching feature has been turned on.
Router Discovery	Indicates whether the Cisco Discovery Protocol has been turned off.
IP output packet accounting	Indicates whether the output packet counter has been turned off.
IP access violation accounting	Indicates whether the feature that counts unauthorized access events on the router has been turned off.
TCP/IP header compression	Indicates whether compression is enabled or disabled.
Probe proxy name replies	Indicates whether HP Probe proxy name replies are generated.
Gateway Discovery	Indicates whether the gateway discovery option has been turned off.
Policy routing	Indicates whether the policy routing option has been turned off.
Network address translation	Indicates whether the status of the network address translation feature has been enabled or disabled.
WCCP Redirect outbound	Indicates whether packets received on an interface are redirected to a Content Engine. This field can be enabled or disabled.
WCCP Redirect exclude	Indicates whether packets targeted for an interface will be excluded from being redirected to a Content Engine. This field can be enabled or disabled.

Related Commands

Command	Description
show ip wccp	Displays global statistics related to the Web Cache Communication Protocol feature.

show ip wccp

To display global statistics related to the Web Cache Communication Protocol (WCCP) feature, use the **show ip wccp** EXEC command.

show ip wccp {web-cache | *service-number***}** [**view** | **detail**]

Syntax Description

web-cache	Directs the router to display statistics for the web cache service.
service-number	Identification number of the Content Engine service group being controlled by a router. The number can be from 0 to 99. The reverse proxy service is indicated by a value of 99. The custom web cache service is indicated by a value of 98.
view	(Optional) Directs the router to display statistics for the WCCP view configuration.
detail	(Optional) Directs the router to display statistics for the WCCP detail configuration.

Defaults

No default behavior or values

Command Modes

EXEC

Command History

Release	Modification
11.1 CA and 11.2 P	This command was introduced.
12.0(3)T	The user was allowed to query the router for the current global configuration information in use by either a single service or all services.

Usage Guidelines

Use the **clear ip wccp** command to reset the counter for the "Total Packets Redirected" information.

Examples

The following example displays output from the **show ip wccp** command:

```
show ip wccp

Global WCCP Information:
Service Name: web-cache:
Number of Content Engines:1
Number of Routers:1
Total Packets Redirected:213
Redirect access-list: no_linux
Total Packets Denied Redirect:88
Total Packets Unassigned:-none-
Group access-list:0
Total Messages Denied to Group:0
Total Authentication failures:0

Service Name: 1
```

```
Number of Content Engines:1
Number of Routers:2
Total Packets Redirected:198
Redirect access-list:-none-
Total Packets Denied Redirect:0
Total Packets Unassigned:0
Group access-list:11
Total Messages Denied to Group:0
Total Authentication failures:0
```

Table B-2 describes the fields shown in the display.

Table B-2 Field Descriptions—show ip wccp Command

Field	Description
Service Name	Service that is detailed in the display output.
Number of Content Engines	Number of Content Engines using the router as their home router.
Number of Routers	Number of routers in the service group.
Total Packets Redirected	Total number of packets redirected by the router.
Redirect access-list	Name or number of the access list that determines which packets will be redirected.
Total Packets Denied Redirect	Total number of packets that were not redirected because they did not match the access list.
Total Packets Unassigned	Number of packets that were not redirected because they were not assigned to any Content Engine. Packets may not be assigned during initial discovery of Content Engines or when a Content Engine is dropped from a cluster.
Group access-list	Content Engine that is allowed to connect to the router.
Total Messages Denied to Group	Number of messages disallowed by the router because they did not meet all the requirements of the service group.
Total Authentication failures	Number of password authentication failures.

The following example displays output from the **show ip wccp web-cache detail** EXEC command. This command displays Content Engine and WCCP router statistics for a particular service group:

```
show ip wccp web-cache detail

WCCP Router information:
IP Address 172.31.88.10
Protocol Version:2.0

WCCP Cache-Engine Information
IP Address:172.31.88.11
Protocol Version:2.0
State:Usable
Initial Hash Info:AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAA
Assigned Hash Info:FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFF
Hash Allotment:256 (100.00%)
Packets Redirected:21345
Connect Time:00:13:46
```

Table B-3 describes the fields shown in the display.

Table B-3 Field Descriptions—show ip wccp web-cache detail Command

Field	Description
WCCP Router information	Header for the area that contains fields for the IP address and version of WCCP associated with the router connected to the Content Engine in the service group.
IP Address	IP address of the router connected to the Content Engine in the service group.
Protocol Version	Version of WCCP being used by the router in the service group.
WCCP Cache-Engine information	Fields for information on Content Engines.
IP Address	IP address of the Content Engine in the service group.
Protocol Version	Version of WCCP being used by the Content Engine in the service group.
State	Indicates whether the Content Engine is operating properly and can be contacted by a router and other Content Engines in the service group.
Initial Hash Info	Initial state of the hash bucket assignment.
Assigned Hash Info	Current state of the hash bucket assignment.
Hash Allotment	Percentage of buckets assigned to the current Content Engine. Both a value and a percentage figure are displayed.
Packets Redirected	Number of packets that have been redirected to the Content Engine.
Connect Time	Length of time the Content Engine has been connected.

The following is sample output from the **show ip wccp view** EXEC command. In this case, the service number 1 has been specified.

```
show ip wccp service 1 view

WCCP Router Informed of:
192.168.88.10
192.168.88.20

WCCP Content Engines Visible
192.168.88.11
192.168.88.12

WCCP Content Engines Not Visible:
-none-
```

If any Content Engine is displayed under the WCCP Content Engines Not Visible field, the Content Engine needs to be reconfigured to add this router to it.

Table B-4 describes the fields shown in the display.

Table B-4 Field Descriptions—show ip wccp service Command

Field	Description
WCCP Routers Informed of	List of routers detected by the current router.

Table B-4 Field Descriptions—show ip wccp service Command (continued)

Field	Description
WCCP Content Engines Visible	List of Content Engines that are visible to the router and other Content Engines in the service group.
WCCP Content Engines Not Visible	List of Content Engines in the service group that are not visible to the router and other Content Engines in the service group.

Related Commands

Command	Description
ip wccp detail	Directs a router to enable or disable the support for a service group.

Symbols

* wildcard character **7-5**
.pac files **16-3**

A

aborting selected objects **6-3**
accelerated WCCP support **4-10**
administrator
  account **1-2**
  login **1-2**
  password **1-2**
Apache-style transaction logging **9-3**
authenticated data caching **6-1**
authentication
  end-to-end
    basic **10-10**
    NTLM **10-10**
  HTTP request **10-3**
  user
    local **10-1**
    TACACS+ **10-1**
  WMT and NTLM **13-5**
  WMT proxy **13-4**
authentication cache
  size adjustments **10-9**
authentication traffic bypass **4-5**

B

block.html
  customized blocking message **8-3**
boomerang content routing
  enabling **16-1**
browser
  autoconfiguration of **16-3**
  for opening management interface **1-2**
  for verifying software configuration **2-11**

C

Cache application
  disabling transparent caching **2-10**
  feature set (table) **3-1**
  overview **1-1**
cache freshness **6-1**
cache-on-abort **6-3**
caching
  disabling for certain clients **B-9**
  of authenticated content **6-1**
  of binary content with cookies **6-2**
  web traffic **1-1**
CDP. *See* Cisco Discovery Protocol
Cisco Discovery Protocol
  implementing **11-5**
CiscoWorks2000 **11-5**
clear ip wccp command **A-6**
CLF
  log file format **9-1**
commands
  WCCP Version 1 **A-5**
  WCCP Version 2 **B-15**
Common Log Format. *See* CLF
configuring
  Cache application through the GUI **1-2**

healing mode **16-3**
primary proxy server **7-1**
proxy-style caching **5-1**
service group **B-7**
static bypass lists **4-5**
connections
management interface **1-2**
Content Engine
and Layer 4-enabled switches **4-12**
communication with router **B-4**
configuring
as content routing agent **16-1**
for reverse proxy services **2-8**
for WCCP **4-4**
for web cache service **2-3, 2-5**
management interface **1-2, 14-8**
overview **1-1**
removing **2-11**
replacing **2-11**
content preloading **16-5**
content routing
enabling boomerang **16-1**
Content Services Switch. *See* CSS 11000 series switch
conventions **xiii**
cookies **6-2**
CSS 11000 series switch
caching configurations **4-11**
Content Engine CLI commands **4-12**
enabling transparent caching **4-12**
transparent caching with **4-11**
sample configuration output **4-13**
custom blocking messages
in URL filtering **8-3**
customer service and support **xv**
custom web cache
enabling **B-8**
router configuration **B-8**

D

data caching
authenticated **6-1**
debug ip wccp events command **A-20**
debug ip wccp packets command **A-21**
disabling
RealMedia caching **14-11**
transparent caching for certain clients **B-9**
displaying
WCCP settings **B-14**
document conventions **xiii**
DSCP
setting values **15-2**
dynamic traffic bypass **4-6**
dynamic web cache service **B-8**

E

enabling
boomerang content routing **16-1**
custom web cache service **B-8**
dynamic authentication bypass **4-5**
reverse proxy service **B-8**
transparent error handling **4-5**
WCCP support on the router **4-4, A-4**
web caching **B-7**
end-to-end authentication **10-9**
basic **10-10**
NTLM **10-10**
exporting log files **9-4**
Extended Squid-style transaction logging **9-3**
external FTP server
exporting transaction logs to **9-4**
permanent error from **9-6**

F

failover **7-2**

force transparently **7-5**
FTP
  proxy caching **5-2**

G

global exclusion from proxy forwarding **7-5**
graphical user interface **1-2**

H

handling proxy-style requests **7-5**
healing mode
  configuring **16-3**
helper addresses
  IP **A-13**
hierarchical caching **10-7**
home router
  removing Content Engine **2-11**
HTTP
  proxy caching **5-2**
  proxy failover **7-2**
  Range request caching **6-4**
  traffic **2-2 to 2-10**
http cache-on-abort command **6-3**
http cluster command **16-4**
HTTP request authentication
  considerations **10-5**
  excluding domains **10-6**
  using LDAP server **10-5**
  using NTLM server **10-3**
  using RADIUS server **10-4**
HTTPS
  proxy features **5-4**

I

If-Range header **6-4**
Internet Cache Protocol **7-6**
IP address
  valid
    informing router of **B-8**
ip route-cache same-interface command **4-4, A-4**
IP routing
  displaying status of interfaces **A-12**
ip wccp command **B-18**
ip wccp enable command **A-8**
ip wccp group-listen command **B-20**
ip wccp redirect exclude in command **B-21**
ip wccp redirect-list command **A-9**
ip wccp redirect out command **B-22**
ip wccp version command **B-24**
ip web-cache redirect command
  description **A-11**
  enabling web traffic redirection **4-4, A-4**

L

LDAP authentication **10-5, 10-6**
live splitting **13-4**
load balancing **B-6**
load shedding **B-6**
log files
  exporting **9-4**
  restarting export of **9-6**
logging command **1-6**

M

management interface
  logging on **1-2**
  opening **1-2**
mapping of RealProxy error level to syslog priority level **1-6**
maximum object size
  for storing in cache **6-3**
MIBs

supported **11-4**
minimum and maximum Time To Live (TTL) settings **6-2**
monitoring
WCCP Version 1 **A-4**
WCCP Version 2 **B-11**
multicast address
registering a router **B-8**
multiple router support **B-2**
multiport transparent redirection **4-9**

N

N2H2 URL filtering **8-4**
near-miss
definition **16-3**
Network Time Protocol (NTP) **1-7**
NTLM
authenticated content **6-1**
authentication support **10-3**
end-to-end authentication
object caching **10-11**
pass-through service **10-10**
HTTP request authentication **10-3**

O

object size
maximum for cache **6-3**
overload bypass **4-7**

P

packet marking **15-2**
parent proxy failover **7-2**
password
setting **B-9**
port 80 **1-1**
preloading
of content **16-5**
primary proxy server **7-1**
proxy authentication **13-4**
proxy failover **7-2**
proxy mode
LDAP authentication **10-6**
operation **5-1**
proxy server
primary **7-1**

R

RADIUS authentication **10-4**
Range requests **6-4**
RealMedia caching
disabling **14-11**
RealPlayer **14-6**
RealProxy
and access control **14-11**
CLI commands **14-3**
configuring **14-3**
conventional RTSP proxy service **14-5**
disabling RealMedia caching **14-11**
error codes **1-7**
features and benefits **14-2**
transparent RTSP proxy service **14-4**
RealSystem administrator page **14-8, 14-9**
redirection process **A-2**
registering router
to a multicast address **B-8**
removing
Content Engine **2-11**
replacing
Content Engine **2-11**
reverse proxy service
configuring **2-7**
enabling **B-8**
router

basic configuration for HTTP traffic and WWCP Version 2 **2-2**
communication with Content Engine **B-4**
configuring
for reverse proxy service **2-8**
for web cache service **2-3, 2-5**
enabling WCCP support **4-4, A-4**
indicating valid IP addresses **B-8**
registering to a multicast address **B-8**
removing Content Engine **2-11**
WCCP configuration example **A-4**
RTSP proxy service
conventional **14-5**
transparent using WCCP-enabled routers **14-4**
Rules Template
actions and patterns **15-2**
configuring **15-1**
order of executing actions **15-5**
order of executing patterns **15-6**
setting ToS or DSCP **15-2**

S

sanitized transaction logs **9-4**
server redundancy **10-7**
service group
configuring using WCCP Version 2 **B-7**
setting password **B-9**
sfadmin console **8-8**
show ip interface command **A-4, A-12, B-25**
show ip wccp command **A-4, A-15, B-17, B-28**
show ip wccp web-caches command **A-4, A-17**
show running-config command **B-9**
SmartFilter software **8-7**
SNMP
key CLI commands **11-3**
overview **11-1**
security models and security levels **11-3**
supported MIBs **11-4**
traps **11-4**
versions supported **11-2**
specifying a domain name **7-5**
Squid-style transaction logging **9-1**
SSL tunneling **5-3**
static bypass **4-8**
streaming media
overview **13-1**
supported protocols **13-2**
syslog
priority level mapping to RealProxy error codes **1-7**
system logging
setting specific parameters for syslog file **1-6**

T

TACACS+
options for user authentication **10-2**
TCP
configuring parameters using Content Engine GUI **12-2**
over satellite extensions **12-3**
port 80 **1-1**
web traffic **1-1**
TCP redirection process **B-5**
technical support **xv**
Time To Live (TTL)
minimum and maximum settings **6-2**
ToS
setting values **15-2**
transaction logging
after authentication using LDAP **10-9**
and WMT **13-5**
Apache-style format **9-3**
Extended squid-style format **9-3**
sanitized **9-4**
Squid-style format **9-1**
transparent caching
configuring **4-1 to 6-5**

disabling **2-10**
transparent mode
hierarchical caching in **10-8**
LDAP authentication **10-6**
operation **4-2**

U

URL **8-4**
url-filter command
order of precedence over rules command **8-1**
URL filtering
configuring **8-1**
with N2H2 server **8-4**
with SmartFilter software **8-7**
with Websense server **8-6**
user authentication
local **10-1**
TACACS+ **10-1**

V

valid IP address
indicating to router **B-8**
Variable **13-3**
variable bit rate **13-3**
verifying
software configuration **2-11**
WCCP configuration settings **B-9**

W

WCCP
and transparent caching **4-1**
displaying settings **B-14**
enabling
support on the router **4-4, A-4**
Version 1 on router **4-4, A-4**
flow protection **4-9**
monitoring **A-4**
Version 1
commands **A-5**
monitoring **A-4**
operation **B-2**
overview **A-1**
prerequisites **4-4, A-3**
supported platforms **4-4, A-3**
Version 2
basic router configuration for HTTP traffic **2-2**
commands **B-15**
configuration examples **B-12**
configuring a service group **B-7**
monitoring **B-11**
operation **B-2**
overview **B-1**
prerequisites **B-6**
restrictions **B-6**
supported platforms **B-6**
verifying configuration settings **B-9**
Web Cache Communication Protocol. *See* WCCP
web cache packet return **B-5**
web cache service
clients and cache on different subnets **2-2**
clients and cache on the same subnet **2-4**
running **B-7**
web caching
enabling **B-7**
web management interface connection **1-2**
Websense server
and URL filtering **8-6**
web server acceleration **2-7**
wildcard character * **7-5**
WMT
authentication using Windows NTLM protocol **13-5**
caching overview **13-3**
conventional proxy service **13-8**
enabling on the Content Engine **13-6**

live splitting **13-4**
multicasting **13-12**
multicast-in and multicast-out **13-14**
multicast-in and unicast-out **13-15**
unicast-in and multicast-out **13-13**
proxy overview **13-2**
support for variable bit rate **13-3**
transaction logging **13-5**
transparent caching using WCCP **13-6**